Node configurations

The Cardano node configurations is a critical part of setting up and running a Cardano node. Node configuration file contains various parameters and settings that control the behavior of the node, such as network connectivity, logging, and protocol-specific configurations.

Key Sections and Parameters in the Node Configuration File

1. General Node Settings

- **`Protocol`**
  - **Description:** Specifies the protocol the node uses (e.g., "Cardano").
  - **Example:** `"Protocol": "Cardano"`

2. Genesis Files

- **`ShelleyGenesisFile`**
  - **Description:** Path to the Shelley genesis file.
  - **Example:** `"ShelleyGenesisFile": "shelley-genesis.json"`
- **`ByronGenesisFile`**
  - **Description:** Path to the Byron genesis file.
  - **Example:** `"ByronGenesisFile": "byron-genesis.json"`
- **`AlonzoGenesisFile`**
  - **Description:** Path to the Alonzo genesis file.
  - **Example:** `"AlonzoGenesisFile": "alonzo-genesis.json"`

3. Network Configuration

- **RequiresNetworkMagic**
  - **Description:** Unique identifier for the network (e.g., testnet, mainnet).
  - **Example:** `"NetworkMagic": 1097911063`

4. Logging configuration

Node configuration for Cardano mainnet breakdown

```
################################################################################
# Mainnet Cardano node configuration

##### Locations #####

AlonzoGenesisFile: mainnet-alonzo-genesis.json
AlonzoGenesisHash: 7e94a15f55d1e82d10f09203fa1d40f8eede58fd8066542cf6566008068ed874
ByronGenesisFile: mainnet-byron-genesis.json
ByronGenesisHash: 5f20df933584822601f9e3f8c024eb5eb252fe8cefb24d1317dc3d432e940ebb
ShelleyGenesisFile: mainnet-shelley-genesis.json
ShelleyGenesisHash: 1a3be38bcbb7911969283716ad7aa550250226b76a61fc51cc9a9a35d9276d81

##### Core protocol parameters #####

Protocol: Cardano

# The mainnet does not include the network magic into addresses. Testnets do.
RequiresNetworkMagic: RequiresNoMagic

##### Update system parameters #####

# This protocol version number gets used by block-producing nodes as part
# of the system for agreeing on and synchronizing protocol updates.
#
# See https://github.com/input-output-hk/cardano-node/blob/master/cardano-node/src/Cardano/Node/Protocol
/Cardano.hs#L199
LastKnownBlockVersion-Major: 3
```

```
LastKnownBlockVersion-Minor: 0
LastKnownBlockVersion-Alt: 0
MaxKnownMajorProtocolVersion: 2

# In the Byron era, some software versions are also published on the chain.
# We do this only for Byron compatibility now.
ApplicationName: cardano-sl
ApplicationVersion: 1

##### Logging configuration #####

# Enable or disable logging overall
TurnOnLogging: True

# Enable the collection of various OS metrics such as memory and CPU use.
# These metrics are traced in the context name: 'cardano.node.metrics' and can
# be directed to the logs or monitoring backends.
TurnOnLogMetrics: True

# Global logging severity filter. Messages must have at least this severity to
# pass. Typical values would be Warning, Notice, Info, or Debug.
minSeverity: Info

# Log items can be rendered with more or less verbose detail.
# Verbosity ranges from MinimalVerbosity, NormalVerbosity to MaximalVerbosity
TracingVerbosity: NormalVerbosity

# The system supports a number of backends for logging and monitoring.
# This setting lists the backends that will be available to use in the
# configuration below. The logging backend is called Katip.
setupBackends:
  - KatipBK

# This specifies the default backends that trace output is sent to if it
# is not specifically configured to be sent to other backends.
defaultBackends:
  - KatipBK

# EKG is a simple metrics monitoring system. Uncomment the following to enable
# this backend and listen on the given local port and point your web browser to
# http://localhost:12788/
hasEKG: 12788

# The Prometheus monitoring system exports EKG metrics. Uncomment the following
# to listen on the given port. Output is provided on
# http://localhost:12789/metrics
# hasPrometheus:
#   - "127.0.0.1"
#   - 12789

# To enable the legacy 'TraceForwarder' backend, uncomment the following setting. Log
# items are then forwarded based on an entry in 'mapBackends' to a separate
# process running a 'TraceAcceptor'.
# Example using UNIX pipes:
# traceForwardTo:
#   tag: RemotePipe
#   contents: "logs/pipe"
#
# Example using Windows named pipes:
# traceForwardTo:
#   tag: RemotePipe
#   contents: "\\\\.\\pipe\\acceptor"
#
# Example using network socket:
# traceForwardTo:
#   tag: RemoteSocket
#   contents:
#     - "127.0.0.1"
#     - "2997"

# For the Katip logging backend we must set up outputs (called scribes)
```

```
