Exit Message Generation & Signing

Keystores or Dirk

If your validator signing keys are in keystores or in Dirk remote keymanager, the easiest method is to use ethdo .

For Keystores:

1. Create an ethdo wallet
2. Import keystores
3. Generate an exit
4. Erase the wallet if it's no longer needed

Create a new wallet:

```
./ethdo --base-dir=./temp wallet create --wallet=wallet
```

Add key from a keystore:

```
./ethdo --base-dir=./temp account import --account=wallet/account --keystore=./ethdo/keystore.json --keystore-passphrase=12345678 --passphrase=pass
```

Generate and sign an exit message:

```
./ethdo --base-dir=./temp validator exit --account=wallet/account --passphrase=pass --json --connection=http://consensus_node:5052
```

ethdo will print out the exit message to stdout. You can save the fileethdo ... > 0x123.json .

After we are done, delete the wallet:

```
./ethdo --base-dir=./temp wallet delete --wallet=wallet
```

If you are looking for a way to automate the process, check out this example .

info Although keystores are encrypted, it is highly recommended to interact with them in a secure environment without internet access. ethdo allows you to prepare everything necessary for offline exit message generation in one convenient file. For this, on a machine with access to a Consensus Node run:

```
./ethdo validator exit --prepare-offline --connection=http://consensus_node:5052 --timeout=300s
```

This command will pull validators info, fork versions, current epoch and other chain data for offline exit message generation and save it tooffline-preparation.json in theethdo directory.

This file can be then transferred to a secure machine along withethdo binary, for example on a encrypted USB drive.

On the secure machine, putoffline-preparation.json into the directoryethdo is ran from, use--offline argument for thevalidator exit command and remove--connection :

```
./ethdo --base-dir=./temp validator exit --account=wallet/account --passphrase=pass --json --offline
```

For Dirk:

```
./ethdo --remote=server.example.com:9091 --client-cert=client.crt --client-key=client.key --server-ca-cert=dirk_authority.crt validator exit --account=Validators/1 --json --connection=http://127.0.0.1:5051
```

ethdo ethdo Docs

For Web3Signer or Proprietary Signers

If you are using the/api/v1/modules/{module_id}/validators/generate-unsigned-exit-messages/{operator_id} endpoint of the KAPI, you can skip getting the epoch and constructing an unsigned exit message in the example below.

Get current epoch:

```
const blockReq =

await

fetch ( CONSENSUS_BLOCK_ENDPOINT ) const blockRes =

await blockReq . json ( ) const blockNumber = blockRes . data . message . slot const currentEpoch =

Math . floor ( blockNumber /

32 )
```

Get fork parameters:

```
const forkReq =
```

await

fetch (CONSENSUS_FORK_ENDPOINT) const forkRes =

await forkReq . json ( ) const fork = forkRes . data Get genesis parameters:

const genesisReq =

await

fetch (CONSENSUS_GENESIS_ENDPOINT) const genesisRes =

await genesisReq . json ( ) const genesis_validators_root = genesisRes . data . genesis_validators_root Construct an exit message:

const voluntaryExit =

{ epoch :

String (currentEpoch) , validator_index :

String (VALIDATOR_INDEX) , } Prepare a signing request:

const body =

{ type :

'VOLUNTARY_EXIT' , fork_info :

{ fork , genesis_validators_root , } , voluntary_exit : voluntaryExit , } Send the request:

const signerReq =

await

fetch (WEB3SIGNER_ENDPOINT ,

{ method :

'POST' , headers :

{

'Content-Type' :

'application/json' ,

Accept :

'application/json'

} , body :

JSON . stringify (body) , }) const signature =

await signerReq . text ( ) Finally, construct a signed exit message:

const signedMessage =

{ message : voluntaryExit , signature , }Complete Example

info It's advised to prepare all the necessary parameters (forks, epoch, etc) ahead of time and communicate with Web3Signer securely, for example via a secure network with no other internet access. Web3Signer API Docs Edit this page

Previous General Information Next Flow Examples