Using the Standard Bridge

The Standard Bridge is a basic token bridging system available on OP Mainnet and all other standard OP Stack chains. The Standard Bridge allows you to easily move ETH and most ERC-20 tokens between Ethereum and OP Mainnet. Transfers from Ethereum to OP Mainnet via the Standard Bridge are usually completed within 1-3 minutes. Transfers from OP Mainnet to Ethereum are completed in 7 days as a result of thewithdrawal challenge period .

The Standard Bridge is fully permissionless and supports standard ERC-20 tokens. Other bridging systems also exist that provide different features and security properties. You may wish to explore some of these options to find the bridge that works best for you and your application.

The Standard Bridgedoes not supportfee on transfer tokens (opens in a new tab)orrebasing tokens (opens in a new tab) because these types of tokens may cause bridge accounting errors.

Design

The Standard Bridge allows users to convert tokens that are native to one chain (like Ethereum) into a representation of those tokens on the other chain (like OP Mainnet). Users can then convert these bridged representations back into their original native tokens at any time.

This bridging mechanism functions identically in both direction — tokens native to OP Mainnet can be bridged to Ethereum just like tokens native to Ethereum can be bridged to OP Mainnet. Here you'll get to understand how the Standard Bridge works when moving tokens from Ethereum to OP Mainnet. Since the bridging mechanism is mirrored on both sides, this will also explain how the bridge works in the opposite direction.

Architecture

The Standard Bridge is composed of two contracts, theL1StandardBridge (opens in a new tab) (onEthereum) and theL2StandardBridge (opens in a new tab) (onOP Mainnet). These two contracts interact with one another via theCrossDomainMessenger system for sending messages between Ethereum and OP Mainnet. You can read more about theCrossDomainMessenger in the guide onSending Data Between L1 and L2 .

Bridged Tokens

The Standard Bridge utilizes bridged representations of tokens that are native to another blockchain. Before a token native to one chain can be bridged to the other chain, a bridged representation of that token must be created on the receiving side.

A bridged representation of a token is an ERC-20 token that implements theIOptimismMintableERC20 (opens in a new tab) interface. This interface includes a few functions that theStandardBridge contracts use to manage the bridging process. All bridged versions of tokensmust implement this interface to be used with theStandardBridge . Native tokens do not need to implement this interface.

A native token may have more than one bridged representation at the same time. Users must always specify which bridged token they wish to use when using the bridge. Different bridged representations of the same native token are considered entirely independent tokens.

Bridging Native Tokens

The Standard Bridge uses a "lock-and-mint" mechanism to convert native tokens into their bridged representations. This means thatnative tokens are locked into the Standard Bridge on one side after whichbridged tokens are minted on the other side. The process for bridging a native token involves a few steps.

User gives the Standard Bridge an allowance

The Standard Bridge must be able to pull tokens from the user to lock them into the bridge contract. To do this, the user must first give the bridge anallowance(opens in a new tab) to transfer the number of tokens that the user wishes to convert into a bridged representation.

User calls the bridging function

After providing a sufficient allowance, the user calls thebridgeERC20To (opens in a new tab) function on theStandardBridge contract on the chain where the native token lives (e.g., theL1StandardBridge contract if the token is native to Ethereum).

The user must provide the following parameters to this function call:

- address _localToken
- : Address of the native token on the sending side.

- address _remoteToken
- : Address of the bridged representation on the receiving side.
- address _to
- : Address of the recipient of these tokens, usually the sender's address.
- uint256 _amount
- : Number of tokens to transfer.
- uint32 _minGasLimit
- : Gas to use to complete the transfer on the receiving side.
- bytes calldata _extraData
- : Optional identify extra data.

Users can also trigger thebridgeERC20 (opens in a new tab) function instead ofbridgeERC20To to avoid needing to specify theaddress _to parameter. Doing so will automatically set theaddress _to parameter to themsg.sender .

ThebridgeERC20 function can be potentially dangerous for users withsmart contract wallets(opens in a new tab) as some smart contract wallets cannot be deployed at the same address on every blockchain. To help users avoid potentially losing access to tokens by accident, thebridgeERC20 function will always revert when triggered from a smart contract. Smart contract wallet users and other smart contracts should therefore use thebridgeERC20To function instead.