# The available types of scribe are:
#   FileSK for files
#   StdoutSK/StderrSK for stdout/stderr
#   JournalSK for systemd's journal system
#   DevNullSK ignores all output
# The scribe output format can be ScText or ScJson. Log rotation settings can
# be specified in the defaults below or overridden on a per-scribe basis here.
setupScribes:
  - scKind: StdoutSK
    scName: stdout
    scFormat: ScText
    scRotation: null

# For the Katip logging backend this specifies the default scribes that trace
# output is sent to if it is not configured to be sent to other scribes.
defaultScribes:
  - - StdoutSK
    - stdout

# The default file rotation settings for katip scribes, unless overridden
# in the setupScribes above for specific scribes.
rotation:
  rpLogLimitBytes: 5000000
  rpKeepFilesNum:  10
  rpMaxAgeHours:   24


##### Coarse grained logging control #####

# Trace output from whole subsystems can be enabled/disabled using the following
# settings. This provides fairly coarse grained control, but it is relatively
# efficient at filtering out unwanted trace output.

TraceAcceptPolicy: True

# Trace BlockFetch client.
TraceBlockFetchClient: False

# Trace BlockFetch decisions made by the BlockFetch client.
TraceBlockFetchDecisions: False

# Trace BlockFetch protocol messages.
TraceBlockFetchProtocol: False

# Serialised Trace BlockFetch protocol messages.
TraceBlockFetchProtocolSerialised: False

# Trace BlockFetch server.
TraceBlockFetchServer: False

# Trace BlockchainTime.
# TraceBlockchainTime: False

# Verbose tracer of ChainDB
TraceChainDb: True

# Trace ChainSync client.
TraceChainSyncClient: False

# Trace ChainSync server (blocks).
TraceChainSyncBlockServer: False

# Trace ChainSync server (headers).
TraceChainSyncHeaderServer: False

# Trace ChainSync protocol messages.
TraceChainSyncProtocol: False

TraceConnectionManager: True

# Trace DNS Resolver messages.
```

```
TraceDNSResolver: True

# Trace DNS Subscription messages.
TraceDNSSubscription: True

TraceDiffusionInitialization: True

# Trace error policy resolution.
TraceErrorPolicy: True

# Trace local error policy resolution.
TraceLocalErrorPolicy: True

# Trace block forging.
TraceForge: True

# Trace Handshake protocol messages.
TraceHandshake: False

# Trace IP Subscription messages.
TraceIpSubscription: True

TraceLedgerPeers: True

TraceLocalRootPeers: True

TraceInboundGovernor: True

TracePeerSelection: True

TracePeerSelectionActions: True

TracePublicRootPeers: True

TraceServer: True


# Trace local ChainSync protocol messages.
TraceLocalChainSyncProtocol: False

# Trace local Handshake protocol messages.
TraceLocalHandshake: False

# Trace local TxSubmission protocol messages.
TraceLocalTxSubmissionProtocol: False

# Trace local TxSubmission server.
TraceLocalTxSubmissionServer: False

# Trace mempool.
TraceMempool: True

# Trace Mux Events.
TraceMux: False

# Trace TxSubmission server (inbound transactions).
TraceTxInbound: False

# Trace TxSubmission client (outbound transactions).
TraceTxOutbound: False

# Trace TxSubmission protocol messages.
TraceTxSubmissionProtocol: False


##### Fine grained logging control #####

# It is also possible to have more fine grained control over filtering of
# trace output and to match and route trace output to particular backends.
# This is less efficient than the coarse trace filters above but provides
# much more precise control.
```

```
options:

  # This routes metrics matching specific names to particular backends.
  # This overrides the 'defaultBackends' listed above. And note that it is
  # an override and not an extension so anything matched here will not
  # go to the default backend, only to the explicitly listed backends.
  mapBackends:
    cardano.node.metrics:
      - EKGViewBK

    cardano.node.resources:
      - EKGViewBK

  # # redirects traced values to a specific scribe which is identified by its
  # # type and its name, separated by "::":
  # mapScribes:
  #   cardano.node.metrics:
  #     - "FileSK::logs/mainnet.log"
  mapSubtrace:
    cardano.node.metrics:
      subtrace: Neutral

hasPrometheus:
  - 127.0.0.1
  - 12798

# Set or unset the mempool capacity override in number of bytes.
#
# This is intended for testing, and for low-resource machines to run with a smaller mempool.
# Please note that running with a large mempool is NOT recommended. The mempool is a
# just a network communication buffer and all the advice on "buffer bloat" applies, see:
# https://en.wikipedia.org/wiki/Bufferbloat
# The default size is two blocks, which is generally enough to ensure full blocks.
#
# MempoolCapacityBytesOverride: Integer | NoOverride
#
# Examples:
#   MempoolCapacityBytesOverride: 1000000 (1MB)
#   MempoolCapacityBytesOverride: NoOverride (default)
```