CoinBridge By Nayax

CoinBridge iOS Solution Implementation Guidelines

Version 2.0

Table of Contents

CoinBridge Solution Overview

General

CoinBridge's patented technology and solution allows for the conversion of any digital asset into real purchases and transactions, at any shop worldwide.

CoinBridge's technology is comprised of two main elements (services):

1. **CoinBridge Mobile SDK** – Embeds into any existing mobile application, and acts as an internal e-Wallet within the host mobile app, providing payment capabilities at points of sale (POS) via the NFC/ EMV protocol (with a Tap & Go user experience), over the credit card scheme.
2. **CoinBridge Back-Office Connector** – Connects loyalty/ user management platforms with the CoinBridge platform, enabling CoinBridge to query user balance and spending policies in real time, whenever a transaction request is received. Once a purchase has been completed, this connector also enables CoinBridge to provide the loyalty/ user management platform with transaction data.

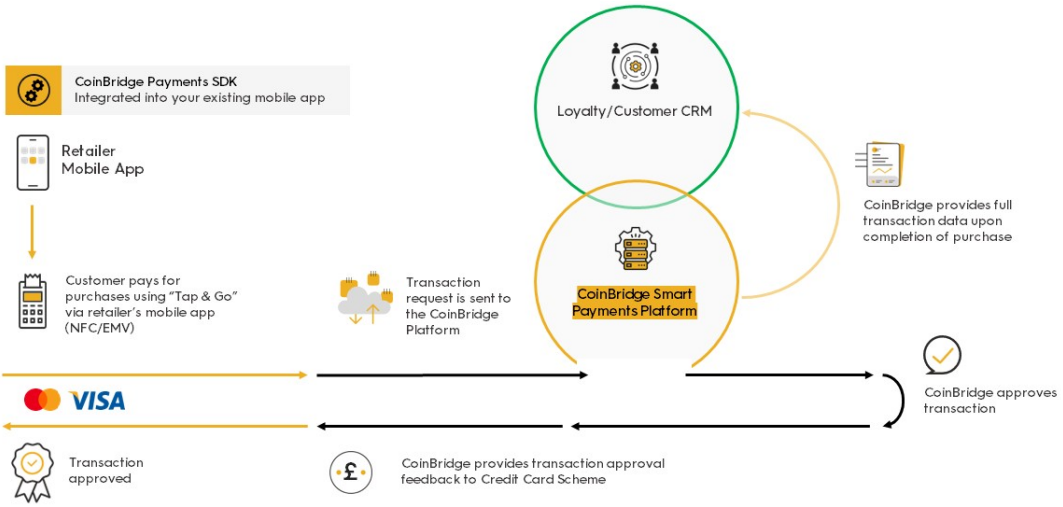


Diagram 1: Transaction Execution Flow

Customer Prerequisites

Below, we describe the two core technology touchpoints that are essential to enable the CoinBridge payment service:

Existing Customer Mobile Application

The customer's existing mobile app will have to be updated to implement the CoinBridge SDK, which will then enable the end-user NFC/ EMV, Tap & Go payment experience. The SDK also provides the mobile application with additional services and features, as described below.

Existing Customer Loyalty or User Management Platform

Either a customer loyalty or user management platform is essential for the use of CoinBridge services. It allows for the dynamic, real-time assessment of user balance and related spending policies, so the CoinBridge service can further evaluate that data against any transaction request, and either approve or decline transactions attempts by a user.

Two-Factor Authentication (2FA) for Tokenization Security

According to Financial Regulations and as an integral part of the CoinBridge Tokenization Process, it is a mandatory requirement to incorporate Two-Factor Authentication (2FA) for enhanced security and for regulation purposes. This crucial measure is designed to fortify the protection of user accounts and sensitive data during the tokenization journey.

CoinBridge strongly encourage to enable 2FA during the login process to your application. Additionally, it is permissible to implement 2FA upon the user's initial joining to the CoinBridge solution.

Apple Prerequisites

iOS Device Security Requirement: Biometric (Face ID, Fingerprint or Passcode) for Apple Wallet Access

To ensure secure access to the CoinBridge solution on iOS devices, users must have either Face ID, or a passcode configured on their phones.

This dual authentication requirement enhances the security of the Apple Wallet within the CoinBridge solution.

Without a configured Face ID or passcode, users may experience limitations in accessing and utilizing Apple Wallet features. Therefore, it is essential for iOS users to have either Face ID, or a passcode set up to guarantee seamless and secure access to the CoinBridge payment feature.

Feature Flag Implementation for CoinBridge Payment Feature

In adherence to Apple's guidelines, the implementation of the CoinBridge Payment feature on iOS must incorporate a feature flag in the backend.
This feature flag serves as a crucial tool, enabling you to exert control over the functionality of the payment feature, and will be used initially when launching the "Add to Apple Wallet" capability for the first time on your application without the need to release a new application version to App Store.
With the feature flag, your application gains the ability to toggle the CoinBridge Payment feature on or off as needed.
It is imperative to carefully consider the user experience when the feature flag is set to the "off" position.
Clear and user-friendly messaging can be implemented to inform your users about the temporary unavailability of the payment feature, ensuring a seamless and transparent experience during periods when the feature is disabled.

Solution Components

The CoinBridge service is enabled by the following components:

1. **CoinBridge Mobile SDK**
2. **CoinBridge Back-Office Connector**

CoinBridge Mobile SDK: Introduction

The SDK

- An SDK is the element of a solution that is embedded into a mobile app.
- The CoinBridge SDK is compatible with any iOS mobile application.
- It enables the conversion of digital assets (such as points and rewards) into real purchases (transactions) through EMV over NFC, directly from the mobile application.

Key Benefits of the CoinBridge SDK

- Easily embeds into any existing mobile application.
- Transparent to the user (does not change the look & feel of the application).

The SDK must be used in tandem with the CoinBridge Back-Office Connector to enable full functionality of the solution.

CoinBridge Back-Office Connector: Introduction

The CoinBridge Back-Office Connector connects the customer's loyalty or CRM (or similar) platform with the CoinBridge platform.

This connection is imperative to enable CoinBridge to dynamically draw information from the customer's platform whenever they make a purchase (driving a transaction), and for the CoinBridge platform to provide feedback about transaction details to the customer's platform.

Connection to the CoinBridge Platform facilitates the following main processes:

- Verification of user service eligibility.
- Retrieval of the user's current balance (or balances - when dealing with multiple products per end-user).
- Retrieval of the user's related spending policy(ies).
    - Spending policy(ies) are an integral part of the program that permits or limits user spending in any way or form.
    - A spending policy is a rule or set of rules that details all allowances and restrictions relevant to a specific user at the time of transaction. These rules are set and managed by the customer on the customer's own platform.
    - Such policies may (but are not limited to):
        - Restrict or allow certain merchant IDs (MID)
        - Restrict or allow certain merchant categories (MCC)
        - Limit spending per transaction
- Data and feedback – CoinBridge updates the customer's loyalty or CRM platform with specific details, according to the transaction type.
    - **Payment Transaction** - Transmits the transaction result, whether it has been approved or declined, and full transaction details.
    - **Refund Transaction** – Transmits refund transaction feedback and additional data.
    - **Transaction Update** – Transmits details of a transaction cancellation, on an already-reported successful transaction, due to a problem that may have occurred.

⚠ NOTE: Full product and technical information about the CoinBridge Back-Office Connector can be found in the dedicated CoinBridge Back-Office Connector documentation.

Product Implementation Guide

(Mobile Application Adaptation)

User Journeys and Prerequisites

Please ensure that the following steps have been completed prior to proceeding with this chapter:

1. Integration of CoinBridge SDK
2. Integration of CoinBridge Back-Office Connector

⚠ **Mandatory Two-Factor Authentication (2FA) for User Login**: In strict accordance with Apple's security mandates for payment applications, it is imperative that your application must implement Two-Factor Authentication (2FA) for every user login. This critical security measure is essential to fortify the protection of user accounts and sensitive payment data.

The user journey is divided into two phases:

- **User Pre-Payment Journey**
- **User Payment Journey**

The **user pre-payment journey** is the process a user goes through when they open your application for the first time (after CoinBridge SDK has been integrated), and while onboarding.
As soon as the user shows intent, CoinBridge issues a virtual card, tokenizes it, and pushes it to the wallet on the user's device.
This is a one-time step, following which their virtual card will be available whenever they want to make a payment.

