

Namespace chia.dotnet

Classes

[AmountFilter](#)

[AmountWithPuzzlehash](#)

[Announcement](#)

[AssetInfo](#)

[AutoClaimSettings](#)

[BlockCountMetrics](#)

[BlockRecord](#)

This class is not included or hashed into the blockchain, but it is kept in memory as a more efficient way to maintain data about the blockchain. This allows us to validate future blocks, difficulty adjustments, etc, without saving the whole header block in memory.

[BlockSpendWithConditions](#)

[BlockchainState](#)

The node's view of the blockchain.

[CATInfo](#)

[CATWallet](#)

Wraps a CAT wallet

[CRCATWallet](#)

Wraps a CRCAT Wallet

[ChallengeChainSubSlot](#)

[ClassgroupElement](#)

Represents a classgroup element (a,b,c) where a, b, and c are 512 bit signed integers. However this is using a compressed representation. VDF outputs are a single classgroup element. VDF proofs can also be one classgroup element(or multiple).

[Coin](#)

This structure is used in the body for the reward and fees genesis coins.

[CoinAnnouncement](#)

[CoinRecord](#)

These are values that correspond to a CoinName that are used in keeping track of the unspent database.

[CoinSpend](#)

This is a rather disparate data structure that validates coin transfers. It's generally populated with data from different sources, since burned coins are identified by name, so it is built up more often than it is streamed.

[Condition](#)

This type doesn't exist in the chia code. This property is serialized into the Json as a tuple, which shows up as a mixed type json array. There is a specialized converter to read the Json and get the ConditionOpCode into [ConditionOpcode](#) conditions: List[Tuple[ConditionOpcode, List[ConditionWithArgs]]]

[ConditionValidTimes](#)

[ConditionWithVars](#)

This structure is used to store parsed CLVM conditions. Conditions in CLVM have either format of(opcode, var1) or(opcode, var1, var2)

[Config](#)

Represents a chia config yaml file and its contents. Used to find the uri and ssl certs needed to connect

[ConnectionInfo](#)

Chia's representation of a connection from node to node

[CrawlerProxy](#)

Proxy that communicates with the crawler

[DAOInfo](#)

[DAORules](#)

[DAOWallet](#)

Wraps a DAO Wallet

[DIDWallet](#)

Wraps a Distributed Identity Wallet

[DaemonProxy](#)

[WebSocketRpcClient](#) for the daemon interface. The daemon can be used to proxy messages to and from other chia services as well as controlling the [PlotterProxy](#) and having its own procedures

[DataLayerOffer](#)

[DataLayerProxy](#)

Proxy that communicates with the Data Layer

[DataLayerSyncStatus](#)

[DataLayerWallet](#)

Wraps a Data Layer Wallet

[DidInfo](#)

[EndOfSubSlotBundle](#)

[EndpointInfo](#)

Information about how to connect and authenticate with the RPC endpoint

[ErrorResponse](#)

Response in error case for all endpoints of the pool protocol

[Extensions](#)

Helper extension methods

[FarmerProxy](#)

Proxy that communicates with the farmer

[FarmerRewards](#)

[FarmerSignagePoint](#)

This type doesn't exist in the chia code but is generated and passed around as a dicitonary (not to be confused with [SignagePoint](#))

[FilterItem](#)

[Foliage](#)

The entire foliage block, containing signature and the unsigned back pointer. The hash of this is the "header hash". Note that for unfinished blocks, the prev_block_hash is the prev from the signage point, and can be replaced with a more recent block

[FoliageBlockData](#)

Part of the block that is signed by the plot key

[FoliageTransactionBlock](#)

Information that goes along with each transaction block that is relevant for light clients

[FullBlock](#)

All the information required to validate a block

[FullNodeProxy](#)

Proxy that communicates with the full node

[FungibleAsset](#)

[HarvesterConfig](#)

[HarvesterConnection](#)

[HarvesterInfo](#)

[HarvesterProxy](#)

Proxy that communicates with the harvester

[HarvesterSummary](#)

[HarvesterSync](#)

[HashFilter](#)

[HttpRpcClient](#)

Class that handles core communication with the rpc endpoint using http(s)

[InfusedChallengeChainSubSlot](#)

[InternalNode](#)

[KVDiff](#)

[KeyData](#)

[KeyDataSecrets](#)

[KeyringStatus](#)

[Layer](#)

[LineageProof](#)

[MempoolItem](#)

[MempoolMinFees](#)

[Message](#)

The messaging data structure for request and response exchange with the RPC endpoint

[Mirror](#)

[NFTBulkMintingInfo](#)

Info for minting NFTs in bulk

[NFTInfo](#)

NFT Info for displaying NFT on the UI

[NFTMintingInfo](#)

Info for minting an NFT

[NFTWallet](#)

Wraps an NFT wallet

[NPC](#)

[NPCResult](#)

[NftCoinInfo](#)

[NftMintEntry](#)

[OfferRecord](#)

The in memory representation of an offer and its record of trade

[OfferStore](#)

[OfferSummary](#)

[PaginatedPlotRequest](#)

[PassphraseRequirements](#)

[PeerCounts](#)

[PlotInfo](#)

Info about a plot file

[PlotInforequestData](#)

[PlotPathrequestData](#)

[PlotterConfig](#)

Configuration settings for the plotter. (equivalent to chia plots create command line args)

