# How to Deploy an AnyTrust chain using the Orbit SDK

This section explains how to to initiate anAnyTrust Orbit chain usingArbitrum's Orbit SDK .

UNDER CONSTRUCTION This document is under construction and may change significantly as we incorporatestyle guidance and feedback from readers. Feel free to request specific clarifications by clicking theRequest an update button at the top of this document. info See the"set-valid-keyset" example in the Orbit SDK repository for additional guidance.

## About AnyTrust Orbit

AnyTrust chains follow a different model than Rollup chains, offering unique features and deployment processes. For a comprehensive understanding of the general principles of Orbit chains and the differences between Orbit chain types, please refer to theOrbit SDK introduction page.

## Deployment Steps

The deployment process of AnyTrust chains is very similar to that oRollup chain but with some differences that we'll go through in this guide.

Here are the steps involved in the deployment process:

1. Setting Up the Chain Parameters
2. Deploying Your AnyTrust chain
3. Getting the AnyTrust Orbit Chain Information After Deployment
4. Setting Valid Keyset on Parent Chain

The deployment of an AnyTrust Orbit chain involves defining and setting up theData Availability Committee (DAC) keyset. This keyset, comprising keys from the appointed members of the DAC, is essential for ensuring data availability and integrity. Once you have selected your committee members and gathered their keys, these keys are then configured into a keyset using the Orbit SDK. This keyset is subsequently embedded into the chain, serving as a verification mechanism to maintain the trust and security of the AnyTrust chain. The proper configuration and deployment of this keyset are vital for the effective operation and reliability of the AnyTrust model within the Arbitrum Orbit framework.

Let's go through each step:

## 1. Setting Up the Chain Parameters

Similarly to the Rollup chain, you'll need to prepare the AnyTrust chain configuration, including the core contracts and operational parameters that govern the chain's functionality, focusing on parameters that are specific to AnyTrust chains.

struct

Config

{ uint64 confirmPeriodBlocks ; uint64 extraChallengeTimeBlocks ; address stakeToken ; uint256 baseStake ; bytes32 wasmModuleRoot ; address owner ; address loserStakeEscrow ; uint256 chainId ; string chainConfig ; uint64 genesisBlockNum ; ISequencerInbox . MaxTimeVariation sequencerInboxMaxTimeVariation ; } ThechainConfig parameter within theConfig structure is an essential element for tailoring your Orbit chain according to specific needs. This parameter is particularly significant when setting up an AnyTrust Orbit chain, as it includes configurations that distinguish it from a Rollup chain. The key parameter that differentiates an AnyTrust chain in this context is theDataAvailabilityCommittee .

Start by setting up the chain configuration parameters. This includes defining key elements like:

Parameter Description chainId Your Orbit chain's unique identifier. It differentiates your chain from others in the ecosystem. DataAvailabilityCommittee Set tofalse , this boolean makes your chain as a Rollup, set totrue configures it as an AnyTrust chain. InitialChainOwner Identifies who owns and controls the chain. MaxCodeSize Sets the maximum size for contract bytecodes on the Orbit chain. e.g. Ethereum mainnet has a limit of 24,576 Bytes. MaxInitCodeSize Similar toMaxCodeSize , defines the maximum size for your Orbit chain'sinitialization code. e.g. Ethereum mainnet limit is 49,152 Bytes. For an AnyTrust chain, you need to set theDataAvailabilityCommittee totrue . This setting is crucial as it indicates the chain's reliance on a committee for data availability, which is a hallmark of the AnyTrust model.

Here’s an example of how to configure thechainConfig for an AnyTrust chain using the Orbit SDK:

import

{ prepareChainConfig }

from

'@arbitrum/orbit-sdk' ;

const chainConfig =

prepareChainConfig ( { chainId :

Some_Chain_ID , arbitrum :

{

InitialChainOwner : deployer_address ,

DataAvailabilityCommittee :

true

} , } ) ; In this example, you set up the chain configuration with a specificchainId , theInitialChainOwner as the deployer's address, and importantly, you configure theDataAvailabilityCommittee astrue . This configuration ensures that your Orbit chain is set up as an AnyTrust chain, utilizing the unique features and operational model of the AnyTrust system within the Arbitrum Orbit framework.

