# CLI reference

caution Thecharon client is under heavy development, interfaces are subject to change until a first major version is published. The following is a reference for charon versionv0.19.1 . Find the latest release onour Github .

The following are the top-level commands available to use.

charon --help Charon enables the operation of Ethereum validators in a fault tolerant manner by splitting the validating keys across a group of trusted parties using threshold cryptography.

Usage: charon [command]

Available Commands: alpha Alpha subcommands provide early access to in-development features combine Combines the private key shares of a distributed validator cluster into a set of standard validator private keys. completion Generate the autocompletion script for the specified shell create Create artifacts for a distributed validator cluster dkg Participate in a Distributed Key Generation ceremony enr Prints a new ENR for this node help Help about any command relay Start a libp2p relay server run Run the charon middleware client version Print version and exit

Flags: -h, --help Help for charon

Use "charon [command] --help" for more information about a command.

## Thecreate

subcommand

Thecreate subcommand handles the creation of artifacts needed by charon to operate.

charon create --help Create artifacts for a distributed validator cluster. These commands can be used to facilitate the creation of a distributed validator cluster between a group of operators by performing a distributed key generation ceremony, or they can be used to create a local cluster for single operator use cases.

Usage: charon create [command]

Available Commands: cluster Create private keys and configuration files needed to run a distributed validator cluster locally dkg Create the configuration for a new Distributed Key Generation ceremony using charon dkg enr Create an Ethereum Node Record (ENR) private key to identify this charon client

Flags: -h, --help Help for create

Use "charon create [command] --help" for more information about a command.

### Creating an ENR for charon

Anenr is an Ethereum Node Record. It is used to identify this charon client to its other counterparty charon clients across the internet.

charon create enr --help Create an Ethereum Node Record (ENR) private key to identify this charon client

Usage: charon create enr [flags]

Flags: --data-dir string The directory where charon will store all its internal data (default ".charon") -h, --help Help for enr

### Create a full cluster locally

charon create cluster creates a set of distributed validators locally, including the private keys, acluster-lock.json file, and deposit and exit data. However, this command should only be used for solo use of distributed validators. To run a Distributed Validator with a group of operators, it is preferable to create these artifacts using thecharon dkg command. That way, no single operator custodies all of the private keys to a distributed validator.

charon create cluster --help Creates a local charon cluster configuration including validator keys, charon p2p keys, cluster-lock.json and a deposit-data.json. See flags for supported features.

Usage: charon create cluster [flags]

Flags: --cluster-dir string The target folder to create the cluster in. (default "./") --definition-file string Optional path to a cluster definition file or an HTTP URL. This overrides all other configuration flags. --fee-recipient-addresses strings Comma separated list of Ethereum addresses of the fee recipient for each validator. Either provide a single fee recipient address or fee recipient addresses for each validator. -h, --help Help for cluster --insecure-keys Generates insecure keystore files. This

should never be used. It is not supported on mainnet. --keymanager-addresses strings Comma separated list of keymanager URLs to import validator key shares to. Note that multiple addresses are required, one for each node in the cluster, with node0's keyshares being imported to the first address, node1's keyshares to the second, and so on. --keymanager-auth-tokens strings Authentication bearer tokens to interact with the keymanager URLs. Don't include the "Bearer" symbol, only include the api-token. --name string The cluster name --network string Ethereum network to create validators for. Options: mainnet, goerli, gnosis, sepolia, holesky. --nodes int The number of charon nodes in the cluster. Minimum is 3. --num-validators int The number of distributed validators needed in the cluster. --publish Publish lock file to obol-api. --publish-address string The URL to publish the lock file to. (default "https://api.obol.tech") --split-existing-keys Split an existing validator's private key into a set of distributed validator private key shares. Does not re-create deposit data for this key. --split-keys-dir string Directory containing keys to split. Expects keys in keystore- * .json and passwords in keystore- * .txt. Requires --split-existing-keys. --testnet-chain-id uint Chain ID of the custom test network. --testnet-fork-version string Genesis fork version of the custom test network (in hex). --testnet-genesis-timestamp int Genesis timestamp of the custom test network. --testnet-name string Name of the custom test network. --threshold int Optional override of threshold required for signature reconstruction. Defaults to ceil(n*2/3) if zero. Warning, non-default values decrease security. --withdrawal-addresses strings Comma separated list of Ethereum addresses to receive the returned stake and accrued rewards for each validator. Either provide a single withdrawal address or withdrawal addresses for each validator.

