Turnkey Service for Zelle®: Mobile SDK

Technical Integration Guide

May 2023

© 2021-2023 Fiserv, Inc. or its affiliates. All rights reserved. This work is confidential and its use is strictly limited. Use is permitted only in accordance with the terms of the agreement under which it was furnished. Any other use, duplication, or dissemination without the prior written consent of Fiserv, Inc. or its affiliates is strictly prohibited. The information contained herein is subject to change without notice. Except as specified by the agreement under which the materials are furnished, Fiserv, Inc. and its affiliates do not accept any liabilities with respect to the information contained herein and are not responsible for any direct, indirect, special, consequential or exemplary damages resulting from the use of this information. No warranties, either express or implied, are granted or extended by this document.

<http://www.fiserv.com>

Fiserv is a registered trademark of Fiserv, Inc.

Zelle® and the Zelle® related marks are wholly owned by Early Warning Services, LLC and are used herein under license.

Other brands and their products are trademarks or registered trademarks of their respective holders and should be noted as such.

Contents

[Revision History 5](#_Toc133844815)

[Document Purpose 7](#_Toc133844816)

[Mobile SDK 8](#_Toc133844817)

[Mobile SDK Capabilities 9](#_Toc133844818)

[Access to Contacts 9](#_Toc133844819)

[Access to Camera 9](#_Toc133844820)

[Access to Photo Gallery 9](#_Toc133844821)

[Access to Share Sheet (SMS, Email, Etc.) 9](#_Toc133844822)

[Implementation Overview 10](#_Toc133844823)

[Sequence to Launch Zelle® Web UI 10](#_Toc133844824)

[Bill Pay Sequence to Launch Zelle® Web UI 11](#_Toc133844825)

[Bill Pay Sequence to Launch Zelle® Web UI (Zelle Deep Link) 11](#_Toc133844826)

[Sample Sequences for Native Functions 12](#_Toc133844827)

[Sample Sequence – QR Code 12](#_Toc133844828)

[Bill Pay Sample Sequence – QR Code 13](#_Toc133844829)

[Sample Sequence – Zelle® Ready Contacts 14](#_Toc133844830)

[Bill Pay Sample Sequence – Zelle® Ready Contacts 15](#_Toc133844831)

[Mobile SDK Parameters 16](#_Toc133844832)

[Sample 16](#_Toc133844833)

[Implementing Mobile SDK Within Android App 18](#_Toc133844834)

[Steps for Quick Start 18](#_Toc133844835)

[Project Setup 18](#_Toc133844836)

[Device Orientation Support (Optional) 18](#_Toc133844837)

[Implementation Steps with Pseudocode (Kotlin) 18](#_Toc133844838)

[Implementation Steps with Pseudocode (Java) 21](#_Toc133844839)

[Supported Versions 23](#_Toc133844840)

[Zelle Mobile SDK Size 23](#_Toc133844841)

[Dependency 23](#_Toc133844842)

[Implementing Mobile SDK Within iOS App 24](#_Toc133844843)

[Steps for Quick Start 24](#_Toc133844844)

[Project Setup 24](#_Toc133844845)

[Implementation Steps with Pseudocode (Swift) 25](#_Toc133844846)

[Implementation Steps with Pseudocode (Objective-C) 26](#_Toc133844847)

[Supported Versions 28](#_Toc133844848)

[Zelle Mobile SDK Size 28](#_Toc133844849)

[Dependency 28](#_Toc133844850)

[Additional Information 29](#_Toc133844851)

[Related Document 29](#_Toc133844852)

[SSO Process 29](#_Toc133844853)

[Terminology 29](#_Toc133844854)