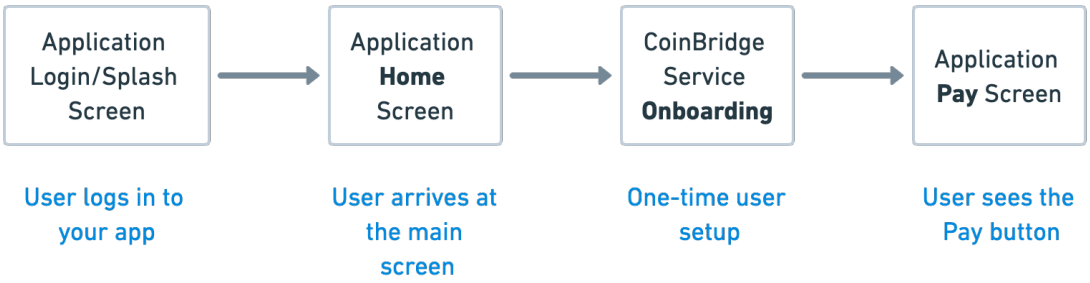
The **user payment journey** is the Tap & Go experience that users undergo whenever they wish to pay using your application.

In the following chapters, we will explain these two steps of the user journey in detail, and provide technical information and design suggestions for your application.

⚙ TECHNICAL INFO: The combination of the `isAuthenticated` method followed by the `initialization` method is crucial and should be applied every time your application starts, ensuring the seamless functioning of the payment process. For a deeper understanding of the initialization method, refer to Appendix 1 in this document.

User Pre-Payment Journey

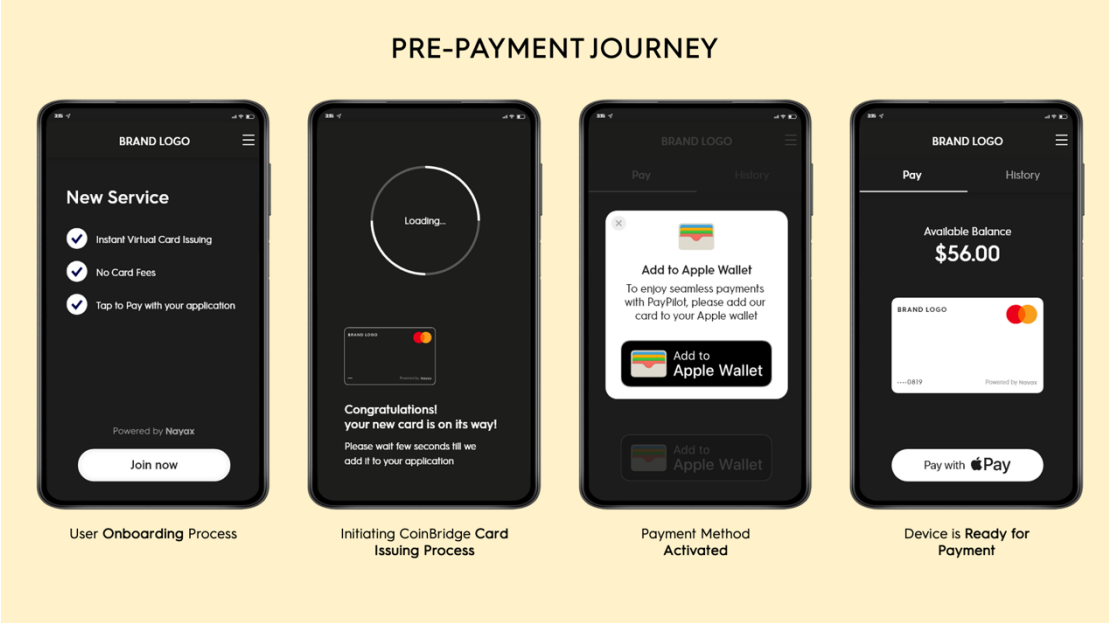
General



The user pre-payment journey covers the following:

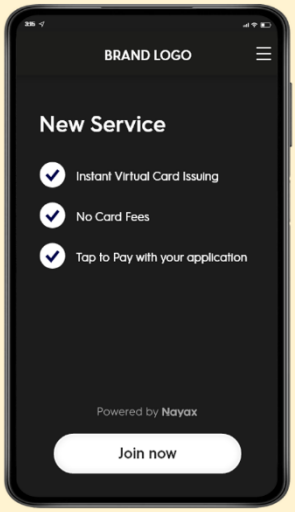
- Steps that the SDK and your application must perform to get to the point where the user can make payments with your app.
- User experience & user interface changes that must be implemented in your application.

Below, you will find a detailed account of the technical steps to be carried out, using various CoinBridge SDK methods, and what the user should see on-screen at each point of the flow.



Onboarding Your User to the CoinBridge Solution

CoinBridge Service - Introduction and Provision



NEW FEATURE INTRODUCTION

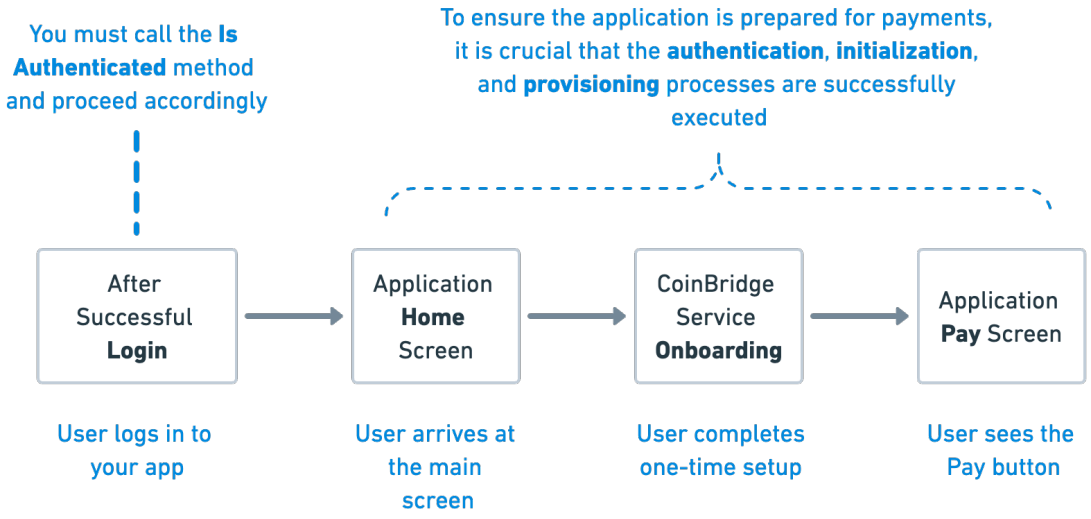
How does the feature work?

What are the benefits of using this feature?

Where can this feature be used?

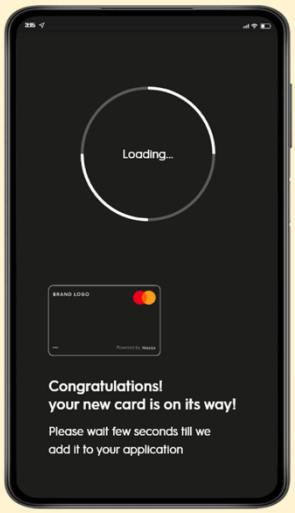
It is crucial to present your user with a proper promotion and introduction to the CoinBridge solution, while emphasizing its benefits and explaining related flows, including the payment flow. It is recommended to do this via a dedicated onboarding page, with a clear CTA for the user to acknowledge.

⚙ TECHNICAL INFO: As mentioned earlier, make sure to use the `isAuthenticated` method every time your application starts. If this call returns ***false***, it means that CoinBridge SDK did not recognize the user/ device, and the Application should call the `Authentication` method. If this call returns ***true***, it means that CoinBridge SDK is authenticated, and the Application should call the `Initialization` method. Further details regarding Initialization & Authentication can be found in Appendices 1 and 2 to this document.



CoinBridge Payment Method Creation Process

For first-time users only, CoinBridge will initiate the payment method creation process.



VIRTUAL CARD CREATION PROCESS

- The virtual card creation process may take up to 10 seconds.
- To ensure a smooth user experience, we suggest notifying the user to wait a few moments until the process is complete.
- This is the time to provide information about any spending policy restrictions.
- The user must stay in your application and not close it during the process.
- Once the virtual card is ready, promptly notify the user to start using it.

