



Visa Sensory Branding

SDK Technical Implementation Guide

January 2024



Welcome to Visa Sensory Branding SDK Technical Implementation Guide

This guide can be used:

- To assist developers with the implementation setup associated with Visa Sensory Branding iOS, Android and Web SDKs.

Contents

- About This Guide..... 6
 - Scope 6
- 1. Visa Sensory Branding 7
 - Overview 7
 - Application..... 7
 - Visa Brand Animation..... 8
 - Visa Brand Sound 14
 - Visa Brand Haptic 15
- 2. Web SDK technical implementation..... 16
 - a. Demo page 16
 - b. Integration 17
 - Animation End Event..... 17
 - View Integration..... 18
 - Sizing and layout 19
 - Init Parameters 20
 - Animation Configuration Parameters 22
 - c. Migration steps for the existing users..... 28
 - d. Appendix 31
 - Full Screen View 31
 - Constrained View 31

Language codes available for integration	32
3. iOS SDK technical implementation	36
a. Prerequisite.....	36
b. Demo App	36
c. Integration	36
Import the SDK module.....	36
Initialization	38
Animate	39
Sizing and layout	40
Animation Configuration Parameters	41
d. Migration steps for existing users.....	43
e. React Native Integration	44
f. Flutter Integration.....	44
Language codes available for integration.....	46
4. Android SDK technical implementation	50
a. Prerequisite.....	50
b. Demo App	50
c. Integration	51
Import the SDK module.....	51
Initialization	51
Animate	52
Sizing and layout	54

Animation Configuration Parameters	55
d. Migration Steps for Existing Users	57
e. React Native Integration	57
f. Flutter Integration.....	58
Language codes available for integration.....	60

About This Guide

The Visa Sensory Branding SDK Technical Implementation Guide provides developers with relevant technical information needed to integrate Visa Sensory Branding using the SDKs.

Scope

This document focuses on iOS, Android and Web SDK integration.

1. Visa Sensory Branding

Overview

Visa has developed sensory brand identity elements that provide the experience of hearing, feeling, and dynamically seeing the Visa brand identity. For use by developers, issuers, merchants, and other partners, the Visa Brand animation, the Visa Brand sound, and the Visa Brand haptic are used to signal a defined Visa event (e.g., successful in-app transactions, using Visa to send or receive money via issuer wallet or third-party apps, successful enrollment for services that require Visa credentials such as Click to Pay, Tap to Phone and more). Visa issuers are responsible for ensuring that the brand identity elements are applied within their applications as provided in the requirements below.

Visa Sensory Branding resources are available as a software development kit (SDK) for iOS, Android, and Web solutions on the [Visa Partner Portal](#).

Application

Effective 1 November 2022 Visa Sensory Branding must be used in all new implementations to signal a defined Visa event on devices capable of supporting sensory branding elements except physical point-of-sale terminals and effective 1 November 2023 Visa Sensory Branding must be used in all implementations except on physical point-of-sale ("POS") terminals. The Visa

Brand animation, the Visa Brand sound, and the Visa Brand haptic must all be used together as one complete experience whenever possible. Each sensory brand element may be used alone, or in combination with one of the other sensory brand elements only in environments with more limited capabilities or functionality (e.g., a client developing a wearable device with visual and haptic capabilities, but no sound capability, would only be required to implement the Visa Brand animation and Visa Brand haptic).

Examples of appropriate application of Visa sensory branding include:

- POS Terminal
- Digital Wallet
- Electronic Payment
- P2P Payment
- Installments
- Click to Pay

Visa Brand Animation

The Visa Brand animation, the “dynamic visual” element of the sensory brand identity, consists of the animated Visa Brand Mark in the new Visa Blue, with the ‘V’ of Visa entering in a checkmark motion in the center of the screen. The ‘ISA’ swiftly follows letter by letter. Meanwhile, a horizontal motion centers the animation. The animation must always conclude with the Visa checkmark, a third-party checkmark, or some sort of confirmation icon/message.

Visa Sensory Branding Animation – Graphic Elements

Animated Visa Brand Mark



The animated Visa Brand Mark appears in the new Visa Blue and its animation capitalizes on the natural checkmark shape of a 'V'.

Confirmation icon/message



An icon, such as a checkmark, or message, such as "Approved," follows the animated Visa Brand Mark as a symbol of confirmation and completion.

Visa Sensory Branding Animation – Behavior

Animated Visa Brand Mark



The 'V' of Visa enters in a checkmark motion in the center of the screen. The 'ISA' swiftly follows letter by letter.

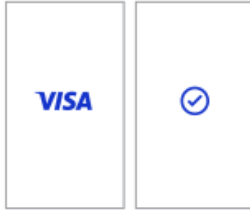
Confirmation icon/message



Quick cut to confirmation screen.

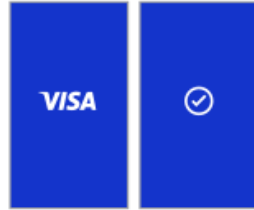
Visa Sensory Branding Animation – Color

Visa Blue against white



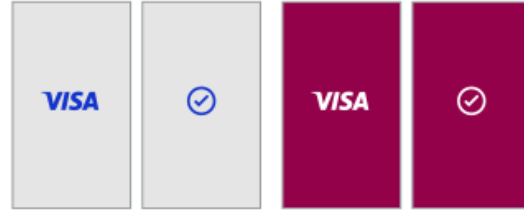
Background
White #FFFFFF
Visa Brand Mark
Visa Blue #1434CB
Checkmark
Visa Blue #1434CB

