



eUTxO Fundamentals: Building Cardano Smart Contracts

Starting from zero

2024

Raul Rosa
aka ElRaulito

I INTRODUCTION TO CARDANO SMART CONTRACTS			3
	1	OVERVIEW OF CARDANO BLOCKCHAIN	4
	1.1	Importance and Applications of Smart Contracts	5
	1.2	Advantages of Cardano for Smart Contract Development	6
II SETTING UP THE DEVELOPMENT ENVIRONMENT			7
DEVELOPMENT TOOLS	2	INSTALLING AND CONFIGURING CARDANO DEVELOPMENT TOOLS	8
	2.1	Installing and Configuring Cardano Development Tools	8
	2.2	Setting Up and Connecting to Cardano Testnet	8
	2.3	Interacting with Cardano Node and Wallet APIs	10

1. OVERVIEW OF CARDANO BLOCKCHAIN

1.0.1. History

Charles Hoskinson, along with Jeremy Wood, co-founded Cardano, both were part of Ethereum before. In 2015, they established Input Output Hong Kong with the aim of creating and developing more sustainable blockchain solutions. Utilizing a peer-reviewed approach to blockchain development and introducing a novel consensus mechanism called **Ouroboros**, Cardano was prepared for its Mainnet launch in 2017.

1.0.2. Ouroboros

Ouroboros relies on a **proof of stake** consensus. Rather than requiring nodes to engage in computationally intensive work as in PoW chains, nodes are randomly selected based on the amount of ADA they hold at stake. This approach serves two purposes: it is more energy-efficient and incentivizes nodes to act responsibly as their stake is at risk in case of misbehavior.

1.0.3. Cardano Architecture

The blockchain model comprises several components. Users interact with the current ledger state by creating transactions, which are then submitted to the mempool until they are included in a block. Blocks are mined by stake pools, which are rewarded for their efforts and share these rewards with their delegators. Increased decentralization is achieved with more stake pool operators.

1.0.4. Improvements from Other Blockchains

Cardano offers several advantages over other chains:

- **Determinism:** This feature enables transaction chaining.
- **Predictable Fees:** There is no risk of pending transactions due to fee increases.
- **Sustainability:** Unlike PoW, which is power-intensive, Cardano's approach is more sustainable.
- **Native Tokens:** Tokens are stored on the ledger, allowing smart contracts to interact with them. Users have control over tokens in their wallets, which cannot be frozen by external parties.

Scaling is currently the most significant challenge for the ecosystem. The ability to handle high volumes of users without being limited by block or transaction size is crucial for increasing adoption.

1.1.1. IMPORTANCE AND APPLICATIONS OF SMART CONTRACTS

Smart contracts are a concept born alongside Ethereum, enabling the execution of code and interactions without a third party. Once initiated, the terms of the contract are set by the parties involved, and no one can stop or interfere thereafter.

However, history teaches us that some protocols have included backdoors within their smart contracts, leading to fund theft or enabling bad actors to access users' funds.

Let's start with the basics.

1.1.1.1. What is a Smart Contract?

A smart contract is decentralized software accessible to users on the blockchain, typically through a website interface. Users interacting with the contract can perform operations (financial, trading, storage) without requiring permission from a third party.

The essential components of a smart contract are:

- **Parties:** Who can interact with the contract? Is it open to everyone, specific users, or owners of particular assets?
- **Actions:** What operations can users perform with the contract? These could include depositing funds, creating NFTs, storing data, reading data, withdrawing funds, and more.
- **Rules:** Define the actions each party can take under specific conditions.
- **Data Fields:** What data is involved in interactions with the contract, and how can each step of the interaction be tracked?

1.1.1.2. Applications

In a typical decentralized exchange (DEX) application, the parties are liquidity providers and traders. Liquidity providers can deposit and withdraw liquidity, while traders can only perform swaps. Rules dictate that liquidity providers must hold LP tokens in their wallets, while traders must have sufficient funds to cover transactions. Data fields stored in the contract typically include LP tokens, fees for liquidity providers, and token data.

