Tokenization & Contactless

More knowledge on ApplePay, GooglePay, HCE and other ways of NFC and eCommerce payment using tokenization (MDES and VTS)

- Issuer Wallet and Apple Pay / Google Pay - Differences
- NFC on iPhone/iOS
- Pay by Account - NFC from bank account
- On-device tokenization in India
- Push Provisioning Step by Step
- Push Provisioning & Web Push Provisioning

Issuer Wallet and Apple Pay / Google Pay - Differences

As we have implemented more than 50 contactless and tokenization projects for banks, fintechs and other payment institutions, let me share a quick view on key differences between various types of contactless payment technologies.

X-Pays

If you are a card issuer in today's world, you usually need to implement **Apple Pay** and **Google Pay** to let your users benefit from various payment activities on mobile phones. We think it is obligatory in today's world for standard card use cases. The power of Apple and Google is so strong that avoiding these platforms really impacts your customers.

In general, implementation of these technologies is not difficult. If you use our **Token Management Platform**, implementation time can be reduced to weeks. Additionally, you have to sign contracts with Apple and Google. In the case of Apple, they charge additional fees for registering a card at Apple Pay. In the case of Google, they collect all transactions of your users to earn money on advertisement and data management. These are key disadvantages. In both cases you have to follow their requirements and changes, but if you cooperate with certified providers, you do not have problems with this, as a processor can solve these problems on your behalf.

Both Google and Apple solutions enable contactless, inApp and eCommerce payments on their browsers. The non-contactless payment transactions are an important part of these projects. You should focus not only on contactless payments.

It is worth mentioning that implementation of tokenization usually gives you access to other **X-Pays** like Fitbit Pay, Garmin Pay or others. They are much smaller companies and we treat them as nice-to-have in card issuing projects today.

Issuer Wallet SDK

Before Apple Pay and Google Pay appeared, both **Mastercard** and **VISA** invented other ways of contactless payments on mobile phones. They were called differently, but today they are mainly called Issuer Wallets. In such cases you do not sign contracts with Apple or Google, but you implement technology (both mobile SDKs and backend) that allows you to go live with contactless payments on mobile without signing contracts with Apple or Google. Actually it was possible for Android only, but recently (2023/2024) Apple allowed non-Apple Pay contactless payments on iPhones in the European Union.

In such cases you need to get and certify SDKs and backend components to go live with **contactless payments**. Such developments usually take 12-24 months and the software must be

kept updated all the time, so it is actually better that you try to use a certified partner for this activity to avoid on-going development costs just for your project. From a contactless use case perspective, transactions work in a very similar way to X-Pays, but you have more flexibility. On Android, for example, you can implement a contactless payment just after unlocking the phone screen. You can - but do not have to - ask for additional authentication. You are also sure that data of your users and their transactions will not be shared with external entities (Apple and Google) for their benefit.

A big advantage of the Issuer Wallet SDK is that it can work not only on Android phones - for example, we have live implementations on Huawei devices. This detail has an important business impact on your users.

In today's world, working with Apple or Google is obligatory in our opinion, but we strongly recommend implementing Issuer Wallets at the same time, as it will give you more flexibility and business security in the long run. The costs and processes do not differ a lot, but the additional benefits of an issuer wallet such as flexibility, more devices, lower transaction costs make it worth implementing.

Read more: **Push Provisioning to Google Pay and Apple Pay**

Thanks for reading.

NFC on iPhone/iOS

What happened?

As part of an agreement between the European Union and Apple, Apple has decided to open access to its NFC module to 3rd party developers. It allows the creation of solutions for contactless payments (HCE), an alternative to Apple Pay.

This article aims to explain the challenges and opportunities related to this technology.

How it works?

- **NFC payments.** Users of participating third-party banking or wallet apps can initiate NFC transactions from within the app with compatible NFC terminals.
- **Default app settings.** Users can choose any eligible app as their default contactless payments app which will enable the app to support Field detect and Double-click features.
- **Field detect.** The default contactless payments app automatically launches when the user places the device in the presence of a compatible NFC terminal and after user authentication (if the iPhone is locked).
- **Double-click.** The default contactless payments app automatically launches when the user double-clicks the side button (for Face ID devices) or the Home button (for Touch ID) and after user authentication (if the iPhone is locked).
- **Payment support for non-default apps.** Eligible apps running in the foreground can prevent the system default contactless app from launching and interfering with the payment.

What are the differences compared to Android?

