Using Data Feeds on StarkNet

StarkNet is a permissionless decentralized ZK-Rollup operating as an L2 network over Ethereum. Unlike other Ethereum L2 networks, StarkNet is not EVM-compatible and usesCairo as its smart contract language. Chainlink Data Feeds are available on the StarkNet testnet as Cairo smart contracts.

You can read Chainlink Data Feeds on StarkNet using anonchain contract that you compile and deploy. Alternatively, you canread the data feed offchain without a StarkNet account. You can complete these steps using only theStarkNet CLI commands , but the example scripts demonstrate how to compile, deploy, and interact with StarkNet contracts programmatically.StarkNet.js ,HardHat , and theStarkNet Hardhat Plugin simplify the processes, which normally require you to keep track of your class hashes and ABI files. Seehttps://docs.starknet.io/ for more information about writing and compiling Cairo contracts for StarkNet.

For a complete list of Chainlink Price Feeds available on StarkNet testnet, see thePrice Feed Contract Addresses page.

Requirements

Set up your environment to run the examples.

- Set up your local StarkNet environment . Note that a Python version in the>=3.6 <=3.9range is required for compiling and deploying contracts onchain. Thecairo-langPython package is not compatible with newer versions of Python as of thecairo-lang0.10.3 package. Checkstarknet.io for the latest requirements.
- Install NodeJS in the version in the>=14 <=18version range.
- Install Yarn .
- Clone and configure the code examples:
- Clone thesmartcontractkit/chainlink-starknet repository, which includes the example contracts for this guide:

gitclone https://github.com/smartcontractkit/chainlink-starknet.git 2. In your clone of thechainlink-starknet repository, change directories to the proxy consumer example:

cd./chainlink-starknet/examples/contracts/proxy-consumer/ 3. Runyarn installto install the required packages includingStarkNet.js ,HardHat , and theStarkNet Hardhat Plugin .

yarninstall * If you want to run the onchain examples, you mustset up a StarkNet account on StarkNet'salpha-goerlinetwork and fund it withtestnet ETH . These examples expect the OpenZeppelin wallet, which stores your addresses and private keys in the following default path:

~/.starknet_accounts/starknet_open_zeppelin_accounts.json

After you prepare the requirements, check to make sure the required tools are configured correctly:

- StarkNet CLI:

starknet -v starknet 0.10.3 * Cairo CLI:

cairo-compile -v cairo-compile 0.10.3 * NodeJS:

node -v v18.12.1 * Yarn:

yarn --version 1.22.19

Running the onchain example

The onchainproxy consumer example uses a local OpenZeppelin wallet as the account to deploy a contract onchain. This contract reads a specified Chainlink data feed and stores the information for the latest round of data. This example has the following components:

- The exampleproxy_consumer.cairo contract: You will compile and deploy this example contract to the StarkNet Goerli testnet where it can read and store values from one of thedata feed proxy contracts . The proxy address is defined in the constructor when you deploy the contract.
- ThedeployConsumer.ts script: This script usesStarkNet.js to identify your OpenZeppelin wallet and deploy the compiled contract.
- TherreadLatestRound.ts script: This script submits an invoke transaction on theget_stored_roundfunction in your contract and prints the result.

Build, deploy, and invoke the example contract:

1. Find the account address and private key for your funded StarkNet testnet account. By default, the OpenZeppelin wallet contains these values at~/.starknet_accounts/starknet_open_zeppelin_accounts.json.
2. Export your address to theDEPLOYER_ACCOUNT_ADDRESSenvironment variable and your private key to theDEPLOYER_PRIVATE_KEYenvironment variable.

exportDEPLOYER_ACCOUNT_ADDRESS=exportDEPLOYER_PRIVATE_KEY= 3. In the./chainlink-starknet/examples/contracts/proxy-consumer/directory, runyarn buildto run Hardhat and create./starknet-artifacts/with the compiled contracts. The@shardlabs/starknet-hardhat-pluginpackage handles the compile step. 4. Runyarn deployto deploy the example consumer contract to the StarkNet Goerli testnet. The deployment might take several minutes depending on network conditions. The console prints the contract address and transaction hash. Record the contract address.

yarn deploy yarn run v1.22.19

Contract address: 0x297d4d4e0dc667c82a452cf809176c50a3e3707408ce39d0f1a1a881d35a83f Transaction hash: 0x545bf14dd55447c95065092a142cdf240806a594af39e25fd283e3059131f7d Done in 110.04s. 5. Runyarn readLatestRound to send an invoke transaction to the deployed contract. Specify the contract address printed by the deploy step. The deployed contract reads the latest round data from the proxy, stores the values, and prints the resulting values.

yarn readLatestRound 0x297d4d4e0dc667c82a452cf809176c50a3e3707408ce39d0f1a1a881d35a83f

Invoking the get_latest_round_data function. Transaction hash: 0x44ce5582f0ae7d144fec2d47ffa879096f7b01a540d7b1d2169aa1fe3798d4f Waiting for transaction... Transaction status is: NOT_RECEIVED Transaction status is: RECEIVED Transaction status is: PENDING Transaction is: ACCEPTED_ON_L2

Stored values are: round_id = 3.402823669209385e+38 answer = 6 block_num = 613094 observation_timestamp = 1673365796 transmission_timestamp = 1673365809 Done in 99.10s.

If the invoke request is successful, you can see the stored values in theStarkScan testnet explorer . Search for your contract by the address. Under theRead Contracttab, run a query on theget_stored_round()method. For example, you can see thecontract methods from the previous output examples.

You can achieve a similar result by running thestarknet-compile,starknet declare,starknet deploy, andstarknet invokeCLI commands , but the scripts are useful for interacting with StarkNet contracts and accounts programmatically.

Running the offchain example

This example reads the proxy contract to get the latest values with no account or contract compiling steps required. ThereadLatestRoundOffChain.ts script usesStarkNet.js to make a call directly to the data feed proxy address. By default, the script reads theLINK / USD feed , but you can change the address to read any of theavailable data feeds on the StarkNet testnet.

In the./chainlink-starknet/examples/contracts/proxy-consumer/directory, runyarn readLatestRoundOffChain.

yarn readLatestRoundOffChain yarn run v1.22.19

round_id = 3.402823669209385e+38 answer = 6 block_num = 613151 observation_timestamp = 1673367749 transmission_timestamp = 1673367801 Done in 3.09s. You can achieve a similar result by running thestarknet callCLI command and specifying the proxy address, function, and the ABI file for the data feed proxy contract. For this example, the ABI file is available in the repository, but you can also generate the ABI yourself from theaggregator_proxy.cairo contract source file.

starknet call--address0x2579940ca3c41e7119283ceb82cd851c906cbb1510908a913d434861fdcb245--functionlatest_round_data--abi./contracts/aggregator_proxy_abi.json The command prints the result:

0x100000000000000000000000000017c51 6 613174 1673368490 1673368507