<https://github.com/Chia-Network/chia-blockchain/wiki/CLI-Commands-Reference> ↗

[PlotterInfo](#)

[PlotterProxy](#)

Class to manage plotting

[PlottingKeys](#)

[PluginStatus](#)

[PoolInfo](#)

[PoolPoint](#)

[PoolState](#)

`PoolState` is a type that is serialized to the blockchain to track the state of the user's pool singleton
`target_puzzle_hash` is either the pool address, or the self-pooling address that pool rewards will be paid to. `target_puzzle_hash` is NOT the p2_singleton puzzle that block rewards are sent to. The `p2_singleton` address is the initial address, and the `target_puzzle_hash` is the final destination.
`relative_lock_height` is zero when in SELF_POOLING state

[PoolStateInfo](#)

This type does not exist in the chia python, but is returned as a dicitonary for the UI to show pool state. Not to be confused with [PoolState](#)

[PoolTarget](#)

[PoolWallet](#)

Wraps a Pool Wallet

[PoolWalletConfig](#)

This is what goes into the user's config file, to communicate between the wallet and the farmer processes.

[PoolWalletInfo](#)

Internal Pool Wallet state, not destined for the blockchain. This can be completely derived with the Singleton's CoinSpends list, or with the information from the WalletPoolStore.

[PrivateKey](#)

[PrivateKeyData](#)

[Proof](#)

[ProofOfSpace](#)

[ProposalInfo](#)

[ProposalState](#)

[PuzzleAnnouncement](#)

[QueuedPlotInfo](#)

An entry on the plotter queue

[ResponseException](#)

Exception thrown when the RPC endpoint returns a response [Message](#) but Data.success is false or there is a communication error on the websocket or http channel

[RewardChainBlock](#)

[RewardChainBlockUnfinished](#)

[RewardChainSubSlot](#)

[Root](#)

[RootHash](#)

[RootHistory](#)

[RoyaltyAsset](#)

[SendPeer](#)

Represents the list of peers that we sent the transaction to, whether each one included it in the mempool, and what the error message (if any) was

[ServiceProxy](#)

Base class that uses an [IRpcClient](#) to send and receive messages to services

[SignagePoint](#)

[SingletonInfo](#)

[SingletonRecord](#)

[SpendBundle](#)

This is a list of coins being spent along with their solution programs, and a single aggregated signature. This is the object that most closely corresponds to a bitcoin transaction (although because of non-interactive signature aggregation, the boundaries between transactions are more flexible than in bitcoin).

[StoreProofs](#)

[SubEpochSummary](#)

[SubSlotProofs](#)

[SyncState](#)

[TerminalNode](#)

[Token](#)

[TradeManager](#)

API wrapper for those wallet RPC methods dealing with trades and offers

[TradeRecord](#)

Used for storing transaction data and status in wallets.

[TransactionRecord](#)

Used for storing transaction data and status in wallets.

[TransactionTypeFilter](#)

[TransactionsInfo](#)

Information that goes along with each transaction block

[UInt32Range](#)

[UInt64Range](#)

[UnfinishedHeaderBlock](#)

Same as a FullBlock but without TransactionInfo and Generator, used by light clients

[VCLineageProof](#)

The covenant layer for exigent metadata layers requires to be passed the previous parent's metadata too

[VCProofs](#)

[VCRecord](#)

[VDFInfo](#)

[VDFProof](#)

[VerifiedCredential](#)

This class serves as the main driver for the entire VC puzzle stack. Given the information below, it can sync and spend VerifiedCredentials in any specified manner. Trying to sync from a spend that this class did not create will likely result in an error.

[VerifiedCredentialManager](#)

API wrapper for those wallet RPC methods dealing with verified credentials

[Wallet](#)

Base class representing a specific wallet (i.e. anything with a WalletID)

[WalletAddress](#)

[WalletBalance](#)

[WalletInfo](#)

This object represents the wallet data as it is stored in DB. ID: Main wallet (Standard) is stored at index 1, every wallet created after done has auto incremented id. Name: can be a user provided or default generated name. (can be modified) Type: is specified during wallet creation and should never be changed. Data: this filed is intended to be used for storing any wallet specific information required for it. This data should be json encoded string.

[WalletProxy](#)

Proxy that communicates with the wallet endpoint

[WebSocketRpcClient](#)

Class that handles core communication with the rpc endpoint using a websocket (wss). Only the daemon endpoint supports websockets, but it can proxy communication to other services. [Destination](#)

Structs

[ServiceNames](#)

The names of chia services. These are used as [Destination](#) values

Interfaces

[IRpcClient](#)

Interface representing rpc communication

Enums

[CoinRecordOrder](#)

[CoinType](#)

[FilterMode](#)

[KSize](#)

Valid plot sizes <https://github.com/Chia-Network/chia-blockchain/wiki/k-sizes>

[MempoolInclusionStatus](#)

[NodeType](#)

[PlotState](#)

[PoolSingletonState](#)

From the user's point of view, a pool group can be in these states: **SELF_POOLING**: The singleton exists on the blockchain, and we are farming block rewards to a wallet address controlled by the user

LEAVING_POOL: The singleton exists, and we have entered the "escaping" state, which means we are waiting for a number of blocks = `relative_lock_height` to pass, so we can leave.

