Overview

The Inter-Blockchain Communication (IBC) protocol is a blockchain interoperability solution that allows blockchains to transfer any type of data encoded in bytes, in a secure and permissionless manner.

The most popular IBC use case arguably is token swaps between different blockchains.

What differentiates IBC from other bridging protocols?

Universal Interoperability . Chains that speak IBC can share any type of data as long as it's encoded in bytes, enabling the industry's most feature-rich cross-chain interactions.

Permissionless access . IBC is completely open-source: Anyone can build with IBC, and there's no in-protocol rent extraction or hidden fees.

Security . IBC's light client-based interoperability removes the need for a trusted third party in cross-chain interactions, securing tens of billions in annual value transfer without a single exploit since launch.

Design Principles

The design principles of IBC drew inspiration from the TCP/IP specification that enabled the creation of the internet. Mirroring the way TCP/IP sets the standard for seamless communication between computers, IBC defines a universal framework of abstractions that lets blockchains communicate.

The IBC protocol stack can be decoupled into two distinct elements: theIBC Transport Layer(opens in a new tab) and theIBC Application Layer(opens in a new tab) .

TheIBC Transport Layer is agnostic to the contents of transferred data packets, much like TCP/IP. The IBC Transport Layer is a foundational layer upon which feature-rich applications are developed, similar to the applications that sit on top of TCP/IP and allow for the flourishing of end-user Internet applications.

The transport layer handles data packet transport, authentication, and ordering. The key components of the transport layer are light clients, connections, channels, and relayers.

Light clients are the heart of the IBC Transport Layer. Alight client(opens in a new tab) is a lightweight representation of destination chain that live within the state machine of the source chain.

For example, chains A and B are connected over IBC via their light clients. Chain A will have a light client representing chain B in its own state machine, and Chain B has a light client of Chain A. Light clients keep track of a counterparty blockchain's consensus algorithm by verifying block headers and Merkle proofs.

Connections are responsible for connecting two different light clients together.

Channels act as a conduit to connect a module/application on the source chain to a module on the destination chain. Data packets between the source and destination chains are sent over this abstraction layer.

Relayers are permissionless off-chain processes that ferry data packets from one chain to anotherRelayer(opens in a new tab) s scan chain states, build transactions based on these states, and submit the transactions to the chains involved in the network. Relayers play a crucial role in IBC because chains do not directly send messages to each other over networking infrastructure. Instead, they create and store the data to be retrieved and used by a relayer to build IBC packets.

Get an in-depth walkthrough of the IBC components and packet flow on thedeveloper documentation.(opens in a new tab)

Asset transfer

Command line

In order to transfer funds from Sei to other chain or other way round we need several bits of information. In particular:

- from wallet e.g. Sei bech32 address
- to wallet e.g. Axelar bech32 address
- the source channel and port id
-
    - These could be located inregistry(opens in a new tab)
-
    - . E.g. foratlantic-2

-
    - Axelar, we would use:
- {
- "counterparty_chain_name"
- :
- "axelar-testnet-lisbon-3"
- ,
- "dst_channel"
- :
- "channel-257"
- ,
- "src_channel"
- :
- "channel-44"
- ,
- "port_id"
- :
- "transfer"
- ,
- "client_id"
- :
- "07-tendermint-80"
- },

With that information on hand we could run a cmd command to perform transfer like:

seid

tx

ibc-transfer

transfer

transfer

channel-44 AXELAR_ADDRESS 200 sei

--from SEI_ADDRESS --fees

4000 usei

--node

https://rpc-testnet.sei-apis.com

--chain-id

atlantic-2

-b

block This would transfer 200 sei from sei address to axelar address.

JS Client

import {ethers} from

"ethers" ; import {IBC_PRECOMPILE_ADDRESS , IBC_PRECOMPILE_ABI} from

"@sei-js/evm" ;

const

transferWithDefaultTimeout

=

async () => { const

signer

```
=
await
getEthSigner ();
if ( ! signer) { toast .error ( 'No signer found' ); return ; } const
contract
=
new
ethers .Contract ( IBC_PRECOMPILE_ADDRESS ,
IBC_PRECOMPILE_ABI , signer); const
axelarAddress
=
AXELAR_ADDRESS
try { console .log ( "Transfer" ) const
result
=
await
contract .transferWithDefaultTimeout (axelarAddress ,
"transfer" ,
"channel-44" ,
"usei" ,
"76000" ,
"memo text" ); console .log ( "TransferResult: " , result); } catch (e) { console .log (e) }
```

}; Last updated onMay 23, 2024 Multi-Sig Accounts IBC Relayer