The virtual card issuing process (and provisioning) usually takes up to 10 seconds, so it is **highly recommended** to show on-screen text along the lines suggested above during this brief waiting time:

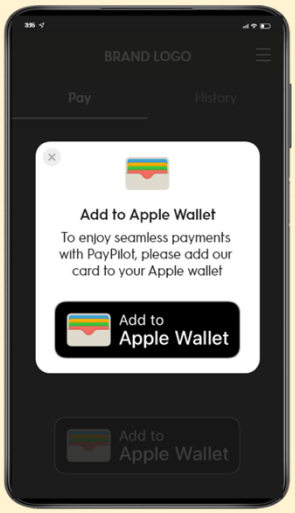
- To ensure a smooth user experience, show a loader animation with appropriate text. We suggest notifying the user to wait a few moments until the process is completed.
- It's important that the user does not leave the application before this process is finalized.

- If the process is aborted, for example by the user leaving the application, it will automatically restart upon reentering the application (and performing a successful initialization or the CoinBridge SDK).

- This is a good opportunity to mention - or refer the user to - any spending policy restrictions.
- Once the process has been completed, the CoinBridge SDK will notify your application.

TECHNICAL INFO: Upon successful completion of the initialization process, you will receive an `initiCompleted` callback providing details about the current state of the virtual card. Within this callback, `hasCard` will be either true or false, indicating whether a virtual card has been issued for the user, or not.
The `CBPaymentStatus` object provides additional information about the Virtual Card. If `hasCard` returns False, it signifies that a virtual card should be issued for the user (either automatically or manually). Conversely, if it returns True, the virtual card has already been issued for this user, and card details will be available within the("`CBPaymentStatus`") object.
Detailed information regarding the `CBPaymentStatus` method can be found in the Appendix 3 to this document.

Completion of the Virtual Card Creation Process



YOUR VIRTUAL CARD IS NOW IN YOUR APPLICATION

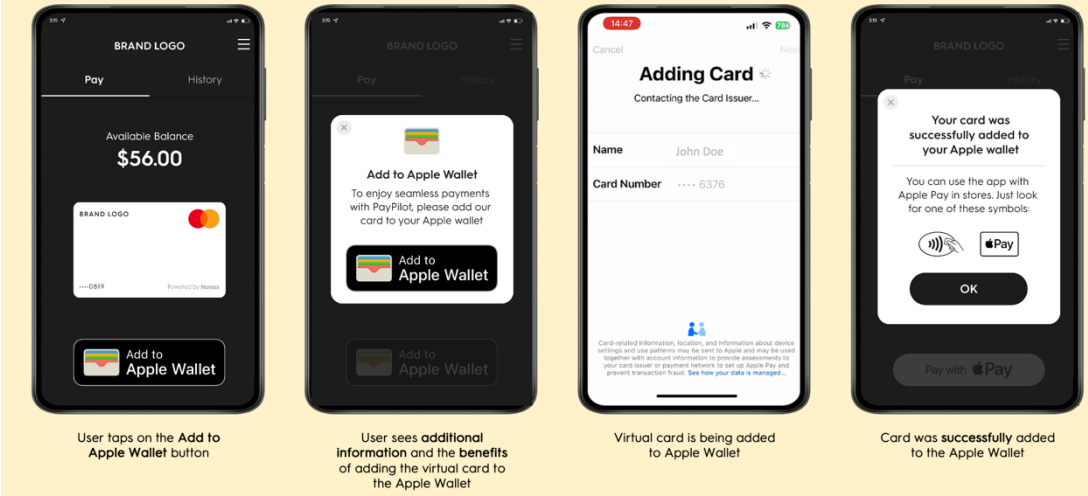
- Congratulate your user once their virtual card is ready for use.
- Provide additional information about what to do next.
- Let them know that they need to add the virtual card to Apple Wallet in order to start using your application to make payments!

Once the payment method creation process has been successfully completed, the CoinBridge SDK will send a notification to your application.

- After completion, it is vital to notify the user that their virtual card is ready, and the only remaining step is to add it to Apple Wallet (a process called "Provisioning").
- It is essential to emphasize to the user that they need to add their card to Apple Wallet in order to use the virtual card to make payments.
  The following section sets out how to do this.

Adding the Virtual Card to the User's Apple Wallet

To complete the pre-payment journey, the user must add the virtual card to their Apple Wallet. This is achieved through a process called 'provisioning' (add to apple wallet) which is similar to the regular process for doing so, but without requiring two-factor authentication.



User taps on the **Add to Apple Wallet** button

User sees **additional information** and the **benefits** of adding the virtual card to the Apple Wallet

Virtual card is being added to Apple Wallet

Card was **successfully** added to the Apple Wallet

As long as the virtual card is issued but not added to the device's wallet, display a CTA button saying 'Add to Apple Wallet' on the screen.

⚠ NOTE: It is crucial to follow the Card Art Style Guide found in Appendix 7 to this document, to properly display the 'Add to Apple Wallet' button.

Once the virtual card has been successfully added to the Apple Wallet, you will be notified by CoinBridge SDK (see PaymentMethodStatus in the Appendix 3 to this document).

From this moment on, the payment home screen will serve as the user's main screen for making Tap & Go payments. (See the next chapter for further details.)

⚠ NOTE: It is important to congratulate the user now that their device is ready to perform payments using your application.

⚠ NOTE: This congratulations screen/popup is a great place to provide your user with any additional information, right before the user makes their first payment!

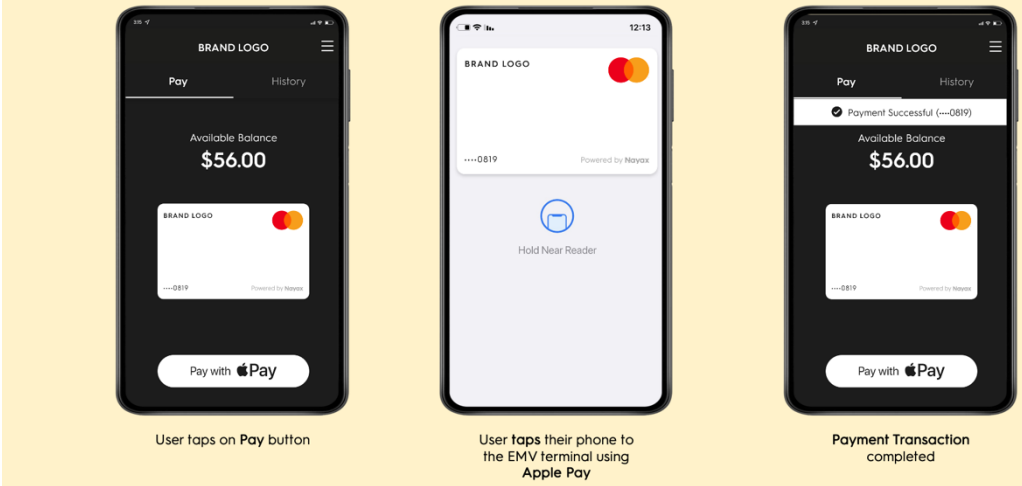
⚙ TECHNICAL INFO: The process of creating a payment method is relevant only for new users of the CoinBridge payment solution. If an existing user chooses to log in to your app using a different mobile device, Authentication will be required during Initialization, but new virtual card issuing won't be required (Only adding the virtual card to the Apple Wallet).

In cases where manual card issuing is the required process, please refer to Appendix 4 'PaymentMethodIssuing Method', for additional information. The Appendix also contains information on design guidelines and non-happy flows.

⚠ NOTE: Apple has established specific communication guidelines to follow when users have a virtual card issued but have not added it to the Apple Wallet on their device. For more detailed information, please consult Appendix 9 to this document.

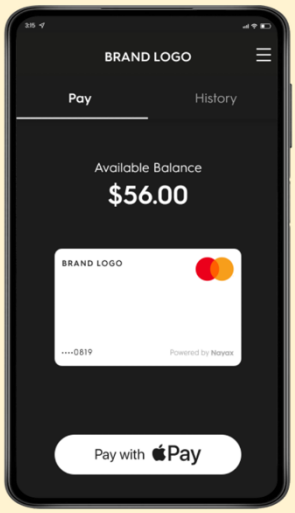
User Payment Journey

General



The user payment journey represents the Tap & Go experience that a user goes through whenever they wish to pay using your application.

Below, you will find a step-by-step account of what the user should see on-screen at each point of the flow.



PAYMENT SCREEN

The Payment screen represents the 'Home Screen' of the CoinBridge solution.

Your users will be able to make payments and view payment-related information/ activities via this screen.