These parameters tailor the chain to your specific requirements and operational context.

The Orbit SDK has a helper to execute this part:

For an easier config preparation, the Orbit SDK provides theprepareChainConfig API, which takes config parameters as arguments and returns achainConfig JSON string. Any parameters not provided will default to standard values, which are detailed in theOrbit SDK documentation .

## 2. Deploying Your AnyTrust Chain

After configuring your chain with thecreateRollupPrepareConfig API, the next step is to use thecreateRollupPrepareTransactionRequest API. This API is designed to take the parameters defined in theRollupDeploymentParams , along with the configuration generated by thecreateRollupPrepareConfig API, to prepare a transaction request. This request is then used to invoke thecreateRollup function of theRollupCreator contract, which effectively deploys and initializes the core contracts of your AnyTrust Orbit chain.

For instance, to deploy using the Orbit SDK with a Config equal toconfig , abatchPoster , and a set of validators such as[validator] , the process would look like this:

import

{ createRollupPrepareTransactionRequest }

from

'@arbitrum/orbit-sdk' ;

const request =

await

createRollupPrepareTransactionRequest ( { params :

{ config , batchPoster , validators :

[ validator ] , } , account : deployer_address , publicClient , } ) ; After creating the raw transaction, you can sign and broadcast it to the network.

## 3. Getting the AnyTrust Orbit Chain Information After Deployment

To extract detailed information about your AnyTrust Orbit chain post-deployment, you will use the same API and steps as you would for a Rollup Orbit chain. Here's a reminder of the example:

import

{ createRollupPrepareTransactionReceipt }

from

'@arbitrum/orbit-sdk' ;

const data =

createRollupPrepareTransactionReceipt ( txReceipt ) ; In this example,txReceipt refers to the transaction receipt you received after deploying the AnyTrust chain. By inputting this receipt into thecreateRollupPrepareTransactionReceipt function, you can access comprehensive data about your deployment, including details about the core contracts and configuration settings.

## 4. Setting Valid Keyset on Parent Chain:

The final step in deploying your AnyTrust Orbit chain is to set up the valid keyset for your Data Availability Committee (DAC) on the parent chain. This keyset is essential for ensuring the integrity of the certificates that the DAC signs when storing Orbit chain's data. The process of generating keys and the keyset for your committee is comprehensively explained in our documentation. Once you have your keyset, it needs to be established on theSequencerInbox contract of your Orbit chain on the parent chain.

To facilitate this, we provide an API in Orbit SDK namedsetValidKeysetPrepareTransactionRequest . This API aids in setting the keyset on the parent chain. To use this API, you need specific information that you gathered in step 3. This includes theupgradeExecutor andsequencerInbox addresses of your Orbit chain, the generated keyset for your committee, and the account of the owner.

Here's an example of how you can use the Orbit SDK to set the keyset:

const txRequest =

await

setValidKeysetPrepareTransactionRequest ( { coreContracts :

{ upgradeExecutor :

'upgradeExecutor_address' , sequencerInbox :

'sequencerInbox_address' , } , keyset , account : deployer . address , publicClient : parentChainPublicClient , } ) ; In this example,upgradeExecutor_address andsequencerInbox_address are placeholders for the actual addresses of the respective contracts in your Orbit chain.keyset is the keyset you generated for your committee, anddeployer.address refers to the owner's account address.

After you create the transaction request using the above API, the next step is to sign and send the transaction. This action will effectively set the keyset on the parent chain, allowing it to recognize and verify the valid keyset for your AnyTrust Orbit chain. This step is crucial for the operational integrity and security of your AnyTrust chain, ensuring that the data verified by the DAC is recognized and accepted by the parent chain. Edit this page Last updatedonApr 2, 2024Previous Deploy a Rollup chain Next Deploy a custom gas token chain