FARMING_TO_POOL: The singleton exists, and it is assigned to a pool.

CLAIMING_SELF_POOLED_REWARDS: We have submitted a transaction to sweep our self-pooled funds.

[Side](#)

[Status](#)

[TradeStatus](#)

[TransactionType](#)

[WalletType](#)

Wallet Types

Class AmountFilter

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record AmountFilter : IEquatable<AmountFilter>
```

Inheritance

[object](#) ← AmountFilter

Implements

[IEquatable](#)<[AmountFilter](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Mode

```
public FilterMode Mode { get; init; }
```

Property Value

[FilterMode](#)

Values

```
public IEnumerable<ulong> Values { get; init; }
```

Property Value

[IEnumerable](#)<[ulong](#)>

Class AmountWithPuzzlehash

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record AmountWithPuzzlehash : IEquatable<AmountWithPuzzlehash>
```

Inheritance

[object](#) ← AmountWithPuzzlehash

Implements

[IEquatable](#) <[AmountWithPuzzlehash](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Amount

```
public ulong Amount { get; init; }
```

Property Value

[ulong](#)

AssetId

```
public string? AssetId { get; init; }
```

Property Value

[string](#)

Memos

```
public IEnumerable<string> Memos { get; init; }
```

Property Value

[IEnumerable](#) <[string](#)>

PuzzleHash

```
public string PuzzleHash { get; init; }
```

Property Value

[string](#)

Class Announcement

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record Announcement : IEquatable<Announcement>
```

Inheritance

[object](#) ← Announcement

Implements

[IEquatable](#) <[Announcement](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Message

```
public string Message { get; init; }
```

Property Value

[string](#)

MorphBytes

```
public string? MorphBytes { get; init; }
```

Property Value

[string](#)

OriginInfo

```
public string OriginInfo { get; init; }
```

Property Value

[string](#) ↗

Class AssetInfo

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record AssetInfo : IEquatable<AssetInfo>
```

Inheritance

[object](#) ← AssetInfo

Implements

[IEquatable](#)<[AssetInfo](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Address

```
public string Address { get; init; }
```

Property Value

[string](#)

Amount

```
public ulong Amount { get; init; }
```

Property Value

[ulong](#)

Asset

```
public string Asset { get; init; }
```

Property Value

[string](#) ↗

Class AutoClaimSettings

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record AutoClaimSettings : IEquatable<AutoClaimSettings>
```

Inheritance

[object](#) ← AutoClaimSettings

Implements

[IEquatable](#) <[AutoClaimSettings](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

BatchSize

```
public ushort BatchSize { get; init; }
```

Property Value

[ushort](#)

Enabled

```
public bool Enabled { get; init; }
```

Property Value

[bool](#)

MinAmount

```
public ulong MinAmount { get; init; }
```

Property Value

[ulong](#) ↗

TxFee

```
public ulong TxFee { get; init; }
```

Property Value

[ulong](#) ↗

Class BlockCountMetrics

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record BlockCountMetrics : IEquatable<BlockCountMetrics>
```

Inheritance

[object](#) ← BlockCountMetrics

Implements

[IEquatable](#)<[BlockCountMetrics](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

CompactBlock

```
public int CompactBlock { get; init; }
```

Property Value

[int](#)

HintCount

```
public int HintCount { get; init; }
```

Property Value

[int](#)

UncompactBlocks

```
public int UncompactBlocks { get; init; }
```

Property Value

[int ↗](#)

Class BlockRecord

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

This class is not included or hashed into the blockchain, but it is kept in memory as a more efficient way to maintain data about the blockchain. This allows us to validate future blocks, difficulty adjustments, etc, without saving the whole header block in memory.

```
public record BlockRecord : IEquatable<BlockRecord>
```

Inheritance

[object](#) ← BlockRecord

Implements

[IEquatable](#) <[BlockRecord](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

ChallengeBlockInfoHash

Hash of challenge chain data, used to validate end of slots in the future

```
public string ChallengeBlockInfoHash { get; init; }
```

Property Value

[string](#)

This class is not included or hashed into the blockchain, but it is kept in memory as a more efficient way to maintain data about the blockchain. This allows us to validate future blocks, difficulty adjustments, etc, without saving the whole header block in memory.

ChallengeVdfOutput

This is the intermediary VDF output at ip_iters in challenge chain

```
public ClassgroupElement ChallengeVdfOutput { get; init; }
```

Property Value

[ClassgroupElement](#)

This class is not included or hashed into the blockchain, but it is kept in memory as a more efficient way to maintain data about the blockchain. This allows us to validate future blocks, difficulty adjustments, etc, without saving the whole header block in memory.

DateTimestamp

```
[JsonIgnore]  
public DateTime? DateTimestamp { get; }
```

Property Value

[DateTime](#)?

This class is not included or hashed into the blockchain, but it is kept in memory as a more efficient way to maintain data about the blockchain. This allows us to validate future blocks, difficulty adjustments, etc, without saving the whole header block in memory.

Deficit

A deficit of 16 is an overflow block after an infusion. Deficit of 15 is a challenge block

```
public byte Deficit { get; init; }
```

Property Value

[byte](#)

This class is not included or hashed into the blockchain, but it is kept in memory as a more efficient way to maintain data about the blockchain. This allows us to validate future blocks, difficulty adjustments, etc, without saving the whole header block in memory.

