Wirecard CEE Integration Documentation

() -

Created: 2019-12-13 17:41

Integration for Mobile Devices

The following chapters are only relevant if you do NOT use the responsive design of Wirecard Checkout Page or Wirecard Checkout Seamless. If you wish to switch to the responsive design, please contact our support teams.

General information

No matter which business case is adopted by merchants or which consumer devices and technologies are involved, when using a mobile browser or a native web view both Wirecard Checkout Page and Wirecard Checkout Seamless can be integrated to your shop.

In general, all functions and features that Wirecard offers for desktop or notebook computers in an online shop, including back-end operations, may also be implemented and used on mobile devices.

Mobile demo shops

Before starting integration you may want to have a look at our mobile demo shops for Wirecard Checkout Page and Wirecard Checkout Seamless.

Wirecard Checkout Page mobile

For integrating Wirecard Checkout Page on mobile devices we offer 2 default layout options, which you may customize for your design and user interface needs:

Wirecard Checkout Page mobile for Smartphones

Wirecard Checkout Page mobile for Tablets

Wirecard Checkout Seamless mobile

Within Wirecard Checkout Seamless you may use your own HTML and CSS for integrating the required UI elements. For an example of a demo integration based on responsive design, see:

Wirecard Checkout Seamless


For all demo shops you may want to use our product demos.

Integration

Supported mobile technologies

Mobile operating systems

Both, Wirecard Checkout Page and Wirecard Checkout Seamless can be used for mobile shops running on all current available mobile operating systems. Typical examples of supported mobile operating systems are e.g.:

- iOS by Apple
- Android by Google
- Windows mobile
- Blackberry
- Tizen by Samsung
- Sailfish
- Ubuntu touch

- ...

Mobile browsers

Browsers on mobile devices are typically built for supporting HTML5, in this case you may use Wirecard Checkout Page and Wirecard Checkout Seamless on all current mobile browsers, e.g.:

- Google Chrome
- Firefox
- Opera
- Safari
- ...

Native apps

Additionally you can integrate Wirecard Checkout Page and Wirecard Checkout Seamless within your native app. Typically this can be done by using a native web view user interface element. Therefore you may integrate into native apps developed in:

- Objective-C in iOS for iPhone, iPad and iPod
- Java for all Android based devices (like Samsung, Sony, HTC, etc.)
- C# for all Windows based mobile devices
- BlackBerry
- and others ...

Available integration possibilities

Browser based integration

Browser based integrations are typically a kind of:

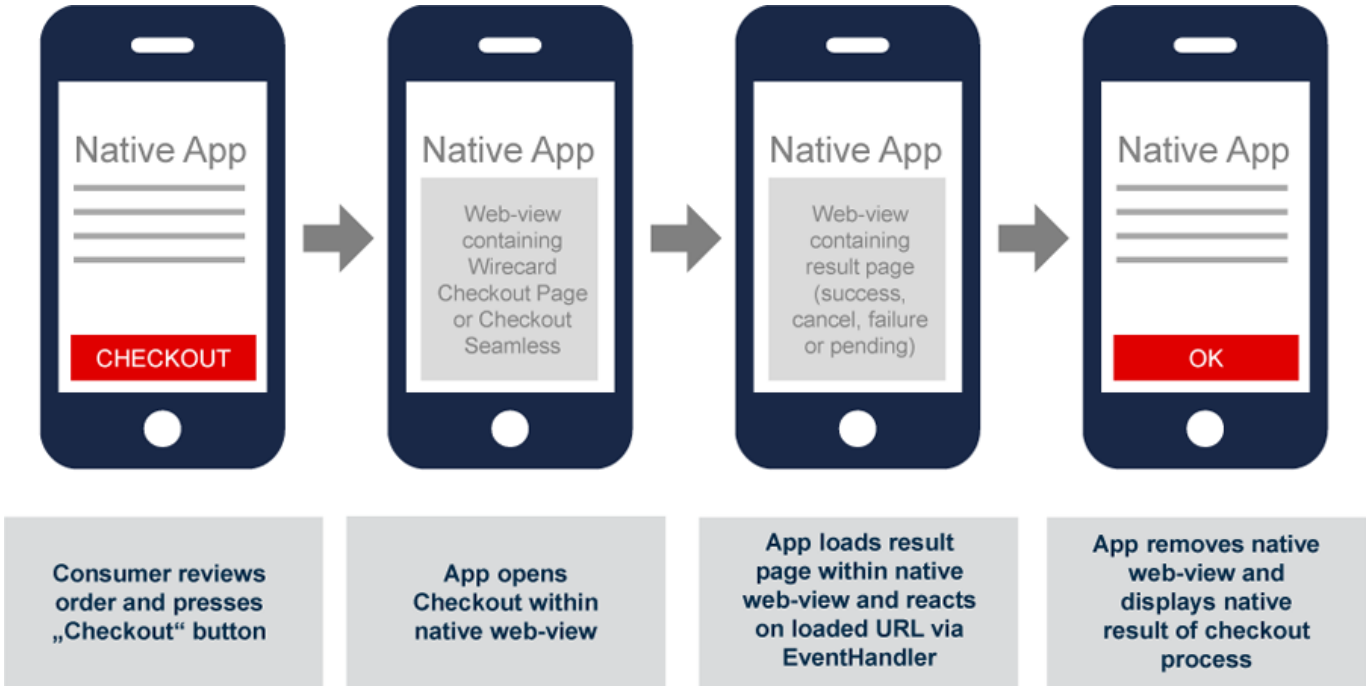
- A mobile web site or
- A mobile web app in HTML or
- A mobile web app in a native wrapper

