IPC CLI

This page lists all the ipc-cli commands to interact with IPC subnets. For background and setup information, make sure to start with the README of IPC on GitHub. If you have IPC installed on your machine, you should be able to runipc-cli --help to check all the available commands.

```

Copy ipc-cli--help

TheIPCagentcommandlinetool

Usage:ipc-cli[OPTIONS] [COMMAND]

Commands: configconfigrelatedcommands subnetsubnetrelatedcommandssuchascreate,joinandetc walletwalletrelatedcommands cross-msgcrossnetworkmessagesrelatedcommands checkpointcheckpointrelatedcommands utilutilcommands helpPrintthismessageorthehelpofthegivensubcommand(s)
```

Configuration

IPC initialization

```

Copy ipc-cliconfiginit
```

This command will initialize a new empty config file under~/.ipc with all parameters required to connect to the IPC rootnet network.

Subnet Management

List active subnets

```

Copy ipc-cli subnet list --subnet=
```

You can check all the active child subnets on a specific parent subnet.

This command only shows subnets that have been registered to the gateway, i.e. that have provided enough collateral to participate in the IPC protocol and haven't been killed. It is not an exhaustive list of all of the subnet actors deployed over the network.

```

Copy
```

Example execution

ipc-clisubnetlist--subnet/r314159 /r314159/t410fmdbc3kcv4gody6drgztmgwnzs2ryzwiazjzu5pq-status:Active,collateral:5.0FIL,circ.supply:21.0FIL,genesis:1069882

```

Create a child subnet

```

Copy ipc-cli subnet create --parent--min-validator --min-validator-stake --bottomup-check-period

```

This command will create a subnet and create a corresponding contract based on the parameters specified with it. Make a note of the subnet-id for the subnet just created.

```

Copy
```

# Example execution

ipc-clisubnetcreate--parent/r314159--min-validators3--min-validator-stake10--bottomup-check-period30 [ipc_cli::commands::subnet::create] created subnet actor with id: /r314159/t410fylyzufn7lfg3q6zxpt6cdvq4yyiwm4tkaged2oy

```

Join a subnet as a validator

```

Copy ipc-clisubnetjoin --subnet --collateral --public-key --initial-balance

```

This command specifies the subnet to join, the amount of collateral to provide, and the public key of the--from address that is joining as a validator.

```

Copy
```

Example execution

ipc-clisubnetjoin--subnet/r314159/t410fh4ywg4wvxcjzz4vsja3uh4f53johc2lf5bpjo6i--collateral10--public-key043385c3b9ab8a697cd7bec6ca623cbdd0fea1293e8b464df825b104eb58a44cc8efacc6a3482b866b85ecdf734b5d4ef5495737deb348625ce6a35536142d2955

```

To join a subnet and also include some initial balance for the validator in the subnet, you can add the--initial-balance flag with the balance to be included in the genesis.

```

Copy

Example execution

ipc-clisubnetjoin--subnet=/r314159/t410fh4ywg4wvxcjzz4vsja3uh4f53johc2lf5bpjo6i--collateral=1--public-key=043385c3b9ab8a697cd7bec6ca623cbdd0fea1293e8b464df825b104eb58a44cc8efacc6a3482b866b85ecdf734b5d4ef5495737deb348625ce6a35536142d2955--initial-balance0.5

```

Leave a subnet

```

Copy ipc-clisubnetleave--subnet

```

Leaving a subnet will release the collateral for the validator and remove all the validation rights from its account. This means that if you have a validator running in that subnet, its validation process will immediately terminate.

```

Copy

Example execution

ipc-clisubnetleave--subnet/r31415926/t4xwzbdu7z5sam6hc57xxwkctciuaz7oe5omipwbq

```

Stake more collateral as a validator

```

Copy ipc-cli subnet stake --subnet--collateral

```

After initially joining a subnet with specified collateral, a validator can stake more collaterals to the subnet.

