Speculo-Bridge (Token Transfer Only)

Bridge Specification

A bridge for cross-chain token transfers enables the value of bAssets to be utilized regardless of the underlying blockchain. This dramatically increases the usefulness of bAssets, as assets are no longer siloed on its own blockchain and can be freely transferred to systems running on other blockchains - such as Anchor. Using the bridge, value locked up in staking within non-Terra blockchains can be held up as collateral on the Anchor system.

While there are multiple approaches to linking blockchains together - including Tendermint's Inter-Blockchain Communication (IBC) - Anchor should be able to support arbitrary assets that do not support generalized bidirectional transaction verification. Such assets include blockchains that do not support Smart Contract functionality, and/or has a block finality mechanism that is difficult to deterministically verify on another blockchain. As such, Speculo's Bridge specification defines the bare minimum requirements for integrating non-Terra assets onto the Anchor system, without having to implement consensus-level message transmission.

Required Properties

Bi-directional token transfers

A token issued on the target chain should be freely transferrable to the Anchor system in the form of wrapped tokens (wTokens), and wTokens should always be redeemable on the target chain.

Signature-level Security

The bridge should provide a mechanism to easily authenticate and verify a signed transaction originating from a remote blockchain. More specifically, the bridge should be able to verify an interchain transfer signed by an account on a remote blockchain, checking whether this transaction was not spoofed in any way during the transfer process, regardless of the remote chain's signature and address format.

Arbitrary Message Transmission
The wToken standard used with bi-directional token transfers should be able to relay relevant messages defined with the bAsset spec through the Bridge. wTokens should also be treated as the same asset with its base vanilla form, regardless of which blockchain it is issued on.

In order to support platform-specific features, such as remote staking control, reward claims from another chain, and others, the bridge should support *arbitrary message transmission* - able to handle any platform-specific messages over a given set of different interchain bridges and blockchains. Note that absolute security guarantees (i.e. consensus-level block proofs) is **not a requirement** for Anchor & Speculo in particular - although taking advantage of such approaches (e.g. IBC between Cosmos-SDK chains, Substrate - Cosmos Bridge by Chorus One) is highly recommended. Message size should be fixed prior to runtime: **it is not a requirement to support dynamic memory allocation.**

Arbitrary Message Transmission may be supported on a two-way token transfer bridge without consensus-level security and/or native message transmission features, as per the Interchain Message Transfer spec listed below.

Reward Claims
A mechanism should exist to bring staking rewards from issued bAssets onto the Anchor system, and vice versa. This should be done in coordination with Proof of Ownership and Arbitrary Message Transmission implementations of the Bridge.

In order to claim rewards from another chain and bring it onto the Anchor system, the bridge should be able to provide proof of ownership, showing that the owner of bAssets on the origin chain and the owner of wbAssets (bAssets wrapped in wTokens) are the same. The bridge should also be able to provide a simple verification scheme for the ownership proof provided.

Interchain Message Transfers

One of Anchor's core requirements is the ability to process **interchain assets**: asset types natively hosted on a remote blockchain. More specifically, an interchain asset being processed on Anchor should satisfy the following two properties:

> **Two-way Token Transfers**: an interchain asset exists on the Terra blockchain in the form of **wTokens** - a token that represents an asset originally issued on a remote blockchain. wTokens should always be one-to-one redeemable with its underlying base asset, and an original interchain asset should always be transferrable to the Anchor system in wToken form.
>
> **Remote Reward Claims**: bAssets being issued on a remote blockchain get wrapped in wTokens when it is transferred to the Anchor system, forming "wbAssets". Because all rewards are accumulated on the blockchain that the bAsset was originally issued, rewards being generated on the remote blockchain should also be wrapped in wTokens and enter the Anchor system. To collect rewards being accumulated on a remote blockchain, the bridge should also be able to periodically keep track of staking rewards on different chains.

Technical Implementation

Abstract

> We will mint a new, valueless token called Anchor Messenger (AM).
>
> The number of AMs being transmitted over a token transfer bridge refers to a particular command or a message.
>
> All message transmissions should be followed with an `ACK`. An `ACK` should be appended with 10 from its original coded message.
>
> Numbers should be appended with a `NUMDEC` command, and the number of tokens being transmitted is the number value being sent over the channel.
>
> This system should have its own command code (opcodes) pre-declared.
>
> Such a system should work with any arbitrary bi-directional token bridges.
>
> To end a communication channel, restore AM balances on both sides to its original state.

Opcode Definitions

1. Channel Control

   `OPEN`: Initiate a new transfer channel (control ordering) - *optional*

   `CONEND`: Finish a transfer channel (control ordering) - *optional*

1. Appended Opcodes

   `ACK`: Acknowledges a previous message, showing that the intended recepient has received and processed the message

   > **Accepts**: `OPCODE`, `SIGNATURE_PROOF`
   >
   > **Returns**: `RESULT`, `NONCE`

`END`: Marks the end of a declarative opcode

1. Declarative Opcodes

   `NUMDEC`: Declares the following data should be interpreted as an integer value, and not an opcode, until an `END`+`NUMDEC` command is issued

   `STRINGDEC`: Declares the following data should be interpreted as a `string`, and not an opcode, until an `END`+`STRINGDEC` command is issued

   `TOKENDEC`: Declares the following data is a token of value, and not an opcode, until an `END`+`TOKENDEC` command is issued - *optional* only issued when control ordering is enabled

1. bAsset Opcodes

   `CLAIMREWARDS`: Executes a `ClaimTx` command on a remote chain, and issues a `TOKENDEC`+(wToken Transfer)+`END`+`TOKENDEC` to transfer all rewards back to the Anchor system

   `DELEGATE`: Delegates wTokens on Anchor to the original chain, issue new bAssets, and send them back to the Anchor system - *optional*

      Accepts: wTokens, `denom`

      Returns: bAssets, `delegationresult`

   `UNDELEGATE`: Sends bAssets on Anchor to the original chain, redeem vanilla Assets, and send them back as wTokens to the Anchor system - *optional*

      Accepts: bAssets, `denom`

      Returns: wTokens, `undelegationresult`

Decimal: 18 → 60bits
Precision: 128bits