RaaS interfaces

Specifications for RaaS interfaces.

RaaS has a base and full interface to enable replication, renew and repair of storage deals.

Self-hosted RaaS

RaaS refers to replication, renewal and repair as a service, for data stored in storage deals on Filecoin. Developers can leverage the self-hosted RaaS to provide RaaS features, within their storage solution, using the RaaS Starter Kit.

Self-hosted RaaS requires 4 components:

- The Client who has data to upload
- An Aggregator platform (a type of storage solution) that receives the Client's data and makes a storage deal on Filecoin
- The RaaS node, hosted by the developer, that checks if the storage deal requires replication, renewal and/or repair
- The RaaS DealStatus smart contract that the RaaS node executes checks with
-

In the example of replication:

1. The client generates aCID
2. for the data and requests the RaaS node to store data.
3. The RaaS node takes the client's data and makes the storage deal onto Filecoin.
4. The client registers a replication job to the RaaS node and defines the number of replicas of their data to store, by calling the/register_jobs
5. API on the RaaS node (e.g. "This data needs to have a minimum of 10 copies on the network").
6. The RaaS node periodically checks the client data'sCID
7. for its deal status on Filecoin.
8.
    - The RaaS node requests deal status with the DealStatus smart contract, viagetActiveDeals(CID)
9.
    - and checks if the client's data is stored with the accurate number of replicas.
10.
    - The DealStatus smart contract returns the information of active deals to the RaaS node.
11. *
12. If the number of replicas does not match the client's requirements, the RaaS node is notified.
13. The RaaS node fetches the data via itsCID
14. and resubmit a request to create new storage deals (repeat step 2).
15. When the client requests for retrieval of data, it queries the RaaS node, which will fetch the data from the storage provider on Filecoin or provide an IPFS pinned copy (depends on how RaaS node is setup to store the data).
16.

In the example of renewal:

1. The client generates aCID
2. for the data and requests the RaaS node to store data.
3. The RaaS node takes the client's data and makes the storage deal onto Filecoin.
4. The client registers a renewal job to the RaaS node and defines the renewal threshold for the data's storage deal, by calling the/register_jobs
5. API on the RaaS node (e.g. renew storage deals that are 1 month away from expiry).
6. The RaaS node periodically checks the client data'sCID
7. for its deal status on Filecoin.
8.
    - The RaaS node requests deal status with the DealStatus smart contract, viagetExpiringDeals(CID)
9.
    - and checks if any of its active deals is expiring.
10.
    - The DealStatus smart contract returns the information of expiring deals to the RaaS node.
11. *
12. If deals with the client's data are expiring, the RaaS node is notified.
13. The RaaS node fetches the data via itsCID
14. and resubmit a request to create new storage deals (repeat step 2).
15. When the client requests for retrieval of data, it queries the RaaS node, which will fetch the data from the storage provider on Filecoin or provide an IPFS pinned copy (depends on how RaaS node is setup to store the data).
16.

In the example of repair

1. The client generates aCID
2. for the data and requests the RaaS node to store data.
3. The RaaS node takes the client’s data and makes the storage deal onto Filecoin.
4. The client registers a repair job to the RaaS node and defines the repair threshold for the data’s storage deal, by calling the/register_jobs
5. API on the RaaS node (e.g. “this deal needs repairing if it is not proven active for X epochs”).
6. The RaaS node periodically checks withLotus
7. , if the deal ID and corresponding miner, is actively being proven by the miner on Filecoin. The node callsStateMarketStorageDeal
8. , with provided deal and miner IDs.
9. If the deal ID and corresponding deal sector are not being actively proven for X epochs, the deal will require repairing.
10. The RaaS node fetches the data via itsCID
11. and resubmit a request to create new storage deals (repeat step 2).
12.

Check out theRaaS starter kit to build your own RaaS interface.

Aggregator-hosted RaaS

The aggregator-hosted RaaS refers to a solution that requires an aggregator to host all the components of RaaS and provides a seamless interface to the client.Lighthouse.storage is the first FVM project to provide an aggregator-hosted RaaS, via its SDK.

Previous Data replication, renewal and repair (RaaS) Next Developing contracts

Last updated5 months ago