How to deploy a custom gas token chain using the Orbit SDK

UNDER CONSTRUCTION This document is under construction and may change significantly as we incorporate style guidance and feedback from readers. Feel free to request specific clarifications by clicking the Request an update button at the top of this document. info See the "create a rollup custom fee token" example in the Orbit SDK repository for additional guidance. Deploying a Custom Gas Token Orbit chain introduces a unique aspect to the standard Orbit chain setup: the ability to pay transaction fees using a specific ERC-20 token instead of ETH . While the setup process largely mirrors that of a standard Rollup Orbit chain , as detailed in the introduction , there are key differences to account for when configuring a Custom Gas Token Orbit chain.

info Custom gas tokens are only supported on Orbit AnyTrust chains, currently. This feature isnot available yet on Rollup Orbit chains.

Key Differences for Custom Gas Token Orbit Chain Deployment

1. Custom gas token specification

The most significant difference is the specification of the ERC-20 token on the parent chain to be used as the gas fee token. This requires selecting an existing ERC-20 token or deploying a new one to be used specifically for transaction fees on your Orbit chain.

Note: Currently, only ERC-20 tokens with 18 decimals are acceptable as gas tokens on Orbit chains.

2. Chain Configuration

You can configure your Orbit chain using the prepareChainConfig method and assigning it to a chainConfig variable.

Example:

import

{ prepareChainConfig }

from

'@arbitrum/orbit-sdk' ;

const chainConfig =

prepareChainConfig ({ chainId :

Some_Chain_ID , nativeToken : yourERC - 20TokenAddress , DataAvailabilityCommittee :

true , }) ; To use the prepareChainConfig method as shown in the example above, some inputs need to be defined:

Parameter Type Description chainId number Your Orbit chain's chainId . nativeToken Address The contract address on the parent chain of the ERC-20 token your chain will use for gas fees. It needs to have 18 decimals to be accepted on Orbit chains. DataAvailabilityCommittee boolean Should be set to true since only AnyTrust chains can accept ERC-20 tokens.

3. Token Approval Before Deployment Process

In Custom gas token Orbit chains, the owner needs to give allowance to the rollupCreator contract before starting the deployment process so that RollupCreator can spend enough tokens for the deployment process. For this purpose, we defined two APIs on the Orbit SDK:

A. createRollupEnoughCustomFeeTokenAllowance

This API gets related inputs and checks if the rollupCreator contract has enough Allowance on the token from the owner:

import

{ createRollupEnoughCustomFeeTokenAllowance }

from

'@arbitrum/orbit-sdk' ;

const allowanceParams =

{ nativeToken , account : deployer . address , publicClient : parentChainPublicClient , } ;

const enough Allowance

=

createRollupEnoughCustomFeeTokenAllowance (allowanceParams) To build theallowanceParams object as shown in the example above, you need to provide with the following:

Parameter Type Description nativeToken Address The contract address on the parent chain of theERC-20 token your chain will use forgas fees. account Address The address Orbit chain's publicClient PublicClient ThePublicClient objectas defined by the Viem library .

B. createRollupPrepareCustomFeeTokenApprovalTransactionRequest

This API gets related inputs and creates the transaction request to secure enough Allowance from the owner to theRollupCreator to spendnativeToken on the deployment process.

Example:

import

{ createRollupEnoughCustomFeeTokenAllowance }

from

"@arbitrum/orbit-sdk" ;

const allowanceParams =

{ nativeToken , account : deployer . address , publicClient : parentChainPublicClient , } ;

const approvalTxRequest = await

createRollupPrepareCustomFeeTokenApprovalTransactionRequest (allowanceParams ,) ;

4. Deployment Process

The overall deployment process, including the use of APIs likecreateRollupPrepareConfig andcreateRollupPrepareTransactionRequest , remains similar to theRollup deployment process. However, attention must be given to incorporating theERC-20 token details into these configurations.

note When using the API, you also need to specifynativeToken as a param. Example:

const txRequest =

await

createRollupPrepareTransactionRequest ({ params :

{ config , batchPoster , validators :

[validator] , nativeToken , } , account : deployer . address , publicClient : parentChainPublicClient , }) ; All other parts would be the same as explained in theRollup Orbit chain deployment page . Edit this page Last updatedonApr 2, 2024 Previous Deploy an AnyTrust chain Next Configure your chain's node