### Creating the configuration for a DKG Ceremony

Thischaron create dkg command creates a cluster_definition file used for thecharon dkg command.

charon create dkg --help Create a cluster definition file that will be used by all participants of a DKG.

Usage: charon create dkg [flags]

Flags: --dkg-algorithm string DKG algorithm to use; default, frost (default "default") --fee-recipient-addresses strings Comma separated list of Ethereum addresses of the fee recipient for each validator. Either provide a single fee recipient address or fee recipient addresses for each validator. -h, --help Help for dkg --name string Optional cosmetic cluster name --network string Ethereum network to create validators for. Options: mainnet, goerli, gnosis, sepolia, holesky. (default "mainnet") --num-validators int The number of distributed validators the cluster will manage (32ETH staked for each). (default 1) --operator-enrs strings [REQUIRED] Comma-separated list of each operator's Charon ENR address. --output-dir string The folder to write the output cluster-definition.json file to. (default ".charon") -t, --threshold int Optional override of threshold required for signature reconstruction. Defaults to ceil(n*2/3) if zero. Warning, non-default values decrease security. --withdrawal-addresses strings Comma separated list of Ethereum addresses to receive the returned stake and accrued rewards for each validator. Either provide a single withdrawal address or withdrawal addresses for each validator.

## Thedkg

subcommand

### Performing a DKG Ceremony

Thecharon dkg command takes acluster_definition.json file that instructs charon on the terms of a new distributed validator cluster to be created. Charon establishes communication with the other nodes identified in the file, performs a distributed key generation ceremony to create the required threshold private keys, and signs deposit data for each new distributed validator. The command outputs thecluster-lock.json file and key shares for each Distributed Validator created.

charon dkg --help Participate in a distributed key generation ceremony for a specific cluster definition that creates distributed validator key shares and a final cluster lock configuration. Note that all other cluster operators should run this command at the same time.

Usage: charon dkg [flags]

Flags: --data-dir string The directory where charon will store all its internal data (default ".charon") --definition-file string The path to the cluster definition file or an HTTP URL. (default ".charon/cluster-definition.json") -h, --help Help for dkg --keymanager-address string The keymanager URL to import validator keyshares. --keymanager-auth-token string Authentication bearer token to interact with keymanager API. Don't include the "Bearer" symbol, only include the api-token. --log-color string Log color; auto, force, disable. (default "auto") --log-format string Log format; console, logfmt or json (default "console") --log-level string Log level; debug, info, warn or error (default "info") --log-output-path string Path in which to write on-disk logs. --no-verify Disables cluster definition and lock file verification. --p2p-allowlist string Comma-separated list of CIDR subnets for allowing only certain peer connections. Example: 192.168.0.0/16 would permit connections to peers on your local network only. The default is to accept all connections. --p2p-denylist string Comma-separated list of CIDR subnets for disallowing certain peer connections. Example: 192.168.0.0/16 would disallow connections to peers on your local network. The default is to accept all connections. --p2p-disable-reuseport Disables TCP port reuse for outgoing libp2p connections. --p2p-external-hostname string The DNS hostname advertised by libp2p. This may be used to advertise an external DNS. --p2p-external-ip string The IP address advertised by libp2p. This may be used to advertise an external IP. --p2p-relays strings Comma-separated list of libp2p relay URLs or multiaddrs. (default [https://0.relay.obol.tech]) --p2p-tcp-address strings Comma-separated list of listening TCP addresses (ip and port) for libP2P traffic. Empty default doesn't bind to local port therefore only supports outgoing connections. --publish Publish lock file to obol-api. --publish-address string The

URL to publish the lock file to. (default "https://api.obol.tech") --shutdown-delay duration Graceful shutdown delay. (default 1s)

# Therun

subcommand

### Run the Charon middleware

Thisrun command accepts acluster-lock.json file that was created either via acharon create cluster command orcharon dkg . This lock file outlines the nodes in the cluster and the distributed validators they operate on behalf of.

charon run --help Starts the long-running Charon middleware process to perform distributed validator duties.