# 

# Revision History

| Date | Revision | Description |
| --- | --- | --- |
| 08-04-2021 | 0.1 | Initial draft |
| 09-21-2021 | 1.0 | Initial version |
| 02-14-2022 | 1.1 | “Mobile SDK Parameters”:   * Added product parameter * Updated descriptions for applicationName and parameters * Updated sample   “Implementing Mobile SDK Within Android App”:   * Updated “Project Setup” (including changing BridgeSDK to ZelleSDK) * Updated “Implementation Steps with Pseudocode (Kotlin)” and “Implementation Steps with Pseudocode (Java)” * Updated minimum supported SDK version * Added Zelle Mobile SDK size and QR Code library dependency   “Implementing Mobile SDK Within iOS App”   * Updated “Project Setup” (including changing BridgeSDK to ZelleSDK) * Updated “Implementation Steps with Pseudocode” * Updated “Supported Versions” * Added Zelle Mobile SDK size and QR Code library dependency |
| 3-15-2022 | 1.2 | Updated “Sequence to Launch Zelle® Web UI,” “Sample Sequence – QR Code,” and “Sample Sequence – Zelle® Ready Contacts” to show the BridgeView class (replacing the previous instance of ZelleView) |
| 5-16-2022 | 1.3 | Clarified that QR codes are a future enhancement.  “Implementing Mobile SDK Within Android App”:   * Changed zxing:core version to 3.3.3 * Added “Device Orientation Support” * Specified that “Launch Zelle® inside another view using code initialization” is the recommended approach * Changed minimum supported SDK version to 24 * Changed minimum supported OS to 7.0 (Nougat)   “Implementing Mobile SDK Within iOS App”   * Removed the key NSLocationWhenInUseUsageDescription from info.plist * Changed “Implementation Steps with Pseudocode” to “Implementation Steps with Pseudocode (Swift)” * Specified that “Launch Zelle® inside another view using code initialization” is the recommended approach * Added “Launch Zelle® inside another view and fill it (for screen orientation)” * Updated “Launch Zelle® as a popup (not inside another view)” * Added “Implementation Steps with Pseudocode (Objective-C)” |
| 8-19-2022 | 1.4 | “Mobile SDK Parameters”:   * Added appData parameter to support prominent disclosures * Updated sample   “Implementing Mobile SDK Within Android App”:   * “Implementation Steps with Pseudocode (Kotlin)”: Updated code sample for creating a Zelle bridge configuration object. Added “Session Timeout.” * “Implementation Steps with Pseudocode (Java)”: Updated code sample for creating a Zelle bridge configuration object. Added “Session Timeout.”   “Implementing Mobile SDK Within iOS App”   * “Implementation Steps with Pseudocode (Swift)”: Added “Session Timeout.” * “Implementation Steps with Pseudocode (Objective-C)”: Added “Session Timeout.” |
| 9-09-2022 | 1.5 | “Implementing Mobile SDK Within Android App”:   * “Implementation Steps with Pseudocode (Java)”: Added note to explain how to pass null for optional parameter.   “Implementing Mobile SDK Within iOS App”   * “Implementation Steps with Pseudocode (Objective-C)”: Added note to explain how to pass null for optional parameter. |
| 01-11-2023 | 1.6 | Added a diagram that shows the high-level view if Zelle® is accessed through Bill Pay.  Added “Bill Pay Sequence to Launch Zelle® Web UI,” “Bill Pay Sequence to Launch Zelle® Web UI (Zelle Deep Link),” “Bill Pay Sample Sequence – QR Code,” and “Bill Pay Sample Sequence – Zelle® Ready Contacts”  “Mobile SDK Parameters”:   * Updated description for parameters to explain that custom name value pairs are optional except if Zelle is accessed through Bill Pay. If Zelle is accessed through Bill Pay, this name value pair is required: flowtype=BillPay * Updated sample   “Implementing Mobile SDK Within Android App”:   * “Implementation Steps with Pseudocode (Kotlin)” and “Implementation Steps with Pseudocode (Java)”: Updated for Bill Pay   “Implementing Mobile SDK Within iOS App”   * “Implementation Steps with Pseudocode (Swift)” and “Implementation Steps with Pseudocode (Objective-C)”: Updated for Bill Pay.   “Additional Information”   * Added the related document *CheckFree® Mobile Web: Mobile Integration Using WebView*. * Added Bill Pay to Terminology. |
| 05-04-2023 | 1.7 | “Mobile SDK Parameters”:   * Added fi\_callback and loaderData parameters * Updated sample   “Implementing Mobile SDK Within Android App”:   * “Implementation Steps with Pseudocode (Kotlin)”: Updated code sample for creating a Zelle bridge configuration object. Added getValue method to “Session Timeout and Intercepting Web Links.” * “Implementation Steps with Pseudocode (Java)”: Updated code sample for creating a Zelle bridge configuration object. Added applicationName and loaderData to note explaining how to pass null for optional parameters. Added getValue method to “Session Timeout and Intercepting Web Links.”   “Implementing Mobile SDK Within iOS App”   * “Implementation Steps with Pseudocode (Swift)”: Updated code sample for creating a Zelle bridge configuration object. Added “Set the GenericTagDelegate Protocol delegate to the existing ViewController.” Added getValue method to “Session Timeout and Intercepting Web Links.” * “Implementation Steps with Pseudocode (Objective-C)”: Updated code sample for creating a Zelle bridge configuration object. Added note explaining how to pass null for optional parameters. Added “Set the GenericTagDelegate Protocol delegate to the existing ViewController.” Added getValue method to “Session Timeout and Intercepting Web Links.” * Updated Zelle Mobile SDK size |