The Standard Bridge locks the transferred tokens

When the user triggers thebridgeERC20To function while transferring a native token, the Standard Bridge will pull the_amount of_localToken tokens from the user's address and lock them inside of the bridge contract. A record of all locked tokens are stored within adeposits mapping(opens in a new tab) that keeps track of the total number of tokens deposited for a given_localToken and_remoteToken pair.

Since a native token may have more than one bridged representation, thedeposits token must keep track of the deposit pools for each_localToken /_remoteToken pair independently.

To illustrate, suppose that two users deposit 100 units of the same native token,Token A , but wish to receive two different bridged tokens,Token B andToken C . Although the Standard Bridge would now have a total balance of 200 units ofToken A , the mapping would show that theToken A /Token B pool and theToken A /Token C pool both have only 100 units.

The Standard Bridge sends a minting message

After locking the native tokens, the Standard Bridge contract on the sending side will trigger a cross-chain message to the Standard Bridge contract on the receiving side via theCrossDomainMessenger system. This message tells the receiving side tomint tokens according to the parameters specified by the user. Specifically, this message is an encoded call to thefinalizeBridgeERC20 (opens in a new tab) function on the other Standard Bridge contract. At this point, execution ends on the sending side.

The minting message is executed

Once the minting message is sent, it must be relayed to the receiving side. Message relaying is automatic when sending from Ethereum to OP Mainnet but requires additional user transactions when sending from OP Mainnet to Ethereum. Read more about the message relaying process in the guide toSending Data Between L1 and L2 .

When the message is relayed, thefinalizeBridgeERC20 function will be triggered on the receiving Standard Bridge contract. This function will receive the_minGasLimit gas defined by the user to execute to completion.

The minting message is authenticated

Upon execution,finalizeBridgeERC20 verifies a number of things about the incoming request:

- The request must have originated from the Standard Bridge contract on the other blockchain(opens in a new tab)
- .
- The Standard Bridge must not be in an emergency paused state(opens in a new tab)
- .
- The bridged token must properly implement theIOptimismMintableERC20 interface(opens in a new tab)
- .
- The bridged token must recognize the original native token as itsremoteToken() (opens in a new tab)
- .

The bridged token is minted

If the minting message is fully verified,finalizeBridgeERC20 willmint tokens to the recipient(opens in a new tab)equal to the number of tokens originally deposited on the other blockchain. For this to work properly, the bridged representation of the native token must correctly implement amint function that allows the Standard Bridge to mint tokens arbitrarily. This is part of theIOptimismMintableERC20 (opens in a new tab) interface.

This completes the process of bridging native tokens. This process is identical in both the Ethereum to OP Mainnet and OP Mainnet to Ethereum directions.

Bridging Non-Native Tokens

The Standard Bridge uses a "burn-and-unlock" mechanism to convert bridged representations of tokens back into their native tokens. This means thatbridged tokens are burned on the Standard Bridge on one side after whichnative tokens are unlocked on the other side. The process for bridging a non-native, bridged representation of a token involves a few steps.

User calls the bridging function

Unlike when bridging native tokens, users do not need to provide an approval to trigger a transfer of a bridged token because the Standard Bridge should already have the ability toburn these tokens. Here, the user calls thebridgeERC20To (opens in a new tab) function on theStandardBridge contract on the chain where the bridged token lives (e.g., theL2StandardBridge contract if the token is bridged to OP Mainnet).

The user must provide the following parameters to this function call:

- address _localToken
- : Address of the bridged token on the sending side.
- address _remoteToken
- : Address of the native token on the receiving side.
- address _to
- : Address of the recipient of these tokens, usually the sender's address.
- uint256 _amount
- : Number of tokens to transfer.
- uint32 _minGasLimit
- : Gas to use to complete the transfer on the receiving side.
- bytes calldata _extraData
- : Optional identify extra data.

The Standard Bridge burns the transferred tokens

When the user triggers thebridgeERC20To function while transferring a bridge tokenthe Standard Bridge will burn the corresponding_amount of tokens from the sender's address(opens in a new tab) .