The Payment screen must include:

- The user's available balance
- An image of the virtual card (see Appendix for card art style guide)
- 'Pay with Pay' CTA button (See Appendix 8 for button style Guide)

This is the main payment screen of your application, from which your users will be able to perform all Tap & Go related activities.

To enable a smooth payment experience, your application must include the following:

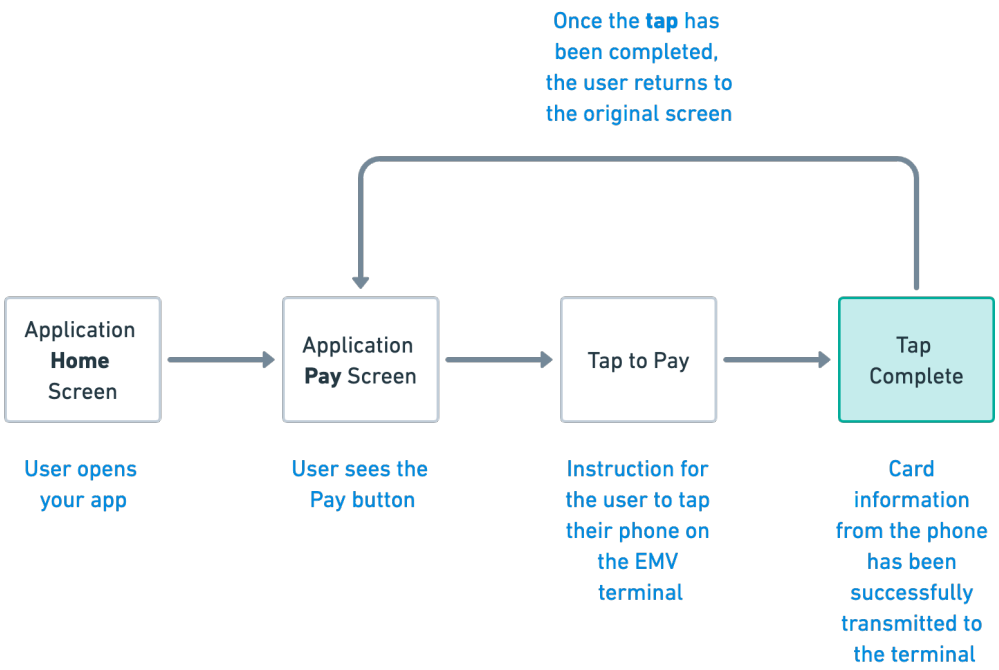
1. **Payment Screen,** showing:
    a. The balance available to the user for making payments (in the relevant currency).
    b. Image of the virtual card - designed according to the Card Art Style Guide found in Appendix 7 to this document.
    c. Payment restrictions/ spending policy guidelines (if applicable).
    d. Payment CTA button, using Apple's 'Pay with Pay' button - it is crucial to follow the Apple Pay style guide found in Appendix 8 to this document.
2. **Payment flow screens** to support:
    a. Happy flow
    b. Cancel payment flow

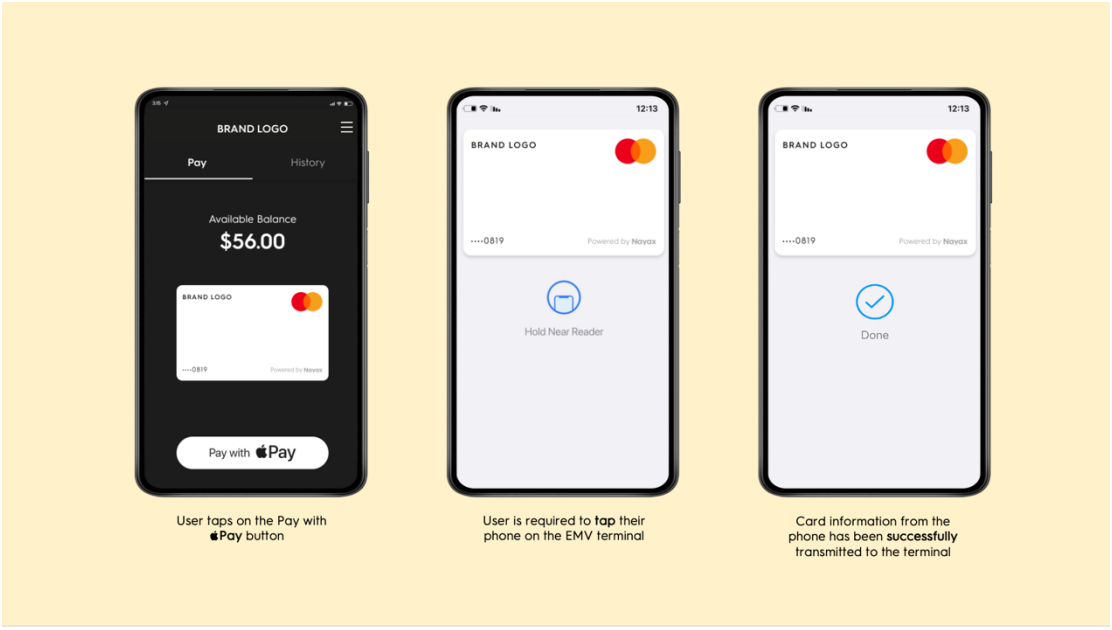
    See below screen mockups for reference.

⚙ TECHNICAL INFO: When your user is showing intent to pay (i.e., they tap the Pay button) – your application should call the `PrepareToPay` method. A successful response from this method (`ReadyToPay`) will automatically redirect the user to their Apple Wallet, where the virtual card will have been selected, to enable payment via the EMV terminal.

The user payment journey contains a number of steps and SDK calls, as set out in the flows below:

Scenario A: Tap Completed (Happy Flow)