```

Copy

Example execution

ipc-clisubnetstake--subnet=/r314159/t410fh4ywg4wvxcjzz4vsja3uh4f53johc2lf5bpjo6i--collateral=10 ipc_provider::manager::evm::manager]interactingwithevmsubnetcontract:0x60c2…d900withcollateral:10000000000000000000

```

Unstake collateral from a subnet

```

Copy ipc-clisubnetunstake--subnet--collateral

Example execution

ipc-clisubnetunstake--subnet/r314159/t410fmdbc3kcv4gody6drgztmgwnzs2ryzwiazjzu5pq--collateral2

```

Claim the unstaked collateral

```

Copy ipc-cli subnet claim --subnet

```

Validators need to claim their collateral after they reduce collateral in the subnet throughunstake

```

Copy

Example execution

ipc-clisubnetclaim--subnet=/r314159/t410fh4ywg4wvxcjzz4vsja3uh4f53johc2lf5bpjo6i

```

Changes in collateral and the power table are not reflected immediately in the parent. They need to be confirmed in the execution of the next bottom-up checkpoint, so until this happens, even if there has been a change in collateral, you may not be the change immediately when runningipc-cli subnet list . This impacts any change to the collateral of validators, i.e.stake ,unstake andleave commands.

To inspect the changes to the power table that have been performed between two epochs you can use the following command:

```

Copy ipc-cli checkpoint list-validator-changes --from-epoch=--to-epoch=

```

Transfer tokens within a subnet

```

Copy ipc-clisubnetsend-value --subnet[--from] --to

```

You can use this command to send tokens between addresses of the same subnet. If--from is not specified,ipc-cli will send tokens from the default wallet address.

```

Copy

Example execution

ipc-clisubnetsend-value--subnet/r31415926/t4xwzbdu7z5sam6hc57xxwkctciuaz7oe5omipwbq--to0x406a7a1d002b71ece175cc7e067620ae5b58e9ec10

```

Wallet Key Management

Theipc-cli has an EVM-compatible wallet that it uses to sign transactions and interact with IPC on behalf of specific addresses. This wallet type can also be used with FEVM.

Create new address

```

Copy ipc-cli wallet new -w

```

This command will create a wallet, and store the key information in~/.ipc/evm_keystore.json . You can create an EVM-compatible wallet which can also used with FVM (Filecoin Virtual Machine).

```

Copy

Sample execution

ipc-cliwalletnew--wallet-typeevm "0x406a7a1d002b71ece175cc7e067620ae5b58e9ec"

```

Check wallet balance

```

Copy ipc-cliwalletbalances--wallet-type--subnet

```

You can check the token balance in your wallet addresses for any active subnet configured in theipc-cli .

```

Copy

Sample execution

ipc-cliwalletbalances--wallet-typeevm--subnet/r314159 0x406a7a1d002b71ece175cc7e067620ae5b58e9ec-Balance:100

```

Set default wallet address

```

Copy ipc-cli wallet set-default --wallet-type--address

```

You can set a default address for your wallet so it is always the one used when the--from flag is not explicitly set.

```

Copy

Sample execution

ipc-cliwalletset-default--wallet-typeevm--address0x406a7a1d002b71ece175cc7e067620ae5b58e9ec

```

Get the default wallet address

```

Copy ipc-cliwalletget-default--wallet-type

Sample execution

ipc-cliwalletset-default--wallet-typeevm "0x406a7a1d002b71ece175cc7e067620ae5b58e9ec"

```

Export a wallet key

```

Copy ipc-cliwalletexport--wallet-type--address>

```

This command will exporte a wallet private key which is stored in theipc-cli keystore~/.ipc/evm_keystore.json .

```

Copy

Sample execution

ipc-cliwalletexport--wallet-typeevm--address0x406a7a1d002b71ece175cc7e067620ae5b58e9ec>/tmp/priv.key
exportednewwalletwithaddress0x406a7a1d002b71ece175cc7e067620ae5b58e9ecinfile"/tmp/priv.key"

```

- Export key encoded in based64 for Fendermint
- ```
- Copy
- ipc-cliwalletexport--wallet-typeevm--address--fendermint>
- ```
- Export key in HEX
- ```
- Copy
- ipc-cliwalletexport--wallet-typeevm--address--hex>
- ```
-

Import a wallet

```

Copy ipc-cli wallet import --wallet-type evm --path--private-key

```

This command will import a wallet from an EVM key file with this format{“address”:,“private_key”} .

```

Copy

Sample execution

ipc-cliwalletimport--wallet-typeevm--path=~/tmp/wallet.key importedwalletwithaddress"0x406a7a1d002b71ece175cc7e067620ae5b58e9ec"

```

Import a wallet from the private key.