FarmerPuzzleHash

```
public string FarmerPuzzleHash { get; init; }
```

Property Value

[string](#) ↗

This class is not included or hashed into the blockchain, but it is kept in memory as a more efficient way to maintain data about the blockchain. This allows us to validate future blocks, difficulty adjustments, etc, without saving the whole header block in memory.

Fees

Transaction block (present iff is_transaction_block)

```
public ulong? Fees { get; init; }
```

Property Value

[ulong](#) ↗?

This class is not included or hashed into the blockchain, but it is kept in memory as a more efficient way to maintain data about the blockchain. This allows us to validate future blocks, difficulty adjustments, etc, without saving the whole header block in memory.

FinishedChallengeSlotHashes

Slot (present iff this is the first SB in sub slot)

```
public IEnumerable<string>? FinishedChallengeSlotHashes { get; init; }
```

Property Value

[IEnumerable<string>](#)

This class is not included or hashed into the blockchain, but it is kept in memory as a more efficient way to maintain data about the blockchain. This allows us to validate future blocks, difficulty adjustments, etc, without saving the whole header block in memory.

FinishedInfusedChallengeSlotHashes

Slot (present iff this is the first SB in sub slot)

```
public IEnumerable<string>? FinishedInfusedChallengeSlotHashes { get; init; }
```

Property Value

[IEnumerable<string>](#)

This class is not included or hashed into the blockchain, but it is kept in memory as a more efficient way to maintain data about the blockchain. This allows us to validate future blocks, difficulty adjustments, etc, without saving the whole header block in memory.

FinishedRewardSlotHashes

Slot (present iff this is the first SB in sub slot)

```
public IEnumerable<string>? FinishedRewardSlotHashes { get; init; }
```

Property Value

[IEnumerable<string>](#)

This class is not included or hashed into the blockchain, but it is kept in memory as a more efficient way to maintain data about the blockchain. This allows us to validate future blocks, difficulty adjustments, etc, without saving the whole header block in memory.

HeaderHash

```
public string HeaderHash { get; init; }
```

Property Value

[string](#)

This class is not included or hashed into the blockchain, but it is kept in memory as a more efficient way to maintain data about the blockchain. This allows us to validate future blocks, difficulty adjustments, etc, without saving the whole header block in memory.

Height

```
public uint Height { get; init; }
```

Property Value

[uint](#)

This class is not included or hashed into the blockchain, but it is kept in memory as a more efficient way to maintain data about the blockchain. This allows us to validate future blocks, difficulty adjustments, etc, without saving the whole header block in memory.

InfusedChallengeVdfOutput

This is the intermediary VDF output at ip_iters in infused cc, if deficit less than or equal to 3

```
public ClassgroupElement? InfusedChallengeVdfOutput { get; init; }
```

Property Value

[ClassgroupElement](#)

This class is not included or hashed into the blockchain, but it is kept in memory as a more efficient way to maintain data about the blockchain. This allows us to validate future blocks, difficulty adjustments, etc, without saving the whole header block in memory.

IsTransactionBlock

```
[JsonIgnore]  
public bool IsTransactionBlock { get; }
```

Property Value

[bool](#) ↗

This class is not included or hashed into the blockchain, but it is kept in memory as a more efficient way to maintain data about the blockchain. This allows us to validate future blocks, difficulty adjustments, etc, without saving the whole header block in memory.

Overflow

```
public bool Overflow { get; init; }
```

Property Value

[bool](#) ↗

This class is not included or hashed into the blockchain, but it is kept in memory as a more efficient way to maintain data about the blockchain. This allows us to validate future blocks, difficulty adjustments, etc, without saving the whole header block in memory.

PoolPuzzleHash

Need to keep track of these because Coins are created in a future block

```
public string PoolPuzzleHash { get; init; }
```

Property Value

[string](#) ↗

This class is not included or hashed into the blockchain, but it is kept in memory as a more efficient way to maintain data about the blockchain. This allows us to validate future blocks, difficulty adjustments, etc, without saving the whole header block in memory.

PrevHash

Header hash of the previous block Transaction block (present iff is_transaction_block)

```
public string PrevHash { get; init; }
```

Property Value

[string](#)

This class is not included or hashed into the blockchain, but it is kept in memory as a more efficient way to maintain data about the blockchain. This allows us to validate future blocks, difficulty adjustments, etc, without saving the whole header block in memory.

PrevTransactionBlockHash

Header hash of the previous transaction block

```
public string? PrevTransactionBlockHash { get; init; }
```

Property Value

[string](#)

This class is not included or hashed into the blockchain, but it is kept in memory as a more efficient way to maintain data about the blockchain. This allows us to validate future blocks, difficulty adjustments, etc, without saving the whole header block in memory.

PrevTransactionBlockHeight

```
public uint PrevTransactionBlockHeight { get; init; }
```

Property Value

[uint](#)

This class is not included or hashed into the blockchain, but it is kept in memory as a more efficient way to maintain data about the blockchain. This allows us to validate future blocks, difficulty

adjustments, etc, without saving the whole header block in memory.

RequiredIters

The number of iters required for this proof of space

```
public ulong RequiredIters { get; init; }
```

Property Value

[ulong](#)

This class is not included or hashed into the blockchain, but it is kept in memory as a more efficient way to maintain data about the blockchain. This allows us to validate future blocks, difficulty adjustments, etc, without saving the whole header block in memory.