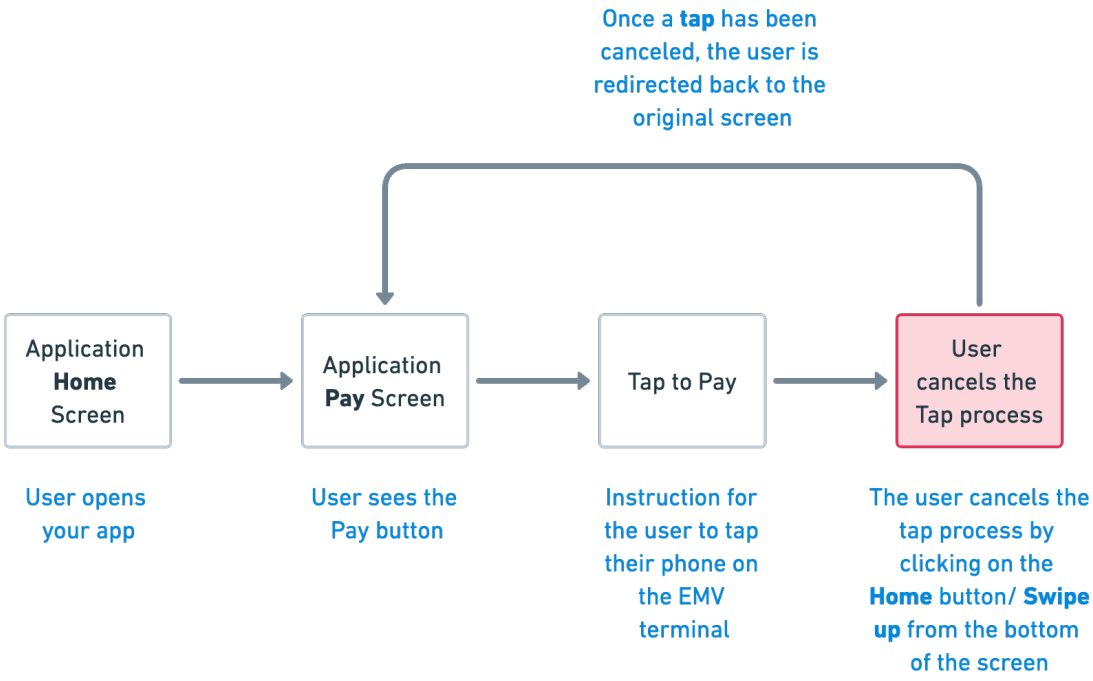


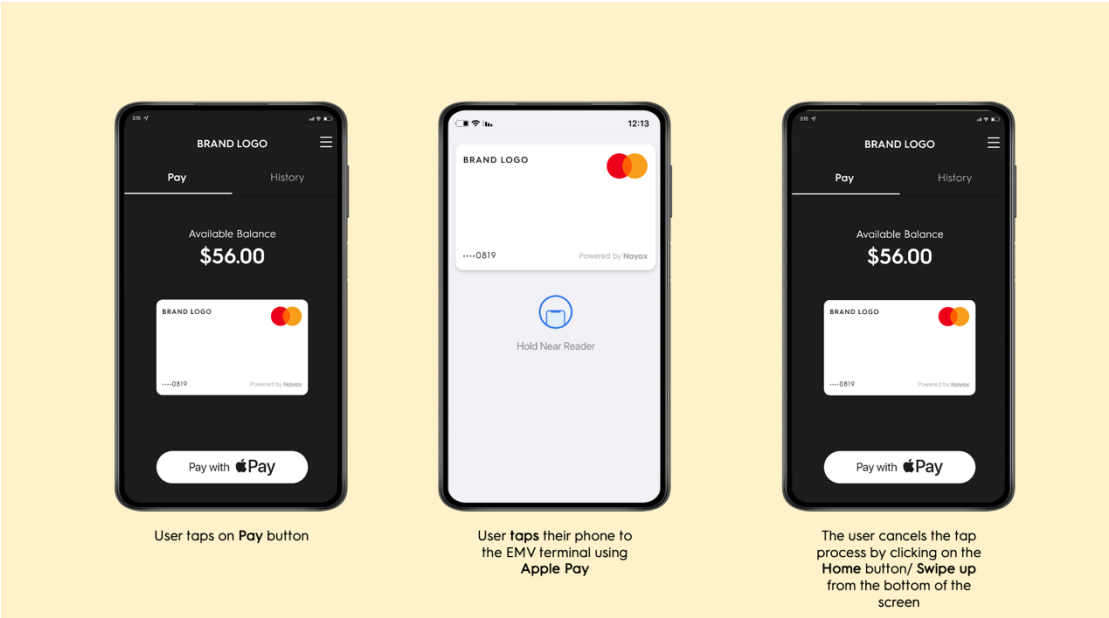


⚠ NOTE: The tap process ends with Tap Complete. However, a successful flow completion does not indicate that the transaction has been approved, only that information from the app was successfully transmitted to the EMV terminal via the device's NFC.

The outcome of the payment flow - known as the **Transaction Resolution** - will be Approved/ Declined. More details on this can be found in the next chapter.

Scenario B: Payment Action Canceled by User





When the SDK is prepared for payment and expects an EMV-over-NFC transaction (before the payment timeout is reached), there might be cases where the app will need to abort the process.

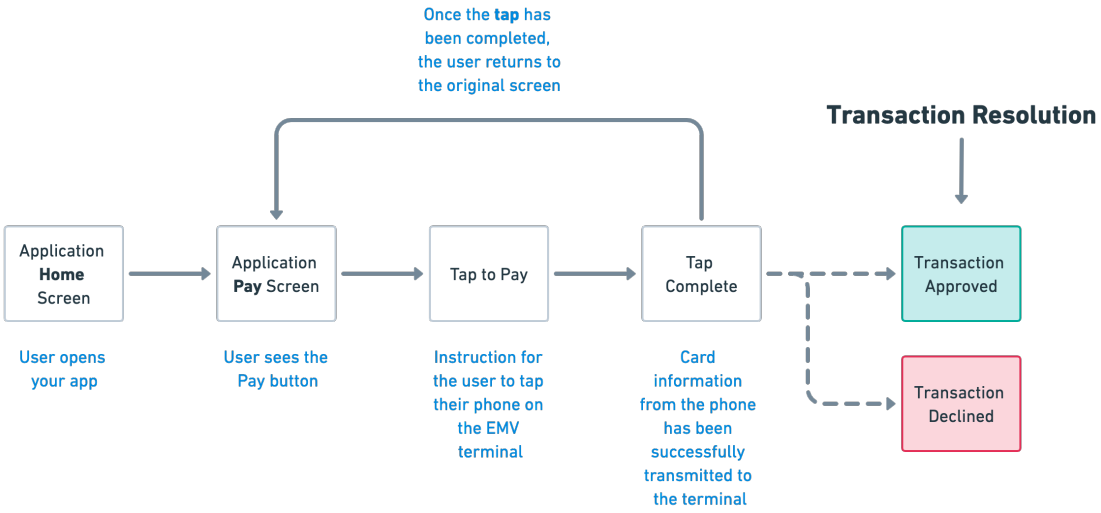
This might happen when the user has aborted the Payment action (by clicking on the **home** button or **swiping up** from the bottom of the screen), or your application decided that it shouldn't allow the user to make a payment attempt at this stage.

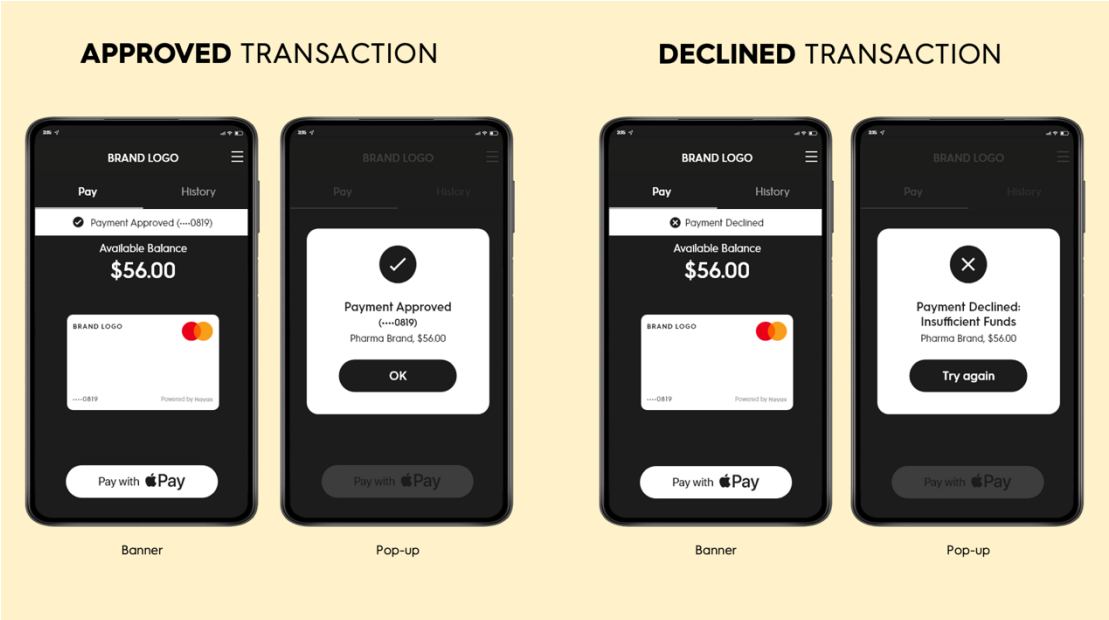
In order to prepare the SDK after a payment action has been aborted, the payment flow should be restarted as a new payment flow (for further details, refer to our SDK documentation).

Transaction Resolution and History

General

The transaction result and all information related to the transaction will be communicated to your loyalty/ user management platform via **CoinBridge's Back-End Connector.**





After a transaction attempt has been made via your application, the merchant's EMV terminal will communicate with the Visa or Mastercard network to authorize the transaction request.

CoinBridge's Back-End Connector will receive a transaction authorization request (from Visa or Mastercard's card network).

CoinBridge's Back-End Connector will compare the transaction details with the user's available balance and spending policies.

Based on the above (and additional extensive account, fraud, and risk related verifications), the CoinBridge Back-End Connector will determine whether to approve or decline the transaction attempt.

The decision (with additional supportive data) will be communicated to your loyalty or user management platform via CoinBridge's Back-End Connector.

⚠ NOTE: Once a transaction is completed, the user should be notified of the transaction resolution.

Transaction notifications in the Apple Wallet provide users with real-time updates and instant alerts about their card transactions. These notifications are sent as push notifications to the user's iOS device, appearing as alerts on the lock screen or in the notification center. They include transaction information such as the merchant's name, transaction amount, and the last few digits of the card used.

However, users may control their notification settings, enabling or disabling notifications for specific cards, and customizing settings based on transaction amounts. Typically, they do not intend to disable their wallet notifications. However, it is important to be aware that if they do, the user will not receive transaction resolution notifications until they open your application and check their payment history.

Below, we provide a suggested approach for implementing additional push notifications from your application, specifically for transaction resolution. It is important to note that this suggestion is optional, and the decision to implement it rests with you.

