How to adopt the bridged USDC standard on your Orbit chain

Circle'sBridged USDC Standard is a specification and process for deploying a bridged form of USDC on EVM blockchains with optionality for Circle to seamlessly upgrade to native issuance in the future.

Why adopt the bridged USDC standard?

When USDC is bridged into an Orbit chain, the default path is to use the chain'scanonical gateway contracts for ERC-20s . By way of example, when a user bridges USDC from Arbitrum One to an Orbit chain, their Arbitrum One USDC tokens are locked into the Orbit chain's parent side bridge, and a representative USDC token is minted to the user's address on the Orbit chain, via the child side bridge.

The challenge with this user flow is twofold.

1. Native vs. non-Native USDC:
2. The USDC tokens issued by Circle ('native USDC') are locked in the parent side bridge contract. Conversely, the USDC tokens on the Orbit chain aren't native USDC but are collateralized by the locked tokens in the bridge. As such, Circle will not recognize these tokens across their product suite.
3. Fragmented UX:
4. If Circle were to provide native support for USDC by deploying a USDC contract on the Orbit chain, there would be two forms of USDC on the chain (native and non-native USDC). This leads to a fragmented user experience, and users with non-native USDC would have to withdraw to the parent chain to be able to turn their tokens into native USDC.

By deploying the bridged USDC standard from the start, all USDC tokens that are bridged are locked in a gateway contract that can be adopted by Circle should a chain upgrade its USDC into native USDC. This allows USDC adoption on Orbit chains today without encountering either of the two problems above.

How to implement the bridged USDC Standard

We provide a custom USDC gateway implementation (for parent and child chains) that follows the Bridged USDC Standard. These contracts can be used by new Orbit chains. This solution will NOT be used in existing Arbitrum chains that are governed by the DAO.

- On a parent chain the contractL1USDCGateway
- is used in case the child chain uses ETH as native currency, orL1OrbitUSDCGateway
- in case the child chain uses a custom fee token.
- On a child chain,L2USDCGateway
- is used.
- For the USDC token contracts, Circle's referenceimplementation
- is used.

This page describes how to deploy a USDC bridge compatible with both the Orbit token bridge and Circle's Bridged USDC Standard.

Steps for a transition to native USDC issuance are also provided. Note that both Circle and the Orbit chain owner must agree to transition to native USDC issuance.

Requirements

- Recommended token bridge version 1.2.0
- No additional dependencies with Nitro or Nitro contract version

Other requirements:

- It is assumed there is already a USDC token deployed and used on the parent chain.
- Also, it is assumed that the standard Orbit chain ownership system is used, i.e.,UpgradeExecutor
- is the owner of theownable
- contracts, and there is an EOA or multi-sig that has the executor role on theUpgradeExecutor
- .
- Refer to thetoken bridge overview page
- for more information about the token bridge design and operational dynamics. You can learn more in ouroverview of gateways operating models
- .

Deployment steps

Throughout the docs and code, the termsL1 andL2 are used interchangeably withparent chain andchild chain . They have the same meaning, i.e., if an Orbit chain is deployed on top of ArbitrumOne, then ArbitrumOne isL1 /parent chain , while Orbit isL2 /child chain .

You can find more details by consulting theusdc bridge deployment script and its README . Checkout target code, install dependencies, and build

cd token-bridge-contracts yarn

install yarn build Populate your.env file based onenv.example in the project's root directory

PARENT_RPC

PARENT_DEPLOYER_KEY

CHILD_RPC

CHILD_DEPLOYER_KEY

L1_ROUTER

L2_ROUTER

INBOX

L1_USDC

OPTIONAL arg. If set, the script will register the gateway. Otherwise, it will store the transaction's payload in a file

ROLLUP_OWNER_KEY

Run the script

yarn deploy:usdc-token-bridge The script will do the following:

- load deployer wallets for L1 and L2
- register L1 and L2 networks in SDK
- deploy new L1 and L2 proxy admins
- deploy bridged (L2) USDC using the Circle's implementation
- init L2 USDC
- deploy L1 USDC gateway
- deploy L2 USDC gateway
- init both gateways
- ifROLLUP_OWNER_KEY
- is provided, register the gateway in the router through the UpgradeExecutor
- ifROLLUP_OWNER_KEY
- is not provided, prepare calldata and store it in theregisterUsdcGatewayTx.json
- file
- set minter role to L2 USDC gateway with max allowance

Now, new USDC gateways can be used to deposit/withdraw USDC. Everything is now in place to support transition to native USDC issuance if Circle and the Orbit chain owner agree to it.

Transitioning to native USDC

Once a transition to native USDC is agreed upon, the following steps are required:

- L1 gateway owner pauses deposits on the parent chain by callingpauseDeposits()
- L2 gateway owner pauses withdrawals on the child chain by callingpauseWithdrawals()
- master minter removes the minter role from the child chain gateway* NOTE: there should be no in-flight deposits when the minter role is revoked. If there are any, they should be finalized first. Anyone can do that by claiming the failed retryable tickets that execute a USDC deposit
- L1 gateway owner sets Circle's account as burner on the parent chain gateway usingsetBurner(address)
- L1 gateway owner reads the total supply of USDC on the child chain and then invokessetBurnAmount(uint256)
- on the parent/child gateway where the amount matches the total supply
- USDCmasterMinter
- gives the minter role with0
- allowance to the L1 gateway so that the burn can be executed
- on the child chain, the L2 gateway owner calls thesetUsdcOwnershipTransferrer(address)
- to set the account (provided and controlled by Circle), which will be able to transfer the bridged USDC ownership and proxy admin
- if not already owned by the gateway, the L2 USDC owner transfers ownership to the gateway, and proxy admin transfers admin rights to the gateway
- Circle uses theusdcOwnershipTransferrer
- account to triggertransferUSDCRoles(address)
- , which will set the caller as USDC proxy admin and will transfer USDC ownership to the provided address
- Circle callsburnLockedUSDC()
- on the L1 gateway using theburner
- account to burn theburnAmount
- of USDC* remaining USDC will be cleared off when remaining in-flight USDC withdrawals are executed, if any
-
    - The L1 gateway owner is trusted to not front-run this transaction to modify the burning amountEdit this page
    Last updatedonOct 30, 2024 Previous Customize ArbOS version Next Enable fast withdrawals