```

Copy

Sample execution

ipc-cliwalletimport--wallet-typeevm--private-key=0x405f50458008edd6e2eb2efc3bf34846db1d6689b89fe1a9f9ccfe7f6e301d8d importedwalletwithaddress"0x406a7a1d002b71ece175cc7e067620ae5b58e9ec"

```

Cross subnet messages

At the moment, theipc-cli only expose commands to perform the basic IPC interoperability primitives for cross-net communication, which is the exchange of FIL (the native token for IPC) between the same address of a subnet. Mainly:

- fund
- , which sends native tokens from one public key address to the address in the child subnet.
- release
- that moves native tokens from one account in a child subnet to its counterpart in the parent.
-

Fund tokens in a child subnet

```

Copy ipc-clicross-msgfund --subnet [--from] [--to]

```

This command includes the cross-net message into the next top-down proof-of-finality. Once the top-down finality is committed in the child, the message will be executed and you should see the funds in your account of the child subnet. If the--to is not set explicitly, the funds are sent to the address of the--from in the subnet.

```

Copy

Example execution

ipc-clicross-msgfund--subnet/r31415926/t4xwzbdu7z5sam6hc57xxwkctciuaz7oe5omipwbq100 fundperformedinepoch1030279

```

Alternatively, we can pass an additional parameter to send the funds to a specific address in the child subnet.

```

Copy

Example execution

ipc-clicross-msgfund--subnet/r31415926/t4xwzbdu7z5sam6hc57xxwkctciuaz7oe5omipwbq--to=0x406a7a1d002b71ece175cc7e067620ae5b58e9ec100 fundperformedinepoch1030279

```

Pre-fund subnet address in genesis

```

Copy ipc-cli cross-msg pre-fund --subnet[--from] she

```

To fund your address in a child subnet genesis before it is bootstrapped, and include some funds on your address in the subnet in genesis, you can use thepre-fund command. This command can only be used before the subnet is bootstrapped and started.

```

Copy

Example execution

./bin/ipc-clicross-msgpre-fund--subnet/r31415926/t4xwzbdu7z5sam6hc57xxwkctciuaz7oe5omipwbq1

```

Release funds from a subnet

```

Copy ipc-clicross-msgrelease --subnet [--from] [--to]

```

This command will release funds to the parent subnet from its subnet. To release funds from a subnet, your account must hold enough funds inside this subnet.

This command includes the cross-net message into a bottom-up checkpoint after the current epoch. Once the bottom-up checkpoint is committed in the parent, you should see the funds in your account in the parent.

```

Copy

Example execution

ipc-clicross-msgrelease--subnet/r31415926/t4xwzbdu7z5sam6hc57xxwkctciuaz7oe5omipwbq100 releaseperformedinepoch1023

```

Alternatively, we can pass an additional parameter to release the funds to a specific address in the parent subnet by setting--to address.

```

Copy

Example execution

ipc-clicross-msgrelease--subnet/r31415926/t4xwzbdu7z5sam6hc57xxwkctciuaz7oe5omipwbq--to0x406a7a1d002b71ece175cc7e067620ae5b58e9ec100
releaseperformedinepoch1030

```

Release initial subnet funds

```

Copy ipc-clicross-msgpre-release --subnet [--from]

```

This command will recover some (or all) of the funds that were sent to a subnet throughpre-fund to be included as a genesis balance for your address.

```

Copy

Example execution

ipc-clicross-msgpre-release--subnet/r31415926/t4xwzbdu7z5sam6hc57xxwkctciuaz7oe5omipwbq0.1

```

Check parent subnet finality

```

Copy ipc-cli cross-msg parent-finality --subnet

```

The epoch in which the message is performed can give you a sense of the time the message will take to be propagated. You can check the current finality in a subnet and wait for the finality height that includes your message to be committed.

```

Copy

Example execution

ipc-clicross-msgparent-finality--subnet/r314159/t410fmdbc3kcv4gody6drgztmgwnzs2ryzwiazjzu5pq
ipc_provider::manager::evm::manager]queryinglatestparentfinality 1070541

```

List top-down messages

```

Copy ipc-clicross-msglist-topdown-msgs--subnet=--epoch=

```

This command will list the top-down messages sent for a subnet from a parent network for a specific epoch.

