# System Design

## Part 1: Payment System

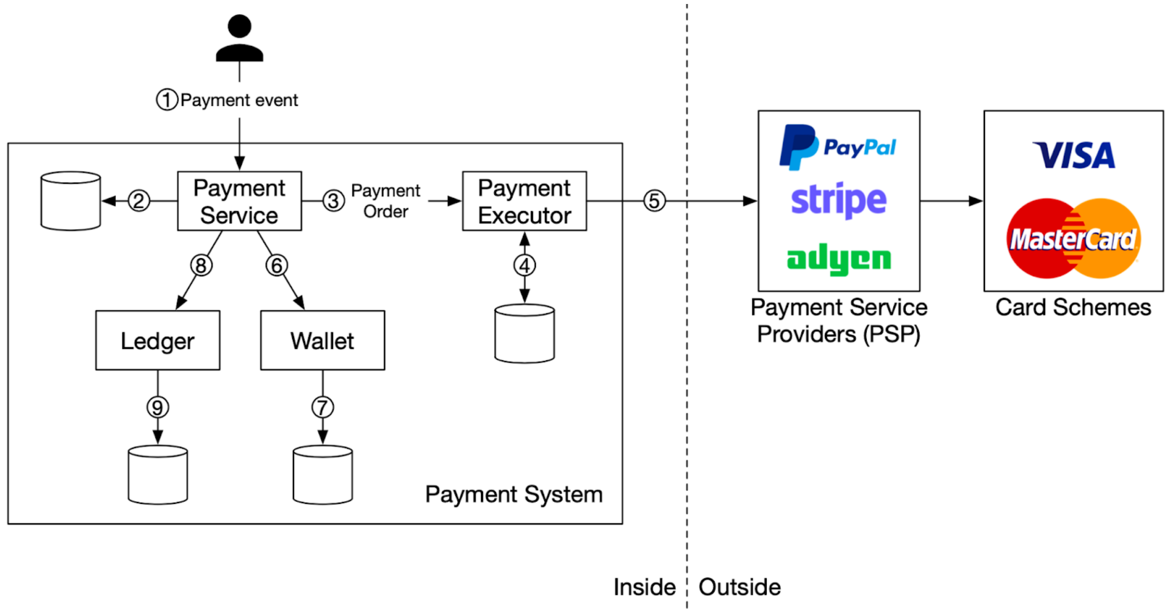
*By Alex Xu*

ByteByteGo

Some of you asked to get a PDF version of payment-related topics. Here you go. You can subscribe to receive updates or new topics here: **blog.bytebytego.com**

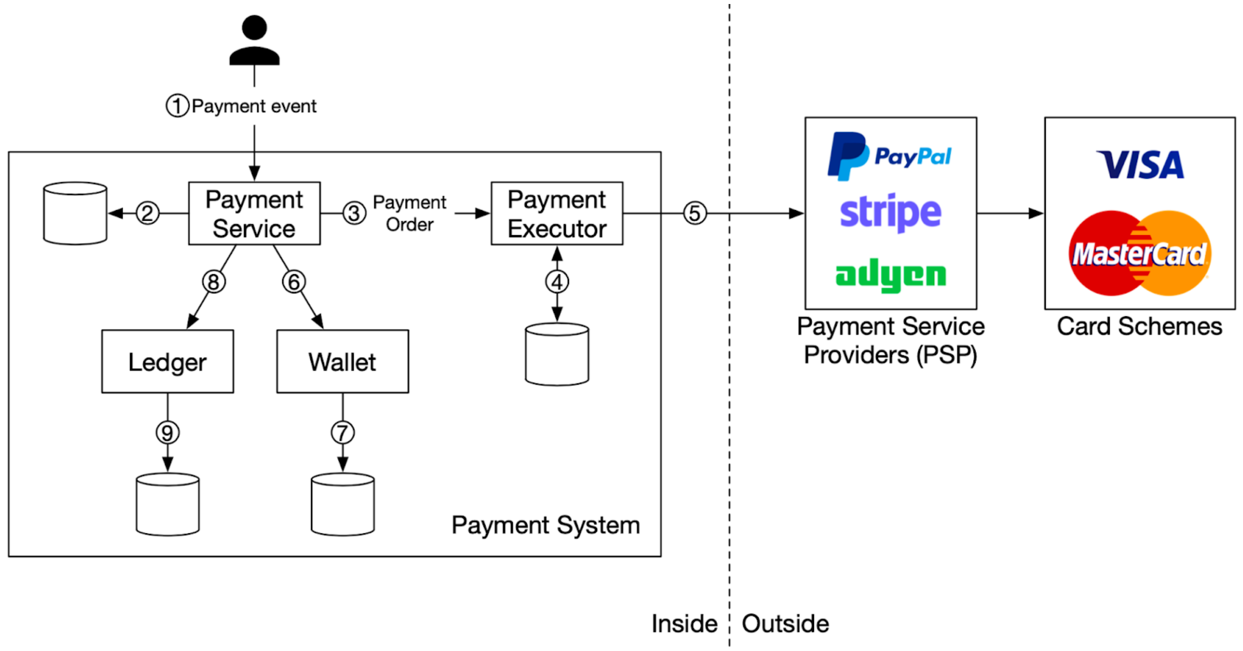
- Payment system
- Payment security
- Double charge
- Reconciliation
- Painful payment reconciliation
- Clearing & settlement
- Foreign exchange
- SWIFT

You can also find detailed payment system design in *Chapter 11 - Payment System* of my book: System Design Interview - An Insider's Guide: Volume 2.