Since 2013 Android allows implementing alternatives to Google's own Google Pay, and there are already a few mature solutions on the market. Apple's NFC API offers very similar capabilities from both a technical and user experience perspective. However, there are a few differences:

- **Not possible to directly ask a user to set your application as the default NFC payment app** - on Android, when users open your app, they can be presented with a dialog window that asks them if they want to use your app, as the default NFC payment app. Apple's documentation doesn't seem to hint at such a functionality.
- **Apple needs to give you special entitlement to access the NFC module** - without Apple's approval, it's not possible to include NFC payments into your app. This entitlement can be requested here: https://developer.apple.com/contact/request/hce-payments-entitlement/
- **Security certification** - every app enabling NFC payments needs the EMVCo certification. As NFC on the Apple platform is a new thing, it's still not clear how exactly security certification will look like, however due to fundamental differences between

Android and iOS we can expect slight differences.

What are the differences compared to Apple Pay?

Apple's API allows 3dr party developers to implement most of the functionality offered by Apple Pay. Two slight differences are:

- Power Reserve Mode - Apple Pay allows payments with the default card for some time after the iPhone battery is depleted.
- Express Transit Mode - allows to pay for public transport tickets in a few areas with compatible cards, without unlocking the iPhone. Full list of locations is available here.

Will it work outside the EU?

Companies registered in the European Economic Area can offer this functionality to customers based in EEA. The table below shows various combinations of companies wanting to offer HCE payments in their App and customers, and the expected outcome according to Apple's requirements.

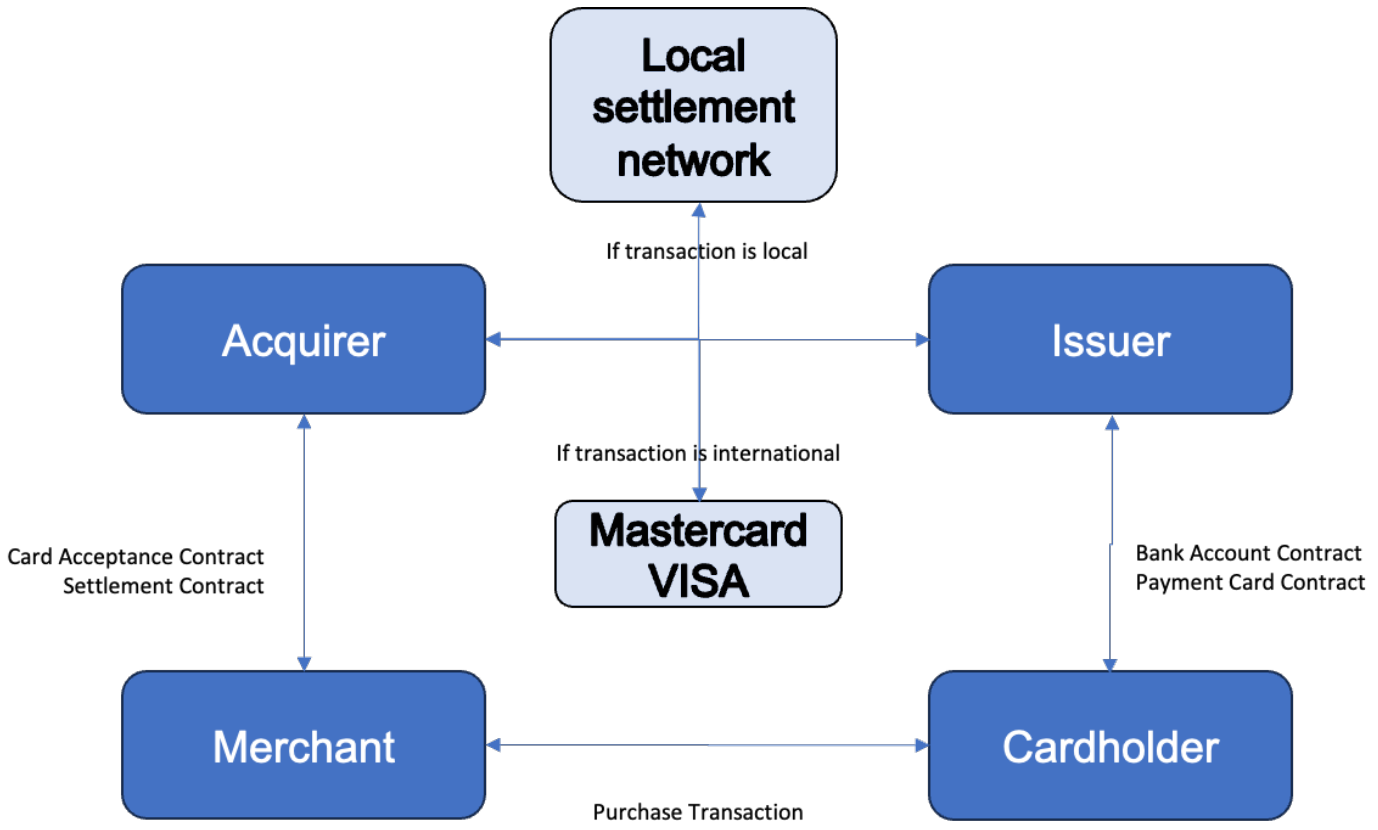
	Company developing App established & licensed for payments in EEA	**Company developing App not present in EEA or not licensed for payments in EEA**
EEA citizen, transacting in EEA country	□ HCE available	□ HCE not Available
EEA citizen, transacting outside EEA (traveling)	□ HCE available	□ HCE not Available
non-EEA citizen, transacting in EEA country (traveling)	□ HCE not Available	□ HCE not Available
non-EEA citizen, transacting outside EEA country	□ HCE not Available	□ HCE not Available