# Document Purpose

This document serves as a technical integration guide for the Turnkey Service for Zelle®: Mobile SDK (Mobile SDK). This technical integration guide is intended to provide the mobile banking application provider with detailed information for upgrading the current Zelle® WebView implementation to include the Mobile SDK.

This guide includes sample code snippets as part of the Mobile SDK upgrade project with Fiserv.

# Mobile SDK

The Turnkey Service for Zelle®: Mobile SDK (Mobile SDK) offers a turnkey solution for enabling an enhanced mobile user experience. The current web based hosted Zelle® UI offers the flexibility to be enabled in both online and mobile banking applications, but it does not enable access to the mobile device capabilities (e.g., contact list, camera, photo gallery, and sharing capabilities) that are expected with a native mobile app experience.

The Mobile SDK (native iOS and Android), integrated with the hosted Zelle® UI and the parent mobile banking application, will enable features that require access to the device capabilities. Those features are:

* Zelle® Ready Contacts
* QR Codes
* Profile Pictures (future enhancement)

In the current state, the Zelle® UI relies on the parent mobile application to access device capabilities. With the Mobile SDK, the SDK is responsible for accessing device capabilities.

This illustration provides a high-level view of the change in approach for the hosted Zelle® UI.

A picture containing graphical user interface

Description automatically generated

This illustration provides a high-level view if Zelle® is accessed through Bill Pay.

Diagram

Description automatically generated with low confidence

## Mobile SDK Capabilities

### Access to Contacts

SDK provides access to contacts on a device and handles the user’s permission for native app to access contacts to add a single contact as a recipient or to pull a list of contacts who are enrolled in Zelle® and display those contacts in a recipient list as “Zelle Ready Contacts.”

### Access to Camera

SDK provides access to the camera on a device and handles the user’s permission for native app to access camera for scanning the QR code of a user’s token for sending or requesting a payment.

### Access to Photo Gallery

SDK provides access to the photo gallery on a device and handles the user’s permission for native app to access photos of the QR code of a user’s token for sending or requesting a payment.

### Access to Share Sheet (SMS, Email, Etc.)

SDK provides the ability to share or print QR codes via the share tray on a device.

# Implementation Overview

The Mobile SDK leverages native app binaries to provide user access to native capabilities (e.g., contact list, camera, etc.).

## Sequence to Launch Zelle® Web UI

The Mobile SDK replaces the existing WebView implementation by replacing WebView with BridgeView.

Note:

Refer to the *Turnkey Service for Zelle® IDD* for a description of the SSO process and a list of all applicable SSO elements.

Diagram

Description automatically generated with low confidence

## Bill Pay Sequence to Launch Zelle® Web UI

Diagram

Description automatically generated

## Bill Pay Sequence to Launch Zelle® Web UI (Zelle Deep Link)

Diagram

Description automatically generated

## Sample Sequences for Native Functions

### Sample Sequence – QR Code

The Mobile SDK supports scanning of a Zelle® QR code within the send and other required flows using a native camera app.