White against Blue



Background
Visa Blue #1434CB
Visa Brand
White #FFFFFF
Checkmark
White #FFFFFF

Against custom background



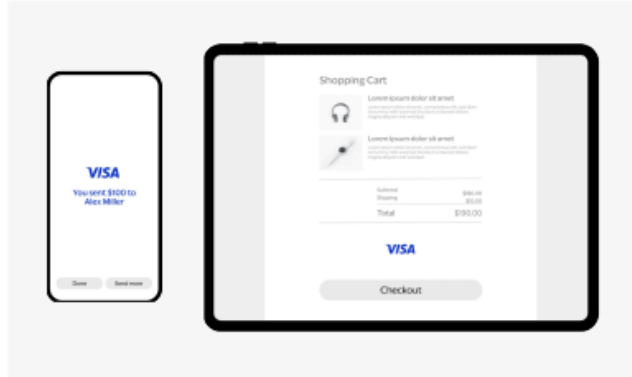
Use the same color for the animated Visa Brand Mark and checkmark.
If a light background color, use Visa Blue (#14134CB). If a dark background color, use white (#FFFFFF). No other colors can be used on these Visa graphic elements.
When selecting background color, ensure there is sufficient contrast (at least 3:1 contrast ratio) between elements and background colors.
For monochrome devices, only use white graphic elements against background colors.

Visa Sensory Branding Animation – Viewports

Full screen view



Constrained view



Use constrained view in spaces where the animation is displayed in context of many other elements. When possible, vertically and horizontally center the animation within the area you're applying the moment. The width of the Visa mark should not exceed 60% of the width of the screen.

Use constrained view in spaces where the animation is displayed in context of many other elements. When possible, vertically and horizontally center of the animation within the area you're applying the moment. The width of the Visa mark should not exceed 60% of the width of the screen.

The animation displays in the same way for all situations:

- The full screen application is used in instances such as mobile screens and other small devices, with no or very few other elements on the screen. The animation should be vertically and horizontally centered.
- The constrained view is used in instances where the animation is displayed in context of many other elements. When possible, vertically and horizontally center the animation within the area where the moment is applied.
- The width of the Visa Brand Mark at the conclusion of the Visa Brand animation should not exceed 60% of the width of the screen.

Visa Sensory Branding Animation – Confirmation Icon/Message

Visa checkmark (Primary use)



Visa checkmark with text
Text weight: Visa Dialect Medium



Sizing relationship



Sizing relationship



You must use the Visa checkmark, a third-party checkmark, or a confirmation icon/message immediately following the appearance of the animated Visa Brand Mark to conclude the Visa Brand animation and signal the completion of a successful Visa event.

If using the Visa checkmark, the checkmark position should be centered on the Visa brand mark.

In some instances, it may be preferable to use a checkmark with text, such as “Approved,” in Visa Dialect Medium font. **Note:** When use of Visa Dialect is not possible (e.g., languages not supported by Visa Dialect), you may use Noto Sans Medium. In cases that Noto Sans Medium is not available for your language, select the weight that nearest matches Visa Dialect Medium.

See Visa Digital Brand Requirements for more details.

Requirements

- The Visa Brand animation must be used with supported devices to signal a defined Visa event, e.g., successful in-app transactions, using Visa to send or receive money via issuer wallet or third-party apps, successful enrollment for services that require Visa credentials such as Click to Pay, Tap to Phone and more
- The Visa Brand animation must appear in the new vibrant Visa Blue against a white or light background, or in white against a Visa Blue background or other custom color backgrounds

Issuer/Co-Brand/Partner Customization

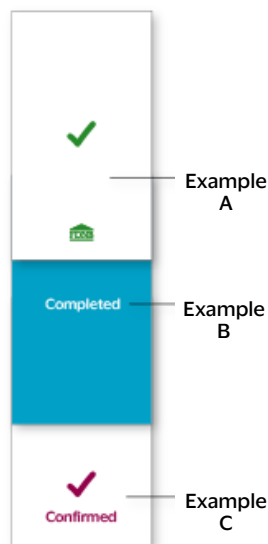
- The confirmation icon/message screen of the Visa Brand animation may be displayed with issuer, issuer third party branding, or partner co-branding on the same screen. Examples of appropriate customized branding include:
 - Issuer-branded Digital Card Art
 - Issuer co-branded Digital Card Art
 - Issuer Logo
 - Partner logo (merchant logo, wallet logo)

Animated Visa Brand Mark



Example A Issuer/co-brand/partner/merchant logo and checkmark
Example B Generic confirmation icon/message
Example C Confirmation icon/message

Example issuer/co-brand/ partner/merchant confirmation icon or message



Visa Brand Sound

The Visa Brand sound, the “audible” element, has been created specifically for use with the Visa Brand animation and Visa Brand haptic.

Requirements

- The Visa Brand sound must be used with supported devices to signal a defined Visa event, e.g., successful in-app transactions, using Visa to send or receive money via issuer wallet or third-party apps, successful enrollment for services that require Visa credentials such as Click to Pay, Tap to Phone and more.
- The Visa Brand sound is initiated when the Visa Brand animations begins.

- If your experience has other audio payment confirmations, play this sound before the spoken confirmation. For example, if a virtual assistant places an order, play the Visa sound after payment is submitted, then play “Your order has been submitted”.

Visa Brand Haptic

For the “feeling” element, Visa has developed a haptic pattern for use in conjunction with the Visa Brand animation and the Visa Brand sound.

Requirements

- The Visa Brand haptic must be used with supported devices to signal a defined Visa event, e.g., successful in-app transactions, using Visa to send or receive money via issuer wallet or third-party apps, successful enrollment for services that require Visa credentials such as Click to Pay, Tap to Phone and more. The Visa Brand haptic pattern consists of the defined sequencing, which must be programmed into a device that is capable of delivering haptic patterns.

