Warp Routes: Fast Native Transfers via Custom Gas Tokens

This document will walk you through how to use Hyperlane to enable fast native transfers to and from your L2 via a custom gas token.

Primary Motives

1. Create a custom gas token to use on your L2 of choice.
2. Use this gas token to enable fast native transfers, instead of long wait times for settlement via canonical bridges.

Pre-Requisites

To complete the following walkthrough, you should have the following available:

1. A L2 network (”yourchain”) on which you are able to specify acustom gas token
2. and would like to enable fast native transfers to and from.
3. A sole deployer or authorized owner to receive mint of the receipt token (more information on this token below).
4. An installed instance of theHyperlane CLI
5. and a wallet private key sourced asHYP_KEY
6. , funded on all relevant networks.

Walkthrough: Fast Native Transfers

info Note that a “receipt token” below is a (one-time) mintable “dummy” token used to collateralize a warp-route on yourchain's corresponding L1. You will burn the minting rights to the token after creating it.

Also, depending on your L2, the token symbol of your custom gas token may be assigned the same symbol as your receipt token. If deploying on ethereum, you may want to set your symbol toETH .

Steps

1) Deploy a Receipt Token

Deploy anETH receipt token to ethereum to become your L2's custom gas token.

1. For example, you may deploy your receipt token viaRemix
2. with a funded wallet and a contract similar to below:
3. // SPDX-License-Identifier: MIT
4. pragma
5. solidity
6. ^
7. 0.8.20
8. ;
9. import
10. "@openzeppelin/contracts/token/ERC20/ERC20.sol"
11. ;
12. contract
13. YourchainETH
14. is
15. ERC20
16. {
17. constructor
18. (
19.)
20. ERC20
21. (
22. "YourchainETH"
23. ,
24. "ETH"
25.)
26. {

27. _mint
28. (
29. msg
30. .
31. sender
32. ,
33. 10000000
34. *
35. (
36. 10
37. **
38. 18
39.)
40.)
41. ;
42. }
43. }
44. Ensure 100% of the total supply of yourETH
45. receipt token is minted to the deployer or authorized address of choice. This must be a one-time mint event in order to avoid value dilution of the ethereum ↔ yourchain native warp route, below.
46. Once deployed, save the address of the contract for use below.

2) Set the Custom Gas Token

Set the custom gas token on yourchain to the newly deployedETH ERC20 receipt token.

1. To set a custom gas token using OP stack:https://docs.optimism.io/builders/chain-operators/features/custom-gas-token
2. To set a custom gas token using arbitrum orbit:https://docs.arbitrum.io/launch-orbit-chain/how-tos/use-a-custom-gas-token

3) Transfer All Receipt Tokens

Use your L2's canonical bridge to transfer 100% of theETH receipt token from ethereum to yourchain.

1. To transfer using the OP stack standard bridgehttps://docs.optimism.io/builders/app-developers/bridging/standard-bridge
2. To transfer using the arbitrum orbit bridge:https://docs.arbitrum.io/launch-orbit-chain/how-tos/add-orbit-chain-to-bridge-ui

4) Deploy a Native Warp Route

Using the Hyperlane CLI, deploy an ethereumETH EvmHypNative ↔ yourchainETH EvmHypNative warp route:

1. Runhyperlane warp init
2. and complete the following flow:
3.
    1. Selectethereum
4.
    1. andyourchain
5.
    1. using space, and hit enter.
6.
    1. For ethereum, selectnative
7.
    1. and accept the mailbox.
8.
    1. For yourchain, selectnative
9.
    1. and accept the mailbox.
10. Before executing the deployment, ensure yourchain’smetadata.yaml
11. has ablockExplorers
12. field defined. This will be needed for contract verification in order to easily collateralize yourchain's warp route.
13. Runhyperlane warp deploy
14. and confirm the warp deploy config is as expected.
15. After execution, youryourchain-ethereum-config.yaml
16. deployment file will look similar to the following:

```yaml
```

17. yaml

18. -
19. language
20. -
21. server
22. :
23. schema
24. =
25. .
26. .
27. /
28. schema
29. .
30. json
31. tokens
32. :
33. -
34. addressOrDenom
35. :
36. "0x3e5bB1a03fef5DB15A320885E6A0C8Bff8b656bd"
37. chainName
38. :
39. yourchain
40. connections
41. :
42. -
43. token
44. :
45. ethereum
46. |
47. ethereum
48. |
49. 0x6d64832bDB4F04721D4F23CCbF17326cb636101e
50. decimals
51. :
52. 18
53. name
54. :
55. Ether
56. standard
57. :
58. EvmHypNative
59. symbol
60. :
61. ETH
62. -
63. addressOrDenom
64. :
65. "0x6d64832bDB4F04721D4F23CCbF17326cb636101e"
66. chainName
67. :
68. ethereum
69. connections
70. :
71. -
72. token
73. :
74. ethereum
75. |
76. yourchain
77. |
78. 0x3e5bB1a03fef5DB15A320885E6A0C8Bff8b656bd
79. decimals
80. :
81. 18
82. name
83. :

84. Ether
85. standard
86. :
87. EvmHypNative
88. symbol
89. :
90. ETH

5) Collateralize Your Warp Route

Deposit yourchainETH as collateral into yourchain's native warp route via thereceive() function. If needed, you may use the verified contract to do this via your scanner's UI, or a wallet such as Metamask

This collateralizes the native warp route, enabling users to bridge their native assets quickly to and from yourchain.

6) Testing

You can initiate a test transfer of a single wei with the following command:

hyperlane warp send --relay

--warp

HOME /.hyperlane/deployments/warp_routes/ETH/ethereum-yourchain-config.yaml tip By Default, your warp core config is found in the local Registry atHOME/.hyperlane/deployments/warp_routes to be used with the--warp flag. You can test in either direction between where you have the warp route set.

Congrats! You have now enabled fast native transfer to and from yourchain via a custom gas token on your L2.

warning Please note that this collateralization strategy takes on certain ISM trust assumptions. info Take a look at the inEVM bridge to see an example of these warp routes in action:https://bridge.inevm.com/

Check out some additional information, published by the Hyperlane supporting team & Injective:

→inEVM Spotlight: Hyperlane

→Connecting Injective: Hyperlane Opens the inEVM Bridge

→Hyperlane and Injective | Expanding opportunities Edit this page Previous Deploy Yield Routes Next How to extend an existing Warp Route