Built-in actors

Built-in actors are how the Filecoin network manages and updates global state. This page contains information on how smart contracts can access built-in actors.

Built-in actors are how the Filecoin network manages and updatesglobal state . Theglobal state of the network at a given epoch can be thought of as the set of blocks agreed upon via network consensus in that epoch. This global state is represented as astate tree , which maps an actor to anactor state . Anactor state describes the current conditions for an individual actor, such as its FIL balance and its nonce. In Filecoin, actors trigger astate transition by sending amessage . Each block in the chain can be thought of as aproposed global state, where the block selected by network consensus sets thenew global state. Each block contains a series of messages, and a checkpoint of the current global state after the application of those messages. The Filecoin Virtual Machine (FVM) is the Filecoin network component that is in charge of execution of all actor code.

A basic example of how built-in actors are used in Filecoin is the process by which storage providers prove storage and are subsequently rewarded. The process is as follows:

1. TheStorageMinerActor
2. processes proof of storage from a storage provider.
3. The storage provider is awarded storage power based on whether the proof is valid or not.
4. TheStoragePowerActor
5. accounts for the storage power.
6. During block validation, theStoragePowerActor
7. ’s state, which includes information on storage power allocated to each storage provider, is read.
8. Using the state information, the consensus mechanism randomly awards blocks to the storage providers with the most power, and theRewardActor
9. sends FIL to storage providers.
10.

Blocks

Each block in the Filecoin chain contains:

- Inline data such as current block height.
- A pointer (CID) to the current state tree.
- A pointer (CID) to the set of messages that, when applied to the network, generated the current state tree.
-

State tree

A Merkle Directed Acyclic Graph (Merkle DAG) is used to map the state tree. and the set of messages. Nodes in the state tree contain information on:

- Actors, like FIL balance, nonce and a pointer (CID) to actor state data.
- Messages in the current block
-

Messages

Like the state tree, a Merkle Directed Acyclic Graph (Merkle DAG) is used to map the set of messages for a given block. Nodes in the messages map contain information on:

- The actor the message was sent to
- The actor that sent the message
- Target method to call on actor being sent the message
- A cryptographic signature for verification
- The amount of FIL transferred between actors
-

Actor code

The code that defines an actor in the Filecoin network is separated into different methods. Messages sent to an actor contain information on which method(s) to call, and the input parameters for those methods. Additionally, actor code interacts with aruntime object, which contains information on the general state of network, such as the current epoch, and cryptographic signatures and proof validations. Like smart contracts in other blockchains, actors must pay agas fee , which is some predetermined amount of FIL to offset the cost (network resources used, etc.) of a transaction. Every actor has a Filecoin balance attributed to it, a state pointer, a code which tells the system what type of actor it is, and a nonce, which tracks the number of messages sent by this actor

Types of built-in actors

The 11 different types of built-in actors are as follows:

- CronActor
- InitActor
- AccountActor
- RewardActor
- StorageMarketActor
- StorageMinerActor
- MultisigActor
- PaymentChannelActor
- StoragePowerActor
- VerifiedRegistryActor
- SystemActor
-

CronActor

TheCronActor sends messages to theStoragePowerActor andStorageMarketActor at the end of each epoch. The messages sent byCronActor indicate to StoragePowerActor and StorageMarketActor how they should maintain internal state and process deferred events. This system actor is instantiated in the genesis block, and interacts directly with the FVM.

InitActor

TheInitActor can initialize new actors on the Filecoin network. This system actor is instantiated in the genesis block, and maintains a table resolving a public key and temporary actor addresses to their canonical ID addresses. TheInitActor interacts directly with the FVM.

AccountActor

TheAccountActor is responsible for user accounts. Account actors are not created by theInitActor , but by sending a message to a public-key style address. The account actor updates the state tree with new actor address, and interacts directly with the FVM.

RewardActor

TheRewardActor manages unminted Filecoin tokens, and distributes rewards directly to miner actors, where they are locked for vesting. The reward value used for the current epoch is updated at the end of an epoch. TheRewardActor interacts directly with the FVM.

StorageMarketActor

TheStorageMarketActor is responsible for processing and managing on-chain deals. This is also the entry point of all storage deals and data into the system. This actor keeps track of storage deals, and the of locked balances of both the client storing data and the storage provider. When a deal is posted on chain through theStorageMarketActor , the actor will first check if both transacting parties have sufficient balances locked up and include the deal on chain. Additionally, theStorageMarketActor holdsStorage Deal Collateral provided by the storage provider to collateralize deals. This collateral is returned to the storage provider when all deals in the sector successfully conclude. This actor does not interact directly with the FVM.

StorageMinerActor

TheStorageMinerActor is created by theStoragePowerActor , and is responsible for storage mining operations and the collection of mining proofs. This actor is a key part of the Filecoin storage mining subsystem, which ensures a storage miner can effectively commit storage to the Filecoin, handles the following:

- Committing new storage
- Continuously proving storage
- Declaring storage faults
- Recovering from storage faults
-

This actor does not interact directly with the FVM.

MultisigActor

TheMultisigActor is responsible for dealing with operations involving the Filecoin wallet, and represents a group of transaction signers, with a maximum of 256. Signers may be external users or theMultisigActor itself. This actor does not interact directly with the FVM.

PaymentChannelActor

ThePaymentChannelActor creates and managespayment channels , a mechanism for off-chain microtransactions for Filecoin dApps to be reconciled on-chain at a later time with less overhead than a standard on-chain transaction, and no gas costs. Payment channels are uni-directional and can be funded by adding to their balance. To create a payment channel and deposit fund, a user calls thePaymentChannelActor . This actor does not interact directly with the FVM.

StoragePowerActor

TheStoragePowerActor is responsible for keeping track of the storage power allocated to each storage miner, and has the ability to create aStorageMinerActor . This actor does not interact directly with the FVM.

VerifiedRegistryActor

TheVerifiedRegistryActor is responsible for managing Filecoin Plus (Fil+) clients. This actor can add a verified client to the Fil+ program, remove and reclaim expired DataCap allocations and manage claims. This actor does not interact directly with the FVM.

SystemActor

For more information onSystemActor , see thesource code .

Access and invoke built-in actors

- Protocol API
- , maintained byProtocol Labs
- , …
- Filecoin.solidity
- , maintained byProtocol Labs
- , is a set of libraries that allows Solidity smart contracts to seamlessly call built-in actors methods.Not all built-in actors and methods are supported
-
    - for a complete list, see theactors and methods supported
- .
-

For information on how invoke and access built-in actors in your smart contracts, see thedevelopers guide .

Last updated5 months ago