2. Web SDK technical implementation

a. Demo page

To view the demo page locally, you can follow the steps below:

- Download Node.js
- Run 'npm install http-server'
- Run 'http-server -p 8080'
- Open <http://localhost:8080/demo/index.html>

Alternatively, you can unzip the file and host the folder in any web server.

Example: <https://domain-name/demo/index.html>

b. Integration

- Place the VisaSensoryBrandingSDK folder from the zip wherever you like
- Import visa-sensory-branding.js in your page

```
<script src="/VisaSensoryBrandingSDK/visa-sensory-branding.js"></script>
```

- Call VisaSensoryBranding.init() with desired configuration parameters for running the branding animation. Parameters are explained under "Animation Configuration Parameters" section.

```
<script>  
  VisaSensoryBranding.init({constrained: true}, './VisaSensoryBrandingSDK');  
</script>
```

- Add an empty div with id "visa-sensory-branding". This is where the branding will display

```
<div id="visa-sensory-branding"></div>
```

- Call VisaSensoryBranding.show() to run the branding animation

```
<script>  
  VisaSensoryBranding.show();  
</script>
```

Animation End Event

When the animation ends, a Custom Event object of type 'visa-sensory-branding-end' event is dispatched from the Dom Element.

To add an event listener, please follow the code snippet below:

```
window.addEventListener("message", e => {  
  if (e.data === 'visa-sensory-branding-end') {  
    console.log(e);  
  }  
});
```

This event listener is useful for third-party confirmation page integration. You can trigger a redirect event to your third-party confirmation page. Please refer to 'documentation/full-screen-with-third-party-checkmark.html' for more examples.

View Integration

To view it, please render it on localhost on a local machine or any web server.

Constrained View:

- <http://localhost:8080/documentation/static.html>
- <http://localhost:8080/documentation/absolute.html>

Full Screen View with Checkmark:

- <http://localhost:8080/documentation/full-screen-with-checkmark-text.html>

- <http://localhost:8080/documentation/full-screen-with-visa-blue-background-checkmark.html>

Full Screen View with Third Party confirmation page:

- <http://localhost:8080/documentation/full-screen-with-third-party-checkmark.html>

Sizing and layout

Integration is flexible. The sizing and placement of the animation within your page is completely up to you.

If you intend to run the animation within a "position: static" element **within the page flow**, refer to **static.html**.

Example: add a CSS (Cascading Style Sheets) class on the Dom Element ID: 'visa-sensory-branding' as shown below:

```
#visa-sensory-branding {  
  width: 400px;  
  height: 398px;  
  position: static;  
}
```

If you intend to run the animation in a "position: absolute" element that **runs outside of the page flow**, refer to **absolute.html**.

Example: add CSS classes on the Dom Element ID: 'visa-sensory-branding' as shown below:

```
#visa-sensory-branding {  
  display: none;  
  position: absolute;  
  top: 0px;  
}  
  
#visa-sensory-branding.show {  
  display: block;  
  width: 700px;  
  height: 500px;  
}
```

Please ensure that the constrained container on the web is bigger than the provided dimensions below.

Minimum width	Minimum height
60px	50px

Init Parameters

```
VisaSensoryBranding.init(animationConfig, brandingFolder);
```

- **animationConfig** (optional): Javascript Object, used to set animation parameters

```
{
  checkmark: "checkmarkWithText",
  checkmarkTextOption: "approved",
  color: "FFFFFF",
  constrained: true,
  language: "en",
  sound: false
}
```

- brandingFolder (required): String, Path to the VisaSensoryBrandingSDK folder

```
'../VisaSensoryBrandingSDK'
```

Sample Configuration:

```
<script>
  VisaSensoryBranding.init({
    constrained: true,
    checkmark: 'checkmarkWithText',
    checkmarkTextOption: 'approved',
    color: "ffffff",
    language: "en",
    sound: false
  },
  '../VisaSensoryBrandingSDK'
);
</script>
```

Animation Configuration Parameters

Parameter	Type	Description
sound	Boolean	<p>Activate the audio as the animation plays.</p> <p>Only available in the desktop browsers, mobile devices disable the sound by default according to the Auto-play policy.</p> <ul style="list-style-type: none">• Required: no• Type: boolean• Default values: false• If the wrong value is entered, no sound will be activated
Color	String	<p>The backdrop color is the background color of the animated logo.</p> <p>If the backdrop color is white (#FFFFFF), a blue Visa Logo, checkmark and text will be shown.</p> <p>If backdrop color is #1434CB (Visa Blue), a white Visa Logo, checkmark and text will be shown.</p> <p>Any other color will result in either blue or white being applied to both Logo and checkmark, depending on the contrast.</p>

Parameter	Type	Description
		<p>Please ensure that the color contrast between custom backdrop color and white/#1434CB should be at least 3:1 to be compliant with ADA (Americans with Disabilities Act).</p> <p>You can test the color contrast at Contrast Checker.</p> <p>color: "FFFFFF"</p> <ul style="list-style-type: none"> • Required: no • Type: string • Default values: "white" • Available values: "hex color codes (without #) with correct color contrast (≥ 3)" • If the wrong value is entered, default white background color with blue visa logo will be shown • If color contrast is less than 3, the animation will still be shown with a warning message printed in the console. <p>⛔ ▶ Your custom color doesn't provide enough contrast. Please enter another color.</p>

