

Multicurrency cards - 3 implementation options

Multi-currency topic is an interesting and important concept of card issuing that usually requires some explanation. Because of the very big market of currency conversion and usually very high fees of universal banks connected with international transactions, it became popular to implement multi-currency cards. Actually the first **Revolut** use case, heavily promoted several years ago, was connected with this topic. So let's go into details.

There is actually one problem that we want to solve when thinking of implementing multi-currency cards - how to enable the best and most **effective card payments in an international environment**? There are various approaches to this problem:

Scenario 1 - multi-currency cards and accounts

In this example we offer users multiple payment accounts in various currencies.

1. The user gets a single payment card connected with all accounts
2. In case the user pays with currency X, the authorisation system recognises transaction currency and debits account of currency X
3. In case there is no money on this account, system debits another (default) currency

This example is very often used, but it has a few disadvantages. The first is that the user must perform currency conversion before. It is an action before his/her travel and actually it is an unnecessary action from the logic's perspective. It should be more convenient for the user to have one account and cheap currency conversion during every transaction. But usually consumers like the solution because they can manage this currency problem in advance, see FX rate and can make decisions on how much money to convert.

Implementation of this scenario is not easy because card issuing companies either need to enable multi-currency functionality with Mastercard / VISA or to implement multiple settlement accounts with payment organizations and manage conversions accordingly based on transaction currency. There are additional fees that Mastercard and VISA charge for this service which can make this implementation costly.

Scenario 2 - currency conversion on a single account

Another way of solving the currency conversion topic is to think about how to enable the cheapest conversion during a transaction. In this example the user does not have to convert currency before his travel. He just uses his card while traveling. I personally like this approach the most because it

is easier for me but in reality many customers prefer scenario 1.

In this scenario, to have dynamic rates, there is a need for online FX API integration and dynamic management of rates during authorisation. Usually card issuers use static conversion rates offered by Mastercard and VISA but this leads to some additional costs and margins. Ensuring dynamic currency conversion during authorization and proper conversion management may be difficult to achieve.

Scenario 3 - multiple cards for different currencies

The third way of managing the multi-currency topic today in the virtual card environment is issuing multiple cards to multiple accounts in various currencies. In today's world this is easily achievable as the cost of card issuing went heavily down. It works in the way that users have several cards, connected with various accounts and card visuals, visible in **Apple Pay** or **Google Pay** with the currency of a particular card. The user can choose a card which is the most convenient for him/her.

In this scenario we need to offer an inexpensive currency conversion mechanism as the user needs to manage balances on each account separately and perform conversion in advance.

This is actually the cheapest scenario of implementation.

While thinking about the multi-currency topic, please consider various scenarios and ways of solving problems. Sometimes the default plan (scenario 1) can be very costly from the transaction processing perspective because of additional fees of payment schemes.

Thanks for reading.

Revision #1

Created 2 June 2025 12:02:48 by Alicja Switkowska

Updated 2 June 2025 13:17:17 by Alicja Switkowska