# Manage Warp Route Limits on xERC20 and FiatToken

## xERC20 Deployments

Warp Routes supportxERC20 tokens . Follow these steps to configurexERC20 for Warp Route usage:

1. Ownership Transfer
2. :
3. Ensure that the ownership of thexERC20 contract
4. is transferred to a secure multisig for security purposes. This step is crucial to prevent unauthorized changes.
5. ThexERC20
6. contract uses OpenZeppelin'sOwnable
7. interface. Use thetransferOwnership
8. function to complete this step.
9. Mint Limit Configuration
10. :
11. The minting and burning limits for the Warp Route contract are managed through thesetLimits
12. function in thexERC20
13. contract. This function must be called by the contract owner.* ThexERC20
14. 
    - contract uses a 24-hour window to manage limits. This is defined by the_DURATION
15. 
    - variable, which is set to 1 days (24 hours). The current available limits are calculated dynamically using the_getCurrentLimit
16. 
    - function.
17. 
    - If 24 hours (_DURATION
18. 
    - ) have passed since the last use, the limit will automatically restore to the fullmaxLimit
19. 
    - .
20. Ensure limits are appropriate for the expected volume to prevent disruptions. Review and adjust the limits based on transaction volumes and expected usage patterns.

## FiatToken Deployments

Warp Routes supportCircle's Bridged USDC in the form of a minter forFiatToken(See more on the repo for documentation). There are three roles that are relevant on theFiatToken andMasterMinter contracts:

1. MasterMinter
2. owner
3. is the account that can set controllers and minters.
4. MasterMinter
5. controller
6. is the account that can set the mint limits for its assigned minters.
7. MasterMinter
8. minter
9. is the account that can actually callmint
10. onFiatToken
11. .

The owner and controller should typically be set to a Safe multisig for enhanced security, the minter is the Warp Route contract address on the local chain. Mint limits forFiatToken are absolute, meaning they do not reset automatically and must be updated if required.

There are three actions that should be set on theMasterMinter contract to be ready for usage:

1. Remove the previous test controller:
2. :

3. As the owner, remove the previous test controller via the removeController(address _controller) function
4. Set the controller and minter:
5. As the owner, you should set the controller and minter via the configureController(address controller, address worker) function
6. . The controller can be the same as the owner, the minter should be the warp route address.
7. Set the mint limit for the minter:
8. As the controller, you should set the mint limit for the minter via the configureMinter(uint256 _newAllowance) function
9. . This limit is not being continuously reset, so either set it to a sufficiently large value (likecast max-uint
10. ) or monitor the usage and adjust accordingly.

Edit this page Previous Deploy an EVM \<-> SVM Warp Route Next How to Go to Production with your Hyperlane Deployment