In a marketplace scenario, the parties are sellers and buyers. Sellers can sell assets, while buyers can buy them. Rules stipulate that sellers must possess the assets they intend to sell, and buyers must have sufficient funds to purchase assets and pay sellers. Data stored includes the seller's address, payment amounts, royalties (if applicable), and platform fees.

On Cardano, specific actions might include Cancel Listing and Buy. Selling/List is more of a smart contract interaction than an action.

A smart contract action involves a transaction where the smart contract is invoked in the inputs. If the smart contract is present only in the outputs, it's considered a smart contract interaction.

1.2. ADVANTAGES OF CARDANO FOR SMART CONTRACT DEVELOPMENT

Two years ago, if you asked me about the advantages of writing smart contracts on Cardano, I would have struggled to answer. However, now I can easily list several:

- **Composability:** The ability to create a transaction involving multiple contracts and perform actions with each of them.
- **User-Friendly:** No longer requiring Haskell, languages like Aiken, Opshin, and more offer a user-friendly experience.
- **Liquid Staking:** Thanks to Cardano staking, smart contracts can delegate ADA or keep funds staked with liquidity providers.
- **UTxO Skills:** While much of the focus has been on Ethereum Virtual Machine (EVM) smart contracts, the UTXO model is ideal for solutions like ZK rollups, as it's easier to implement compared to the account model.

If you're still interested in becoming a Cardano smart contract wizard after this introduction, we can continue in the next chapter, where we'll install the components needed to **build on Cardano**.

2. INSTALLING AND CONFIGURING CARDANO DEVELOPMENT TOOLS

2.1. INSTALLING AND CONFIGURING CARDANO DEVELOPMENT TOOLS

The purpose of this book is to gather all the information for developing on Cardano that currently is scattered around. What are we going to use for our project?

- **A hot Wallet:** We are going to use a wallet to test our contracts, this wallet will be used in to receive tADA. We'll never store our main ADA holdings in this wallet: Wallets recommended for testnet are Nami on desktop and Vesper for mobile.
- **A indexer account:** Indexers are the ones that will provide us the APIs in order to interact with the chain, we won't need to run a node for testing, let's use services and projects already there like **Maestro**, let's setup an account and get the API key.
- **Lucid library:** Lucid is not maintained anymore as function and has being replaced by COMING SOON, however for testing and understanding the flow of Cardano transactions it can be really useful.
- **tADA:** How are we going to test without having testnet ADA? let's not mess up real ADA
- **IDE:** Personally I use Visual Studio Code as IDE, but any other editor is ok since we are going to
- **Cardano Node:** This is NOT mandatory at all, however as homework we could try to setup a cardano node and interact with the chain using cardano-cli (command line), however this is something we can do in our free time, there are other hobbies out there better than this, swimming, dancing or reading a book.