Parameter	Type	Description
constrained	Boolean	<p>The constrained animation will be shown within the parent Dom Element, otherwise it will be shown in full screen.</p> <pre>constrained: true</pre> <ul style="list-style-type: none"> • Required: no • Type: boolean • Default Value: false • If the wrong value is entered, the default value (false) will be assigned to the parameter
language	String	<p>Define the language of the texts present below the checkmark.</p> <pre>language: "en"</pre> <ul style="list-style-type: none"> • Required: no • Type: string • Default Value: "en" • Example values (ISO-639-1):

Parameter	Type	Description
		<ul style="list-style-type: none"> ○ ar ○ en ○ es ○ zh_tw ○ zh_hk ○ zh_cn <p>Please refer to "language codes available for integration" table under Appendix section.</p> <p>If the wrong value is entered, the text will be defaulted to English.</p>
checkmark	String	<p>This parameter enables/disables checkmark screen after visa logo animation</p> <p>Example:</p> <ul style="list-style-type: none"> • "none": no checkmark at the end of the animation • "checkmark": shows checkmark at the end of the animation • "checkmarkWithText": shows checkmark and checkmark text at the end of the animation

Parameter	Type	Description								
		<div>checkmark: "checkmark"</div> <ul style="list-style-type: none">Required: noType: stringDefault Value: "none"Available values:<ul style="list-style-type: none">"none""checkmark""checkmarkWithText"If the wrong value is entered, please refer to the table below<table><tr><th>Wrong Value</th><th>Effect</th></tr><tr><td>true</td><td>Checkmark is shown</td></tr><tr><td>false</td><td>Checkmark is not shown</td></tr><tr><td>Any random string</td><td>Checkmark is shown</td></tr></table>If checkmark = "checkmarkWithText" and checkmarkTextOption is not defined, checkmark will be displayed without text	Wrong Value	Effect	true	Checkmark is shown	false	Checkmark is not shown	Any random string	Checkmark is shown
Wrong Value	Effect									
true	Checkmark is shown									
false	Checkmark is not shown									
Any random string	Checkmark is shown									
checkmarkText Option	String	<p>If 'checkmark' attribute is equal to "checkmarkWithText", it defines the text to render below.</p> <p>Available Checkmark text options</p>								

Parameter	Type	Description								
		<table><tr><th>Text Key</th><th>Text Value</th></tr><tr><td>approved</td><td>Approved</td></tr><tr><td>success</td><td>Success</td></tr><tr><td>complete</td><td>Complete</td></tr></table> <p>Attribute Name - checkmarkTextOption</p> <div><pre>checkmarkTextOption: "approved"</pre></div> <ul style="list-style-type: none">• Required: no• Type: string• Default Value: undefined• Available values:<ul style="list-style-type: none">○ "approved"○ "success"○ "complete"• If the wrong value is entered, no text will be shown.	Text Key	Text Value	approved	Approved	success	Success	complete	Complete
Text Key	Text Value									
approved	Approved									
success	Success									
complete	Complete									

c. Migration steps for the existing users

STEP 1: Download the latest web zip

STEP 2: Replace the existing /VisaSensoryBrandingSDK/ folder with the new /VisaSensoryBrandingSDK/ from the zip file

STEP 3: Migrate the existing configurations **highlighted** with italic formatting in the below table

Method	Original	New Changes
Initialization	<pre><script> VisaSensoryBranding.init({ checkmark: true, color: "ffffff", constrained: true, sound: false }, '../VisaSensoryBrandingSDK'); </script></pre>	<pre><script> VisaSensoryBranding.init({ checkmark: 'checkmark', color: "ffffff", constrained: true, sound: false }, '../VisaSensoryBrandingSDK'); </script></pre> <p>*Note: the checkmark attribute changes from Boolean to String.</p>

Method	Original	New Changes
Animation End Event	<pre>window.addEventListener(' visa-sensory-branding-end', function(e) { console.log(e); });</pre>	<pre>window.addEventListener("mess age", e => { if (e.data === 'visa-sensory- branding-end'){ console.log(e); } });</pre>

STEP 4: Add the new configurations **highlighted** with italic formatting in below table if it is applicable

Method	Original	New Changes
Initialization	<p>No text in the original animation</p> <pre><script> VisaSensoryBranding.init({ checkmark: true, color: "ffffff", constrained: true, sound: false }, './VisaSensoryBrandingSDK');</pre>	<p>Text 'Approved' / 'Success' are available to insert below the checkmark icon</p> <pre><script> VisaSensoryBranding.init({ checkmark: 'checkmarkWithText', checkmarkTextOption: 'approved', language: 'en', color: "ffffff", constrained: true,</pre>

Method	Original	New Changes
	</script>	sound: false }, './VisaSensoryBrandingSDK'); </script>

d. Appendix

Full Screen View

Standard width	Standard height
40% of the screen width	Auto

The width of the logo will not exceed **60% of screen width**.

Constrained View

Standard width	Standard height
90% of the constrained containers	Auto

If aspect ratio (width/height) of the constrained container is > 1.5

Standard width	Standard height
135% of the constrained container height	Auto

Language codes available for integration

Language Name	Language Code
English	En
Arabic	Ar
Azeri	Az
Bahasa Indonesia	Id
Bosnian	Bs
Bulgarian	Bg
Canadian English	en_ca
Canadian French	fr_ca
Croatian	Hr
Czech	Cs
Danish	Da
Dutch	Nl
Finnish	Fi
French	Fr
Georgian	Ka

Language Name	Language Code
German	De
Greek	El
Hebrew	He
Hungarian	Hu
Icelandic	Is
Italian	It
Latvian	Lv
Lithuanian	Lt
Japanese	Ja
Kazakh	Kk
Khmer	Km
Kinyarwanda	Rw
Korean	Ko
Mongolian	Mn
Norwegian	No
Polish	Pl