Diagram

Description automatically generated

### Bill Pay Sample Sequence – QR Code

Diagram

Description automatically generated

### Sample Sequence – Zelle® Ready Contacts

The Mobile SDK supports displaying Zelle® Ready contacts within the Select Recipient screen if the user has provided permissions to access their phone contacts.

Diagram

Description automatically generated

### Bill Pay Sample Sequence – Zelle® Ready Contacts

Diagram

Description automatically generated

## Mobile SDK Parameters

| Parameter | Description | Example |
| --- | --- | --- |
| applicationName | Name of the application shown on custom popups. This is an optional parameter. | “Demo Bank” |
| baseURL | Base URL to launch Web UI | CERT URL: https://certtransfers.fta.cashedge.com/popnet/faces/loginServlet  The PROD URL differs based on client implementation. |
| institutionId | Financial institution identifier | 88850000 |
| product | Product name | Zelle |
| ssoKey | SSO key received from SSO response | 70770198b7340d8a2b0431052b8da3ff |
| fi\_callback | If the value of this parameter is true, web links on the Zelle® UI such as "Privacy Policy" or "Contact Us" can be intercepted and then handled by the parent mobile application | true/false |
| loaderData | Data used for custom loader  Default loaderColor= “#FFFFFF” Default bgColor= “#747474” | loaderColor=value, bgColor=value |
| appData | Application data for app customization | pd\_contact=value, pd\_camera=value, pd\_gallery=value |
| parameters | Custom name value pairs (optional except if Zelle® is accessed through Bill Pay) | key1=value1, key2=value2, etc. |
|  | If Zelle® is accessed through Bill Pay, this name value pair is **required**:  **flowtype=BillPay** | flowtype=BillPay, key1=value1, etc. |

### Sample

val pdContact = mapOf(  
 **"title"** to **"We would like to access your phone contacts"**,  
 **"message"** to **"We only sync phone numbers and email addresses from your contact list to help you add and pay a new recipient in Zelle®"**  
)

val pdCamera = mapOf(  
 **"title"** to **"We would like to access your camera"**,  
 **"message" to "We only access your camera to help you add and pay a new recipient in Zelle®"**  
)

val pdPhotos = mapOf(  
 **"title"** to **"We would like to access your photos"**,  
 **"message" to "We only access your photos to help you add and pay a new recipient in Zelle®"**  
)

val zelle = Zelle (   
 applicationName = **"Demo Bank"**, **// Optional**  
 baseURL = **"https://certtransfers.fta.cashedge.com/popnet/faces/loginServlet"**,  
 institutionId = **"88850000"**,  
 product = **"zelle"**,  
 ssoKey = **"e78abf35705a6d9b51fbf3939aa82489"**, **// Optional**   
 fi\_callback = **true**, // Mandatory when handling getValue method otherwise optional  
 loaderData= mapOf(  
 **"loaderColor"** to **"**hex color**"**,  
 **"bgColor"** to **"**hex color**"**  
 ), **// Optional**

appData= mapOf(  
 **"pd\_contact"** to pdContact,  
 **"pd\_camera"** to pdCamera,  
 **"pd\_gallery"** to pdPhotos  
 ), **// Optional**

parameters = mapOf( **// Name value pairs are optional unless Zelle**  
 **"key1"** to **"value1"**, **// is accessed through Bill Pay**  
 **"key2"** to **"value2"**,  
 **"key3"** to **"value3"**   
 **"flowtype"** to **"BillPay" // Required if Zelle is accessed through Bill Pay**  
 )

)

# Implementing Mobile SDK Within Android App

## Steps for Quick Start

### Project Setup

* Add ZelleSDK.aar to your project libs folder:

A screenshot of a computer

Description automatically generated with medium confidence

* Add the following line to the app module Gradle file dependencies:

|  |
| --- |
| implementation files ('libs/ZelleSDK.aar')  //QR Code reader  implementation('com.journeyapps:zxing-android-embedded:4.2.0') {transitive = false}  implementation 'com.google.zxing:core:3.3.3' |

* Import ZelleSDK where needed in any source code file:

|  |
| --- |
| import com.fiserv.dps.mobile.sdk.bridge.model.\* import com.fiserv.dps.mobile.sdk.bridge.controller.Bridge |