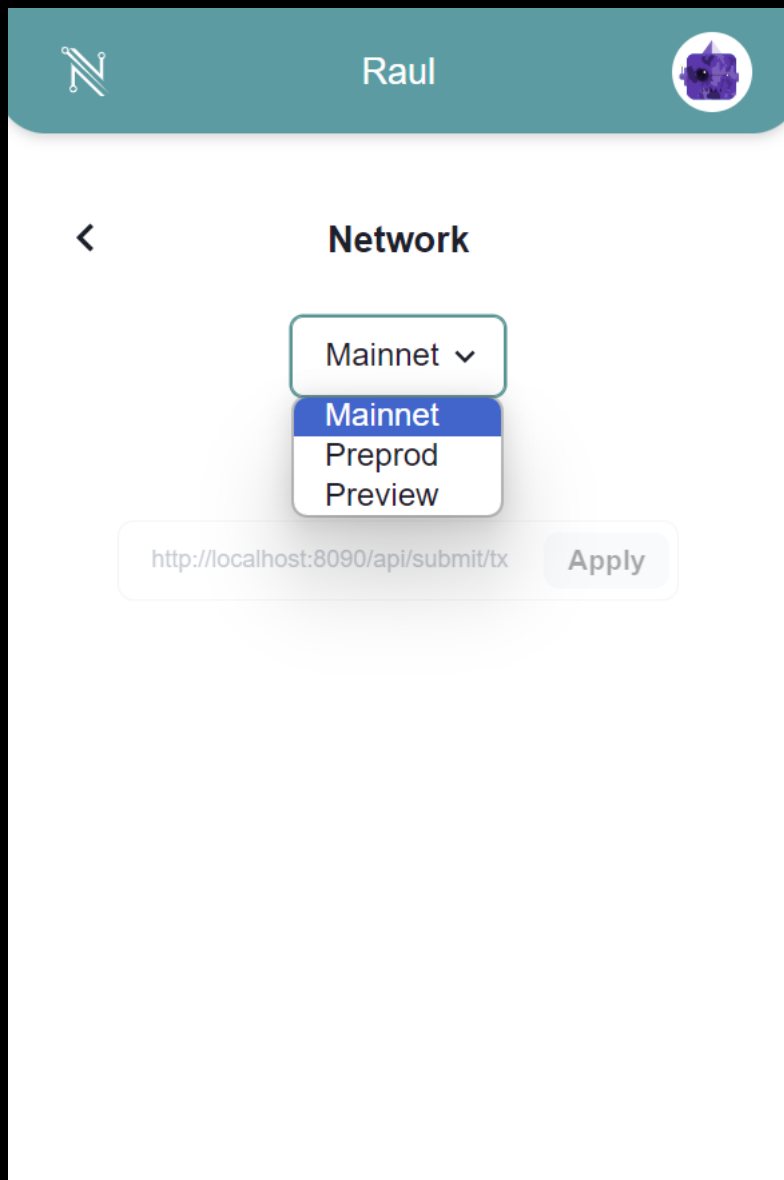
2.2. SETTING UP AND CONNECTING TO CARDANO TESTNET

So let's install a wallet and config for testnet

In this example we'll install Nami wallet that we can find [here](#)

Once we install the wallet we'll get 24 seed phrase words

Never share the seedphrase or store it on a cloud, use a paper or different ways to store, software can keep track of your seedphrase and you could lose the funds.



Let's set Nami to **testnet preview** and we'll finally get our wallet in testnet









In order to receive tADA we can use the official faucet from Cardano at the following **link**

2.2.1. CIP30

In order to connect our wallet with any webpage we'll use CIP30 reference, we can find the list of methods to connect and invoke the functions of the wallets at this <https://www.cardano-caniuse.io/>page

walletName









This is the field the wallet will inject itself as into the `window.cardano` object

 Flint	 Nami	 Eternl	 Gero Gero	 NuFi	 Begin	 Lace	 Yoroi
flint	nami	eternl	gerowallet	nufi	begin	lace	yoroi

cardano.{walletName}.enable(): Promise<API>

Errors: `APIError`

This is the endpoint to start communication with the user's wallet. The wallet should request the user's permission to connect the web page to the user's wallet, and if permission has been granted, the full API will be returned to the dApp to use. The wallet can choose to maintain a whitelist to not necessarily ask the user's permission every time access is requested but this behavior is up to the wallet and should be transparent to web pages using this API. If a wallet is already connected this function should not request access a second time but instead just return the `API` object.

 Flint	 Nami	 Eternl	 Gero Gero	 NuFi	 Begin	 Lace	 Yoroi
SUPPORTED	SUPPORTED	SUPPORTED	SUPPORTED	SUPPORTED	SUPPORTED	SUPPORTED	SUPPORTED

The steps to interact with a wallet following cip30 are:

- **cardano.walletName.enable()**: we get an API object as promise, this will create a popup message to allow the wallet to connect to the current website
- **api.getBalance()**: using the api object we got before, we get the total amount of lovelace in the wallet (1 ADA = 1000,000 lovelace)
- **api.signTx**: Signing a tx that was build with Lucid or any other library we sign and interact with the blockchain

EXCERCISE 1: Create a webpage with 2 buttons, 1 to enable the wallet connection, second button to view the amount of ADA in the wallet.