Language Name	Language Code
Portuguese (Brazilian)	pt_br
Portuguese (Regular)	Pt
Romanian	Ro
Russian	Ru
Serbian	Sr
Simplified Chinese	zh_cn
Slovakian	Sk
Slovenian	Sl
Spanish (AR)	es_ar
Spanish (MX)	es_mx
Spanish (LA)	es_la
Spanish (ES)	Es
Swedish	Sv
Thai	Th
Traditional Chinese (HK)	zh_hk
Traditional Chinese (TW)	zh_tw

Language Name	Language Code
Turkish	Tr
UK English	en_gb
Ukrainian	Ua
Vietnamese	Vi

3.iOS SDK technical implementation

a. Prerequisite

The .xcframework will work with

- iOS 13 or above
- Xcode 13 or above
- Swift 5.5 or above

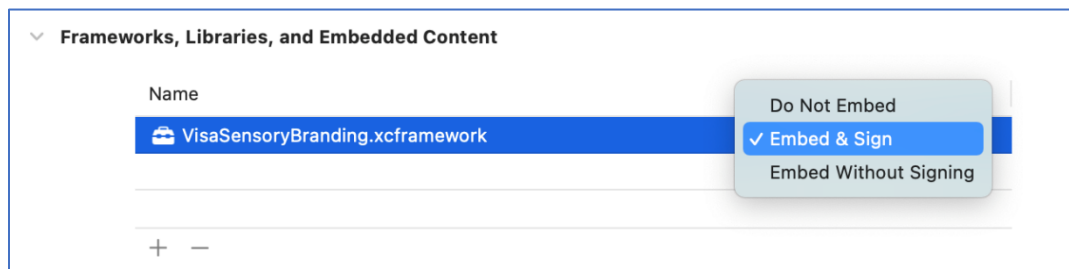
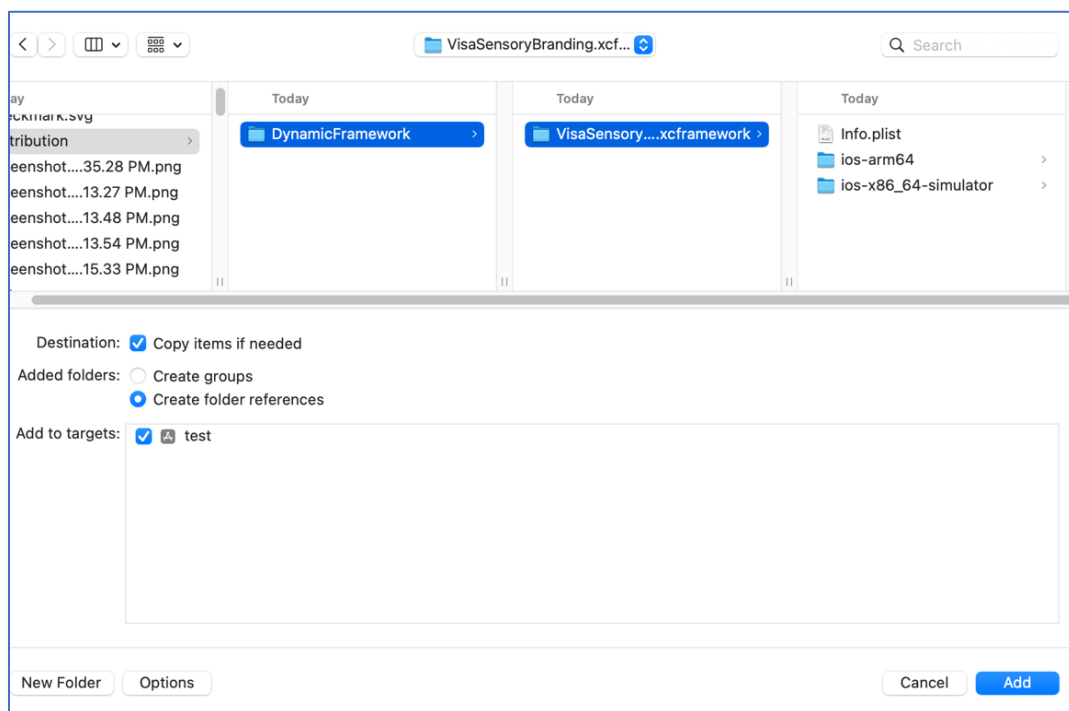
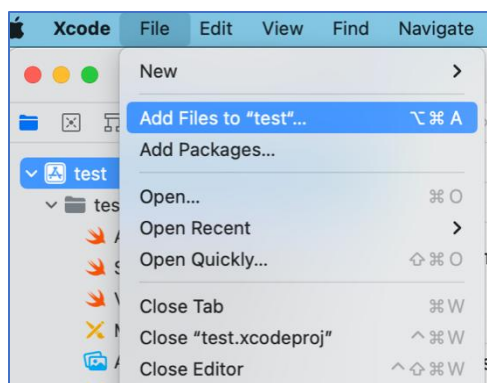
b. Demo App

1. To run demo app, please install [CocoaPods](#).
2. Run "pod install" in "VisaSensoryBrandingDemo" where "Podfile" is.
3. Open the VisaSensoryBranding workspace and run the active scheme.
4. Select the options to test for in the main screen and press the play button on the top right to see the logo animation.

c. Integration

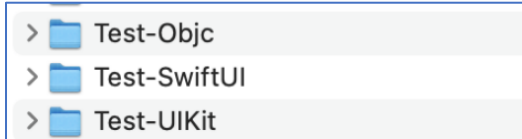
Import the SDK module

1. Download VisaSensoryBranding.xcframework file.
2. Drag and drop VisaSensoryBranding.framework into Embedded Binaries section of Project. (Leave "Copy items if needed" checked.)