RewardClaimsIncorporated

Transaction block (present iff is_transaction_block)

```
public IEnumerable<Coin>? RewardClaimsIncorporated { get; init; }
```

Property Value

[IEnumerable](#)<[Coin](#)>

This class is not included or hashed into the blockchain, but it is kept in memory as a more efficient way to maintain data about the blockchain. This allows us to validate future blocks, difficulty adjustments, etc, without saving the whole header block in memory.

RewardInfusionNewChallenge

The reward chain infusion output, input to next VDF

```
public string RewardInfusionNewChallenge { get; init; }
```

Property Value

string

This class is not included or hashed into the blockchain, but it is kept in memory as a more efficient way to maintain data about the blockchain. This allows us to validate future blocks, difficulty adjustments, etc, without saving the whole header block in memory.

SignagePointIndex

```
public byte SignagePointIndex { get; init; }
```

Property Value

byte

This class is not included or hashed into the blockchain, but it is kept in memory as a more efficient way to maintain data about the blockchain. This allows us to validate future blocks, difficulty adjustments, etc, without saving the whole header block in memory.

SubEpochSummaryIncluded

Sub-epoch (present iff this is the first SB after sub-epoch)

```
public SubEpochSummary? SubEpochSummaryIncluded { get; init; }
```

Property Value

SubEpochSummary

This class is not included or hashed into the blockchain, but it is kept in memory as a more efficient way to maintain data about the blockchain. This allows us to validate future blocks, difficulty adjustments, etc, without saving the whole header block in memory.

SubSlotIters

Current network sub_slot_iters parameter

```
public ulong SubSlotIters { get; init; }
```

Property Value

[ulong](#)

This class is not included or hashed into the blockchain, but it is kept in memory as a more efficient way to maintain data about the blockchain. This allows us to validate future blocks, difficulty adjustments, etc, without saving the whole header block in memory.

Timestamp

Transaction block (present iff is_transaction_block)

```
public ulong? Timestamp { get; init; }
```

Property Value

[ulong?](#)

This class is not included or hashed into the blockchain, but it is kept in memory as a more efficient way to maintain data about the blockchain. This allows us to validate future blocks, difficulty adjustments, etc, without saving the whole header block in memory.

TotalIters

Total number of VDF iterations since genesis, including this block

```
public BigInteger TotalIters { get; init; }
```

Property Value

[BigInteger](#)

This class is not included or hashed into the blockchain, but it is kept in memory as a more efficient way to maintain data about the blockchain. This allows us to validate future blocks, difficulty adjustments, etc, without saving the whole header block in memory.

Weight

Total cumulative difficulty of all ancestor blocks since genesis

```
public BigInteger Weight { get; init; }
```

Property Value

[BigInteger](#) ↗

This class is not included or hashed into the blockchain, but it is kept in memory as a more efficient way to maintain data about the blockchain. This allows us to validate future blocks, difficulty adjustments, etc, without saving the whole header block in memory.

Class BlockSpendWithConditions

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record BlockSpendWithConditions : IEquatable<BlockSpendWithConditions>
```

Inheritance

[object](#) ← BlockSpendWithConditions

Implements

[IEquatable](#) <[BlockSpendWithConditions](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

CoinSpend

```
public CoinSpend CoinSpend { get; init; }
```

Property Value

[CoinSpend](#)

Conditions

```
public IEnumerable<ConditionWithVars> Conditions { get; init; }
```

Property Value

[IEnumerable](#) <[ConditionWithVars](#)>

Class BlockchainState

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

The node's view of the blockchain.

```
public record BlockchainState : IEquatable<BlockchainState>
```

Inheritance

[object](#) ← BlockchainState

Implements

[IEquatable](#) <[BlockchainState](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

AverageBlockTime

```
public uint AverageBlockTime { get; init; }
```

Property Value

[uint](#)

The node's view of the blockchain.

BlockMaxCost

```
public long BlockMaxCost { get; init; }
```

Property Value

[long](#)

The node's view of the blockchain.

Difficulty

```
public ulong Difficulty { get; init; }
```

Property Value

[ulong](#)

The node's view of the blockchain.

GenesisChallengeInitiated

```
public bool GenesisChallengeInitiated { get; init; }
```

Property Value

[bool](#)

The node's view of the blockchain.

MempoolCost

```
public long MempoolCost { get; init; }
```

Property Value

[long](#)

The node's view of the blockchain.

MempoolFees

```
public ulong MempoolFees { get; init; }
```

Property Value

[ulong](#)

The node's view of the blockchain.

MempoolMaxTotalCost

```
public long MempoolMaxTotalCost { get; init; }
```

Property Value

[long](#)

The node's view of the blockchain.

MempoolMinFees

```
public MempoolMinFees MempoolMinFees { get; init; }
```

Property Value

[MempoolMinFees](#)

The node's view of the blockchain.

MempoolSize

```
public long MempoolSize { get; init; }
```

Property Value

[long](#)

The node's view of the blockchain.

NodeId

```
public string NodeId { get; init; }
```

Property Value

[string](#)

The node's view of the blockchain.

Peak

```
public BlockRecord? Peak { get; init; }
```

Property Value

[BlockRecord](#)

The node's view of the blockchain.

Space