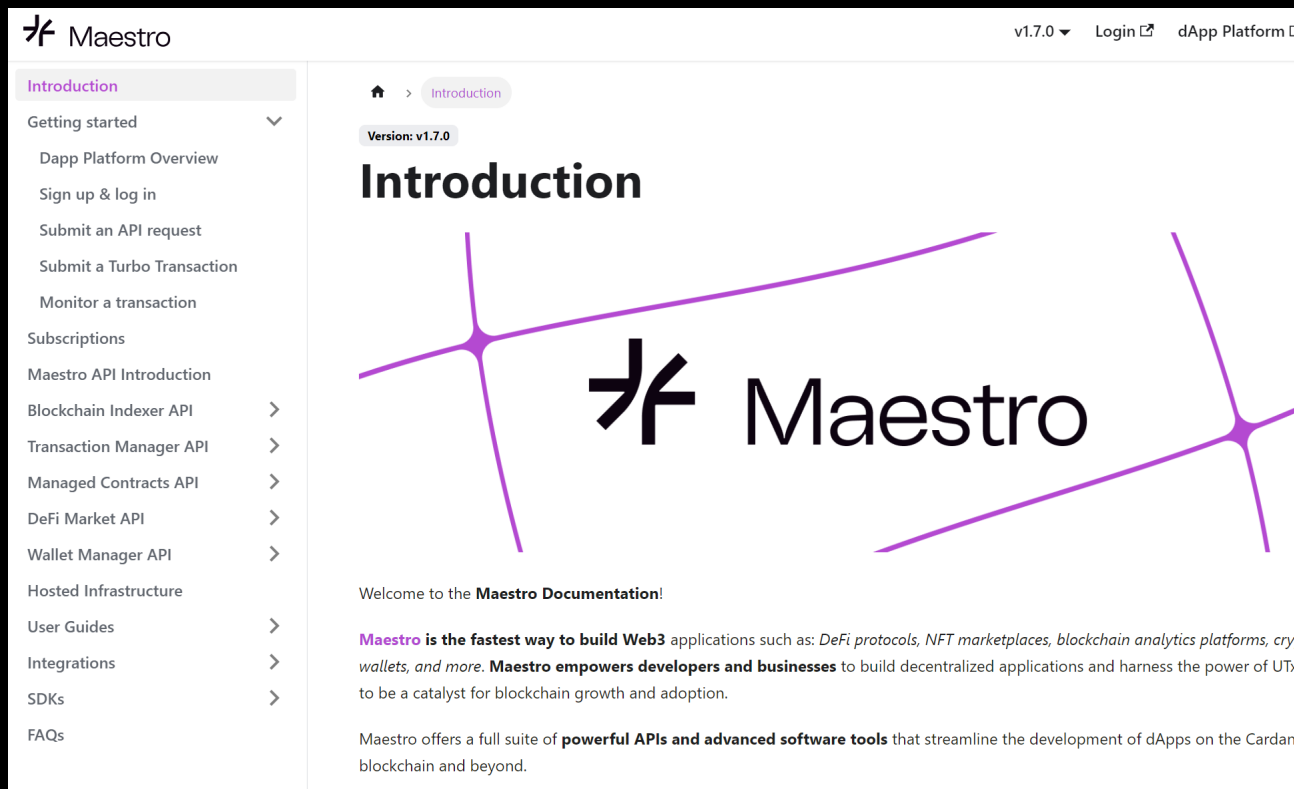
2.3. INTERACTING WITH CARDANO NODE AND WALLET APIS

CIP30 is not enough, what if we want to get the informations regarding a specific NFT in our wallet? How to get the list of tokens inside the wallet and get information regarding their circulation supply?

We need an indexer. We could setup one on our own or use a service, in this book we'll use **Maestro** as service provider so the first thing to do is:

2.3.1. Create a Maestro account

Head over **Maestro login** page and create an account, here we'll be able to get the API keys to interact with Cardano.



Maestro is going to be our key to get all the possible APIs in order to interact with Cardano, here the possible things we can do with these APIs:

- Get the history of an address with this **API**
- Get all the assets of a specific policy
- Get the address holding a specific ada handle
- Get the history of holders for a specific NFT
- and much more

Now that we have a way to interact with a wallet and APIs to query the Cardano blockchain we are ready to put our hands on the real coding part.

Let's code smart contracts.

EXCERCISE 2: Head over <http://cnftlab.party/> and connect your testnet wallet, mint a collection of NFTs and then use Maestro to get the information of each NFT of your collection.