Using RaaS

This page is a tutorial on how to replicate, renew and repair storage deals on Filecoin.

RaaS refers to replication, renewal and repair as a service, for data stored in storage deals on Filecoin. You can read morehere .

In this article, we will cover how to create storage deals, with replication, renew and repair requirements, via a smart contract. This utilizes theRaaS Starter Kit .

Interacting with the Smart Contract

First, you need to:

- EITHER start an instance of the BaseInterface by deploying a contract that inherits fromIAggregatorOracle
- (you can do so via.yarn deploy
- in theRaaS Starter Kit
-)
- OR use an existing instance of the FullInterface located at
-
    - Calibration Testnet:0x6ec8722e6543fB5976a547434c8644b51e24785b
- *
-

Interact with the smart contract by submitting a CID of your choice to thesubmit function. This will create a new deal request that will be picked up by the RaaS services.

```

Copy // contractInstance is the address of the contract you deployed or the FullInterface address above. constdealStatus=awaitethers.getContractAt("DealStatus",contractInstance); // Submit the CID of the file you want to upload to the Filecoin network in the following way. awaitdealStatus.submit(ethers.utils.toUtf8Bytes(newJob.cid));

```

TheRaaS Starter Kit provides you with a frontend that allows you to upload your file to Lighthouse, get a CID for the uploaded file, then seamlessly submit the CID to the smart contract (accessible viayarn start).

Before that, you need to know how to register the various RaaS workers. Note that RaaS functionality will NOT function automatically if deals are only created using submit function.

Add Replication, Renewal, Repair Workers

You can add workers to perform replication, renewal, and repair jobs by having them listen to theSubmitAggregatorRequest . The methods for doing so differ between the Base and Full interfaces.

If you are running a base interface (specifically, the one in theRaaS Starter Kit), there's an event listener inside the RaaS service node that you can use to listen for new deal requests.

This event listener performs processing for each job submitted to the contract to add RaaS service workers and eventually to callcomplete on the contract.

```

Copy // Initialize the listener for the Deal Creation event asyncfunctioninitializeDealCreationListener() { constdealStatus=awaitethers.getContractAt(contractName,contractInstance);

/// Logic for handling SubmitAggregatorRequest events functionhandleEvent(transactionId,cid) { console.log(Received SubmitAggregatorRequest event: (Transaction ID:{transactionId}, CID:{cid})); // ... other code to handle the event emission

(async()=>{ // ... other code

// After processing this event, reattach the event listener if(dealStatus.listenerCount("SubmitAggregatorRequest")===0) { dealStatus.once("SubmitAggregatorRequest",handleEvent); } })(); }

// Start listening to the first event and recursively handle the next events if(dealStatus.listenerCount("SubmitAggregatorRequest")===0) { dealStatus.once("SubmitAggregatorRequest",handleEvent); } }

```

To use this, callyarn service in the terminal and proceed through the frontend, as you normally would when uploading a file. Register the workers using the autocompleted CID that appears in the box.

If you want to register the workers manually for a job that you didn't upload, paste in the known CID of your file into the box and register the jobs.

Previous Direct deal-making with Client contract Next Verify a contract

Last updated4 months ago