cBridge requirements

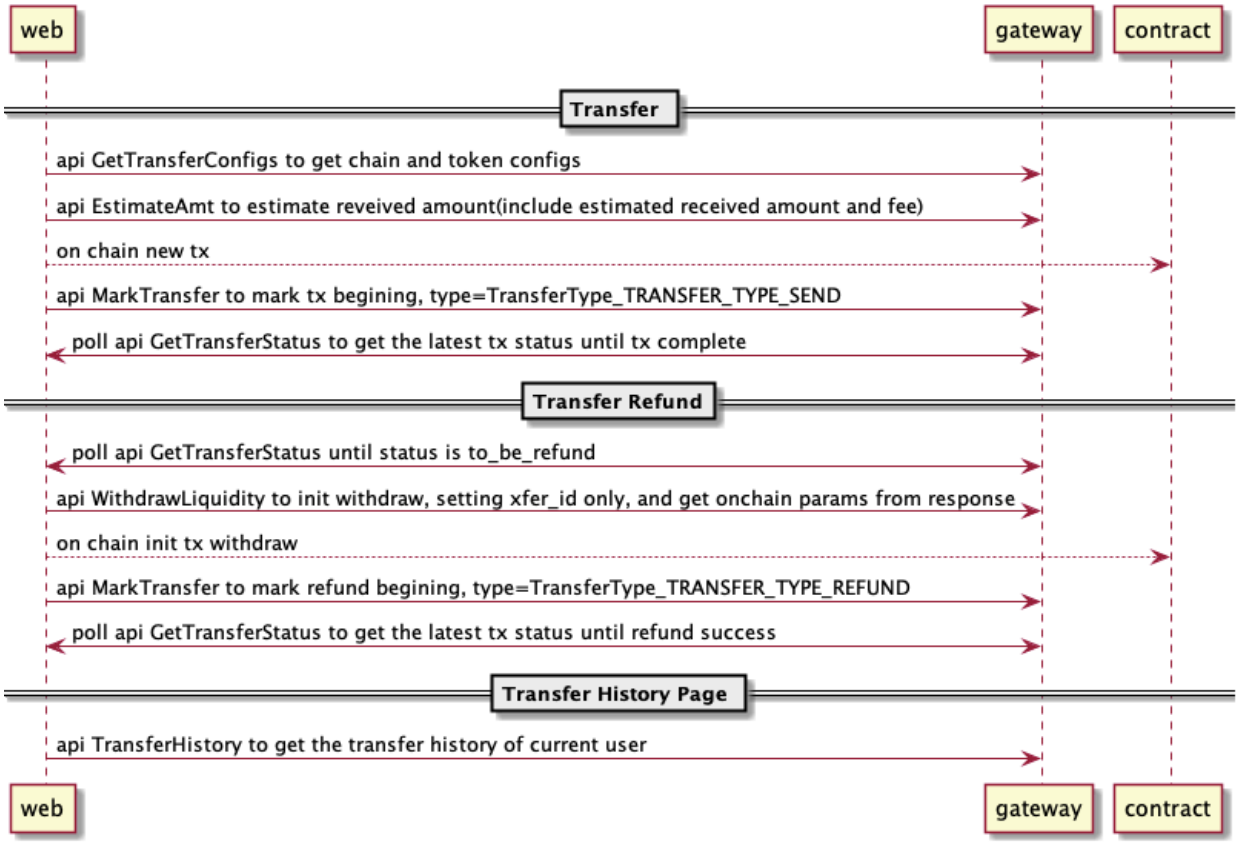
Introduction

cBridge 2.0 provides a simple liquidity provider experience and high liquidity efficiency for users when they manage their funds in different chains with lower cost. Celer State Guardian Network will use the entire pool's liquidity to calculate the slippage and pricing when users migrate funds from liquidity-abundant chainA to liquidity-scarce chainB.

cBridge Transfer Terminology

- src_chain: Source Chain
- dst_chain: Destination Chain
- WithdrawLiquidity: Refund for Failed Token Transfer

Generic Flow Chart



Flows

User Flow

1. Transfer
    a. User Input: token_info, token_amount, src_chain_info, dst_chain_info, slippage, walllet_info
    b. FE: Check allowance for selected src_chain, let user approve if needed
    c. FE -> BE: Get Transfer Estimate for User Input
        i. Input: src_chain_id, dst_chain_id, token_symbol, token_amount, user_address
        ii. Output: eq_value_token_amt, bridge_rate, fee, slippage_tolerance
    d. FE -> User: Show Transfer Estimate and Wait for User final approve
    e. FE -> Source Chain: send user's token to cBridge contract
        i. Input: receiver, token_address, token_amount, dst_chain_id, slippage
        ii. Output: transaction_hash
    f. FE -> BE: Mark Transfer(On-chain Transaction Notice for cBridge BE)

        i. Input: transfer_id, send_info, dst_min_receive_info, user_address, on_chain_transaction_hash,
    g. FE: Pooling Transfer Status Query => Waiting for TRANSFER_COMPLETED
        i. Input: transfer_id
        ii. Output: status, wd_onchain, sorted_sigs, singers, powers, src_block_tx_link, dst_block_tx_link

2. Request Refund (Triggered when status is TRANSFER_TO_BE_REFUNDED)
    a. FE -> Ethereum: Sign Message
        i. Input: transfer_id, req_id, withdraw_type
        ii. Output: sig
    b. FE -> BE: Withdraw Liquidity
        i. Input: withdraw_req, sig, estimated_received_amount, method_type
    c. FE: Pooling Transfer Status Query => Waiting for TRANSFER_REFUND_TO_BE_CONFIRMED
        i. Input: transfer_id
    d. User -> FE: Final Confirmation For Withdraw
    e. FE -> Source Chain: withdraw user's liquidity from cBridge contract
        i. Input: wd_onchain, sorted_sigs, singers, powers
        ii. Output: transaction hash
    f. FE -> BE: Mark Transfer
        i. Input: transfer_id, src_tx_hash, type
    g. FE: Pooling Transfer Status Query => Waiting for TRANSFER_REFUNDED
        i. Input: transfer_id