# Nov 29, 2021 - Payment system

Today is Cyber Monday. Here is how money moves when you click the Buy button on Amazon or any of your favorite shopping websites.



1. When a user clicks the “Buy” button, a payment event is generated and sent to the payment service.

2. The payment service stores the payment event in the database.

3. Sometimes a single payment event may contain several payment orders. For example, you may select products from multiple sellers in a single checkout process. The payment service will call the payment executor for each payment order.

4. The payment executor stores the payment order in the database.

5. The payment executor calls an external PSP to finish the credit card payment.

6. After the payment executor has successfully executed the payment, the payment service will update the wallet to record how much money a given seller has.

7. The wallet server stores the updated balance information in the database.

8. After the wallet service has successfully updated the seller’s balance information, the payment service will call the ledger to update it.

9. The ledger service appends the new ledger information to the database.

10. Every night the PSP or banks send settlement files to their clients. The settlement file contains the balance of the bank account, together with all the transactions that took place on this bank account during the day.

—
Please follow Alex if you want to receive weekly updates about tech interviews, system design, or book writing. : https://bit.ly/linkedinaxu

# Jan 3, 2022 - Payment security

A few weeks ago, I posted the high-level design for the payment system. Today, I'll continue the discussion and focus on payment security.

The table below summarizes techniques that are commonly used in payment security. If you have any questions or I missed anything, please leave a comment.

| Problem | Solution |
|---|---|
| Request/response eavesdropping | Use HTTPS |
| Data tampering | Enforce encryption and integrity monitoring |
| Man-in-the-middle attack | Use SSL and authentication certificates |
| Data loss | Database replication across multiple regions and take snapshot of data |
| Distributed denial-of-service attack (DDoS) | Rate limiting and firewall |
| Card theft | Tokenization. Instead of using real card numbers, tokens are stored and used for payment |
| PCI compliance | PCI DSS is an information security standard for organizations that handle branded credit cards |
| Fraud | Address verification, card verification value (CVV), user behavior analysis, etc. |

—

Please follow Alex if you want to receive weekly updates about tech interviews, system design, or book writing. : https://bit.ly/linkedinaxu

# Jan 4, 2022 - Double charge

One of the most serious problems a payment system can have is to **double charge a customer**. When we design the payment system, it is important to guarantee that the payment system executes a payment order exactly-once.

At the first glance, exactly-once delivery seems very hard to tackle, but if we divide the problem into two parts, it is much easier to solve. Mathematically, an operation is executed exactly-once if:

1. It is executed at least once.

2. At the same time, it is executed at most once.

We now explain how to implement at least once using retry and at most once using idempotency check.

### Retry

Occasionally, we need to retry a payment transaction due to network errors or timeout. Retry provides the at-least-once guarantee. For example, as shown in Figure 10, the client tries to make a $10 payment, but the payment keeps failing due to a poor network connection. Considering the network condition might get better, the client retries the request and this payment finally succeeds on the fourth attempt.