### Device Orientation Support (Optional)

* Add the configChanges below to the launching ZelleLaunchingActivity class in the Manifest file:

|  |
| --- |
| android:configChanges="keyboardHidden|orientation|screenSize|layoutDirection|uiMode" |

### Implementation Steps with Pseudocode (Kotlin)

* Create a Zelle bridge configuration object.

|  |
| --- |
| val pdContact = mapOf(  "title" to "We would like to access your phone contacts",  "message" to "We only sync phone numbers and email addresses from your contact list to help you add and pay a new recipient in Zelle®" ) val pdCamera = mapOf(  "title" to "We would like to access your camera",  "message" to "We only access your camera to help you add and pay a new recipient in Zelle®" ) val pdPhotos = mapOf(  "title" to "We would like to access your photos",  "message" to "We only access your photos to help you add and pay a new recipient in Zelle®" )  val zelle = Zelle(  applicationName = "Demo Bank", //Optional  baseURL = "https://certtransfers.fta.cashedge.com/popnet/faces/loginServlet",  institutionId = "88850000",  product = "zelle",  ssoKey = "e78abf35705a6d9b51fbf3939aa82489",  fi\_callback = true, // Mandatory when handling getValue method otherwise optional  loaderData= mapOf(  "loaderColor" to "hex color",  "bgColor" to "hex color"  ), //Optional  appData= mapOf(  "pd\_contact" to pdContact,  "pd\_camera" to pdCamera,  "pd\_gallery" to pdPhotos  ), //Optional  parameters = mapOf(  "key1" to "value1",  "key2" to "value2",  "key3" to "value3"  ) //Optional unless Zelle is accessed through Bill Pay ) |

Note:

ZelleSDK will **automatically** add the appropriate default values for these parameters to the URL: product.version and container (mobile\_sdk\_android).

* Create a Bridge object (optional lazy implementation).

Note:

Pass the appropriate parent appActivity (type Activity) to activity.

|  |
| --- |
| private val bridge: Bridge by lazy {   Bridge(  activity = this,   config = zelle   ) } |

#### Launch Zelle® inside another view using code initialization (Recommended)

If replacing an existing WebView, replace existing WebView with BridgeView.

|  |
| --- |
| // optionally: set the contact pre-caching flag (default: false) zelle.preCacheContacts = true  val zelleView = bridge.view() supportFragmentManager.beginTransaction().apply {  replace(R.id.desiredLocation, zelleView)   commit () } |

#### Launch Zelle® as a popup (not inside another view)

|  |
| --- |
| // optionally: set the contact pre-caching flag (default: false) zelle.preCacheContacts = true  val zellePopup = bridge.popup() zellePopup.show(supportFragmentManager, zellePopup.tag) |

#### Session Timeout and Intercepting Web Links

|  |
| --- |
| //Implement the GenericTag with Activity/Fragment  Class ZelleLaunchFragment: Fragment (), GenericTag{  override fun onCreateView(inflater: LayoutInflater, container ViewGroup?, savedInstanceState: Bundle?) {  \_binding = FragmentZelleLaunchBinding.inflater.inflate(inflater, container, false)  //Initialize generic tag  genericTag = this  return binding.root  } }  //Inside the class override the sessionTag method & getValue method  override fun sessionTag(name: String) {  if (name == “Landing”){  //Here navigates the application to the desired screen. (This function will be triggered after the session expires)   } }  // If the parent app has passed true for the fi\_callback parameter, if the user  // clicks on a web link such as the "Privacy Policy" link on the Zelle UI, then  // the getValue method will be triggered and pass "privacy policy" as the value  // for the name parameter. The parent app handles this callback on their side.  override fun getValue(name: String) {      if (name == “TAG\_NAME”) {          //Here navigates the application to the desired screen. (This function will help to communicate between Zelle UI and parent app)      }  } |

### Implementation Steps with Pseudocode (Java)

* Create a Zelle bridge configuration object.

