**WASM Smart Contract**

Created by Hgminerva

**Smart Contracts in Polkadot**

One of the first questions we typically get when somebody learns about Substrate, Polkadot, or Kusama is when to develop a parachain vs. when to develop a smart contract.

The distinction here is that in the context of Polkadot and Kusama a parachain leases a slot for a couple of months for up to two years. The deal with a lease is that the parachain gets a fixed slot for executing its business logic (typically referred to as its state transition function) and can persist its modified state in a block. In Substrate terminology this state transition function is called the chain's runtime.

The distinction to other ecosystems here is that, in the context of Polkadot, parachains and smart contracts exist at different layers of the stack: smart contracts sit on top of parachains. Parachains would usually be described as layer-1 blockchains ‒ except for that they don't have to build their own security, are upgradable, and interoperable.

It's noteworthy that a parachain's state transition function doesn't get further validated ‒ it's up to the parachain how it utilizes its slot time. The parachain was already pre-paid for its slot when it won the slot auction on Polkadot or Kusama. This means the parachain can build its own (blockchain) world! For example, it can decide on how transaction fees are charged ‒ or even if transaction fees are charged at all. These options are crucial when building new or more user-friendly business models. Other distinguishing factors between parachains that we observe in the wild are differences in how governance works or the crypto-economics. There are some constraints on how the parachain can build its world though. Like physics in the real world it has to adhere to certain ground rules. For Polkadot and Kusama that's for example the consensus algorithm for the Relay Chain to communicate with the parachain. From those ground rules the advantages of Polkadot and Kusama emerge. Advantages like the aforementioned shared security, cross-chain communication, or guaranteed execution slot time.

For smart contracts, on the other hand, an existing parachain has to include the pallet-contracts for users to deploy smart contracts. The deployed smart contract is always untrusted code. Anyone (or any program) that has tokens of the chain can upload a smart contract without requiring permission. Smart contracts allow permissionless deployment of untrusted programs on a blockchain. The pallet-contracts have to assume that these programs are adversarial, it has to put a number of safety pillars in place to ensure that the contract can not e.g. stall the chain or cause state corruption of other contracts. For pallet-contracts those safety pillars include mechanisms like gas metering or deposits for storing data on-chain.

To restate this important distinction: developing a parachain runtime is different from developing a smart contract ‒ a smart contract sits on top of a parachain.

The trade-off is that with a parachain one has the freedom to decide on (nearly) all the rules that make up the parachain. With a smart contract one is constrained by what the chain allows and the safety pillars that necessarily have to be in place. A smart contract can never be as fast as a native pallet built in the parachain runtime ‒ there is too much logic in between. A smart contract on the other hand has less friction for developing and deploying it.

Developers don't have to take care of governance, crypto-economics, etc. One just needs a few tokens and can go on its merry way deploying a smart contract. It's as simple as that.

**Disclaimer on Randomness in pallet-contracts**

Codes using the randomness functionality cannot be uploaded, and contracts cannot be instantiated from existing codes that use this deprecated functionality. It will eventually be removed. Hence, for new pallet-contracts deployments, it is acceptable to supply a dummy implementation for this type (because it is never used).

Source: <https://github.com/paritytech/polkadot-sdk/blob/f60318f68687e601c47de5ad5ca88e2c3f8139a7/substrate/frame/contracts/src/lib.rs#L237>