FIS Digital Payments Mobile SDK Integration Guide

Updated: October 2021

Notice

The information contained in or supplied with this document is provided solely for the purpose of utilizing the products and services of Fidelity National Information Services, Inc, and/or its affiliates and subsidiaries (collectively “FIS”). Such information may not be copied by or disclosed to any person or entity without the prior written consent of FIS.

The information contained or supplied with this document shall be used in accordance with the terms of the agreement that currently governs your receipt and use of FIS software, products, and/or services, and shall in no way be used by any party to the competitive disadvantage of FIS.

FIS makes no warranty, express or implied, with respect to the quality, accuracy or completeness of this document. FIS makes no representation or warranty with respect to the contents of this document and specifically disclaims any implied warranties of fitness for any particular purpose and liability for any direct, indirect, incidental or consequential, special or exemplary damages, including but not limited to, lost profits resulting from the use of the information in the document or from the use of any products described in this document.

**Trademarks**

All trademarks are the property of their respective owners.

Company, product, and service names used by FIS within, or supplied with this document may be trademarks or service marks of other persons or entities.

**Copyright**

Copyright©2021 FIS

All Rights Reserved

This document is protected under copyright and trade secret laws.

Unauthorized copying, reproduction, modification, display, use, and distribution of this document are strictly prohibited.

Table of Contents

[Introduction 5](#_Toc83802438)

[Integration Overview 6](#_Toc83802439)

[Security 6](#_Toc83802440)

[Whitelisting 6](#_Toc83802441)

[Cross-site Scripting (XSS) 7](#_Toc83802442)

[Permissions 7](#_Toc83802443)

[Error Handling 7](#_Toc83802444)

[Event Handling (Optional) 7](#_Toc83802445)

[SDK Testing 8](#_Toc83802446)

[Platform-Specific Integration 9](#_Toc83802447)

[Android 9](#_Toc83802448)

[Event Handling 10](#_Toc83802449)

[Manifest/Permissions 10](#_Toc83802450)

[Prominent Disclosure 11](#_Toc83802451)

[Back Button 11](#_Toc83802452)

[iOS 12](#_Toc83802453)

[Event Handling 12](#_Toc83802454)

[Permissions 13](#_Toc83802455)

# Introduction

The FIS Digital Payments Mobile SDK is a library that allows the FIS Zelle web application to use device contacts to prepopulate forms when adding contacts to Zelle.

The SDK works by attaching interfaces to web view components. The attached interfaces allow communication between the Zelle web page and the device.

# Integration Overview

There are minor differences in how each OS integrates with web applications but the SDK integration steps are similar:

1. Import the module or framework.
2. Create and configure the platform-specific web view component.
3. Create the SDK with a reference to the web view.
4. Optionally implement event handlers.
5. Load the web application into the web view.
6. Dispose of the SDK when done with the web application.

## Security

### Whitelisting

When creating an SDK bridge connection object, a list of whitelisted domains must be provided. This is the list of domains that are allowed to access native device features. This list should only include domains from which Digital Payments web applications are served. This is the domain that is returned by the SingleSignOn web service call. This list must be populated to support any test and production environments.

Each domain in the whitelist must be the full domain including any subdomains. For example, if the web application is loaded from <https://epayments.epayui.billdomain.com/FIS/RemoteAccess>, the entry in the whitelist should be epayments.epayui.billdomain.com.

Before access to a native device feature is granted, the domain that is currently loaded in the web view component is validated against the whitelist. This ensures native functionality and information is only granted to approved domains and will not be available if the web view component is redirected.

### Cross-site Scripting (XSS)

All data passing through the SDK to the web application is validated and encoded to protect against XSS vulnerabilities.

### Permissions

Contact permission is acquired using guidelines specified by Apple and Google.

## Error Handling

The SDK should never throw an exception or cause an error condition for which the native app needs to respond. All errors will be handled and logged by the web application. Error information is propagated to the native app using the event handling mechanism described in this document.

## Event Handling (Optional)

Any communication between the native app and SDK is done by registering an event handler. Platform-specific details are found in the following sections.

An event object is provided whenever an event occurs. The event type can be inspected to take appropriate action. The SDK will raise events with the following types:

| **Event Type** | **Description** |
| --- | --- |
| STARTED | Raised when a process starts, typically after a web page loads after SingleSignOn. This can be used to hide progress indicators when the site is ready. |
| COMPLETED | Raised when a process completes. Can be used to navigate to the native app when a user indicates they are done with the web application. |
| BACK\_BUTTON | Raised when the user presses the back button and the web application is at the beginning of a flow. This event will only be raised on Android devices. |

## SDK Testing

An external site for testing SDK functionality can be found at <https://epayments-crossdomain-fnc.billdomain.com/Mobile/Index.html>.

This site can be loaded in a web view to quickly ensure the SDK is integrated properly.