### Idempotency

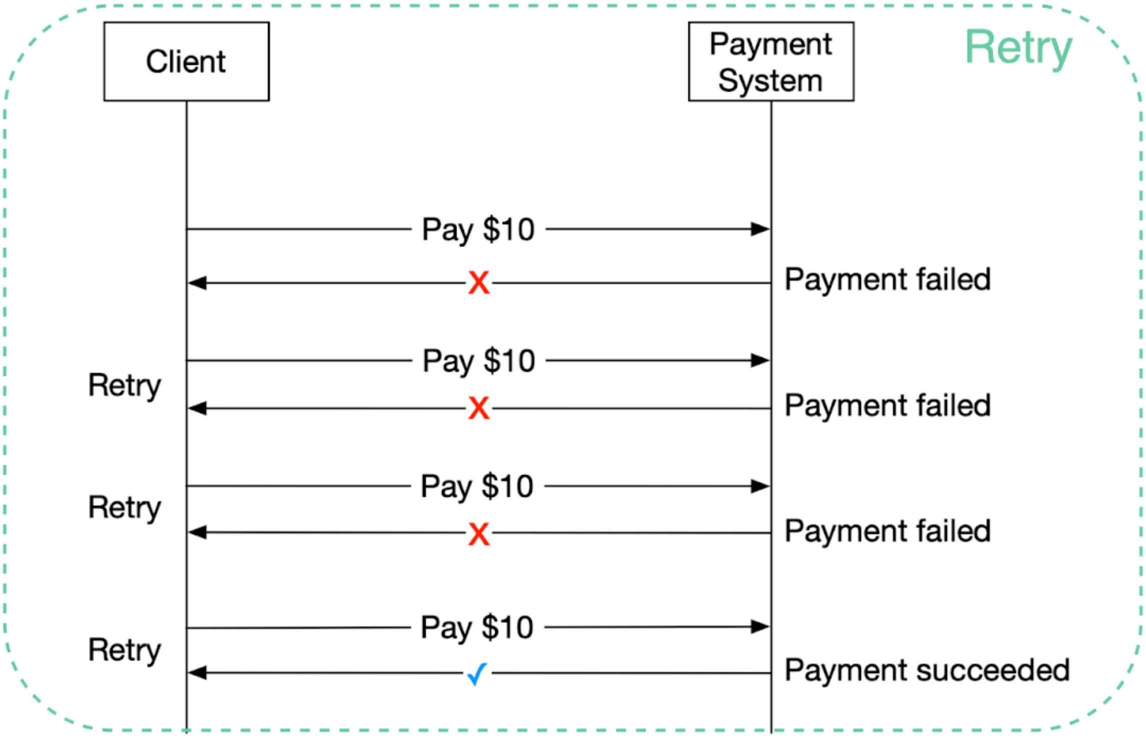
From an API standpoint, idempotency means clients can make the same call repeatedly and produce the same result.

For communication between clients (web and mobile applications) and servers, an idempotency key is usually a unique value that is generated by clients and expires after a certain period of time. A UUID is commonly used as an idempotency key and it is recommended by many tech companies such as Stripe and PayPal. To perform an idempotent payment request, an idempotency key is added to the HTTP header: <*idempotency-key: key_value*>.

—

Please follow Alex if you want to receive weekly updates about tech interviews, system design, or book writing. : https://bit.ly/linkedinaxu