The Standard Bridge sends an unlock message

After burning the bridged tokens, the Standard Bridge contract on the sending side will trigger a cross-chain message to the Standard Bridge contract on the receiving side via theCrossDomainMessenger system. This message tells the receiving side tounlock tokens according to the parameters specified by the user. Specifically, this message is an encoded call to thefinalizeBridgeERC20 (opens in a new tab) function on the other Standard Bridge contract. At this point, execution ends on the sending side.

The unlock message is executed

Once the unlock message is sent, it must be relayed to the receiving side. Message relaying is automatic when sending from Ethereum to OP Mainnet but requires additional user transactions when sending from OP Mainnet to Ethereum. Read more about the message relaying process in the guide toSending Data Between L1 and L2 .

When the message is relayed, thefinalizeBridgeERC20 function will be triggered on the receiving Standard Bridge contract. This function will receive the_minGasLimit gas defined by the user to execute to completion.

The unlock message is authenticated

Upon execution,finalizeBridgeERC20 verifies a number of things about the incoming request:

- The request must have originated from the Standard Bridge contract on the other blockchain(opens in a new tab)
- .
- The Standard Bridge must not be in an emergency paused state(opens in a new tab)
- .

The native token is unlocked

If the minting message is fully verified,finalizeBridgeERC20 willunlock and transfer tokens to the recipient(opens in a new tab) equal to the number of tokens originally burned on the other blockchain.

This completes the process of bridging native tokens. This process is identical in both the Ethereum to OP Mainnet and OP

Mainnet to Ethereum directions.

Bridging ETH

The Standard Bridge contracts can also be used to bridge ETH from Ethereum to OP Mainnet and vice versa. The ETH bridging process is generally less complex than the ERC-20 bridging process. Users simply need to trigger and send ETH to thebridgeETH (opens in a new tab) orbridgeETHTo (opens in a new tab) functions on either blockchain.

Tutorials

- Learn how to bridge ERC-20 tokens with the Optimism SDK
- Learn how to bridge ETH with the Optimism SDK
- Learn how to create a standard bridged token
- Learn how to create a custom bridged token

Superchain Token List

TheSuperchain Token List exists to help users discover the right bridged token addresses for any given native token. Consider checking this list to make sure that you're not using the wrong bridged representation of a token when bridging a native token.

Developers who are creating their own bridged tokens should consideradding their token(opens in a new tab) to the Superchain Token List. Tokens on the Superchain Token List will automatically appear on certain tools like theOptimism Bridge UI(opens in a new tab) .

Searching the Token List

You should strongly consider using the Superchain Token List to verify that you're using the correct bridged representation of a token when bridging a native token. Doing so can help you avoid accidentally bridging to the wrong token and locking up your native token permanently.

You can easily find the bridged representation of a token for OP Mainnet on theBridged Token Addresses page. If you want to find the bridged representation of a token for another chain, use the following steps.

Find the token you want to bridge

The Superchain Token List is organized by the token's address and native blockchainSearch the token list(opens in a new tab) for the token you want to bridge to confirm that it's included in the list. Make sure that the chain ID in the entry matches the chain ID of the blockchain you're bridging from. Retrieve the token's name and symbol from the list.

Find the bridged representation of the token

Once you've found the token you want to bridge, look for the token's name and symbol in the list. Find the entry that matches the name and symbol of the token you want to bridge and where the chain ID matches the chain ID of the blockchain you're bridging to. The address of this entry is the address of the bridged representation of the token you want to bridge.

Special Considerations

USDC

Circle(opens in a new tab) , the issuer ofUSDC(opens in a new tab) , natively issues USDC on OP Mainnetas of September 2023(opens in a new tab) . Before this service was made available, USDC had to be bridged to OP Mainnet via the Standard Bridge. This bridged representation of USDC is referred to asUSDC.e .

Circle recommends that users and developers make use of the native version of USDC issued by Circle over the bridgedUSDC.e token. Please note that the bridgedUSDC.e token will be deprecated in the near future.

Information about the bridgedUSDC.e token and native USDC token can be found below.

Symbol Description Address USDC.e Bridged USDC from Ethereum0x7f5c764cbc14f9669b88837ca1490cca17c31607 (opens in a new tab) USDC Native USDC issued by Circle0x0b2C639c533813f4Aa9D7837CAf62653d097Ff85 (opens in a new tab)

Basics of Bridging Custom Token Bridges