Bridging L1 and L2 (Cross Domain Messenger)

Titan offers Cross Domain Messenger functionality through the Titan SDK, enabling seamless asset transfers between L1 and L2. Transferring assets from L1 to L2 is referred to as a deposit, while moving assets from L2 back to L1 is known as withdraw.
This article will provide a Javascript code example demonstrating how to deposit or withdraw ETH using the Titan SDK. Titan SDK makes it easy for service developers to implement bridging functions to move assets between L1 and L2 without calling contracts directly.
You can find executable example code here.

Setup

- Titan SDK can be imported to use the supported methods/types in the library. The `ethers` library is imported to use the functionality required to interact with the Ethereum blockchain.

```
// Transfers between L1 and L2 using the Titan SDK

const ethers = require("ethers")
const titanSDK = require("@tokamak-network/titan-sdk")
```

- Using the `ethers` library, you can create a provider and wallet instance to generate signatures and send transactions on the L1 and L2 networks.
  - `l1RpcProvider` : `ethers.providers.JsonRpcProvider` instance for L1
  - `l2RpcProvider` : `ethers.providers.JsonRpcProvider` instance for L2
  - `l1Wallet` : Create a wallet for the L1 network using `ethers.Wallet`
  - `l2Wallet` : Create a wallet for the L2 network using `ethers.Wallet`

```
const l1RpcProvider = new ethers.providers.JsonRpcProvider(l1Url)
const l2RpcProvider = new ethers.providers.JsonRpcProvider(l2Url)
const l1Wallet = new ethers.Wallet(privateKey, l1RpcProvider)
const l2Wallet = new ethers.Wallet(privateKey, l2RpcProvider)
```

- You create a `BatchCrossChainMessenger` instance of the Titan SDK. The parameters for creation are as follows. The created instance is used to transfer ETH.
  - l1ChainId: L1 network chain ID
  - l2ChainId: L2 network chain ID
  - l1SignerOrProvider: L1 network wallet
  - l2SignerOrProvider: L2 network wallet

```
crossChainMessenger = new titanSDK.BatchCrossChainMessenger({
      l1ChainId: 5,    // Goerli value, 1 for mainnet
      l2ChainId: 5050,  // Goerli value, 55004 for mainnet
      l1SignerOrProvider: l1Signer,
      l2SignerOrProvider: l2Signer,
  })
```

- The `l1ChainId` and `l2ChainId` should be the Chain ID of the network supported by the Titan SDK. For example, to make deposits and withdrawals on the Titan Mainnet, `l1ChainId` should be

Deposit

- 1, the chain ID of the Ethereum Mainnet, and `l2ChainId` should be 55004, the chain ID of the Titan Mainnet. (See 'Connection information' section.)
- It is very simple to request deposit transaction to move ETH from L1 to L2. You can bridge the desired amount of ETH by calling `depositETH()` on the `crossChainMessenger` instance you created during the setup phase. (L1 account must have at least the amount of ETH you are depositing + the appropriate gas fee to send the transaction).
- After sending the deposit transaction, execute `wait()`. You have to wait for the transaction to be included in the block and for the transaction to be received.

```
// 0.000001 ETH (= 1,000 gwei)
// gwei = 10^9 wei
const response = await crossChainMessenger.depositETH(1000n * gwei)
await response.wait()
```

- The deposit process internally calls functions of the contracts deployed on L1 and L2. The unit of transfer that a contract passes to another contract is called a `message`.
- `waitForMessageStatus()` is a method that waits until the `message`, which is the first parameter, is changed to the message state defined in the second parameter. When the deposit is completed and the ETH is moved from L1 to L2, the status of message changes to `RELAYED`.
- You should check if the deposit has been completed successfully by checking if `response` has changed to `RELAYED`.

```
// message to wait for RELAYED
await crossChainMessenger.waitForMessageStatus(response, titanSDK.MessageStatus.RELAYED
```

Withdraw

- Titan SDK makes it easy to move ETH from L2 to L1. You can call `withdrawETH()` to bridge as much ETH as you want. (L2 account must have at least the amount of ETH you are withdrawing + the appropriate gas fee to send the transaction).

```
// 0.000001 ETH
const response = await crossChainMessenger.withdrawETH(1000n * gwei)
await response.wait()
```

- When you request a withdraw transaction from L2, the generated L2 transactions are rolled-up by the sequencer and proposer, which roll-up the transaction and State Root from L2 to L1. You wait for the message's status to change to `READY_FOR_RELAY` by calling `waifForMessageStatus()`.
- When the State Root is successfully rolled up to L1 by the proposer, the message's status will change to `READY_FOR_RELAY`. It will take 7 days for the status to change to `READY_FOR_RELAY` because Optimistic Rollup on the mainnet defines 7 days as the validation period for fraud proof.

```
await crossChainMessenger.waitForMessageStatus(response, titanSDK.MessageStatus.READY_F
```

- You can call `finalizeBatchMessage()` to request a relay transaction from L2 to L1. The method requires an array of messages as a parameter. The `finalizeBatchMessage()` will put the messages stored in the array into one transaction and relay them all simultaneously.
- Once the withdraw is successfully completed and the desired amount of ETH has been moved from L2 to L1, the message's status will change to `RELAYED`.