Data replication, renewal and repair (RaaS)

Replicate, Renew, and Repair your storage deals with a service

RaaS refers to replication, renew and repair of storage deals. It is a feature of programmatic storage on Filecoin, enabled by FVM. This page covers the basic features of RaaS and handy tutorial to help you get started.

Introduction

What does it mean to replicate, renew, or repair data storage deals through a Filecoin smart contract?

- RaaS refers to replication, renewal and repair as a service, for data stored in storage deals on Filecoin:
- Replication refers to the option to store a user-defined number of replicas of your data.
- Renewal refers to the option to automatically observe on-chain storage deals until the expiry of their deal term, and automatically renew the deal.
- Repair refers to the automatic observation of storage deals, to ensure they are not in a faulted sector. If they are, these workers repair them automatically.
-

The motivation behind RaaS is to enable perpetual data storage on Filecoin.

“As a service" refers to the opportunity to provide these RaaS features as services, to incentivize clients to use them and/or storage platforms to enable them. These are a part of the Filecoin programmable storage market with FVM.

RaaS modules

There are two ways available to use with RaaS, for different purposes - the self-hosted RaaS where the Client runs all components and aggregator-hosted RaaS, which refers to a fully hosted RaaS service that is provided to the Client by aggregators.

At the RaaS, any contract or storage platform that has the metadata of the stored data, can request the RaaS node to perform its functions. However, the RaaS node has to be self-hosted in order to function. Storage platforms have the opportunity to build an aggregator-hosted RaaS node, to provide hosted replication, renewal, and repair services to clients.

A RaaS node, which monitors deals done through aggregators on Filecoin, will take action if replication, renew, or repair requirements are not observed.

RaaS nodes are only able to monitor deals from aggregators, since the nodes listen to the SubmitAggregatorRequest event in order to pick these deals up.

Here is a brief outline of the RaaS process:

1. An Aggregator sends a CID to the RaaS node, requesting for replication, renewal and/or repair as needed.
2. The RaaS smart contracts maintains the information of deals created by the RaaS, including deal_id and miner_id.
3. The RaaS node periodically checks the deal status on the Filecoin network, by interacting with the RaaS` smart contract.
4. If the deal requires replication, renewal and/or repair, the RaaS node resubmit deals to aggregators and requests the aggregator to make a new deal.
5.

Tutorial

We're now going to quickly cover how to create storage deals with replication, renew and repair requirements using smart contracts. 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 theFullInterface
- located at0x6ec8722e6543fB5976a547434c8644b51e24785b
- . This instance is on the Calibration testnet only, for the moment.

-

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.

Replicate, renew, and repair

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 Next RaaS interfaces

Last updated4 months ago