Direct deal-making with Client contract

This page covers the client contract, and includes a tutorial on how developers can use the client contract to create storage deals programmatically.

In this tutorial we will cover the background of creating storage deals using smart contracts, and how to create storage deals with smart contracts on theFilecoin virtual machine .

You can find a video form of this walkthrough onETHGlobal's YouTube Channel .

Prerequisites

Before continuing, make sure you have the following software installed and prerequisites ready:

- Git
- NodeJS
- Yarn or NPM (Yarn is used in this tutorial)
- A code editor such as VS Code
- A wallet with Calibration testnet FIL
-

Steps

Let's run through how to create storage deals using smart contracts.

Setup

First, let's grab the kit and set up the development environment.

1. Clone the Filecoin virtual machine deal-making kit, including all submodules:
2. ```
3. Copy
4. gitclone--recurse-submoduleshttps://github.com/filecoin-project/fvm-starter-kit-deal-making.git
5. ```
6. This will copy the FVM deal-making kit into your current directory and initiate thego-generate-car
7. submodule.
8. Move into thefvm-starter-kit-deal-making
9. directory and grab all the dependencies usingyarn
10. :
11. ```
12. Copy
13. cdfvm-starter-kit-deal-making
14. yarn
15. ```
16. Now that all the packages are downloaded, you need to create a.env
17. file with your private key. This is so the Hardhat kit knows what wallet to use for transactions. Open up the repo in your code editor of choice and find the file titled.env.example
18. . Rename the file to.env
19. . You can do this in your terminal by running:
20. ```
21. Copy
22. mv.env.example.env
23. ```
24. Within the.env
25. file, replace the example private key with your actual private key. If you are using Metamask, followthis tutorial to get your private key
26. .
27.

Take precautions and never share your private key with anyone! Also make sure to not check your private key into Git. The.gitignore of the Hardhat kit is already set to ignore.env files. 1. Deploy the contracts withhardhat 2. : 3. 4. Copy 5. yarnhardhatdeploy 6. 7. This should compile and deploy all the contracts, including the client contract, which is the one we will be interacting with. Take note of the address of the deployed contract; we'll be using this later. 8.

Preparing a file for storage

Before storing a file with a storage provider it needs to be prepared by turning it into a.car file. The metadata also needs to be recorded. The Hardhat kit has a toolwhich can do this for you . However, to keep things nice and simple, we're going to use theFVM Data Depot website . This website will automatically convert files to the.car format, output all the necessary metadata, and act as an HTTP retrieval point for the storage providers.

1. Go to theFVM Data Depot website
2. and create an account.
3. ClickUpload File
4. and select a file you wish to upload.
5. Select theFile ID
6. of the file to read all the relevant metadata. Make a note of the:
7.
    - Piece CID
8.
    - Payload CID
9.
    - Car size
10.
    - Piece size
11.
    - URL
12.
    -
13. We'll use this information in the next step when invoking theMakeDealProposal
14. method.
15.

Invoke theMakeDealProposal method

Now that we have the.car file prepared in the data depot, we can invoke the MakeDealProposal method on the smart contract we deployed earlier. To do this, we will run themake-deal-proposal task in Hardhat. There are quite a few parameters to include in this call:

- contract
- : the address of your deployedClientContract.sol
- .
- `piece-cid: gathered from the previous step.
- piece-size
- : gathered from the previous step.
- car-size
- : gathered from the previous step.
- start-epoch
- : The block number you want the deal to begin on. It should be a block in the future. You can find the current block number onFilFox Calibration
- .
- end-epoch
- : The block number you want the deal to end on. It should be a block in the future and after theStart-Epoch
- . You can find the current block number onFilFox Calibration
- .
- location-ref
- : The location of where the storage provider can find the .car file. This is theURL
- from the previous step.
-

When calling themake-deal-proposal task in Hardhat, your command will look something like this:

```

Copy yarnhardhatmake-deal-proposal\ --contract0x0219eB1740C315fe5e20612D7E13AE2A883dB3f4\ --piece-cidbaga6ea4seaqn4eomxfk3ttog7lnvlvedu7nia377w4gotw2pm746k6kq7gwe6ga\ --piece-size2097152\ --verified-dealfalse\ --car-size1439368\ --labelbaga6ea4seaqn4eomxfk3ttog7lnvlvedu7nia377w4gotw2pm746k6kq7gwe6ga\ --start-epoch180000\ --end-epoch700000\ --storage-price-per-epoch0\ --provider-collateral0\ --client-collateral0\ --extra-params-version1\ --location-ref"https://bafybeidguwwno5ohjss7g4l6ygvyy3dzxxrkkgtxqkobnnxnu62aw4ipxa.ipfs.w3s.link/ipfs/bafybeidguwwno5ohjss7g4l6ygvyy3dzxxrkkgtxqkobnnxnu62aw4ipxa/baga6ea4seaqn4eomxfk3ttog7lnvlve
\ --skip-ipni-announce true \ --remove-unsealed-copy true \

```

Parameters such as thecollateral andprice-per-epoch are set to0 . On mainnet, these would be determined by storage providers, but since this is on the Calibration testnet, the storage providers should pick up the jobs even with these parameters set to0 .

Storage provider picks up the job

Now if you've invoked the task with all the correct parameters, the method will execute on-chain and emit an event that Boost storage providers will be listening for. If the deal is well-formed and the parameters are acceptable, they will download the.car file, double-check to ensure thepiece-cid andpiece-size match the deal, and publish your storage deal! This could take up to a day. Once the deal is published, you'll be able to find it on a Calibration testnetblock explorer . The client in the deal should be thet4 address of the smart contract we calledMakeStorageDeal on.

Monitoring deal proposal acceptance

After emitting an event and waiting for storage providers to accept your deal, you can monitor its status on a provided Boost logs dashboard. This feature is only available on the Calibration testnet. See this guide on GitHub for help diagnosing why deals might not be accepted and adjusting your proposal for re-submission.

Conclusion

During this tutorial, we have shown the significance of making deals using smart contracts and then walked through making a storage deal using the FVM deal-making kit and web3.storage. Developers can make use of this workflow to integrate decentralized storage on Filecoin with their smart contracts and decentralized applications.

Previous Filecoin.sol Next Using RaaS

Last updated4 months ago