# How to avoid double payment



Retry

Client

Payment System

Pay $10

Payment failed

Retry

Pay $10

Payment failed

Retry

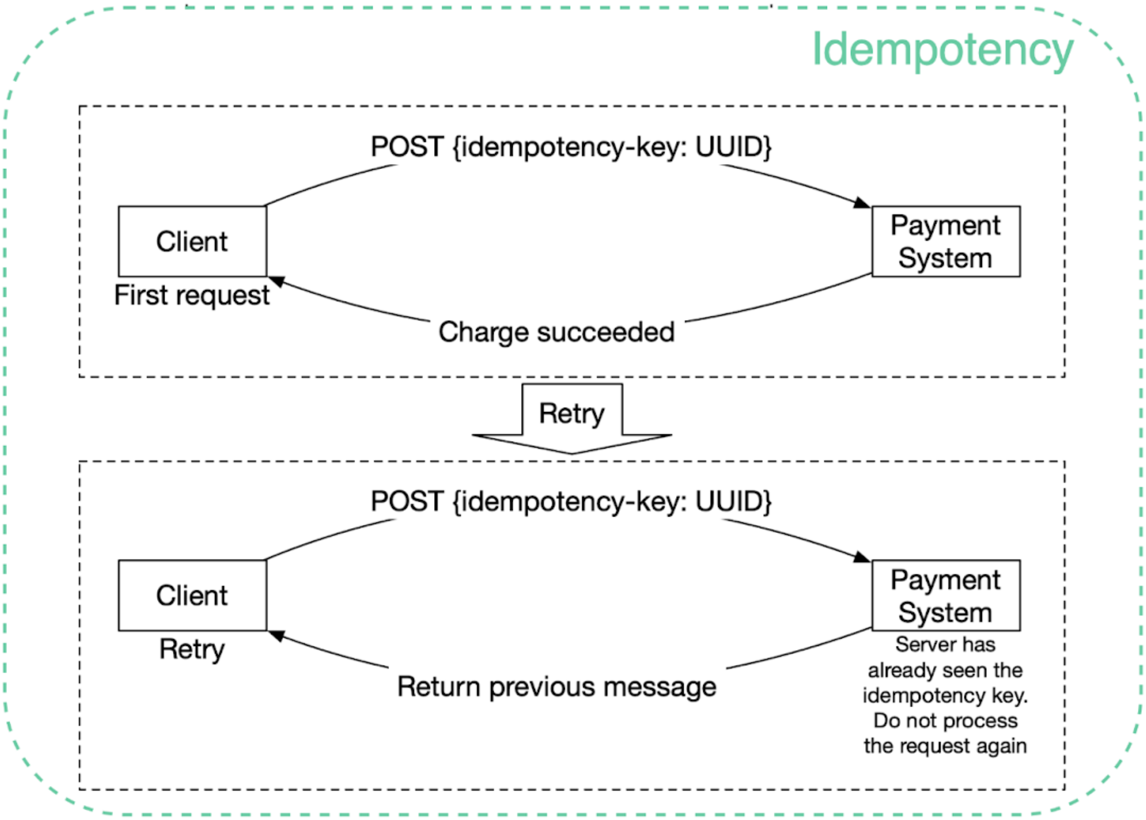
Pay $10

Payment failed

Retry

Pay $10

Payment succeeded



Idempotency

POST {idempotency-key: UUID}

Client

First request

Payment System

Charge succeeded

Retry

POST {idempotency-key: UUID}

Client

Retry

Payment System

Return previous message

Server has already seen the idempotency key. Do not process the request again

# Jan 11, 2022 - Reconciliation

**Reconciliation** might be the most painful process in a payment system. It is the process of comparing records in different systems to make sure the amounts match each other.

For example, if you pay $200 to buy a watch with Paypal:
- The eCommerce website should have a record of the purchase order of $200.
- There should be a transaction record of $200 in Paypal (marked with 2 in the diagram).
- The Ledger should record a debit of $200 dollars for the buyer, and a credit of $200 for the seller. This is called double-entry bookkeeping (see the table below).

Let's take a look at some pain points and how we can address them:

**Problem** 1: Data normalization. When comparing records in different systems, they come in different formats. For example, the timestamp can be "2022/01/01" in one system and "Jan 1, 2022" in another.
**Possible solution**: we can add a layer to transform different formats into the same format.

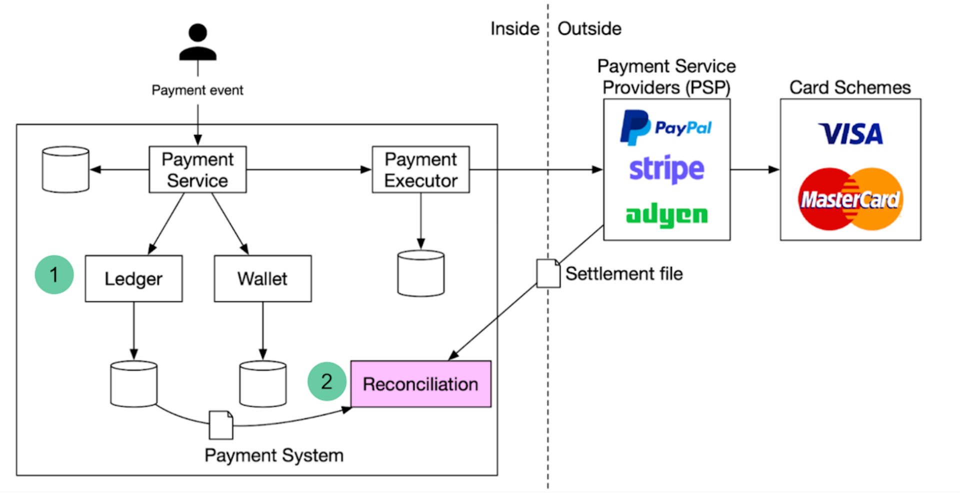
**Problem** 2: Massive data volume
**Possible solution**: we can use big data processing techniques to speed up data comparisons. If we need near real-time reconciliation, a streaming platform such as Flink is used; otherwise, end-of-day batch processing such as Hadoop is enough.

**Problem** 3: Cut-off time issue. For example, if we choose 00:00:00 as the daily cut-off time, one record is stamped with 23:59:55 in the internal system, but might be stamped 00:00:30 in the external system (Paypal), which is the next day. In this case, we couldn't find this record in today's Paypal records. It causes a discrepancy.
**Possible solution**:  we need to categorize this break as a "temporary break" and run it later against the next day's Paypal records. If we find a match in the next day's Paypal records, the break is cleared, and no more action is needed.

You may argue that if we have exactly-once semantics in the system, there shouldn't be any discrepancies. But the truth is, there are so many places that can go wrong. Having a reconciliation system is always necessary. It is like having a safety net to keep you sleeping well at night.

—
Please follow Alex if you want to receive weekly updates about tech interviews, system design, or book writing. : https://bit.ly/linkedinaxu

# Reconciliation in Payment System



## Double-entry Bookkeeping in Ledger

| Account | Debit | Credit |
|---|---|---|
| buyer | $200 | |
| seller | | $200 |

# Jan 17, 2022 - Painful payment reconciliation

My previous post about painful payment reconciliation problems sparked lots of interesting discussions. One of the readers shared more problems we may face when working with intermediary payment processors in the trenches and a potential solution:

1. Foreign Currency Problem: When you operate a store globally, you will come across this problem quite frequently. To go back to the example from Paypal - if the transaction happens in a currency different from the standard currency of Paypal, this will create another layer, where the transaction is first received in that currency and exchanged to whatever currency your Paypal is using. There needs to be a reliable way to reconcile that currency exchange transaction. It certainly does not help that every payment provider handles this differently.

