Introduction to Solana Development

Welcome to the Solana developer docs!

This guide contains step-by-step instructions on how to get started. Before the hands-on part of the guide, we'll cover basic concepts that all developers need to be familiar with to build on Solana:

- Transactions
- Accounts
- Programs

What you will learn#

- What the developer workflows look like
- What transactions, accounts, and programs are
- Test networks and other tools

An overview of Solana developer workflows#

The Solana network can be thought of as one massive global computer where anyone can store and execute code for a fee. Deployed code is called a program, often referred to as a "smart contract" on other blockchains. To interact with a program, you need to send a transaction on the blockchain from a client.

Here's a high level representation of this. It’s important to note that this is an oversimplification of the Solana network for the purposes of learning in an easy-to-understand way.

Solana developer workflows program-client model

Program development#

The first development workflow allows you to create and deploy custom Rust, C and C++ programs directly to the blockchain. Once these programs are deployed, anyone who knows how to communicate with them can use them.

You can communicate with these programs by writing dApps with any of the available client SDKs (or the CLI), all of which use the JSON RPC API under the hood.

Client development#

The second development workflow is the dApp side where you can write dApps that communicate with deployed programs. Your apps can submit transactions with instructions to these programs via a client SDK to create a wide variety of applications such as wallets, exchanges and more. The most popular apps are browser extension wallets and web apps, but you can build mobile/desktop apps or anything that can communicate with the JSON RPC API .

These two pieces work together to create a network of dApps and programs that can communicate with each other to update the state and query the blockchain.

Wallets#

A wallet is a pair of public and private keys that are used to verify actions on the blockchain. The public key is used to identify the account and the private key is used to sign transactions.

Transactions#

A transaction is the fundamental unit of activity on the Solana blockchain: it's a signed data structure that contains instructions for the network to perform a particular operation like transferring tokens.

You need a transaction to create, update or delete data on-chain. You can read data without a transaction.

All transactions interact with programs on the network - these can be system programs or user built programs. Transactions tell the program what they want to do with a bunch of instructions, and if they're valid, the program will execute them and update the state of the blockchain. Think of it like a write command that can be rejected if certain conditions aren't met.

Here's a visual representation of what a transaction contains:Visual layout of a transaction

- Signatures: An array of digital signatures from the transaction's signers.
- Message: The actual instructions that the transaction is issuing to the
- network.* Message header: 3uint8s

-
    - describing how many accounts will sign the
-
    - payload, how many won’t, and how many are read-only.
-
    - Account addresses: an array of addresses of the accounts that will be used
-
    - in the transaction.
-
    - Recent blockhash: a unique value that identifies a recent block - this
-
    - ensures the transaction is not too old and is not re-processed.
-
    - Instructions: which program to call, which accounts to use, and any
-
    - additional data needed for the program to execute the instruction.

Transactions can be created and signed using clients via SDKs, or even on-chain programs.

You can learn more about transactionshere .

Instructions#

Instructions are the most basic operational unit on Solana. A transaction can contain one or more instructions. Instructions are executed sequentially in the order they are provided in the transaction by programs on the blockchain. If any part of an instruction fails, the entire transaction will fail.

Here's what an instruction looks like:

Item Description Program ID The ID of the program being called Accounts The accounts that the instruction wants to read or modify Data Input data provided to the program as additional information or parameters in the format of a byte array You can read more about instructionshere .

Transaction Fees#

Every time you submit a transaction, somebody on the network is providing space and processing power to make it happen. To facilitate this, transactions on Solana require a fee to be paid in Lamports, which are the smallest units of SOL (like cents to a dollar or paise to a rupee). One SOL is equal to 1,000,000,000 Lamports, and one Lamport has a value of 0.000000001 SOL. This fee is paid to the validators who process the transaction.

Transactions fees are calculated based on two main parts:

- a statically set base fee per signature, and
- the computational resources used during the transaction, measured in
- "compute units
- "

The more work a transaction requires, the more compute units it will use, and the more it will cost.

You can read more about transaction feeshere .

Accounts#

Accounts on Solana are storage spaces that can hold arbitrary data up to 10MB. They're used to store data, user programs, and native system programs.

If a program needs to store state between transactions, it does so using accounts. This means that all programs on Solana are stateless - they don't store any state data, only code. If an account stores program code, it's marked "executable" and can process instructions.

The easiest way to think of an account is like a file. Users can have many different files. Developers can write programs that can "talk" to these files. In the same way that a Linux user uses a path to look up a file, a Solana client uses an address to look up an account. The address is a 256-bit public key. Also like a file, an account includes metadata that tells the runtime who is allowed to access the data and how. This prevents unauthorized changes to the data in the account.

Unlike a file, the account includes metadata for the lifetime of the file. Solana accounts have a unique lifecycle. When an account is created, it needs to be assigned some space, and tokens are required to rent this space. If an account doesn't have enough tokens to cover the rent, it will be removed. However, if the account does hold enough tokens to cover the rent for two years, it's considered "rent-exempt" and won't be deleted.

You can read more about accountshere .

Programs#

Programs are the foundation of the Solana blockchain. They're responsible for everything that happens on the network: creating accounts, processing transactions, collecting fees, and more.

Programs process instructions from both end users and other programs. All programs are stateless: any data they interact with is stored in separate accounts that are passed in via instructions.

There are two sets of programs that are maintained by the Solana Labs team: Native Programs and the Solana Program Library (SPL) . These serve as core building blocks for on-chain interactions. Native programs are used for core blockchain functionality like creating new accounts, assigning ownership, transferring SOL, and more. SPL programs are used for creating, swapping, and lending tokens, as well as generating stake pools and maintaining an on-chain name service.

You can interact with both native programs and SPL programs easily using the Solana CLI and the SDKs, allowing you to create complete dApps without writing Rust. You can also build on top of any user programs that have been deployed to the network - all you need is the program's address and how it works: the account structures, instructions, and error codes.

Developers most commonly write programs in Rust using frameworks such as Anchor. However, programs can be written in any language that compiles to BPF, including C++ and Move.

You can learn more about programshere .

Testing and developing environments#

When developing on Solana you have a few options for environments.

The easiest and quickest way to get started is the Solana Playground - a browser based IDE that allows you to write, deploy, and test programs.

The most popular setup is local development with a local validator that you run on your machine - this allows you to test your programs locally before deploying them to any network.

In each environment, you'll be using one of three networks:

- Mainnet Beta - the "production" network where all the action happens.
- Transactions cost real money here.
- Testnet - used for stress testing recent releases. Focused on network
- performance, stability, and validator behavior.
- Devnet - the primary network for development. Most closely resembles Mainnet
- Beta, but tokens are not real.

Devnet has a faucet that allows you to get free SOL to test with. It costs 0 to do development on Solana.

Check out the clusters page for more information on these.

Next steps#

You're now ready to get started building on Solana!

- Deploy your first Solana program in the browser
- Setup your local development environment
- Get started building programs locally with Rust
- Overview of writing Solana programs