User Query

1. Get Transfer History
    a. User Input: Ethereum Address
    b. FE -> BE: Get Transfer History:
        i. Input: wallet_address, page_size, next_page_token
        ii. Output: next_page_token, current_size, transfer_history_lists

cBridge Transfer Related APIs

1. GetTransferConfigs

a. Input: none
b. Output:
    i. chains: Array<Chain> /// Chains supported by cBridge

ii. chain_token: Map<Chain, TokenInfos> /// Supported tokens for each chain
iii. farming_reward_contract_addr: String /// Farming Reward Contract

2. EstimateAmt

a. Input:
   i. src_chain_id: Number /// Source chain id
   ii. dst_chain_id: Number /// Destination chain id
   iii. token_symbol: String /// Token to be transferred
   iv. amt: String /// Token amount to be transferred
   v. usr_addr: String /// User wallet address
b. Output:
   i. eq_value_token_amt: String /// Token amount on destination chain
   ii. bridge_rate: Number /// bridge rate from src_chain to dst_chain
   iii. perc_fee: String /// cBridge LP and Celer SGN staker bonus
   iv. base_fee: String /// On-chain transaction gas fee
   v. slippage_tolerance: Number
   vi. max_slippage: Number /// The higher max_slippage, the more transaction success possibility.
c. Minimum received amount calculation:
   i. minimum received amount = transfer_amount * current_bridge_rate * (1-slippage_tolerance) - perc_fee - base_fee

3. On-chain cBridge send

a. Input:
   i. receiver: String /// User's wallet address
   ii. token_address: String /// Token address on source chain
   iii. value: BigNumber /// Token amount
   iv. dst_chain_id: BigNumber /// Destination Chain
   v. nonce: BigNumber /// Date().getTime()
   vi. max_slippage: BigNumber /// Estimate Amount Response max_slippage
b. Output:
   i. transaction_hash: String

4. MarkTransfer for Transfer

a. Input:
   i. transfer_id: String /// Ethereum generation Hash, see below for detail sample
   ii. src_send_info: TransferInfo /// Send info

1. chain: Chain /// src_chain
2. token: Token /// token info on src_chain
3. amount: String /// token amount

iii. dst_min_received_info: TransferInfo /// Receive info
1. chain: Chain /// dst_chain
2. token: Token /// token info on dst_chain
3. amount: String /// eq_value_token_amt - perc_fee - base_fee in EstimateInfo

iv. addr: String /// User's wallet Address

v. src_tx_hash: /// transaction_hash in On-chain cBridge send tx

b. Transfer_id Generation:

```
const transfer_id = ethers.utils.solidityKecaak256(
        [
         "address",
         "address",
         "address",
         "uint256",
         "uint64",
         "uint64",
         "uint64"
        ],
        [
         userWalletAddress,
         userWalletAddress,
         token.address,
         value.toString(),
         dest_chain_id,
         nonce,  /// same nonce as on-chain cBridge send transaction
         src_chain_id
        ],
)
```

5. GetTransferStatus

a. Input:

i. transfer_id: String /// same as MarkTransfer request

b. Output:

i. status: TransferHistoryStatus

ii. wd_onchain: String

iii. sorted_sigs: Array<String>

iv. signers: Array<String>

v. powers: Array<String>

vi. src_block_tx_link: String

vii. dst_block_tx_link: String

6. Sign WithdrawRequest data on Ethereum

a. Input:
   i. withdraw_request: WithdrawReq
      1. req_id: Number /// current timestamp
      2. xfer_id: String /// transfer_id as above
      3. withdraw_type: WithdrawType /// WITHDRAW_TYPE_REFUND_TRANSFER

b. Output:
   i. sig: String

c. Code Sample:

```
sig = await
        signer.signMessage(
            ethers.utils.arrayify(
                ethers.utils.keccak256(withdrawReqProto.serializeBinary())))
```

7. WithdrawLiquidity

a. Input:
   i. withdraw_req: Array<UInt8> /// serialized WithdrawReq above
   ii. sig: Array<UInt8> /// serialized sig above
   iii. estimated_received_amt: String /// send amount in this failed transfer
   iv. method_type: WithdrawMethodType /// WD_METHOD_TYPE_ONE_RM

8. On-chain cBridge withdraw tx

a. Input: parameters retrieved in GetTransferStatus response with format change
   i. wdmsg
   ii. sigs
   iii. signers
   iv. Powers

b. Output:
   i. transaction_hash

c. Code Sample:
   i. import { base64, getAddress, hexlify } from “ethers/lib/utils”;
   ii. const wdmsg = base64.decode(wd_onchain)
   iii. const sigs = sorted_sigs.map(item => {
            return base64.decode(item);
        })
   iv. const signers = signers.map(item => {
            const decodeSigner = base64.decode(item);

```
            const hexSigner = hexlify(decodeSigner);
            return getAddress(hexSigner)
        })
v.      const powers = powers.map(item => {
            return base64.decode(item);
vi.     })
```

9. MarkTransfer for Transfer Refund

a. Input:
   i. transfer_id: String /// same transfer_id as above
   ii. src_tx_hash: String /// On-chain withdraw transaction hash
   iii. type: MarkTransferTypeRequest.TRANSFER_TYPE_REFUND