3. Change "Do not embed" to "Embed & Sign"

4. Please see the integration examples under below folders inside the package.



Initialization

Import the Visa Sensory Branding library into any source files as necessary.

Swift:

```
import SensoryBrandingView
```

ObjC:

```
#import "VisaSensoryBranding/VisaSensoryBranding-Swift.h"
```

Swift:

```
let sensoryBrandingView = SensoryBranding()
```

SwiftUI:

```
func makeUIView(context: Context) -> SensoryBranding {  
    SensoryBranding()  
}
```

ObjC:

```
SensoryBranding * sb = [SensoryBranding new];  
[[self view] addSubview: sb];
```

Animate

Activates the Visa Sensory Branding animation.

Swift:

```
sensoryBrandingView.animate { result, error in  
    // handling code  
}
```

ObjC:

```
[sb animateWithCompletion:^(BOOL, NSError * err) {  
    printf("error: %s", err);  
}];
```

Few possible error codes/scenarios:

Error code	NOTE
.animationInProgress ("Animation is in progress")	Please wait for a while till animation is completed.
.fileNotFound ("File not found: (file name)")	Please refer to "Import the SDK module" section
.unexpected ("Unexpected error : (error details)")	Please contact Visa team by providing error log and screenshot for support.

Error code	NOTE
<code>.invalidColor</code> ("Invalid background color selected, contrast levels are below 3:1 against #FFFFFF and #1434CB")	Please choose different background color.

Sizing and layout

You can choose to have the Visa animation render in full screen on a device, or in a constrained view in context of other UI elements. To render in full screen, simply initialize the container to fit device height and width and set background color to same as `backdropColor`. For constrained view, specify the width of `SensoryBrandingView` during initialization.

- The width of the animation object will not be larger than 60% of the screen's width. If the width is set to be larger, it will be resized to be 60% of the screen's width.
- The width of the animation object will not be smaller than 20% of the screen's width. If the width is set to be smaller, it will be resized to be 20% of the screen's width.

Animation Configuration Parameters

Language Code

Define the language of the texts present below the checkmark.

Swift & ObjC:

```
sensoryBrandingView.languageCode = "en"
```

Please refer to “Available language codes for integration” table under Appendix section.

Backdrop Color

The color behind the animation, used to determine the Visa logo colors. Specify Visa Blue(#1434CB) for a white Visa logo. Specify White(#FFFFFF) for a blue Visa logo. Any other color will result in either blue or white color applied to the Logo, depending on contrast.

Note: Do not set backdropColor to clear, will default to Visa Blue if done so.

Swift:

```
sensoryBrandingView.backdropColor = .black
```

ObjC:

```
sensoryBrandingView.backdropColor = [UIColor blackColor];
```

Sound

Set to true to enable sound. (true by default)

Swift & ObjC:

```
sensoryBrandingView.isSoundEnabled = true
```

Haptic Feedback

Set to true to enable haptic feedback. (true by default)

Swift & ObjC:

```
sensoryBrandingView.isHapticFeedbackEnabled = true
```

Checkmark

This parameter enables/disables checkmark screen after visa logo animation

Example:

- "checkmark": shows checkmark at the end of the animation without text (default)
- "checkmarkWithText": shows checkmark and checkmark text at the end of the animation
- "none": no checkmark at the end of the animation

Swift:

```
sensoryBrandingView.checkmarkMode = .checkmarkWithText  
sensoryBrandingView.checkmarkTextOption = .success
```

Objc:

```
sensoryBrandingView.checkmarkMode = CheckmarkModeCheckmarkWithText  
sensoryBrandingView.checkmarkTextOptionForObjc =  
CheckmarkTextOptionForObjcComplete
```

Checkmark text includes:

- .success (default)
- .approve
- .complete

d. Migration steps for existing users

STEP 1: Replace existing SDK, refer to Requirements above.

STEP 2: Migrate existing configurations in the table below.

Option	Original	New changes
Backdrop Color	backdropColor: String	backdropColor: UIColor
Checkmark options	isCheckmarkShown: Boolean	checkmarkMode = enum checkmarkTextOption = enum checkmarkTextOptionForObjc = enum //ObjC
Constrained Flags	hasConstrainedFlags: Boolean	Obsolesced

Option	Original	New changes
Language Code	N/A	LanguageCode: String //e.g "en"

e. React Native Integration

React Native serves as a bridge to construct native views, grounded in React's declarative UI framework, thereby enabling integration with Visa Sensory Branding's native SDKs for both iOS and Android platforms.

Currently, there are two methodologies available to encapsulate native views for embedding into a React Native application:

- *Native Components*: refer to the official documentation <https://reactnative.dev/docs/native-components-ios>
- *Fabric Native Components*: refer to the official documentation <https://reactnative.dev/docs/the-new-architecture/pillars-fabric-components>

Developers are advised to select a method that aligns with their React Native version and the status of their ongoing project.

f. Flutter Integration

Flutter's "Platform Views" allow you to embed native views in a Flutter app so you can apply transforms, clips, and opacity to the native view from Dart.

iOS only uses Hybrid composition, which means that the native 'UIView' is appended to the view hierarchy.

The Visa Sensory Branding SDK is compliant with accessibility standards. For optimal accessibility support, we suggest using the hybrid composition method. Consult the official documentation below for a detailed guide on how to integrate the iOS native view:

<https://docs.flutter.dev/platform-integration/ios/platform-views>

Language codes available for integration

Please refer <https://support.apple.com/en-us/HT206175> for languages supported by VoiceOver.