2. Payment providers are only intermediaries. Each purchase does not trigger two events for a company, but actually at least 4. The purchase via Paypal (where both the time and the currency dimension can come into play) triggers the debit/credit pair for the transaction and then, usually a few days later, another pair when the money is transferred from Paypal to a bank account (where there might be yet another FX discrepancy to reconcile if, for example, the initial purchase was in JPY, Paypal is set up in USD and your bank account is in EUR). There needs to be a way to reconcile all of these.

3. Some problems also pop up on the buyer side that is very platform-specific. One example is shadow transaction from Paypal: if you buy two items on Paypal within 1 week of time between the two transactions, Paypal will first debit money from your bank account for transaction A. If at the time of transaction B, transaction A has not gone through completely or is canceled, there might be a world where Paypal will use the money from transaction A to partially pay for transaction B, which leads to only a partial amount of transaction B being withdrawn from the bank account.

In practice, this usually looks something like this:
1) Your shop assigns an order number to the purchase
2) The order number is carried over to the payment provider
3) The payment provider creates another internal ID, which is carried over across transactions within the system
4) The payment ID is used when you get the payout on your bank account (or the payment provider bundles individual payments, which can be reconciled within the payment provider system)
5) Ideally, your payment provider and your shop have an integration/API with the tool you use to (hopefully automatically) create invoices. This usually carries over the order id from the shop (closing the loop) and sometimes even the payment id to match it with the invoice id, which you then can use to reconcile it with your accounts receivable/payable. :)

Credit: A knowledgeable reader who prefers to stay private. Thank you!

—

Please follow Alex if you want to receive weekly updates about tech interviews, system design, or book writing. : https://bit.ly/linkedinaxu

## Jan 18, 2022 - Clearing & settlement

One picture is worth more than a thousand words. This is what happens when you buy a product using Paypal/bank card under the hood.

To understand this, we need to digest two concepts: **clearing & settlement**. Clearing is a process that calculates who should pay whom with how much money; while settlement is a process where real money moves between reserves in the settlement bank.

Let's say Bob wants to buy an SDI book from Claire's shop on Amazon.

- Pay-in flow (Bob pays Amazon money):
1.1 Bob buys a book on Amazon using Paypal.
1.2 Amazon issues a money transfer request to Paypal.
1.3 Since the payment token of Bob's debit card is stored in Paypal, Paypal can transfer money, on Bob's behalf, to Amazon's bank account in Bank A.
1.4 Both Bank A and Bank B send transaction statements to the clearing institution. It reduces the transactions that need to be settled. Let's assume Bank A owns Bank B $100 and Bank B owns bank A $500 at the end of the day. When they settle, the net position is that Bank B pays Bank A $400.
1.5 & 1.6 The clearing institution sends clearing and settlement information to the settlement bank. Both Bank A and Bank B have pre-deposited funds in the settlement bank as money reserves, so actual money movement happens between two reserve accounts in the settlement bank.

- Pay-out flow (Amazon pays the money to the seller: Claire):
2.1 Amazon informs the seller (Claire) that she will get paid soon.
2.2 Amazon issues a money transfer request from its own bank (Bank A) to the seller bank (bank C). Here both banks record the transactions, but no real money is moved.
2.3 Both Bank A and Bank C send transaction statements to the clearing institution.
2.4 & 2.5 The clearing institution sends clearing and settlement information to the settlement bank. Money is transferred from Bank A's reserve to Bank C's reserve.

Notice that we have three layers:
- Transaction layer: where the online purchases happen
- Payment and clearing layer: where the payment instructions and transaction netting happen
- Settlement layer: where the actual money movement happens

The first two layers are called information flow, and the settlement layer is called fund flow.

You can see the **information flow and fund flow are separated**. In the info flow, the money seems to be deducted from one bank account and added to another bank account, but the actual money movement happens in the settlement bank at the end of the day.