```
public BigInteger Space { get; init; }
```

Property Value

[BigInteger](#)

The node's view of the blockchain.

SubSlotIter

```
public ulong SubSlotIter { get; init; }
```

Property Value

[ulong](#) ↗

The node's view of the blockchain.

Sync

```
public SyncState Sync { get; init; }
```

Property Value

[SyncState](#)

The node's view of the blockchain.

Class CATInfo

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record CATInfo : IEquatable<CATInfo>
```

Inheritance

[object](#) ← CATInfo

Implements

[IEquatable](#) <[CATInfo](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

AssetId

```
public string AssetId { get; init; }
```

Property Value

[string](#)

Name

```
public string Name { get; init; }
```

Property Value

[string](#)

Symbol

```
public string Symbol { get; init; }
```

Property Value

[string](#) ↗

Class CATWallet

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

Wraps a CAT wallet

```
public sealed class CATWallet : Wallet
```

Inheritance

[object](#) ← [Wallet](#) ← CATWallet

Inherited Members

[Wallet.WalletId](#) , [Wallet.WalletProxy](#) , [Wallet.GetWalletInfo\(CancellationToken\)](#) ,
[Wallet.GetBalance\(CancellationToken\)](#) ,
[Wallet.SelectCoins\(ulong, IEnumerable<Coin>, IEnumerable<ulong>, ulong?, ulong?, CancellationToken\)](#) ,
'
[Wallet.GetTransactions\(string, TransactionTypeFilter, bool, string, uint, uint, bool?, CancellationToken\)](#) ,
[Wallet.GetSpendableCoins\(ulong?, ulong?, IEnumerable<ulong>, IEnumerable<Coin>,
IEnumerable<string>, CancellationToken\)](#) ,
[Wallet.GetNextAddress\(bool, CancellationToken\)](#) ,
[Wallet.GetTransactionCount\(TransactionTypeFilter, CancellationToken\)](#) ,
[Wallet.DeleteUnconfirmedTransactions\(CancellationToken\)](#) ,
[Wallet.SendTransaction\(string, ulong, IEnumerable<string>, IEnumerable<ulong>, IEnumerable<string>,
ulong?, ulong?, bool, ulong, CancellationToken\)](#) ,
[Wallet.SendTransactionMulti\(IEnumerable<Coin>, IEnumerable<Coin>, ulong, CancellationToken\)](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Remarks

ctor

Constructors

CATWallet(uint, WalletProxy)

Wraps a CAT wallet

```
public CATWallet(uint walletId, WalletProxy walletProxy)
```

Parameters

walletId [uint](#)

The wallet_id to wrap

walletProxy [WalletProxy](#)

Wallet RPC proxy to use for communication

Remarks

ctor

Methods

GetAssetId(CancellationToken)

Get the asset id of a wallet's CAT

```
public Task<string> GetAssetId(CancellationToken cancellationToken = default)
```

Parameters

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[string](#)>

The asset id

GetName(CancellationToken)

Get the name of a wallet's CAT

```
public Task<string> GetName(CancellationToken cancellationToken = default)
```

Parameters

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) < string >

The CAT name

SetName(string, CancellationToken)

Set the name of the CAT

```
public Task SetName(string name, CancellationToken cancellationToken = default)
```

Parameters

name [string](#)

The new name

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

An awaitable [Task](#)

Spend(string, ulong, IEnumerable<string>?, ulong?, ulong?, IEnumerable<ulong>?, bool?, ulong, CancellationToken)

Spend a CAT

```
public Task<TransactionRecord> Spend(string innerAddress, ulong amount, IEnumerable<string>? memos = null, ulong? minCoinAmount = null, ulong? maxCoinAmount = null, IEnumerable<ulong>? excludeCoinAmounts = null, bool? reusePuzhash = null, ulong fee = 0, CancellationToken cancellationToken = default)
```

Parameters

innerAddress [string](#)

The inner address for the spend

amount [ulong](#)

The amount to put in the wallet (in units of mojos)

memos [IEnumerable](#)<[string](#)>

Optional list of byte string memos to include in the transaction

minCoinAmount [ulong](#)?

maxCoinAmount [ulong](#)?

excludeCoinAmounts [IEnumerable](#)<[ulong](#)>

reusePuzhash [bool](#)?

fee [ulong](#)

The fee to create the wallet (in units of mojos)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[TransactionRecord](#)>

A [TransactionRecord](#)

Validate(CancellationToken)

Validates that [WalletId](#) is a [CAT](#)

```
public override Task Validate(CancellationToken cancellationToken = default)
```

Parameters

cancellationToken [CancellationToken](#)

Wraps a CAT wallet

Returns

[Task](#)

