dune-node Cheatsheet (July 11, 2019)

dune-node

General options (GENARGS)

–config-file=FILE	
–data-dir=DIR	

config options (CONFARGS)

–binary-chunks-size=NUM	Size limit (in kB) of messages
–bootstrap-threshold=NUM	Set the number of peers for bootstrapped
–connections=N	min_connections=N/2 expected_connections=N max_connections=(3*N)/2 peer_table_size=8*N bootstrap_threshold=min(N/4,2)
–disable-mempool	Do not propagate pending operations.
–discovery-addr=ADDR:PORT	The UDP address for local peer discovery.
–expected-pow=FLOAT	Proof-of-work for peers identity.
–max-download-speed=NUM	Maximum bytes read per second.
–max-upload-speed=NUM	Maximum bytes sent per second.
–net-addr=ADDR:PORT	The TCP address for this node.
–no-bootstrap-peers	Ignore hard-coded and config bootstrap peers
–peer=ADDR:PORT	Add bootstrap peer
–peer-table-size=NUM	Maximum size of internal peer tables
–private-mode	Do not accept connections
–cors-header=HEADER	Header reported by Access-Control-Allow-Headers for CORS
–cors-origin=ORIGIN	CORS origin allowed by the RPC server
–history-mode=History mode	History mode: archive, full (default), experimental-rolling
–rpc-addr=ADDR:PORT	The TCP address for the RPC server
–rpc-tls=crt,key	Enable TLS for this RPC server
–config-file=FILE	The main configuration file.
–data-dir=DIR	The node directory
–log-output=OUTPUT	Log output. Either stdout, stderr, syslog:<facility> or a file path.

dune-node identity [GENARGS]

identity show	print peer id
identity generate [difficulty]	generate peer id
identity check [difficulty]	check peer id

dune-node run [CONFARGS]

–checkpoint=<level>, <block_hash>	Only accept the chains that contains that block and those that might reach it
–sandbox[=FILE.json]	Run the daemon in sandbox mode. P2P to non-localhost addresses are disabled, and constants of the economic protocol can be altered with an optional JSON file.
-v	Increase log level. Using -v is equivalent to using DUNE_LOG='* -> info', and -vv is equivalent to using DUNE_LOG='* -> debug'.

dune-node config [CONFARGS]

config show	print config
config init	initialize config
config reset	reset config to defaults
config update	modify config

Debug

The environment variable **DUNE_LOG** is used to fine-tune what is going to be logged. The syntax is DUNE_LOG='<section> -> <level> [; ...]' where section is one of:

base node.distributed_db.p2p_peer_id node.state rpc p2p.fd p2p.io-scheduler p2p.connection p2p.connection-pool p2p.discovery p2p.maintenance p2p.welcome p2p db updater validation_process.sequential node.distributed_db.scheduler.Operation_hash node.distributed_db.scheduler.block_hash node.distributed_db.scheduler.operation_hashes node.distributed_db.scheduler.operations node.distributed_db.scheduler.Protocol_hash node.distributed_db.p2p_reader node.validator.block node.validator.bootstrap_pipeline node.chain_validator node.validator node.worker shell.snapshots demo 000-Ps9mPmXa 004-Pt24m4xi node.main

and level is one of fatal, error, warn, notice, info or debug. A * can be used as a wildcard in sections, i.e. client* -> debug. The rules are matched left to right, therefore the leftmost rule is highest priority .

dune-node snapshot

snapshot export FILE	export a snapshot to file
snapshot import FILE	import a snapshot from file
–block=<block_hash>	Block hash of the block to export/import
–rolling	Force export command to dump a minimal snapshot based on the rolling mode

dune-node upgrade

upgrade storage	upgrade the node disk storage

dune-admin-client

Event logging framework

query events from <SinkName> [–names <LIST>] [–sections <LIST>] [–since <DATE>] [–until <DATE>] [–as-json] [–dump-unknown] [–for-script <FORMAT>]	Query the events from an event sink
show event-logging	Display configuration/state information about the internal-event logging framework.
output schema of <event-name> **to** <file-path>	Output the JSON schema of an internal-event.

Commands for managing protocols

list protocols	List protocols known by the node.
inject protocol <dir>	Inject a new protocol into the node.
dump protocol <protocol hash>	Dump a protocol from the node's record of protocol.
fetch protocol <protocol hash>	Fetch a protocol from the network.

Privileged operations on the node

unmark invalid [<block>...]	Make the node forget its decision of rejecting blocks.
unmark all invalid blocks	Make the node forget every decision of rejecting blocks.
show current checkpoint	Retrieve the current checkpoint and display it in a format compatible with node argument '–checkpoint'.

