Algorand Development

Intro to building on Algorand

September 16, 2022 Joe Polny

Algorand™ FOUNDATION

Tools

- Sandbox
  - Leverages docker for quick node spinup
- PyTeal
  - Python language for generating TEAL
- Beaker
  - Python framework used with PyTeal for writing contract
- SDKs
  - JavaScript, Python, Go, Java
- IPFS
  - Decentralized storage network

Resources

- Developer Portal
  - Documentation, articles, and challenges
- Algorand Forums
- Algorand Discord
- Algorand YouTube channels
  - Algorand, AlgorandFoundation
- @AlgoDevs Twitter

Encoding

- Smart contracts work with raw bytes
  - All smart contract arguments are bytes
- Multiple ways to encode bytes
- SDKs/languages provide encoding methods
  - JavaScript has
    - `Buffer.from(data, encoding)`
    - `algosdk.encodeUint64(number)`

Encoding Examples

- utf-8: `byte "Hello World"`
- hexadecimal: `byte 0x48656C6C6F20576F726C64`
- base64: `byte b64 SGVsbG8gV29ybGQ=`
- base32: `byte b32 JBSWY3DPEBLW64TMMQ======`

Accounts

- Algorand accounts are Ed25519 key pairs
- Address is derived from public key
    - a. Checksum of first 4 bytes is added to the end
    - b. Encode with base32
- 25-word Mnemonic is derived from private key
    - Uses BIP-0039 wordlist in a non-standard algorithim
- Smart contracts use raw public key

HTTP Endpoints

- Indexer provides HTTP endpoints for entire blockchain history
  - Applications
  - Assets
  - Balances
  - Transactions
- Algod provides HTTP endpoints for current blockchain state
  - Pending transactions
  - Sending transactions
  - Suggested transaction parameters
- KMD provides HTTP endpoints for managing accounts

Connection Options

- Public Providers (algod/indexer)
  - Algo Explorer
  - AlgoNode
  - PureStake
- Connection Information
  - URL
  - Port
  - Access token