|  |
| --- |
| HashMap<String, String> params = new HashMap<String, String> (); params.put("key1", "value1"); params.put("key2", "value2"); params.put("key3", "value3");  HashMap<String, String> loaderData = new HashMap<String, String> (); loaderData.put("loaderColor", "hex color"); loaderData.put("bgColor", "hex color");  Map<String, String> pdContact= new HashMap<String, String> (); pdContact.put("title", "We would like to access your phone contacts"); //prominent disclosure title pdContact.put("message", "We only sync phone numbers and email addresses from your contact list to help you add and pay a new recipient in Zelle®"); //prominent disclosure message  Map<String, String> pdCamera= new HashMap<String, String> (); pdCamera.put("title", "We would like to access your camera"); //prominent disclosure title pdCamera.put("message", "We only access your camera to help you add and pay a new recipient in Zelle®"); //prominent disclosure message  Map<String, String> pdPhotos= new HashMap<String, String> (); pdPhotos.put("title", "We would like to access your photos"); //prominent disclosure title pdPhotos.put("message", "We only access your photos to help you add and pay a new recipient in Zelle®"); //prominent disclosure message  Map<String, Map<String, String>> appData = new HashMap<String, HashMap<String, String>> ();  appData.put(“pd\_contact”, pdContact); appData.put(“pd\_camera”, pdCamera); appData.put(“pd\_gallery”, pdPhotos);  Zelle zelle = new Zelle(  "Demo Bank", //applicationName (Optional)  "https://certtransfers.fta.cashedge.com/popnet/faces/loginServlet", //baseURL   "88850000", //institutionId  "zelle", //product  "e78abf35705a6d9b51fbf3939aa82489", // ssoKey  true, //fi\_callback is true when handling getValue method otherwise false  loaderData, //Optional (Nullable)  appData, //Optional (Nullable)  params //Optional (Nullable) unless Zelle is accessed through Bill Pay ); |

Note:

Since applicationName, loaderData, and appData are optional parameters, you may pass null (e.g., **appData: null**).

Note:

params is an optional parameter unless Zelle® is accessed through Bill Pay. If Zelle® is accessed through Bill Pay, the name value pair **flowtype=BillPay** is required. Otherwise, you may pass null (e.g., **params: null**).

Note:

ZelleSDK will **automatically** add the appropriate default values for these parameters to the URL: product.version and container (mobile\_sdk\_android).

* Create a Bridge object.

Note:

Pass the appropriate parent appActivity (type Activity) to activity.

|  |
| --- |
| Bridge bridge = new Bridge(this, zelle); |

#### Launch Zelle® inside another view using code initialization (Recommended)

|  |
| --- |
| // optionally: set the contact pre-caching flag (default: false) zelle.preCacheContacts = true  BridgeView zelleView = bridge.view(); //user supportFragmentManager to push zelleView to desired location |

#### Launch Zelle® as a popup (not inside another view)

|  |
| --- |
| // optionally: set the contact pre-caching flag (default: false) zelle.preCacheContacts = true  BridgePopup zellePopup = bridge.popup(); popup.show(supportFragmentManager, zellePopup.getTag()); |

#### Session Timeout and Intercepting Web Links

|  |
| --- |
| //Implement the GenericTag with Activity/Fragment  public class ZelleLaunchFragment extends Fragment implements GenericTag{  @Override  public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState){  //Initialize generic tag  Bridge.genericTag = this  return inflater.inflate(R.layout.fragment\_zelle\_launch, container, false)  } }  //Inside the class override the sessionTag & getValue method  @Override  public void sessionTag (String name) {  if (name == “Landing”) {  //Here navigates the application to the desired screen. (This function will be triggered after the session expires)   } }  // If the parent app has passed true for the fi\_callback parameter, if the user  // clicks on a web link such as the "Privacy Policy" link on the Zelle UI, then  // the getValue method will be triggered and pass "privacy policy" as the value  // for the name parameter. The parent app handles this callback on their side.  @Override  public void getValue(String name) {  if (name == “TAG\_NAME”) {  //Here navigates the application to the desired screen. (This function will help to communicate between Zelle UI and parent app)   } } |

### Supported Versions

Minimum SDK: 24

Minimum OS: Android 7.0 Nougat

### Zelle Mobile SDK Size

224 KB

### Dependency

zxing:core - 3.3.3 (QR Code library)