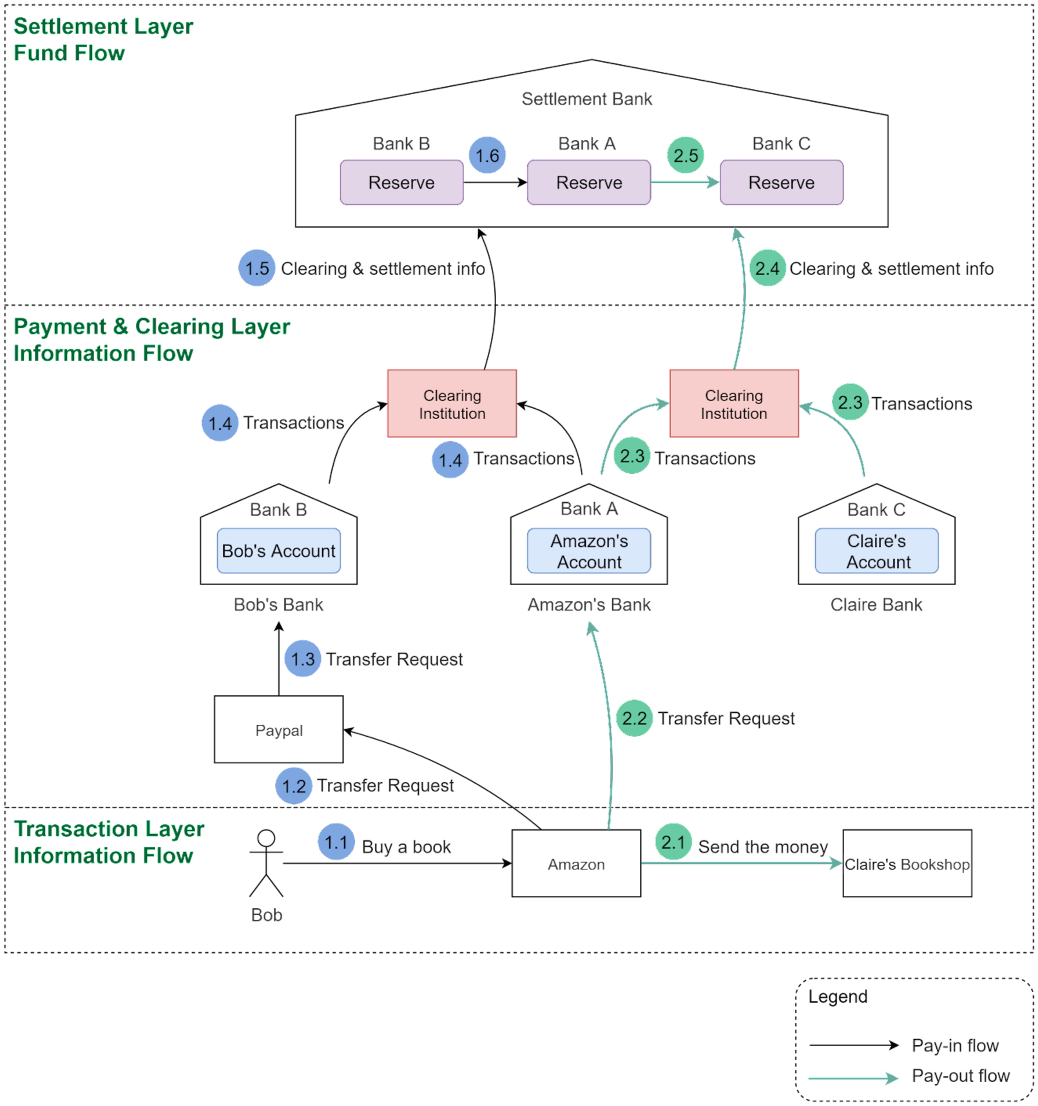
Because of the asynchronous nature of the info flow and the fund flow, reconciliation is very important for data consistency in the systems along with the flow.

It makes things even more interesting when Bob wants to buy a book in the Indian market, where Bob pays USD but the seller can only receive INR.