Pay by Account - NFC from bank account

As we have done several Pay by Account projects, I would like to share some information on what is the best way to implement such a solution.

Pay by Account or - in other words - **mobile NFC payments directly from a bank account** is an important product development step for many local schemes. Usually it is not enough to enable money transfers, QR payments or bill payments, and it would be beneficial to make use of the global authorisation and clearing network of Mastercard and VISA. In such a case, users of local schemes like BLIK in Poland or PIX in Brazil could be using or are using mobile phones to pay globally.

The solution is not difficult to implement using virtual cards and some local settlement ideas. There are the following architectural components:



The key implementation steps are as follows:

1. Every user interested in using contactless, eCommerce, or inApp payments will get a hidden virtual payment token
2. The token will be tokenised at a mobile phone of the user (usually Issuer Wallet SDK) and will be used for payments on the standard Mastercard or VISA acceptance network
3. In case the token is used to do domestic transactions at acquirers and terminals integrated with a local scheme, the authorisation and clearing will be routed to a local scheme
4. In case the token is used for international transactions, globally, at any terminal in the world, the authorisation and clearing will happen via the standard Mastercard or VISA settlement network

To enable this project you need to have a virtual card issuer or card issuing processor with super low fees per card - test us :) You will also need some support from **Mastercard or VISA** to agree on such a processing mechanism. Additionally, certified SDKs and backend components for contactless payments will be necessary (we can provide as well).

There have already been several implementations of similar schemes in the world. We are happy to discuss this setup in more detail.

Thanks for reading.

On-device tokenization in India

Payment law in India required a few years ago that server based card-on-file systems are not allowed. Instead, there is a necessity to store user data on-device to perform transactions.

This is an interesting topic that impacted a lot of local players like big merchants, Cred, Phonepe or Amazon so let me describe it in some details because maybe it will be implemented in a few years in other countries as well.

Requirement for storing tokens in a secure way on customer devices forces us to implement and certify secure SDK in which payment token data will be saved. It is a similar concept to standard HCE (Host Card Emulation) implementation for mobile NFC payments. While registering to a merchant or wallet system, the user adds a payment card, performs tokenization with approval (One-Time-Passwords) of the issuing bank and the token connected with his card is stored in this wallet / SDK.

Once we have a secure place of storing the user's token on his mobile phone we can use this token for multiple purposes:

1. NFC / contactless payments - the user uses his phone to perform transactions on a contactless acceptance network. Token and payment keys are transferred through the contactless interface to the payment terminal and acquirer for authorization.
2. inApp - the user chooses products and adds them to the basket in the merchant app, clicks that he wants to pay with a particular wallet or payment brand and confirms the transaction in a wallet app. The token is taken from an SDK, transferred to the cooperating acquirer in the form of a DSRP message (Dynamic Secured Remote Payment) and processed in standard way
3. web purchase - the user chooses products and adds them to the basket on the merchant website, clicks that he/she wants to pay with a particular wallet or payment brand and receives a push notification in his/her mobile app to finalize the transaction. As above, the token is taken from an SDK, transferred to the cooperating acquirer in the form of a DSRP message (Dynamic Secured Remote Payment) and processed in standard way. There could be a possibility to store the token inside the browser of the user on his laptop / PC but this requires more discussion with Mastercard and VISA.

To enable these use cases, several implementation points needs to be considered:

- types of devices - an inApp and web purchase can work on all devices (iOS and Android) but NFC is enabled on Android only. Apple is blocking the NFC access outside of the European Union today.
- VISA vs Mastercard - do you need a solution working for both schemes? Are there any local certification requirements?

