Custom analytic tools

warning This article requires a revision. Check plugins for some easy addition of analytical tools

info Learn more about Plugins You can also read more about some useful interfaces below:

There are multiple extension points where you can add custom analytics to your Nethermind node if you know some C#. Below you will find an example of using two very useful interfaces -IBlockVisitor andITreeVisitor .

Just to execute the code I have added one new initialization step that invokes two custom verifiers that I have used for calculating total supply in two different ways - by calculating mining rewards and by summing up all account balances:

[RunnerStepDependencies(typeof(ReviewBlockTree))] public class RunCustomTools : IStep { private readonly EthereumRunnerContext _context;

public RunCustomTools(EthereumRunnerContext context) { _context = context; }

public Task Execute(CancellationToken cancellationToken) { ILogger logger = _context.LogManager.GetClassLogger(); IInitConfig initConfig = _context.Config();

switch (initConfig.DiagnosticMode) { case DiagnosticMode.VerifySupply: { logger.Info("Genesis supply:"); SupplyVerifier supplyVerifier = new SupplyVerifier(logger); StateDb stateDb = new StateDb(_context.DbProvider.StateDb.Innermost); StateDb codeDb = new StateDb(_context.DbProvider.StateDb.Innermost); StateReader stateReader = new StateReader(stateDb, codeDb, _context.LogManager); stateReader.RunTreeVisitor(supplyVerifier, _context.BlockTree!.Genesis.StateRoot);

Block head = _context.BlockTree!.Head; logger.Info("Head ({head.Number}) block supply:"); supplyVerifier = new SupplyVerifier(logger); stateReader.RunTreeVisitor(supplyVerifier, head.StateRoot); break; } case DiagnosticMode.VerifyRewards: _context.BlockTree!.Accept(new RewardsVerifier(_context.LogManager), cancellationToken); break; }

return Task.CompletedTask; } } Below you will see an example of using ITreeVisitor that allows to check all the blocks, including some of the discarded branches if you wish so:

public class RewardsVerifier : IBlockTreeVisitor { private ILogger _logger; public bool PreventsAcceptingNewBlocks => true; public long StartLevelInclusive => 0; public long EndLevelExclusive => 10618000;

private UInt256 _genesisAllocations = UInt256.Parse("72009990499480000000000000"); private UInt256 _uncles; private UInt256 _blockRewards;

public RewardsVerifier(ILogManager logManager) { _logger = logManager.GetClassLogger(); }

private RewardCalculator _rewardCalculator = new RewardCalculator(MainnetSpecProvider.Instance);

public Task VisitBlock(Block block, CancellationToken cancellationToken) { BlockReward[] rewards = _rewardCalculator.CalculateRewards(block); for (int i = 0; i < rewards.Length; i++) { if (rewards[i].RewardType == BlockRewardType.Uncle) { _uncles += rewards[i].Value; } else { _blockRewards += rewards[i].Value; } }

_logger.Info("Visiting block {block.Number}, total supply is (genesis + miner rewards + uncle rewards) | {_genesisAllocations} + {_blockRewards} + {_uncles}"); return Task.FromResult(BlockVisitOutcome.None); }

public Task VisitLevelStart(ChainLevelInfo chainLevelInfo, CancellationToken cancellationToken) => Task.FromResult(LevelVisitOutcome.None);

public Task VisitMissing(Keccak hash, CancellationToken cancellationToken) => Task.FromResult(true);

public Task VisitHeader(BlockHeader header, CancellationToken cancellationToken) => Task.FromResult(true);

public Task VisitLevelEnd(CancellationToken cancellationToken) => Task.FromResult(LevelVisitOutcome.None); } And here you will find an example of a tree visitor that sums up all the account balances:

public class SupplyVerifier : ITreeVisitor { private readonly ILogger _logger; private UInt256 _balance = UInt256.Zero; private int _accountsVisited;

public SupplyVerifier(ILogger logger) { _logger = logger; }

public bool ShouldVisit(Keccak nextNode) { return true; }

public void VisitTree(Keccak rootHash, TrieVisitContext trieVisitContext) { }

public void VisitMissingNode(Keccak nodeHash, TrieVisitContext trieVisitContext) { }

public void VisitBranch(TrieNode node, TrieVisitContext trieVisitContext) { }

public void VisitExtension(TrieNode node, TrieVisitContext trieVisitContext) { }

public void VisitLeaf(TrieNode node, TrieVisitContext trieVisitContext, byte[] value = null) { if (trieVisitContext.IsStorage) { return; }

AccountDecoder accountDecoder = new AccountDecoder(); Account account = accountDecoder.Decode(node.Value.AsRlpStream()); _balance += account.Balance; _accountsVisited++;

_logger.Info("Balance after visiting {_accountsVisited}: {_balance}"); }

public void VisitCode(Keccak codeHash, TrieVisitContext trieVisitContext) { } }Edit this page Last updatedonFeb 17, 2024
Previous Plugins Next FAQ