—

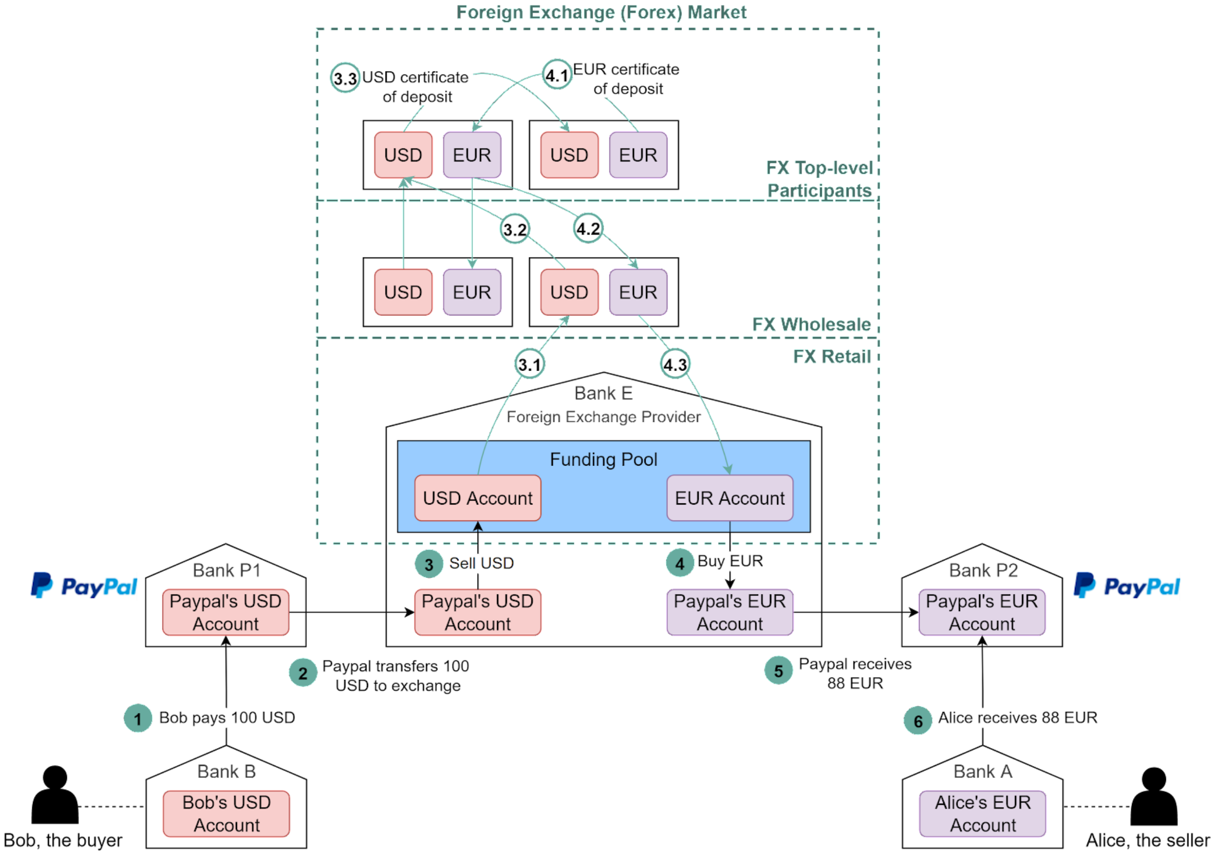
Please follow Alex if you want to receive weekly updates about tech interviews, system design, or book writing. : https://bit.ly/linkedinaxu

Money Movement



# Feb 16, 2022 - Foreign exchange

## Foreign Exchange in Payments



Have you wondered what happens under the hood when you pay with USD online and the seller from Europe receives EUR (euro)? This process is called foreign exchange.

Suppose Bob (the buyer) needs to pay 100 USD to Alice (the seller), and Alice can only receive EUR. The diagram below illustrates the process.

1. Bob sends 100 USD via a third-party payment provider. In our example, it is Paypal. The money is transferred from Bob's bank account (Bank B) to Paypal's account in Bank P1.

2. Paypal needs to convert USD to EUR. It leverages the foreign exchange provider (Bank E). Paypal sends 100 USD to its USD account in Bank E.

3. 100 USD is sold to Bank E's funding pool.

4. Bank E’s funding pool provides 88 EUR in exchange for 100 USD. The money is put into Paypal’s EUR account in Bank E.

5. Paypal’s EUR account in Bank P2 receives 88 EUR.

6. 88 EUR is paid to Alice’s EUR account in Bank A.