Usage: charon run [flags]

Flags: --beacon-node-endpoints strings Comma separated list of one or more beacon node endpoint URLs. --builder-api Enables the builder api. Will only produce builder blocks. Builder API must also be enabled on the validator client. Beacon node must be connected to a builder-relay to access the builder network. --debug-address string Listening address (ip and port) for the pprof and QBFT debug API. --feature-set string Minimum feature set to enable by default: alpha, beta, or stable. Warning: modify at own risk. (default "stable") --feature-set-disable strings Comma-separated list of features to disable, overriding the default minimum feature set. --feature-set-enable strings Comma-separated list of features to enable, overriding the default minimum feature set. -h, --help Help for run --jaeger-address string Listening address for jaeger tracing. --jaeger-service string Service name used for jaeger tracing. (default "charon") --lock-file string The path to the cluster lock file defining distributed validator cluster. If both cluster manifest and cluster lock files are provided, the cluster manifest file takes precedence. (default ".charon/cluster-lock.json") --log-color string Log color; auto, force, disable. (default "auto") --log-format string Log format; console, logfmt or json (default "console") --log-level string Log level; debug, info, warn or error (default "info") --log-output-path string Path in which to write on-disk logs. --loki-addresses strings Enables sending of logfmt structured logs to these Loki log aggregation server addresses. This is in addition to normal stderr logs. --loki-service string Service label sent with logs to Loki. (default "charon") --manifest-file string The path to the cluster manifest file. If both cluster manifest and cluster lock files are provided, the cluster manifest file takes precedence. (default ".charon/cluster-manifest.pb") --monitoring-address string Listening address (ip and port) for the monitoring API (prometheus). (default "127.0.0.1:3620") --no-verify Disables cluster definition and lock file verification. --p2p-disable-reuseport Disables TCP port reuse for outgoing libp2p connections. --p2p-external-hostname string The DNS hostname advertised by libp2p. This may be used to advertise an external DNS. --p2p-external-ip string The IP address advertised by libp2p. This may be used to advertise an external IP. --p2p-relays strings Comma-separated list of libp2p relay URLs or multiaddrs. (default [https://0.relay.obol.tech]) --p2p-tcp-address strings Comma-separated list of listening TCP addresses (ip and port) for libP2P traffic. Empty default doesn't bind to local port therefore only supports outgoing connections. --private-key-file string The path to the charon enr private key file. (default ".charon/charon-enr-private-key") --private-key-file-lock Enables private key locking to prevent multiple instances using the same key. --simnet-beacon-mock Enables an internal mock beacon node for running a simnet. --simnet-beacon-mock-fuzz Configures simnet beaconmock to return fuzzed responses. --simnet-slot-duration duration Configures slot duration in simnet beacon mock. (default 1s) --simnet-validator-keys-dir string The directory containing the simnet validator key shares. (default ".charon/validator_keys") --simnet-validator-mock Enables an internal mock validator client when running a simnet. Requires simnet-beacon-mock. --synthetic-block-proposals Enables additional synthetic block proposal duties. Used for testing of rare duties. --testnet-chain-id uint Chain ID of the custom test network. --testnet-fork-version string Genesis fork version in hex of the custom test network. --testnet-genesis-timestamp int Genesis timestamp of the custom test network. --testnet-name string Name of the custom test network. --validator-api-address string Listening address (ip and port) for validator-facing traffic proxying the beacon-node API. (default "127.0.0.1:3600")

# Thecombine

subcommand

### Combine distributed validator key shares into a single Validator key

Thecombine command combines many validator key shares into a single Ethereum validator key.

charon combine --help Combines the private key shares from a threshold of operators in a distributed validator cluster into a set of validator private keys that can be imported into a standard Ethereum validator client.

Warning: running the resulting private keys in a validator alongside the original distributed validator cluster * will * result in slashing.

Usage: charon combine [flags]

Flags: --cluster-dir string Parent directory containing a number of .charon subdirectories from the required threshold of nodes in the cluster. (default ".charon/cluster") --force Overwrites private keys with the same name if present. -h, --help Help

for combine --no-verify Disables cluster definition and lock file verification. --output-dir string Directory to output the combined private keys to. (default "./validator_keys") To run this command, one needs at least a threshold number of node operator's.charon directories, which need to be organized into a single folder:

tree ./cluster cluster/ ├── node0 │ ├── charon-enr-private-key │ ├── cluster-lock.json │ ├── deposit-data.json │ └── validator_keys │ ├── keystore-0.json │ ├── keystore-0.txt │ ├── keystore-1.json │ └── keystore-1.txt ├── node1 │ ├── charon-enr-private-key │ ├── cluster-lock.json │ ├── deposit-data.json │ └── validator_keys │ ├── keystore-0.json │ ├── keystore-0.txt │ ├── keystore-1.json │ └── keystore-1.txt ├── node2 │ ├── charon-enr-private-key │ ├── cluster-lock.json │ ├── deposit-data.json │ └── validator_keys │ ├── keystore-0.json │ ├── keystore-0.txt │ ├── keystore-1.json │ └── keystore-1.txt └── node3 ├── charon-enr-private-key ├── cluster-lock.json ├── deposit-data.json └── validator_keys ├── keystore-0.json ├── keystore-0.txt ├── keystore-1.json └── keystore-1.txt That is, each operator '.charon' directory must be placed in a parent directory, and renamed to avoid conflicts.

If for example the lock file defines 2 validators, eachvalidator_keys directory must contain exactly 4 files, a JSON and TXT file for each validator.

Those files must be named with an increasing index associated with the validator in the lock file, starting from 0.

The chosen folder name does not matter, as long as it's different from.charon .

At the end of the processcombine will create a new set of directories containing one validator key each, named after its public key:

# charon combine --cluster-dir

"./cluster" --output-dir = "./combined" tree ./combined combined ├── keystore-0.json ├── keystore-0.txt ├── keystore-1.json └── keystore-1.txt By default, thecombine command will refuse to overwrite any private key that is already present in the destination directory.

To force the process, use the--force flag.

caution The generated private keys are in the standardEIP-2335 format, and can be imported in any Ethereum validator client that supports it.

Ensure your distributed validator cluster is completely shut down for at least two epochs before starting a replacement validator or you are likely to be slashed.

## Host a relay

Relays run a libp2pcircuit relay server that allows charon clusters to perform peer discovery and for charon clients behind strict NAT gateways to be communicated with. If you want to self-host a relay for your cluster(s) the following command will start one.

charon relay --help Starts a libp2p relay that charon nodes can use to bootstrap their p2p cluster

Usage: charon relay [flags]

Flags: --auto-p2pkey Automatically create a p2pkey (secp256k1 private key used for p2p authentication and ENR) if none found in data directory. (default true) --data-dir string The directory where charon will store all its internal data (default ".charon") -h, --help Help for relay --http-address string Listening address (ip and port) for the relay http server serving runtime ENR. (default "127.0.0.1:3640") --log-color string Log color; auto, force, disable. (default "auto") --log-format string Log format; console, logfmt or json (default "console") --log-level string Log level; debug, info, warn or error (default "info") --loki-addresses strings Enables sending of logfmt structured logs to these Loki log aggregation server addresses. This is in addition to normal stderr logs. --loki-service string Service label sent with logs to Loki. (default "charon") --monitoring-address string Listening address (ip and port) for the prometheus and pprof monitoring http server. (default "127.0.0.1:3620") --p2p-advertise-private-addresses Enable advertising of libp2p auto-detected private addresses. This doesn't affect manually provided p2p-external-ip/hostname. --p2p-allowlist string Comma-separated list of CIDR subnets for allowing only certain peer connections. Example: 192.168.0.0/16 would permit connections to peers on your local network only. The default is to accept all connections. --p2p-denylist string Comma-separated list of CIDR subnets for disallowing certain peer connections. Example: 192.168.0.0/16 would disallow connections to peers on your local network. The default is to accept all connections. --p2p-disable-reuseport Disables TCP port reuse for outgoing libp2p connections. --p2p-external-hostname string The DNS hostname advertised by libp2p. This may be used to advertise an external DNS. --p2p-external-ip string The IP address advertised by libp2p. This may be used to advertise an external IP. --p2p-max-connections int Libp2p maximum number of peers that can connect to this relay. (default 16384) --p2p-max-reservations int Updates max circuit reservations per peer (each valid for 30min) (default 512) --p2p-relay-loglevel string Libp2p circuit relay log level. E.g., debug, info, warn, error. --p2p-relays strings Comma-separated list of libp2p relay URLs or multiaddrs. (default [https://0.relay.obol.tech]) --p2p-tcp-address strings Comma-separated list of listening TCP addresses (ip and port) for libP2P traffic. Empty default doesn't

bind to local port therefore only supports outgoing connections. You can also consider addingalternative public relays to your cluster by specifying a list ofp2p-relays incharon run . Edit this page Previous Charon networking Next DV Launchpad