Language Name	Language Code
English	en
Arabic	ar
Azeri	az
Bahasa Indonesia	id
Bosnian	bs
Bulgarian	bg
Canadian English	en_ca
Canadian French	fr_ca
Croatian	hr
Czech	cs
Danish	da
Dutch	nl
Finnish	fi

Language Name	Language Code
French	fr
Georgian	ka
German	de
Greek	el
Hebrew	he
Hungarian	hu
Icelandic	is
Italian	it
Latvian	lv
Lithuanian	lt
Japanese	ja
Kazakh	kk
Khmer	km
Kinyarwanda	rw
Korean	ko
Mongolian	mn

Language Name	Language Code
Norwegian	no
Polish	pl
Portuguese (Brazilian)	pt_br
Portuguese (Regular)	pt
Romanian	ro
Russian	ru
Serbian	sr
Simplified Chinese	zh_cn
Slovakian	sk
Slovenian	sl
Spanish (AR)	es_ar
Spanish (MX)	es_mx
Spanish (LA)	es_la
Spanish (ES)	es
Swedish	sv
Thai	th

Language Name	Language Code
Traditional Chinese (HK)	zh_hk
Traditional Chinese (TW)	zh_tw
Turkish	tr
UK English	en_gb
Ukrainian	ua
Vietnamese	vi

4. Android SDK technical implementation

a. Prerequisite

Download the latest Android SDK zip file, including the Demo App project and SDK module.

The SDK requires:

1. Android 5.0 or above.
2. Gradle 5 or above.
3. Android Gradle Plugin 4 or above.
4. androidx.appcompat:appcompat:1.2.0 or above (the Android Support library is no longer supported).
5. Java 8 or above, Kotlin 1.6 or above

To run the demo app, it requires:

1. Android Studio 2021.2.1 or above.
2. Gradle 7.4 or above.
3. Android Gradle Plugin 7.2 or above.

b. Demo App

1. Please open `${sdkFolder}/VisaSensoryBrandingDemo` project by Android Studio, wait a while for the Gradle Sync.
2. Make sure you connected the emulators or physical devices to the computer, then run the main demo application by typing the command `./gradlew :jetpack-compose:installDebug`.

3. Select the options to test for in the main screen, and press the play button at the top right to see the logo animation.

c. Integration

Import the SDK module

1. Copy the `${sdkFolder}/visa-sensory-branding-sdk` module to your project, it's a pre-configured Gradle module for `visa-sensory-branding-${sdkVersion}.aar`.
2. Add the SDK module to your own modules' `build.gradle(.kts)` by below code:

```
dependencies {  
    ...  
    implementation(project(":visa-sensory-branding-sdk"))  
}
```

3. Click Gradle Sync button of Android Studio to load the SDK.
4. You can find the integration examples under folders inside the package.

Initialization

Import the Visa Sensory Branding class reference into source files as necessary.

Java/Kotlin:

```
import com.visa.SensoryBrandingView
```

XML Layout:

```
<com.visa.SensoryBrandingView  
    android:id="@+id/vsb"
```

```
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_centerInParent="true"/>
```

Java + View:

```
SensoryBrandingView vsb = (SensoryBrandingView) findViewById(R.id.vsb);
```

Kotlin + View:

```
val vsb = findViewById<SensoryBrandingView>(R.id.vsb)
```

Kotlin + Jetpack Compose:

```
// Wrap the SensoryBrandingView with AndroidView
AndroidView(
    modifier = Modifier
        .width(Dp(width))
        .wrapContentHeight(),
    factory = { ctx ->
        val vsb = SensoryBrandingView(ctx, null)
        // More configurations...
        // vsb.languageCode = "en"
        vsb
    },
    update = { vsb -> })
```

Animate

Activates the Visa Sensory Branding animation.

Java + View:

```
vsb.animate(error -> {
    Log.d("SensoryBrandingView", error == null ? "OK" : error.getMessage());
});
```

```
return null;  
});
```

Kotlin + View:

```
vsb.animate { error ->  
    Log.d("SensoryBrandingView", error?.message ?: "OK")  
}
```

Kotlin + Jetpack Compose:

```
// We need a property to trigger animation  
var playingTheAnimation by rememberSaveable { mutableStateOf(false) }  
  
// Wrap the SensoryBrandingView with AndroidView  
AndroidView(  
    modifier = Modifier  
        .width(Dp(width))  
        .wrapContentSize(),  
    factory = { ctx ->  
        val vsb = SensoryBrandingView(ctx, null)  
        // More configurations...  
        // vsb.languageCode = "en"  
        vsb  
    },  
    update = { vsb ->  
        if (playingTheAnimation) {  
            vsb.animate { result ->  
                Log.d("SensoryBrandingView", result?.message ?: "OK")  
            }  
        }  
    })
```

3 possible error messages (Error#getMessage()):

Error message	NOTE
Previous animation still in progress, cannot start a new animation."	Please wait for a while till animation displays.
Alpha channel is not supported for backdrop color.	Please choose different background color.
Invalid background color selected, contrast levels are below 3:1 against #FFFFFF and #1434CB.	

Sizing and layout

You can choose to have the Visa animation render in full screen on a device, or in a constrained view in context of other UI elements. To render in full screen, simply initialize the container to fit device height and width and set background color to same as backdropColor. For constrained view, specify the width of SensoryBrandingView during initialization.

- The width of the animation object will not be larger than 60% of the screen's width. If the width is set to be larger, it will be resized to be 60% of the screen's width.
- The width of the animation object will not be smaller than 20% of the screen's width. If the width is set to be smaller, it will be resized to be 20% of the screen's width.