We suggest using push notifications in addition to pop-ups to notify the user of the transaction resolution.
The transaction result (**approved** or **declined**) must be shown to the user and must contain the following:

In the case of an **approved** transaction:

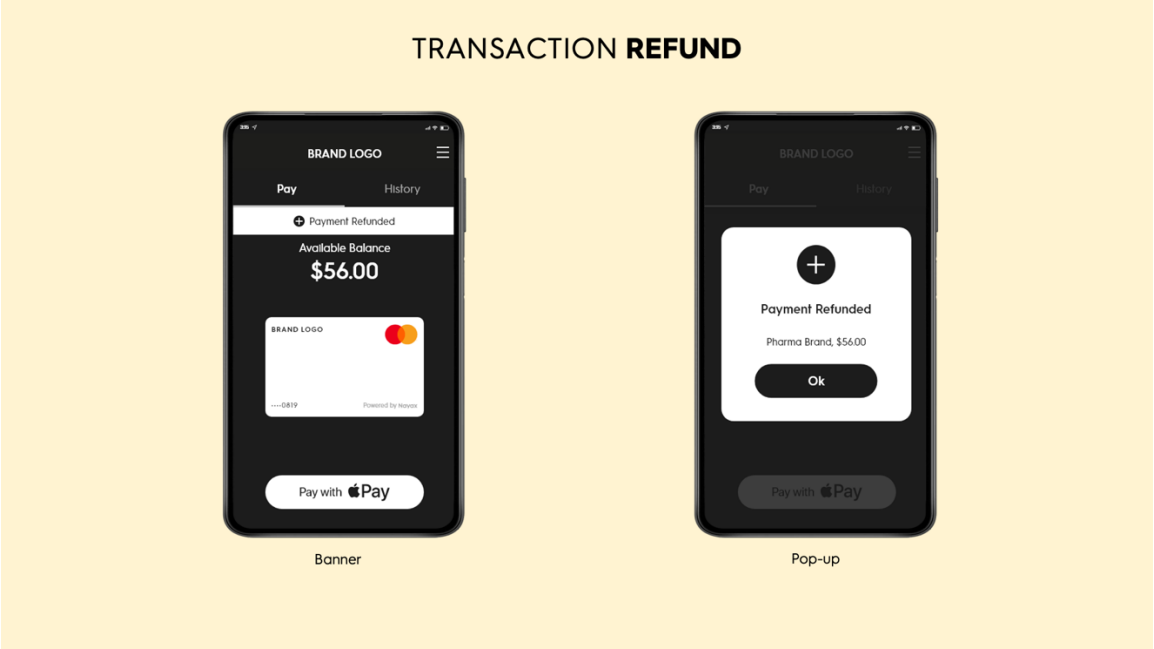
- An indication that the transaction has been approved.
- Additional transactional data (transaction amount, merchant name, etc.).

In the case of a **declined** transaction:

- An indication that the transaction was declined.
- The reason(s) for declining the transaction, using the data communicated from the CoinBridge Back-End Connector to your platform (see Appendix 5 to this document for further information regarding decline reasons).

Additional Transaction Reporting Types

Transaction Refund refers to a case where the user has been refunded by the merchant.



The refund transaction will appear as an independent transaction, with a unique transaction ID, and won't be connected to any historical transaction that may exist.

As in a regular transaction, the refund amount will always have a non-negative value, with a minimum value of zero, but in this case, you will have to top up the user's balance accordingly.

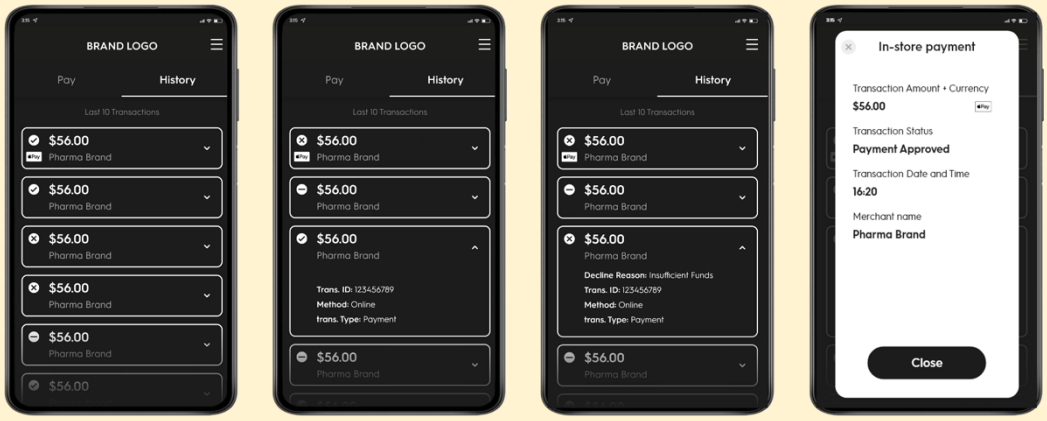
For further information about refund transactions, please see the CoinBridge Back-End Connector Manual.

In the event of a refund transaction, we suggest you show an appropriate message in the **Transaction History** (see the following section), such as: "You have been refunded the amount of [xx]."

Transaction Update refers to an update sent to your application regarding a case in which we sent a transaction for approval that actually failed due to a card network error. In such a case, the user will have seen that this transaction failed, although we would have reported it as a success.

For further information about this transaction reporting type, please see our dedicated CoinBridge Back-End Connector documentation.

Transaction History



Transaction History
List

Transaction History
(Approved Transaction Details)

Transaction History
(Declined Transaction Details)

Transaction History Specifics
(Details of a specific Approved Transaction)

It is important for users to be able to access their transaction history.

We recommend creating a dedicated transaction history screen to display end-user payment actions and provide end-users with additional information for each transaction.
In most cases, a payment/ transaction history module already exists within your application. If so, only minor adjustments will be required.

CoinBridge will provide you with detailed information about the transaction once it has been completed. Please note that the transaction details in the examples above are for illustration purposes only.

⚠ NOTE: The user must be able to differentiate between approved and declined transactions.

⚠ NOTE: In the event of declined transactions, it is crucial to display the decline reason to the user.

⚠ NOTE: **Apple** requires that each transaction made using Apple Wallet be marked with an indication that the payment was made using Apple Pay. This requirement helps ensure transparency and clarity for users regarding their payment method within the Apple Wallet ecosystem.

User Logout

This section covers what needs to happen when a user logs out of your application.
It's crucial to quickly let the CoinBridge SDK know when this happens.

Initiating the logout method within the SDK triggers the systematic clearance of all user-specific payment data. This action ensures that the SDK is left in a clean state, ready for the subsequent login sequence.

⚠ NOTE: It's mandatory to do this. If you forget to inform the CoinBridge SDK when a user logs out, it could cause mismatch with user identification and transaction information.

Appendices

Appendix #1 – Initialization Method

Initialization method is an asynchronous operation that may take a few seconds to complete.

each time your application starts, but only after the SDK has been successfully authenticated (when the IsAuthenticated call returns **True**).

It is strongly advised to perform SDK initialization in the onCreate() method of the payment activity. Once the initialization process wraps up, the OnInitCallback will hold details about the user's payment method readiness state, incorporating the "HasCard" and "CBPaymentStatus" classes.

If the SDK integration has been completed correctly, the initialization method should be failproof.

For more technical information, please refer to our SDK documentation.

Please see below a list of errors that may occur:

Initialization Call Error Values	Error Description
`SDK_NOT_AUTHENTICATED`	The CoinBridge SDK is not authenticated, SDK should first authenticate before initialization
`DEVICE_NOT_ELIGIBLE`	The device is not eligible for payment (this may be because the device is not NFC equipped or is rooted, etc.)
`PERMISSIONS_NOT_GRANTED`	The CoinBridge SDK did not receive one or more of the mandatory permissions
`INIT_ALREADY_STARTED`	Not an error! Please wait for a corresponding callback
`NETWORK_ERROR`	The CoinBridge SDK could not access the network. This may be because there is no internet connectivity, or because the CoinBridge SDK lacks permission to access the network
`GENERAL_ERROR`	An unknown error has occurred. Please contact CoinBridge support for further information

Appendix #2 – Authentication Method

Authentication, commonly referred to as Auth, is a critical process that involves synchronizing the unique identifier of a user with the CoinBridge SDK and the CoinBridge Back-End Connector.

It is important to note that this method must be called only when required, namely when the IsAuthenticated call returns False.