# Implementing Mobile SDK Within iOS App

## Steps for Quick Start

### Project Setup

* Open the file info.plist (right-click > Open As > Source Code). Add the following keys to the file:

|  |
| --- |
| <key>NSContactsUsageDescription</key> <string>[PERMISSION\_DESCRIPTION]</string>  <key>NSCameraUsageDescription</key> <string>[PERMISSION\_DESCRIPTION]</string>  <key>NSPhotoLibraryUsageDescription</key>  <string>[PERMISSION\_DESCRIPTION]</string> |

* Import ZelleSDK.xcframework for the required target under project settings (Project > Target > Frameworks, Libraries, and Embedded Contents).
* Set ZelleSDK.xcframework to “Embed & Sign”:

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

* Import ZelleSDK as needed in any source code files.

### Implementation Steps with Pseudocode (Swift)

* Create a Zelle bridge configuration object.

|  |
| --- |
| private let zelle = Zelle(  applicationName: "Demo Bank", // Optional  baseURL: "https://certtransfers.fta.cashedge.com/popnet/faces/loginServlet",  institutionId: "88850000",  product = "zelle",  ssoKey: "e78abf35705a6d9b51fbf3939aa82489",  fi\_callback: true, //Mandatory when handling getValue method otherwise optional  loaderData: [  "loaderColor" : "hex color",  "bgColor" : "hex color",  ], // Optional (Nullable)  parameters: [  "key1" : "value1",  "key2" : "value2",  "key3" : "value3"  ] // Optional (Nullable) unless Zelle is accessed through Bill Pay ) |

Note:

ZelleSDK will **automatically** add the appropriate default values for these parameters to the URL: product.version and container (mobile\_sdk\_ios).

* Create a Bridge object (optional lazy implementation).

Note:

Pass the appropriate parent ViewController (type UIViewController) to self.

|  |
| --- |
| private lazy var bridge: Bridge = {  Bridge(  config: zelle,   viewController: self  ) } |

#### Launch Zelle® inside another view using code initialization (Recommended)

|  |
| --- |
| let zelleFrame = CGRect(x:0, y:0, width:view.frame.width, height:view.frame.height) // Custom Frame Size  let zelleView = bridge.view(frame:zelleFrame)  view.addSubview(zelleView) |

#### Set the GenericTagDelegate Protocol delegate to the existing ViewController

|  |
| --- |
| Bridge.genericTag = **self** |

#### Launch Zelle® inside another view and fill it (for screen orientation)

|  |
| --- |
| let zelleView = bridge.view()  zelleView.fill(view) |

#### Launch Zelle® as a popup (not inside another view)

|  |
| --- |
| let zellePopup = bridge.popup()  zellePopup.anchor(to: view) |

#### Session Timeout and Intercepting Web Links

|  |
| --- |
| //Implement the GenericTagDelegate with ParentViewController  Class ZelleViewController: UIViewController, GenericTagDelegate{  //Inside the class override the sessionTag & getValue method  func sessionTag(name: String){  if (name == “Landing”) {  //Here navigates the application to the desired screen. (This function will be triggered after the session expires)   } }  // If the parent app has passed true for the fi\_callback parameter, if the user  // clicks on a web link such as the "Privacy Policy" link on the Zelle UI, then  // the getValue method will be triggered and pass "privacy policy" as the value  // for the name parameter. The parent app handles this callback on their side.  func getValue(name: String) {  if (name == “TAG\_NAME”) {  //Here navigates the application to the desired screen. (This function will help to communicate with the Zelle UI and parent app)   } }  } |

### Implementation Steps with Pseudocode (Objective-C)

* Create a Zelle bridge configuration object.

|  |
| --- |
| Zelle\* zelle = [[Zelle alloc] initWith applicationName:@"Demo Bank" // Optional baseURL:@"https://certtransfers.fta.cashedge.com/popnet/faces/loginServlet" institutionId:@"88850000" product:@"zelle" ssoKey:@"e78abf35705a6d9b51fbf3939aa82489"  fi\_callback:true //true when handling getValue method otherwise false/nil  loaderData:@{  @"loaderColor": @"hex color",  @"bgColor": @"hex color"} // Optional (Nullable) parameters:@{  @"key1": @"value1",  @"key2": @"value2",  @"key3": @"value3"}]; // Optional (Nullable) unless Zelle is accessed via Bill Pay |

Note:

Since loaderData is an optional parameter, you may pass nil or empty (e.g., **loaderData: nil**).