# Platform-Specific Integration

## Android

The Android SDK is packaged with an associated pom file so it can be hosted in a maven repository. This is the recommended approach for including modules. When using maven, the SDK can be added as a dependency in build.gradle with a statement such as:

implementation 'com.fis:digital-payments-sdk:2.4.3'

If maven is not available, the SDK can be included through flat directory repository.

Once the SDK module is imported, it needs to be initialized with the WebView through the SDKManager.create static method. This method has the following parameters:

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| context | Context of the application or activity. |
| webView | WebView hosting a Digital Payments web application.  **Note**: WebView must be JavaScript enabled before calling the register method or an exception will be thrown. |
| domainWhitelist | Array of whitelisted domains allowed to access SDK functionality. |
| enableDisclosure | Optional; defaults to true but can be disabled by passing false. |
| webEventListener | Optional implementor of WebEventListener interface for event handling. |

When done with the Zelle web application it is good practice to call the dispose method of the SDKManager to remove the interfaces from the web view. Note that the iOS dispose method is a static method of SDKManager and takes a WKWebView instance while the Android dispose method is an instance method of SDKManager. This is due to differences in how the native device features are supported.

### Event Handling

To handle events, include an implementation of the WebEventListener interface in the SDKManager.create call.

Below is a sample implementation of this interface:

@Override

public void onEventReceived**(**WebEvent event**)** **{**

WebEventType eventType **=** event**.**getEventType**();**

**if** **(**eventType **==** WebEventType**.**STARTED**)** **{**

// A web application process has started. This event can be used to stop a progress indicator

**}**

**else** **if** **(**eventType **==** WebEventType**.**COMPLETED**)** **{**

// A web application process has completed. This event can be used to trigger navigation back to the native app. Typically only used with photo bill pay.

**}**

**}**

When an event is raised, this method will be called and the event object can be queried to determine the appropriate action.

### Manifest/Permissions

The SDK module has its own manifest describing all required features, permissions, and activities. The manifest is merged with the main application manifest during the build process.

READ\_CONTACTS permission is merged with the manifest when including the contacts module.

### Prominent Disclosure

Google requires users be informed of any data collected by an app that is not essential to its basic functions. Device contacts fall in this category and their usage must be disclosed. The SDK contains a disclosure with default language that will appear before a user is prompted for contact access.

Some apps may not need this disclosure as it may be present elsewhere within the app. In this case, the enableDisclosure flag can be set to false when calling SDKManager.create (see Android section).

The text can be customized by overriding the following strings in the app’s strings.xml file:

|  |  |
| --- | --- |
| **Name** | **Default Language** |
| fis\_contact\_usage\_disclosure\_title | Here's how we'll use your contacts |
| fis\_contact\_usage\_disclosure\_message | We would like to access your contact names, emails, and phone numbers to help make it easier to find and pay people you trust. This information is used to prepopulate forms when adding a Zelle contact for the purposes of making a payment to someone you know. This information is not shared with any other party or used for any other purposes. You can turn off contact permission for this app in Settings at any time. |
| fis\_contact\_usage\_disclosure\_allow | Allow |
| fis\_contact\_usage\_disclosure\_do\_not\_allow | Don’t Allow |

### Back Button

Android users may be accustomed to using the back button for navigation. The SDKManager object has an optional utility method named checkNavigateBack that can be used in the onKeyDown method of an activity to allow the web application to respond to back button presses. The web application will handle back button presses and navigate back until the beginning of a flow. If the back button is pressed at the beginning of a flow, an event of type BACK\_BUTTON will be raised. This event can be handled to navigate away from the web application.

## iOS

iOS integration uses an xcframework in conjunction with WKWebView.

The SDK framework must be included as an embedded binary within the hosted app.

Once the SDK framework is embedded, it must be initialized with the WKWebView through the FDSSDKManager.create static method. This method has the following parameters:

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| webView | WKWebView hosting a Digital Payments web application. |
| domainWhitelist | Array of whitelisted domains allowed to access SDK functionality. |
| webEventListener | Optional implementor of FDSWebEventListening interface for event handling. |

When done with the Zelle web application it is good practice to call the dispose method of the SDKManager to remove the interfaces from the web view. Note that the iOS dispose method is a static method of SDKManager and takes a WKWebView instance while the Android dispose method is an instance method of SDKManager. This is due to differences in how the native device features are supported.

### Event Handling

To handle events, include an implementation of the FDSwebEventListener protocol to the FDSSDKManager.register call.

The following is a sample implementation of this protocol:

func onEventReceived**(**webEvent**:** FDSWebEvent**)** **{**

let webEventType **=** webEvent**.**webEventType

**if** eventType **==** WebEventType**.**STARTED

// A web application process has started. This event can be used to stop a progress indicator

**}**

**else** **if** eventType **==** WebEventType**.**COMPLETED **{**

// A web application process has completed. This event can be used to trigger navigation back to the native app. Typically only used with photo bill pay.

**}**

**}**

### Permissions

The app’s info.plist file must contain a usage description for contacts to or it will crash when prompting for permission.