```

Copy

Example execution

ipc-clicross-msglist-topdown-msgs--subnet/r314159/t410fmdbc3kcv4gody6drgztmgwnzs2ryzwiazjzu5pq--epoch100450

```

CheckPoint

List checkpoints for a subnet
```

```

Copy ipc-clicheckpointlist-bottomup --from-epoch --to-epoch --subnet

```

Subnets are periodically committing checkpoints to their parent everybottomup-check-period (parameter defined when creating the subnet). You can use this command to inspect the information of a range of bottom-up checkpoints committed in the parent for a subnet.

```

Copy

# Example execution

ipc-clicheckpointlist-bottomup--from-epoch0--to-epoch100--subnet/r31415926/t4xwzbdu7z5sam6hc57xxwkctciuaz7oe5omipwbq epoch0-prev_check=
{"/":"bafy2bzacedkoa623kvi5gfis2yks7xxjl73vg7xwbojz4tpq63dd5jpfz757i"},cross_msgs=null,child_checks=null epoch10-prev_check=
{"/":"bafy2bzacecsatvda6lodrorh7y7foxjt3a2dexxx5jiyvtl7gimrrvywb7l5m"},cross_msgs=null,child_checks=null epoch30-prev_check=
{"/":"bafy2bzaceauzdx22hna4e4cqf55jqmd64a4fx72sxprzj72qhrwuxhdl7zexu"},cross_msgs=null,child_checks=null

```

Check quorum-reached bottom-up checkpoints

```

Copy ipc-cli checkpoint quorum-reached-events --from-epoch--to-epoch --subnet

```

As with bottom-up messages, you can get a sense of the time that your message will take to get to the parent by looking at the epoch in which your bottom-up message was triggered (the output of the command) and listing the latest bottom-up checkpoints to see how far it is from being propagated.

This command will list the bottom-up checkpoints populated, signed, and agreed on their validity by a majority of validators in the child subnet.

```

Copy

# Sample execution

ipc-clicheckpointquorum-reached-events--from-epoch600--to-epoch680--subnet/r314159/t410ffumhfeppdjixhkxtgagowxkdu77j7xz5aaa52vy

```

Check if bottom-up checkpoints are submitted

```

Copy ipc-cli checkpoint has-submitted-bottomup-height --subnet--submitter

```

This command can be used to check the state of the checkpoints submitted from a subnet relayer. Once subnet validators have agreed on the bottom-up checkpoint to be submitted in the parent for a specific epoch, relayers need to pick up the checkpoint and submit it in the parent.

```

Copy

# Sample execution

ipc-clicheckpointhas-submitted-bottomup-height--subnet/r314159/t410ffumhfeppdjixhkxtgagowxkdu77j7xz5aaa52vy--
submitter0x406a7a1d002b71ece175cc7e067620ae5b58e9ec

```

List submitted bottom-up checkpoints

```

Copy ipc-cli checkpoint list-bottomup-bundle --subnet--from-epoch --to-epoch

```

This command can be used to check the list of the bundle of bottom-up checkpoints and signatures populated and already signed by a child subnet for their submission to the parent on a window of heights.

```

Copy

# Sample execution

ipc-clicheckpointlist-bottomup-bundle--subnet/r314159/t410ffumhfeppdjixhkxtgagowxkdu77j7xz5aaa52vy--from-epoch600--to-epoch680

```

Run a relayer

```

Copy ipc-cli checkpoint relayer --subnet--submitter

```

IPC relies on the role of a specific type of peer on the network called therelayers that are responsible for submitting bottom-up checkpoints that have been finalized in a child subnet to its parent. Without relayers, cross-net messages will only flow from the top levels of the hierarchy to the bottom, but not the other way around.

```

Copy

# Example execution

ipc-clicheckpointrelayer--subnet/r31415926/t4xwzbdu7z5sam6hc57xxwkctciuaz7oe5omipwbq

```

To run the relayer from a different address you can use the--submitted flag.

```

Copy

# Example execution

ipc-clicheckpointrelayer--subnet/r31415926/t4xwzbdu7z5sam6hc57xxwkctciuaz7oe5omipwbq--submitter0x406a7a1d002b71ece175cc7e067620ae5b58e9ec

```

Relayers are rewarded through cross-net message fees for the timely submission of bottom-up checkpoints to the parent. Relayers can claim the checkpointing rewards collected for a subnet.

```

Copy

# Example execution

ipc-clisubnetclaim--subnet=/r31415926/t4xwzbdu7z5sam6hc57xxwkctciuaz7oe5omipwbq--reward

```

Previous Networks Next Troubleshooting Last updated26 days ago On this page *Configuration * Subnet Management * Wallet Key Management * Cross subnet messages * CheckPoint

Was this helpful? Edit on GitHub