Additionally, the authentication call should contain the user's specific identifier, which will later be used for communication between the CoinBridge Back-End Connector and your loyalty or user management platform.

For further information, please refer to the SDK documentation provided by CoinBridge.

Authentication Call Error Values	Error Description
`INVALID_AUTHENTICATION`	Authentication failed. This may happen due to an invalid User ID
`ALREADY_AUTHENTICATED`	Not an error! Please action this as a successful call – Next step is to call Initialization method
`NETWORK_ERROR`	The CoinBridge SDK could not access the network. This may be because there is no internet connectivity, or because the CoinBridge SDK lacks permission to access the network
`GENERAL_ERROR`	An unknown error has occurred. Please contact CoinBridge support for further information

Appendix #3 – PaymentMethodStatus Method

General

The process for creating the CoinBridge payment method is comprised of two main flows that must take place in order for the user to be able to make payments:

- Virtual card issuing – Issuing a virtual card for the specific user.
- Virtual card provisioning - Adding the token of this virtual card to the CoinBridge SDK (integrated within the specific instance of the mobile application).

⚠ NOTE: The above processes are performed by the CoinBridge SDK, in the background.

Timing

Once the Initialization flow has been successfully completed, CoinBridge will initiate card issuing & card provisioning automatically. You will then be notified of the CoinBridge virtual card issuing & provisioning status.

⚠ NOTE: The virtual card creation process will be initiated automatically (by CoinBridge), or manually (by the user, using the `CreatePaymentMethod` method).

⚠ NOTE: Please consult the CoinBridge onboarding team to find out which flow applies to your application.

Status of Payment Method and Readiness to Make Payments

The payment method (virtual card) status will be provided by the CoinBridge SDK in the following scenarios:

- Following successful (completed) initialization.
- Following successful authentication.
- In the event of an update in payment method status, for example the card status has changed, the last 4 digits of the card have been updated, or the card has been provisioned.

The payment method status (`CBPaymentStatus`) will always include the following information:

Response Component	Response Description	Possible answers
`CBCardStatus`	Virtual Card Status	- **Active** - **notActive**
`lastfourDigits`	Last 4 Digits	- **Empty**: there is no card number - **4 Digits**: the last 4 digits of the card
`isProvisioned`	Card In Wallet	**- True:** When the card is in the wallet, it indicates that the user & device can be used to make payments **- False**: When the card is not yet in the device's Apple Wallet

Provision to Apple Watch

Please be aware that if a user has an Apple Watch and has provisioned their card to the Apple Watch wallet **only**, the "isProvisioned" status will still be set to False in your application. In this case, you should continue displaying the option for the user to Add to Apple Wallet, until they successfully provision their virtual card to the device itself.

It's important to highlight that, in correspondence with Apple's Mandatory Requirements, your application should provide users with the capability to add their virtual card to their Apple Wallet on additional paired devices, including devices like the Apple Watch.
To check whether the user has other paired devices where the virtual card can be provisioned, refer to the "Can Add Card to Apple Wallet" method in the SDK documentation.

CoinBridge strongly suggests incorporating a dedicated section within your application to address this scenario. This section can serve as a centralized hub for handling Apple-related functionalities, including the ability to add the virtual card to additional Apple devices, Apple Pay FAQs and any other relevant Apple-related information. This dedicated section will allow the best convenience and clarity for your users.

Appendix #4 – PaymentMethodIssuing Method

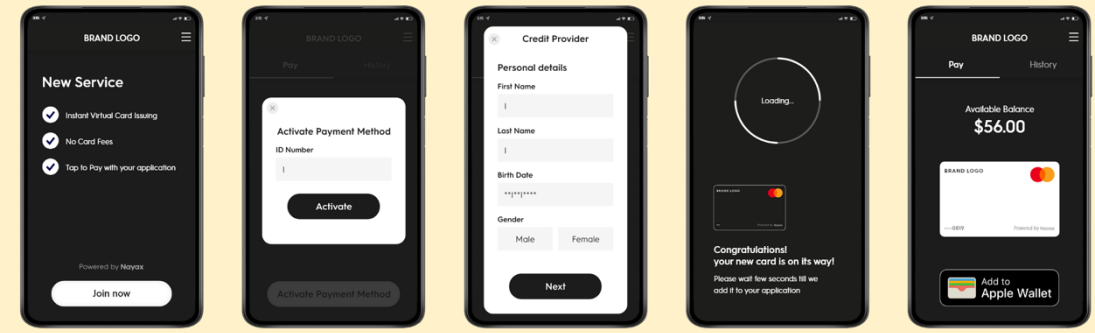
Manual vs. Automatic Issuing of Payment Method

Based on the regulations in your country, and as agreed in the MOU between you and CoinBridge, the CoinBridge payment method may be issued either automatically or manually.

- **Automatic** – CoinBridge will create the payment method and push it to the SDK wallet in the background, without any action being required from the user side (other than adding the card to Apple Wallet).
- **Manual** – The user will have to initiate the issuing process and may have to provide certain details in order to complete the creation of the payment method.

⚠ NOTE: The manual process may require your user to insert some personal details, and most may not understand why they need to provide personal information. Please make sure to properly onboard the user and engage them with the new CoinBridge feature to ensure a high onboarding rate.

Manual Virtual Card Issuing Flow (using the Create Payment method)



This method should be used only in the event that **all** of the following applies:

1. Your application is configured to support manual virtual card issuing.
2. The user has successfully completed Initialization & Authentication.
3. `initiCompleted(HasCard = False)`

Note: In some cases, manual card issuing may require additional user-specific information to be provided, such as SSN / National ID number, depending on regulations/ financial authority requirements in your country. In such cases, please make sure to provide this information as part of the Create Payment method payload. In the event that your application does not have this information, make sure to ask for it from your user in an appropriate manner (i.e., explain why this information is needed after introducing the new service).

Based on the card issuing flow, a successful Create Payment method may require the following (managed by the CoinBridge SDK):

1. User to provide additional details via a dedicated card-issuer form.
2. User to consent to additional terms (based on issuer/ regulations).

A dedicated **onSuccess** `createPaymentMethod` callback will be returned once the manual virtual card issuing flow has been finalized.

At this stage, the user should be notified, and flow should continue as explained above in the Completing the Pre-Payment Journey section.

Appendix #5 – Transaction Resolution and History

Reasons for Transaction Decline

Whenever a transaction is declined, CoinBridge's Back-End Connector will transmit a decline reason code and a description to your loyalty/ user management platform.

A decline reason code provides a high-level explanation for the decline, while a decline reason description provides a more detailed description and may include parameters (shown below in {} brackets) relevant to the declined transaction.

Whatever the decline reason transmitted by the CoinBridge Back-End Connector, it is important that your app represents your own messaging syntax aimed at your users.

Please see below the list of our standard decline reasons (as a function of your spending policies):

Decline Reason Code	Decline Reason Description
CB_SYSTEM_ERROR	Unexpected CoinBridge Error
INSUFFICIENT_END_USER_FUNDS	Transaction Amount {Transaction_Amount & Transaction_Currency} exceeds end-user's available balance
INSUFFICIENT_APPLICATION_FUNDS	Transaction amount {Transaction_Amount & Transaction_Currency} exceeds application's available balance
MCC_NOT_ALLOWED	This MCC is restricted {MCC_Code} under this policy
MID_NOT_ALLOWED	This MID is restricted {MID_Number} under this policy
COUNTRY_NOT_ALLOWED	This country is restricted {Country_Code} under this policy
AMOUNT_BELOW_THRESHOLD	Transaction amount {Transaction_Amount & Transaction_Currency} is below the minimum allowed amount
AMOUNT_EXCEEDS_THRESHOLD	Transaction amount {Transaction_Amount & Transaction_Currency} exceeds the maximum allowed amount

⚠ NOTE: This is not an exhaustive list of decline reasons. In the event of specific dedicated spending policies that you have chosen to define, other reasons may exist. In such cases, CoinBridge Support will provide the specific Code and Description.

Appendix #6 – Apple Wallet Behavior

In this section, we will explore the anticipated behavior of both Apple Wallet and your application across various scenarios and use cases. We aim to provide a comprehensive understanding of how these two entities interact and function together.

Scenario A – User removes the card from the wallet

If the user chooses to remove the virtual card from their Apple Wallet, the CoinBridge SDK's CBPaymentStatus method will indicate **IsProvisioned=False**.

As a result, the user will need to re-provision their virtual card to their Apple Wallet. To assist with this process, please ensure that the Add to Apple Wallet button is prominently displayed to the user in your application.