True if the wallet is a CAT wallet

Class CRCATWallet

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

Wraps a CRCAT Wallet

```
public sealed class CRCATWallet : Wallet
```

Inheritance

[object](#) ← [Wallet](#) ← CRCATWallet

Inherited Members

[Wallet.WalletId](#) , [Wallet.WalletProxy](#) , [Wallet.GetWalletInfo\(CancellationToken\)](#) ,
[Wallet.GetBalance\(CancellationToken\)](#) ,
[Wallet.SelectCoins\(ulong, IEnumerable<Coin>, IEnumerable<ulong>, ulong?, ulong?, CancellationToken\)](#) ,
,
[Wallet.GetTransactions\(string, TransactionTypeFilter, bool, string, uint, uint, bool?, CancellationToken\)](#) ,
[Wallet.GetSpendableCoins\(ulong?, ulong?, IEnumerable<ulong>, IEnumerable<Coin>,
IEnumerable<string>, CancellationToken\)](#) ,
[Wallet.GetNextAddress\(bool, CancellationToken\)](#) ,
[Wallet.GetTransactionCount\(TransactionTypeFilter, CancellationToken\)](#) ,
[Wallet.DeleteUnconfirmedTransactions\(CancellationToken\)](#) ,
[Wallet.SendTransaction\(string, ulong, IEnumerable<string>, IEnumerable<ulong>, IEnumerable<string>,
ulong?, ulong?, bool, ulong, CancellationToken\)](#) ,
[Wallet.SendTransactionMulti\(IEnumerable<Coin>, IEnumerable<Coin>, ulong, CancellationToken\)](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Remarks

ctor

Constructors

CRCATWallet(uint, WalletProxy)

Wraps a CRCAT Wallet

```
public CRCATWallet(uint walletId, WalletProxy walletProxy)
```

Parameters

walletId [uint](#)?

The wallet_id to wrap

walletProxy [WalletProxy](#)

Wallet RPC proxy to use for communication

Remarks

ctor

Methods

ApprovePending(ulong?, bool?, ulong, CancellationToken)

Moving any "pending approval" CR-CATs into the spendable balance of the wallet.

```
public Task<IEnumerable<TransactionRecord>> ApprovePending(ulong? minAmountToClaim = null,  
bool? reusePuzhash = null, ulong fee = 0, CancellationToken cancellationToken = default)
```

Parameters

minAmountToClaim [ulong](#)?

The minimum amount to claim (in units of mojos)

reusePuzhash [bool](#)?

fee [ulong](#)

Fee (in units of mojos)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <IEnumerable<[TransactionRecord](#)>>

A list of [TransactionRecord](#)

Validate(CancellationToken)

Validates that [WalletId](#) is a [CRCAT](#)

```
public override Task Validate(CancellationToken cancellationToken = default)
```

Parameters

cancellationToken [CancellationToken](#)

Wraps a CRCAT Wallet

Returns

[Task](#)

True if the wallet is a pooling wallet

Class ChallengeChainSubSlot

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record ChallengeChainSubSlot : IEquatable<ChallengeChainSubSlot>
```

Inheritance

[object](#) ← ChallengeChainSubSlot

Implements

[IEquatable](#)<[ChallengeChainSubSlot](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

ChallengeChainEndOfSlotVdf

```
public VDFInfo ChallengeChainEndOfSlotVdf { get; init; }
```

Property Value

[VDFInfo](#)

InfusedChallengeChainSubSlotHash

Only at the end of a slot

```
public string? InfusedChallengeChainSubSlotHash { get; init; }
```

Property Value

[string](#)

NewDifficulty

Only at the end of epoch, sub-epoch, and slot

```
public ulong? NewDifficulty { get; init; }
```

Property Value

[ulong](#)?

NewSubSlotIters

Only at the end of epoch, sub-epoch, and slot

```
public ulong? NewSubSlotIters { get; init; }
```

Property Value

[ulong](#)?

SubepochSummaryHash

Only once per sub-epoch, and one sub-epoch delayed

```
public string? SubepochSummaryHash { get; init; }
```

Property Value

[string](#)?

Class ClassgroupElement

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

Represents a classgroup element (a,b,c) where a, b, and c are 512 bit signed integers. However this is using a compressed representation. VDF outputs are a single classgroup element. VDF proofs can also be one classgroup element(or multiple).

```
public record ClassgroupElement : IEquatable<ClassgroupElement>
```

Inheritance

[object](#) ← ClassgroupElement

Implements

[IEquatable](#) <[ClassgroupElement](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Data

```
public string Data { get; init; }
```

Property Value

[string](#)

Represents a classgroup element (a,b,c) where a, b, and c are 512 bit signed integers. However this is using a compressed representation. VDF outputs are a single classgroup element. VDF proofs can also be one classgroup element(or multiple).

Class Coin

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

This structure is used in the body for the reward and fees genesis coins.

```
public record Coin : IEquatable<Coin>
```

Inheritance

[object](#) ← Coin

Implements

[IEquatable](#) <Coin>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Amount

```
public BigInteger Amount { get; init; }
```

Property Value

[BigInteger](#)

This structure is used in the body for the reward and fees genesis coins.

AmountHex

The [Amount](#) as a hex string

```
[JsonIgnore]  
public string AmountHex { get; }
```

Property Value

[string](#) ↗

This structure is used in the body for the reward and fees genesis coins.

Name

SHA256 hash of [ParentCoinInfo](#), [PuzzleHash](#), and [AmountHex](#)

```
[JsonIgnore]  
public string Name { get; }
```

Property Value

[string](#) ↗

This structure is used in the body for the reward and fees genesis coins.

ParentCoinInfo

```
public string ParentCoinInfo { get; init; }
```

Property Value

[string](#) ↗

This structure is used in the body for the reward and fees genesis coins.

PuzzleHash

```
public string PuzzleHash { get; init; }
```

Property Value

[string](#) ↗

This structure is used in the body for the reward and fees genesis coins.

Class CoinAnnouncement

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record CoinAnnouncement : IEquatable<CoinAnnouncement>
```