Note:

parameters is an optional parameter unless Zelle® is accessed through Bill Pay. If Zelle® is accessed through Bill Pay, the name value pair **flowtype-BillPay** is required. Otherwise, you may pass nil or empty (e.g., **parameters: nil** or **parameters: @{}**.

Note:

ZelleSDK will **automatically** add the appropriate default values for these parameters to the URL: product.version and container (mobile\_sdk\_ios).

* Create a Bridge object (optional lazy implementation).

Note:

Pass the appropriate parent ViewController (type UIViewController) to self.

|  |
| --- |
| Bridge\* bridge = [[Bridge alloc] initWithConfig:zelle viewController:self ]; |

#### Launch Zelle® inside another view using code initialization (Recommended)

|  |
| --- |
| CGRect zelleFrame = CGRectMake(0, 0, self.view.bounds.size.width, self.view.bounds.size.height); UIView\* zelleView = [bridge viewWithFrame:zelleFrame]; [view addSubview:zelleView]; |

#### Set the GenericTagDelegate Protocol delegate to the existing ViewController

|  |
| --- |
| Bridge.genericTag = **self** |

#### Launch Zelle® inside another view and fill it (for screen orientation)

|  |
| --- |
| BridgeView\* bridgeView = [bridge view]; [bridgeView fill:self.view]; |

#### Launch Zelle® as a popup (not inside another view)

|  |
| --- |
| BridgePopup\* bridgePopup = [bridge popup]; [bridgePopup anchorTo:self.view]; |

#### Session Timeout and Intercepting Web Links

|  |
| --- |
| //Implement the GenericTagDelegate with ParentViewController  Class ZelleViewController: UIViewController, GenericTagDelegate{  //Inside the class override the sessionTag & getValue method  -(void)sessionTagWithName:(NSString\*) name {  if ([name isEqualToString: @ “Landing”]) {  //Here navigates the application to the desired screen. (This function will be triggered after the session expires)   } }  // If the parent app has passed true for the fi\_callback parameter, if the user  // clicks on a web link such as the "Privacy Policy" link on the Zelle UI, then  // the getValue method will be triggered and pass "privacy policy" as the value  // for the name parameter. The parent app handles this callback on their side.  -(void)getValueWithName:(NSString\*) name {  if ([name isEqualToString: @ “TAG\_NAME”]) {  //Here navigates the application to the desired screen. (This function will help to communicate Zelle UI and parent app)   } }  } |

### Supported Versions

Minimum Xcode Version: 11

Minimum OS: iOS 13

### Zelle Mobile SDK Size

2.6 MB (size includes simulator)

### Dependency

QRCodeReader - 10.1.0 (QR Code library)

# Additional Information

## Related Document

* See the “Turnkey Service for Zelle® Mobile SDK Integration” section of *CheckFree® Mobile Web: Mobile Integration Using WebView*.

## SSO Process

There are no changes to the existing SSO process. Refer to the appropriate implementation guide for Fiserv Turnkey Service for Zelle®.

## Terminology

Bill Pay

Bill Pay refers to Fiserv’s CheckFree® RXP® bill pay solution. For some Fiserv clients, Zelle® is implemented as part of Bill Pay where the user either:

* Navigates to Zelle® by first accessing Bill Pay and then selecting Zelle®, or
* Navigates to Zelle® from the main mobile banking navigation by selecting Zelle® and “deep linking” into the Zelle® section of Bill Pay.

FTK

Fiserv Turnkey Service for Zelle®. Fiserv offers a turnkey set of services designed to simplify the implementation of Zelle® for financial institutions of all sizes.

QR Code

Quick Response Code. A machine-readable code consisting of an array of black and white squares, typically used for storing URLs or other information for reading by the camera on a mobile device.

Sample QR Code:

Qr code

Description automatically generated

SDK

Software Development Kit. An SDK is a collection of software used for developing applications (for example, mobile applications).