Animation Configuration Parameters

Language Code

Define the language of the texts present below the checkmark.

Java:

```
vsb.setLanguageCode("en");
```

Kotlin:

```
vsb.languageCode = "en"
```

Please refer to “Available language codes for integration” table in the Appendix.

Backdrop Color

The color behind the animation, used to determine the Visa logo colors. Specify Visa Blue(#1434CB) for a white Visa logo. Specify White(#FFFFFF) for a blue Visa logo. Any other color will result in either blue or white color applied to the Logo, depending on contrast.

Note: Do not set backdropColor with simi-transparent color or make it transparent.

Java:

```
vsb.setBackdropColor(Color.parseColor("#123333"));
```

Kotlin:

```
vsb.backdropColor = Color.parseColor("#123333")
```

Sound

Set to true to enable sound. (true by default)

Java:

```
vsb.setSoundEnabled(true);
```

Kotlin:

```
vsb.soundEnabled = true
```

Haptic Feedback

Set to true to enable haptic feedback. (true by default)

Java:

```
vsb.setHapticEnabled(true);
```

Kotlin:

```
vsb.hapticEnabled = true
```

Checkmark

This parameter enables/disables checkmark screen after visa logo animation:

Checkmark Mode includes:

- CheckmarkMode.CHECKMARK: shows checkmark at the end of the animation without text
- CheckmarkMode.CHECKMARK_WITH_TEXT: shows checkmark and checkmark text at the end of the animation (default)
- CheckmarkMode.NONE: no checkmark at the end of the animation

Checkmark Text includes:

- CheckmarkTextOption.APPROVE
- CheckmarkTextOption.SUCCESS
- CheckmarkTextOption.COMPLETE (default)

Java:

```
vsb.setCheckmarkMode(CheckmarkMode.CHECKMARK_WITH_TEXT);  
vsb.setCheckmarkText(CheckmarkTextOption.APPROVE);
```

Kotlin:

```
vsb.checkmarkMode = CheckmarkMode.CHECKMARK_WITH_TEXT  
vsb.checkmarkText = CheckmarkTextOption.APPROVE
```

d. Migration Steps for Existing Users

STEP 1: Replace existing SDK, refer to Requirements above.

STEP 2: Migrate existing configurations in the table below.

Option	Original	New changes
Constrained Flags	setConstrainedFlags(Boolean)	Obsolesced
Haptic Feedback	setHapticFeedbackEnabled(Boolean)	setHapticEnabled(Boolean)
Checkmark Options	setCheckMarkShown(Boolean)	setCheckmarkMode(CheckmarkMode.XXX) setCheckmarkText(CheckmarkTextOption.XXX)
Language Code	N/A	setLanguageCode("xx")

e. React Native Integration

React Native serves as a bridge to construct native views, grounded in React's declarative UI framework, thereby enabling integration with Visa Sensory Branding's native SDKs for both iOS and Android platforms.

Currently, there are two methodologies available to encapsulate native views for embedding into a React Native application:

- *Native Components*: refer to the official documentation <https://reactnative.dev/docs/native-components-android>
- *Fabric Native Components*: refer to the official documentation <https://reactnative.dev/docs/the-new-architecture/pillars-fabric-components>

Developers are advised to select a method that aligns with their React Native version and the status of their ongoing project.

f. Flutter Integration

Flutter's "Platform Views" allow you to embed native views in a Flutter app, it supports two modes: **hybrid composition** and **virtual displays**:

- Hybrid composition appends the native `android.view.View` to the view hierarchy. Therefore, keyboard handling, and accessibility work out of the box.
- Virtual displays render the `android.view.View` instance to a texture, so it's not embedded within the Android Activity's view hierarchy. Certain platform interactions such as keyboard handling and accessibility features might not work.

The Visa Sensory Branding SDK is compliant with accessibility standards. For optimal accessibility support, we suggest using the hybrid composition method. Consult the official documentation below for a detailed guide on how to integrate the Android native view: <https://docs.flutter.dev/platform-integration/android/platform-views>

Appendix

Language codes available for integration

Please refer to

<https://support.google.com/accessibility/android/answer/11101402?hl=en> for languages supported by Talkback service.

Language Name	Language Code
English	En
Arabic	Ar
Azeri	Az
Bahasa Indonesia	Id
Bosnian	Bs
Bulgarian	Bg
Canadian English	en_ca
Canadian French	fr_ca
Croatian	Hr
Czech	Cs
Danish	Da
Dutch	Nl

Language Name	Language Code
Finnish	Fi
French	Fr
Georgian	Ka
German	De
Greek	El
Hebrew	He
Hungarian	Hu
Icelandic	Is
Italian	It
Latvian	Lv
Lithuanian	Lt
Japanese	Ja
Kazakh	Kk
Khmer	Km
Kinyarwanda	Rw
Korean	Ko
Mongolian	Mn

Language Name	Language Code
Norwegian	No
Polish	Pl
Portuguese (Brazilian)	pt_br
Portuguese (Regular)	Pt
Romanian	Ro
Russian	Ru
Serbian	Sr
Simplified Chinese	zh_cn
Slovakian	Sk
Slovenian	Sl
Spanish (AR)	es_ar
Spanish (MX)	es_mx
Spanish (LA)	es_la
Spanish (ES)	Es
Swedish	Sv
Thai	Th
Traditional Chinese (HK)	zh_hk

Language Name	Language Code
Traditional Chinese (TW)	zh_tw
Turkish	Tr
UK English	en_gb
Ukrainian	Ua
Vietnamese	Vi