Report node's status

list heads [-o –output <path>]	The last heads that have been considered by the node.
list rejected blocks [-o –output <path>]	The blocks that have been marked invalid by the node.

dune-node Cheatsheet (July 2019)

dune-admin-client

Commands for monitoring p2p-layer	
p2p stat	show global network status
connect address <address>	Connect to a new point.
kick peer <peer>	Kick a peer.
ban address <address>	Add an IP address and all its ports to the blacklist and kicks it. Remove the address from the whitelist if it was previously in it.
unban address <address>	Remove an IP address and all its ports from the blacklist.
trust address <address>	Add an IP address to the whitelist. Remove the address from the blacklist if it was previously in it.
untrust address <address>	Removes an IP address from the whitelist.
is address banned <address>	Check if an IP address is banned.
is peer banned <peer>	Check if a peer ID is banned.
ban peer <peer>	Add a peer ID to the blacklist and kicks it. Remove the peer ID from the blacklist if was previously in it.
unban peer <peer>	Removes a peer ID from the blacklist.
trust peer <peer>	Add a peer ID to the whitelist. Remove the peer ID from the blacklist if it was previously in it.
untrust peer <peer>	Remove a peer ID from the whitelist.
clear acls	Clear all access control rules.

dune-baker-PROTO

Commands related to the baker daemon	
run with local node <context_path> [<name>...] [-P –pidfile <filename>] [–max-priority <slot>] [–minimal-fees <amount>] [–minimal-nanodun-per-gas-unit <amount>] [–minimal-nanodun-per-byte <amount>] [–no-waiting-for-late-endorsements]	Launch the baker daemon

dune-endorser-PROTO

Commands related to the endorser daemon	
run [<name>...] [-P –pidfile <filename>] [–endorsement-delay <seconds>]	Launch the endorser daemon

dune-accuser-PROTO

Commands related to the accuser daemon	
run [-P –pidfile <filename>] [–preserved-levels <threshold>]	Launch the accuser daemon

dune-signer

Same as dune-client for managing crypto keys and ledger devices.

Commands specific to the signing daemon	
[-P –pidfile <filename>] [-M –magic-bytes <0xHH,0xHH,...>] [-W –check-high-watermark]	DAEMONARGS
[-a –address <host|address>] [-p –port <port number>]	ADDRARGS
launch socket signer [DAEMONARGS] [ADDRARGS]	Launch a signer daemon over a TCP socket.
launch local signer [DAEMONARGS] [-s –socket <path>]	Launch a signer daemon over a local Unix socket.
launch http signer [DAEMONARGS] [ADDRARGS]	Launch a signer daemon over HTTP.
launch https signer <cert> <key> [DAEMONARGS][ADDRARGS]	Launch a signer daemon over HTTPS.
add authorized key <pk> [-N –name <name>]	Authorize a given public key to perform signing requests.

Node Configuration File

```
{ "data-dir": string,
  "rpc":
  { "listen-addr": string,
    "cors-origin": [ string ... ],
    "cors-headers": [ string ... ],
    "crt": string,
    "key": string
  },
  "p2p":
  { "expected-proof-of-work": number,
    "bootstrap-peers": [ string ... ],
    "listen-addr": string,
    "discovery-addr": string || null,
    "private-mode": boolean,
    "limits":
    { "connection-timeout": number,
      "authentication-timeout": number,
      "min-connections": int16,
      "expected-connections": int16,
      "max-connections": int16,
      "backlog": uint8,
      "max-incoming-connections": uint8,
      "max-download-speed": int31,
      "max-upload-speed": int31,
      "swap-linger": number,
      "binary-chunks-size": uint8,
      "read-buffer-size": int31,
      "read-queue-size": int31,
      "write-queue-size": int31,
      "incoming-app-message-queue-size": int31,
      "incoming-message-queue-size": int31,
      "outgoing-message-queue-size": int31,
      "known_points_history_size": int16,
      "known_peer_ids_history_size": int16,
      "max_known_points": [ int16, int16 ],
      "max_known_peer_ids": [ int16, int16 ],
      "greylist-timeout": number,
      "maintenance-idle-time": number
    },
    "disable_mempool": boolean,
    "disable_testchain": boolean
  },
  "log":
  { "output": string,
    "level": "info|debug|error|fatal|warning|notice",
    "rules": string,
    "template": string
  },
  "internal-events": { "activate": [ string ... ] },
  "shell":
  { "peer_validator":
    { "block_header_request_timeout": number,
      "block_operations_request_timeout": number,
      "protocol_request_timeout": number,
      "new_head_request_timeout": number,
      "worker_backlog_size": int16,
      "worker_backlog_level":
        "info|debug|error|fatal|warning|notice" },
    "block_validator":
    { "protocol_request_timeout": number,
      "worker_backlog_size": int16,
      "worker_backlog_level"?:
        "info|debug|error|fatal|warning|notice" },
    "prevalidator"?:
    { "operations_request_timeout"?: number,
      "max_refused_operations"?: int16,
      "worker_backlog_size"?: int16,
      "worker_backlog_level"?:
        "info|debug|error|fatal|warning|notice" },
    "chain_validator"?:
    { "bootstrap_threshold"?: uint8,
      "worker_backlog_size"?: int16,
      "worker_backlog_level"?:
        "info|debug|error|fatal|warning|notice" },
    "history_mode"?: "full|archive|rolling"
  }
}
```