Our products may be integrated in almost the same way as in your desktop-based online shop.

Typically for Wirecard Checkout Page your online shop is based on a responsive or adaptive design and you are using the `smartphone` or `tablet` option within the request parameter `layout`. For Wirecard Checkout Seamless you may integrate the payment specific UI elements seamless to your online shop.

Native app or cross-platform app integration

For this purpose you may integrate the checkout process within a native web view in your app. The flow from native to web view and back to native can be implemented using the following pattern:



Implementation pattern for web view in native apps

1. A native “Checkout” button is displayed for the consumer in the native GUI of the app.
2. The consumer taps on this button.
3. A web view containing Wirecard Checkout Page or Wirecard Checkout Seamless opens in the native app.
4. The consumer is guided through the checkout process presented within the native web view.
5. The return URL at the end of the checkout process is called in the web view. This can be one of the URLs you set in the following required request parameters: `successUrl`, `cancelUrl`, `failureUrl` or `pendingUrl`.
6. The native app responds to the loading of the URL and returns to native control and displays the result of the checkout process to the consumer within the native GUI of the native app.

You may detect the value of the return URL by using callback functions offered by the native web view, which are called each time a new URL is loaded in the web view.

- For Android you can use the method `onPageStarted` of the WebViewClient.
- For iOS you may want to have a look at `webViewDidStartLoad:` of the UIWebViewDelegate Protocol.

Please be aware that we strongly recommend to use the `confirmUrl` to ensure that your web server is able to receive the result of the checkout process independently of the consumer's device and the behavior of your consumer in the native app.

Improving usability

If you are able to identify your consumer in your mobile application (e.g. by their username/password), you have the possibility to simplify the checkout process for recurring consumers.

This can be done by using the feature “recurring payment” which allows you to re-use an already successful payment of your consumer as a base for a new payment with the same payment type and data as the consumer used before. This has the advantage that your consumer does not need to re-enter the sensitive payment information for any further checkouts in your shop.

You can find more details regarding recurring payments within the description of the operation `recurPayment` in:

- Transaction-based operations for Wirecard Checkout Page and Wirecard Checkout Seamless

Additional considerations

iframes and pop-up windows

Please be aware that for usability reasons it is not recommended to use either iframes or pop-up windows. Additionally within native web views it may not be possible to open pop-up windows out-of-the-box for security reasons.

Specific configuration of your mobile shop

Additionally we recommend you to use a different value for the request parameter `shopId`, so that our support can configure the behavior of the checkout process to your needs.

Typically all external pages which are required by various financial service providers are done by a redirect of the current page and not opened in a new window.

Another option you may want to use is a different set of payment methods you offer on the mobile device. This may especially be required if a specific payment method is not allowed to be used on mobile devices by the corresponding financial service provider.

To set up a specific `shopId` for your mobile shop, please contact our Support teams.

"Non-mobile" external pages of relevant financial service provider

Some external web sites of financial service providers, which need to be opened to your consumer (like entering 3-D Secure passwords), may be not suitably designed for some mobile devices.

Native web view

When you are using Wirecard Checkout Page or Wirecard Checkout Seamless within a web view in a native app, please take into consideration that some payment methods require the opening of a pop-up window by the corresponding financial service provider. Due to the fact that opening a pop-up window is not supported within web views by default, these payment methods cannot be used in such configuration.

Currently the following payment methods are not suited to be used within a native web view:

- eps-Überweisung
- TatraPay
- Trustly

iDEAL needs some additional steps to be implemented in a native app. Open Wirecard Checkout Page in the system default browser. After the payment process your consumer is redirected to the app. The return URL at the end of the checkout process is called in web view. This can be one of the URLs you set in the required request parameters: successUrl, cancelUrl, failureUrl or pendingUrl. The native app responds to the loading of the URL and returns to native control and displays to the consumer the result of the checkout process within the native GUI of the native app.

If you still need these payment methods in a mobile solution you can either switch to a browser-based integration or by opening a mobile browser and then returning back from the mobile browser to your native app within the result page.