- Issuers - are banks ready to connect to your new X-Pay wallet? Which banks are enabled on local markets?
- local on-soil requirements - is there a need to store data in the country? What are the legal impacts?

This is an interesting development and area of work. We are live with a few partners and are happy to work with new ones. Please contact us if you are interested.

Thanks for reading.

Push Provisioning Step by Step

Android - Push Provisioning to Google Pay

1. Request Access

Before enabling Push Provisioning in the Android application, you'll need to be whitelisted by Google. To initiate this process:

- Visit the **Google Pay Push Provisioning** launch page: https://developers.google.com/pay/issuers/apis/push-provisioning/android/launch-process
- Click "**Request access**" and complete the provided form.

2. Provide Application Details (Upon Approval)

Once your access request is approved, Google will contact you to gather additional information about your application:

- **Screenshots:** Submit screenshots that showcase the placement and functionality of the "Google Pay" button within your app.
- **User Flow Recording:** Provide a screen recording demonstrating the user experience for adding a card to Google Pay.

Important Note: Google Pay integration does not require any additional paid certifications.

3. SDK Configuration

- **Verestro Application Ownership:** If Verestro develops the application, our development team will handle the internal configuration of 3rd party Google Pay SDK.
- **Customer Application Ownership:** If the customer owns the application, customer's development team needs to get the Google Pay SDK and follow official documentation.

iOS - Push Provisioning to Apple Pay

1. Contact Apple

To begin enabling Push Provisioning for **Apple Pay**, initiate contact with Apple directly. Use the following email addresses:

- apple-pay-provisioning@apple.com
- applepayentitlements@apple.com

2. Application Information and Testing

Provide Apple with the details of your App Store application. Verestro will conduct integration testing using the same application (including the same ADAM ID) within the TestFlight platform.

3. FIME Certification (Optional)

While not mandatory, FIME can provide an optional certification process for Apple Pay Push Provisioning. Contact FIME directly for current pricing and instructions. This cost was approximately €3125 at the time of writing.

4. SDK Configuration

- **Verestro Application Ownership:** If Verestro develops the application, our development team will handle the internal configuration of the 3rd party Passkit SDK.
- **Customer Application Ownership:** If the customer owns the application, customer's development team needs to get the SDK of their choice (Apple Pay, Passkit, etc.) and follow their documentation.

Additional Notes

These instructions provide a high-level overview.

For detailed technical guidance and code implementation, please refer to the official Google Pay and Apple Pay developer documentation.

It will be necessary to implement the 3rd party **SDKs** and **APIs** provided by **Verestro** and used to encrypt card data and support additional processes in xPay's.

Push Provisioning & Web Push Provisioning

In this article we will focus on the process of pushing card data to X-Pays (like Apple Pay or Google Pay) – a process called Push Provisioning.

In many innovative card issuing use cases it is important to launch tokenization of cards to enable better customer experience and contactless payments with mobile. There are actually a few ways of registering card data at Apple or Google Pay from a user's perspective:

1. Manual provisioning – which means that the user provides card data one by one. After that the user accepts T&C, confirms this action with One Time Password received from the issuer / bank and gets the card tokenized.

2. Scanning card – the above mentioned process can be improved if the user can scan their payment card with their phone, so that the card data and expiry date appear automatically in the wallet. Not a difficult thing to do nowadays. This process is also very useful if a card visual can appear on the website and the user scans the card visual and data directly from their laptop.

3. Push Provisioning – in this case, the process starts with opening a mobile banking application. The user can click the “Add to wallet” or “Register to wallet” button and can be redirected to Apple or Google Pay to finalize registrations. Various SDKs enable quick registration of the user and card data. It is a very user-friendly process.

However, in some cases it is necessary that the process of Push Provisioning starts from the website – not from the mobile app. This is called Web Push Provisioning and it is a very innovative way of starting tokenization. There are available APIs that streamline such a process and it can be used in many cases. However, to make it happen special approvals of Google or Apple are needed, which is not easy to receive.

This use case can be interesting in multiple projects, for example:

- **Gift cards delivery** – the user opens a gift card on the website widget on their phone and can immediately add this gift card to the wallet.

- **Expense management cards** – the user opens a card generated by their employer on their phone and can immediately tokenize the card in ApplePay.

- **Emergency cards** – the user gets a card from their insurance or bank and can use it immediately on their phone for emergency transactions.

Please contact us if you are interested in testing those use cases.