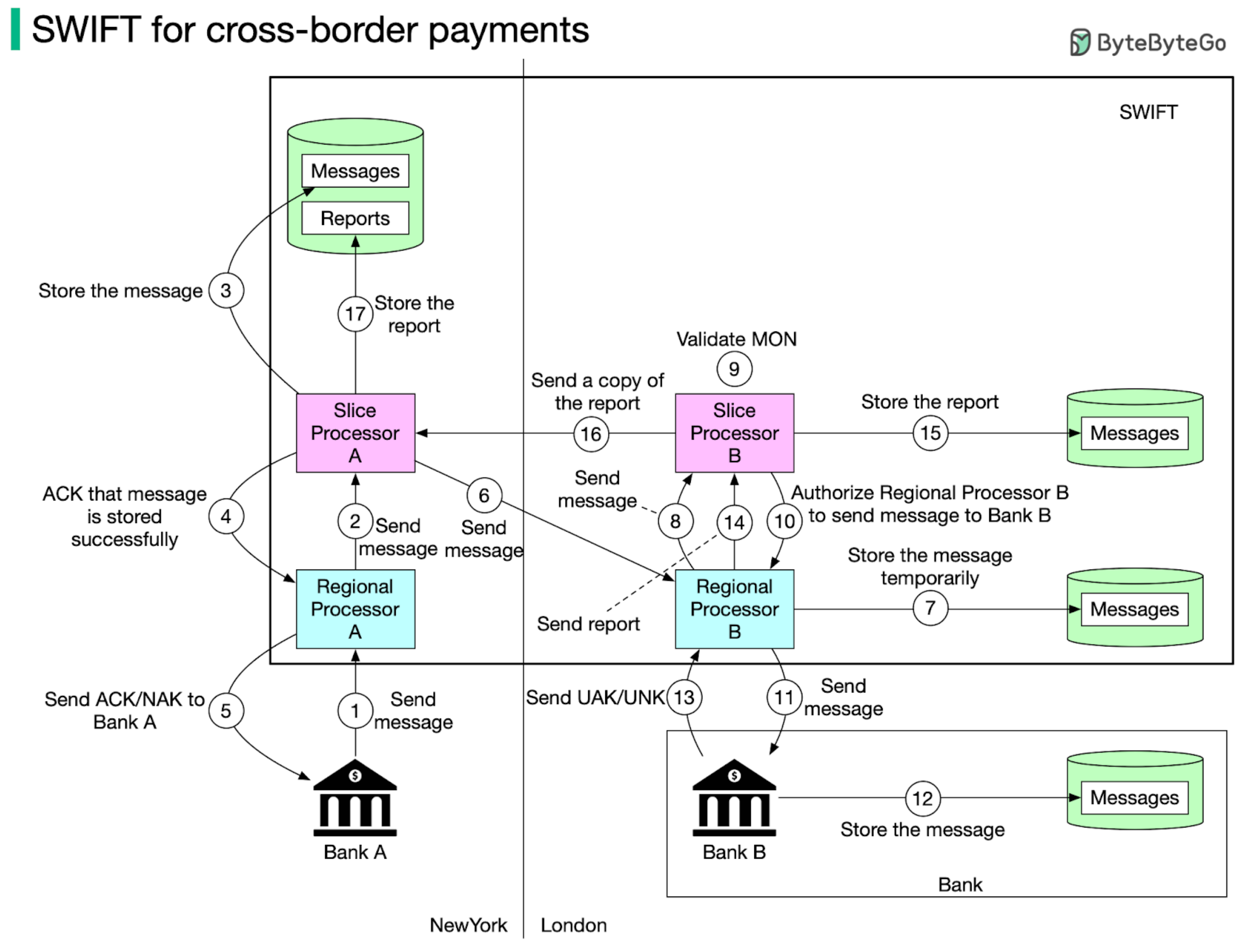
Now let’s take a close look at the foreign exchange (forex) market. It has 3 layers:

🔹 Retail market. Funding pools are parts of the retail market. To improve efficiency, Paypal usually buys a certain amount of foreign currencies in advance.
🔹 Wholesale market. The wholesale business is composed of investment banks, commercial banks, and foreign exchange providers. It usually handles accumulated orders from the retail market.
🔹 Top-level participants. They are multinational commercial banks that hold a large number of certificates of deposit from different countries. They exchange these certificates for foreign exchange trading.

When Bank E’s funding pool needs more EUR, it goes upward to the wholesale market to sell USD and buy EUR. When the wholesale market accumulates enough orders, it goes upward to top-level participants. Steps 3.1-3.3 and 4.1-4.3 explain how it works.

—
Please follow Alex if you want to receive weekly updates about tech interviews, system design, or book writing. : https://bit.ly/linkedinaxu

# Feb 25, 2022 - SWIFT



You probably heard about **SWIFT**. What is SWIFT? What role does it play in cross-border payments? You can find answers to those questions in this post.

The Society for Worldwide Interbank Financial Telecommunication (SWIFT) is the main secure **messaging system** that links the world's banks.

The Belgium-based system is run by its member banks and handles millions of payment messages per day. The diagram below illustrates how payment messages are transmitted from Bank A (in New York) to Bank B (in London).

Step 1: Bank A sends a message with transfer details to Regional Processor A in New York. The destination is Bank B.

Step 2: Regional processor validates the format and sends it to Slice Processor A. The Regional Processor is responsible for input message validation and output message queuing. The Slice Processor is responsible for storing and routing messages safely.

Step 3: Slice Processor A stores the message.

Step 4: Slice Processor A informs Regional Processor A the message is stored.

Step 5: Regional Processor A sends ACK/NAK to Bank A. ACK means a message will be sent to Bank B. NAK means the message will NOT be sent to Bank B.

Step 6: Slice Processor A sends the message to Regional Processor B in London.

Step 7: Regional Processor B stores the message temporarily.

Step 8: Regional Processor B assigns a unique ID MON (Message Output Number) to the message and sends it to Slice Processor B

Step 9: Slice Processor B validates MON.

Step 10: Slice Processor B authorizes Regional Processor B to send the message to Bank B.

Step 11: Regional Processor B sends the message to Bank B.

Step 12: Bank B receives the message and stores it.

Step 13: Bank B sends UAK/UNK to Regional Processor B. UAK (user positive acknowledgment) means Bank B received the message without error; UNK (user negative acknowledgment) means Bank B received checksum failure.

Step 14: Regional Processor B creates a report based on Bank B’s response, and sends it to Slice Processor B.

Step 15: Slice Processor B stores the report.

Step 16 - 17: Slice Processor B sends a copy of the report to Slice Processor A. Slice Processor A stores the report.

—

Please follow Alex if you want to receive weekly updates about tech interviews, system design, or book writing. : https://bit.ly/linkedinaxu