Names and Addresses

Almost all blockchains use addresses to identify external actors via a hash of a public key, and many newer ones extended this to identify on-chain "smart contracts" with unique addresses as well.

On-chain addresses are represented by the use of a concise, immutable binary format, typically 20 or 32 bytes long, often derived from a hashing function. However, there are many human-readable representations of these binary addresses, which are shown to clients.

For example,

- Bech32
- bc1qc7slrfxkknqcq2jevvvkdgvrt8080852dfjewde450xdlk4ugp7szw5tk9
- , hex0x8617E340B3D01FA5F11F306F4090FD50E238070D
- ,
- checksummed hex
- 0x5aAeb6053F3E94C9b9A09f33669435E7Ef1BeAed
- ,
- and evenlarge integers
- 3040783849904107057L
- .

Addr

Addresses in Cosmos SDK are 20-character long strings and contain security checks - such as chain-prefix in Bech32, checksums in Bech32, and checksummed hex (EIP55). Since CosmWasm is an extension of Cosmos SDK, it follows the same address rules; wallets, smart contracts, and modules have an identifier address with a defined prefix.cosmos1... for gaia,wasm1... for CosmWasm testnets.

For passing address to contracts, pass it as a string and then validate the input to an:AddrAddr is just a wrapper around a plain string that provides useful helper functions such as string validation to an address.

There is another representation of an address, calledCanonical Addresses , that tackles the case of change in human representation, but this is rare.

For example Bitcoinmoved from Base58 to Bech32 encoding along with SegWit, and Rise is alsomoving from Lisk format to Bech32 in the v2 upgrade.

This means that ifmessage.signer is always the string address that signed the transaction and I use it to look up your account balance, if this encoding ever changed, then you lose access to your account. We need a stable identifier to work with internally.

This is where we define aCanonical Address . This is defined to be stable and unique. That is, for one given account, there is only ever one canonical address (for the life of the blockchain). We define acanonical address format that potentially multiple string addresses can be converted to. It can be transformed back and forth without any changes

We define theCanonical Address as the binary representation used internally in the blockchain. This is what the native tokens are indexed by and therefore must never change for the life of an account. This is the representation that can be used for allstorage lookups (if you use part of an address as the key in the storage).

Naming

More and more,human readable names are increasingly important in blockchainsand beyond .

At one point, we considered making names a first-class citizen of CosmWasm and using them everywhere in messages. Until we realized that accounts can have no name until initialized, and we need a permanently stableAddress . However, we would still like names to be as central to the blockchain as possible. To this end, we can consider names as just another form ofAddress albeit one that requires a contract query (with storage access) to resolve, not just a call to a pure function.

This actual format and interfaces are still under discussion, and we are still working on atutorial version of a name service . However, for sake of argument, imagine we agree everyAddress that begins with: is a name to lookup with the name service, and other ones can be resolved directly with the built-in blockchain resolution function. So when creating a transaction for an escrow, you could use either{"arbiter": "cosmos1qqp837a4kvtgplm6uqhdge0zzu6efqgujllfst"} or{"arbiter": ":alice"} , performing the resolution inside the contract rather than only in the client. Of course, we would need a standard query format for name service, and the calling contract would need to somehow know the address of the canonical name service contract to resolve, which is why this feature is still under discussion.

DIDs

Note: it will likely be quite some time before this is fully implemented. It serves as design inspiration

Let's keep imagining what is possible withHuman Names , once we develop a solution to the name service issue. We could not just use a reference to resolve a user address, but resolve a contract as well. Maybe we could dispatch a message to an "ERC20" token contract not by its name, but by itsuniquely registered token ticker . We would soon need to use some way to distinguish the scope or context of a name. This is whereDecentralized Identifiers (DIDs) could come in. Imagine the following message format, that could be used either by an end client or by a smart contract "actor":

{ "destination" :

"did:token:XRN" , "msg" :

{ "transfer" :

{ "from" :

"did:account:alice" , "to" :

"did:account:bob" , "amount" :

"13.56" } } } This would not be some spec to be resolved client-side, but the actual interchange format used on the blockchain. So one smart contract could also dispatch such a message in the course of execution. Previous Actor Model Next Querying * Addr * Naming * * DIDs