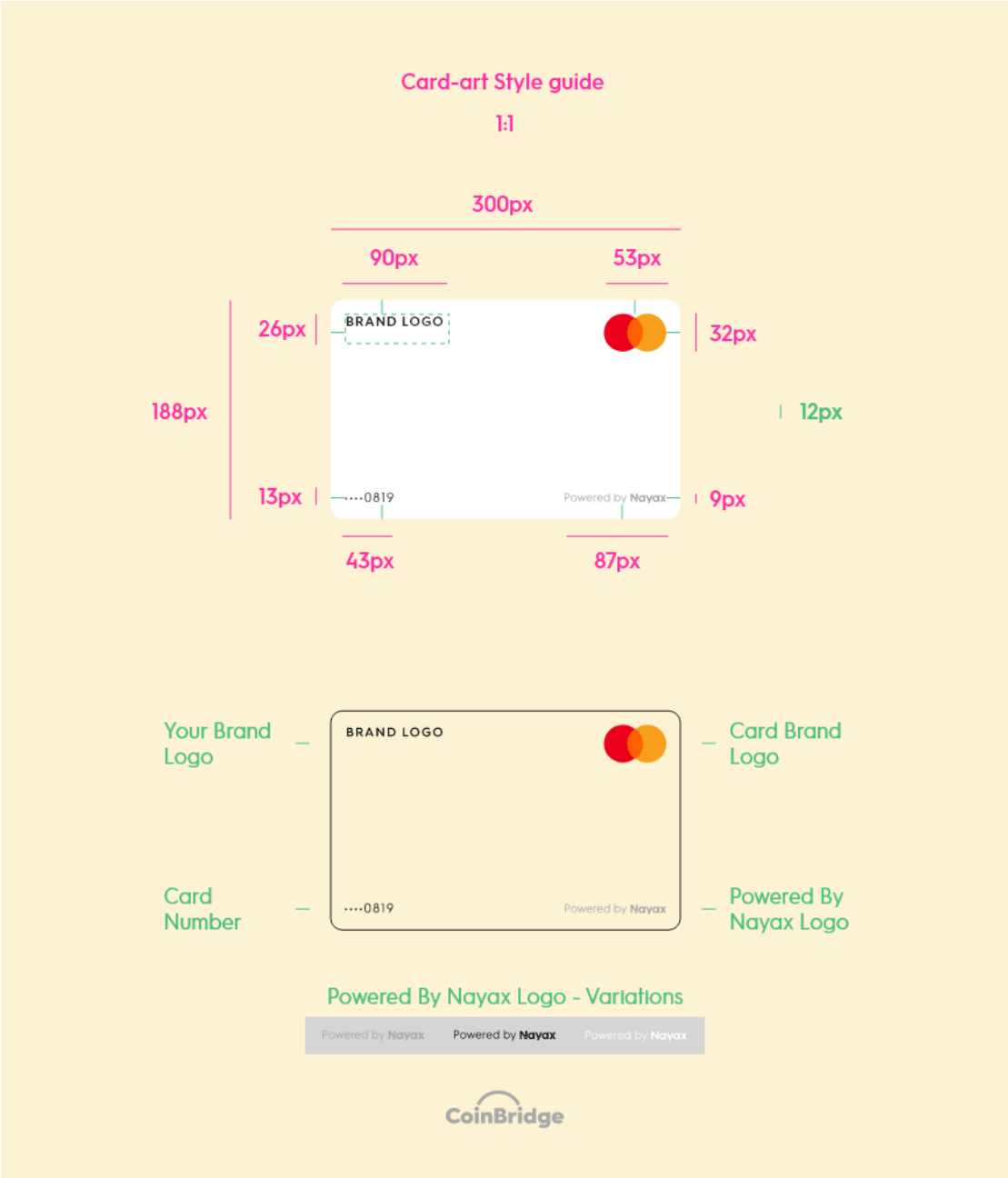
Scenario B – User moves to a new device

In this scenario, when a user transitions to a new iPhone and restores their data from the old device, Apple will prompt them to transfer their cards from the Apple Wallet to the new device. However, if the user agrees to this transfer, they will be required to enter the CV2 details (CVV).

The challenge arises when the user doesn't have access to this information, leading to a roadblock. To resolve this situation, the user needs to choose *not* to transfer the card to the new device. When they subsequently open your application, you will receive a IsProvisioned=False notification, indicating that the card is not provisioned in the Apple Wallet on the new device. At this point, you can provide a solution by allowing the user to add the card to their Apple Wallet within your application on the new device.

Appendix #7 – Card Art Style Guide

All design elements can be found via this link: Card Art Design Elements



Appendix #8 – Apple Buttons Style Guide

Please use the Apple Button design guides in the following links:

- Add to Apple Wallet button
- Pay with Apple Pay button

Appendix #9 – Apple Communication Guide

Below are different scenarios in which Apple requires you to communicate with your users. For further information, refer to the Apple Guide provided by CoinBridge.

Scenario A - Provisioning Confirmation

Within one hour of provisioning, users must receive a notification confirming **successful provisioning activation**.

Successful Provisioning Activation is the conclusion of the provisioning process, when the virtual card has been both issued AND pushed to the Apple Wallet.

Component	Details
'From' address	Use a 'from' address that makes it clear that the message is from the Card Issuer. This address should typically be the standard servicing communication 'from' address
Subject	"You can now use Apple Pay"
Content	"Your [Insert Card Issuer] card ending in [XXXX] is ready for Apple Pay. You can use it wherever you see the contactless symbol or the Apple Pay mark."
Timing	Within **1 hour** of activation

Scenario B - Inactive Customers

The term **Inactive Customers** refers to any customer who, within 7 days of a Successful Provisioning Activation, has not used their virtual card to make any payments.

An automated email notification must be sent to all marketing opt-in users, advising them to notify **Inactive Customers** about the benefits of using Apple Pay.

Component	Details
'From' address	Use a 'from' address that makes it clear that the message is from the Card Issuer. This address should typically be the standard servicing communication 'from' address
Subject	"You can now use Apple Pay"
Content	"Your card ending [XXXX] is now enabled with Apple Pay. You can use Apple Pay in-store, on the web using Safari, and in apps. Just look out for one of these symbols: [Insert contactless symbol and Apple Pay mark]."
Timing	Within **7 days** of successful provisioning

Scenario C - Activation without Provisioning

The term **Activation without Provisioning** refers to customers that completed the Card Issuing Activation, without provisioning the virtual card to their Apple Wallet. In the app, we show the Add to Apple Wallet button in this instance.

In the event of Activation without Provisioning, you should send an email to all relevant cardholders within 2-3 days (and at least once per quarter).

Component	Details
'From' address	Use a 'from' address that makes it clear that the message is from the Card Issuer. This address should typically be the standard servicing communication 'from' address
Subject	"Set up Apple Pay for your virtual card ending in [XXXX]"
Content	"We note that you have activated your virtual card In the XXX app, but have not yet added it to your Apple Wallet. To set up Apple Pay, in the [insert Application name] app, select the virtual card and follow the instructions. You can use Apple Pay in-store, on the web using Safari and in apps. Just look out for one of these symbols: [Insert contactless symbol and Apple Pay mark.]" + Link directly to the In-App Provisioning Process
Timing	Within **2-3 days** of successful provisioning, and once a quarter as a matter of routine

Scenario D – Offering an Incentive

In the following cases, you should offer a cardholder an incentive of a minimum of at least 20 Israeli Shekels (ILS 20):

- To complete provisioning - where the cardholder has commenced provisioning of a virtual card onto the Apple Pay platform, but has not completed the provisioning within seven (7) days; or
- To make a successful Apple Pay transaction - where the cardholder has completed provisioning of a virtual card but has not made a successful transaction with Apple Pay within fourteen (14) days of provisioning.

⚠ NOTE: This incentive may be sent only once per User_id

Component	Details
'From' address	Use a 'from' address that makes it clear that the message is from the Card Issuer. This address should typically be the standard servicing communication 'from' address
Subject	"Start using Apple Pay and enjoy 20 ILS on us!"
Content	"We noticed that you haven't used your card with Apple Pay yet, and we wanted to remind you about this secure and convenient way to pay. To get you started, we have added 20 ILS to your wallet, which you can use to make purchases with the [insert Application name] app."
Timing	1. **7 days** after the user started provisioning, without completing it 2. **14 days** after provisioning was successfully completed, without the user making a payment with the virtual card