Inheritance

[object](#) ← CoinAnnouncement

Implements

[IEquatable](#) <[CoinAnnouncement](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

CoinId

```
public string CoinId { get; init; }
```

Property Value

[string](#)

Message

```
public string Message { get; init; }
```

Property Value

[string](#)

MorphBytes

```
public string? MorphBytes { get; init; }
```

Property Value

[string](#) ↗

Class CoinRecord

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

These are values that correspond to a CoinName that are used in keeping track of the unspent database.

```
public record CoinRecord : IEquatable<CoinRecord>
```

Inheritance

[object](#) ← CoinRecord

Implements

[IEquatable](#)<[CoinRecord](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Coin

```
public Coin Coin { get; init; }
```

Property Value

[Coin](#)

These are values that correspond to a CoinName that are used in keeping track of the unspent database.

Coinbase

```
public bool Coinbase { get; init; }
```

Property Value

[bool](#)

These are values that correspond to a CoinName that are used in keeping track of the unspent database.

ConfirmedBlockIndex

```
public uint ConfirmedBlockIndex { get; init; }
```

Property Value

[uint](#)

These are values that correspond to a CoinName that are used in keeping track of the unspent database.

DateTimestamp

Timestamp of the block at height confirmed_block_index

```
[JsonIgnore]  
public DateTime DateTimestamp { get; }
```

Property Value

[DateTime](#)

These are values that correspond to a CoinName that are used in keeping track of the unspent database.

Spent

```
public bool Spent { get; init; }
```

Property Value

[bool](#)

These are values that correspond to a CoinName that are used in keeping track of the unspent database.

SpentBlockIndex

```
public uint SpentBlockIndex { get; init; }
```

Property Value

[uint](#)

These are values that correspond to a CoinName that are used in keeping track of the unspent database.

Timestamp

Timestamp of the block at height confirmed_block_index

```
public ulong Timestamp { get; init; }
```

Property Value

[ulong](#)

These are values that correspond to a CoinName that are used in keeping track of the unspent database.

Enum CoinRecordOrder

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public enum CoinRecordOrder : byte
```

Fields

ConfirmedHeight = 1

SpentHeight = 2

Class CoinSpend

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

This is a rather disparate data structure that validates coin transfers. It's generally populated with data from different sources, since burned coins are identified by name, so it is built up more often than it is streamed.

```
public record CoinSpend : IEquatable<CoinSpend>
```

Inheritance

[object](#) ← CoinSpend

Implements

[IEquatable](#)<[CoinSpend](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Coin

```
public Coin Coin { get; init; }
```

Property Value

[Coin](#)

This is a rather disparate data structure that validates coin transfers. It's generally populated with data from different sources, since burned coins are identified by name, so it is built up more often than it is streamed.

PuzzleReveal

```
public string PuzzleReveal { get; init; }
```

Property Value

[string ↗](#)

This is a rather disparate data structure that validates coin transfers. It's generally populated with data from different sources, since burned coins are identified by name, so it is built up more often than it is streamed.

Solution

```
public string Solution { get; init; }
```

Property Value

[string ↗](#)

This is a rather disparate data structure that validates coin transfers. It's generally populated with data from different sources, since burned coins are identified by name, so it is built up more often than it is streamed.

Enum CoinType

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public enum CoinType : byte
```

Fields

Clawback = 1

CrCat = 3

CrCatPending = 2

Normal = 0

Class Condition

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

This type doesn't exist in the chia code. This property is serialized into the Json as a tuple, which shows up as a mixed type json array. There is a specialized converter to read the Json and get the ConditionOpCode into [ConditionOpcode](#) conditions: List[Tuple[ConditionOpcode, List[ConditionWithArgs]]]

```
[JsonConverter(typeof(ConditionConverter))]  
public record Condition : IEquatable<Condition>
```

Inheritance

[object](#) ← Condition

Implements

[IEquatable](#)<[Condition](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Args

```
public IEnumerable<ConditionWithVars> Args { get; init; }
```

Property Value

[IEnumerable](#)<[ConditionWithVars](#)>

This type doesn't exist in the chia code. This property is serialized into the Json as a tuple, which shows up as a mixed type json array. There is a specialized converter to read the Json and get the ConditionOpCode into conditions: List[Tuple[ConditionOpcode, List[ConditionWithArgs]]]

ConditionOpcode

```
public string ConditionOpcode { get; init; }
```

Property Value

[string](#) ↗

This type doesn't exist in the chia code. This property is serialized into the Json as a tuple, which shows up as a mixed type json array. There is a specialized converter to read the Json and get the ConditionOpCode into conditions: List[Tuple[ConditionOpcode, List[ConditionWithArgs]]]

Class ConditionValidTimes

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record ConditionValidTimes : IEquatable<ConditionValidTimes>
```

Inheritance

[object](#) ← ConditionValidTimes

Implements

[IEquatable](#)<[ConditionValidTimes](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

MaxBlocksAfterCreated

```
public uint? MaxBlocksAfterCreated { get; init; }
```

Property Value

[uint](#)?

MaxHeight

```
public uint? MaxHeight { get; init; }
```

Property Value

[uint](#)?

MaxSecAfterCreated

```
public ulong? MaxSecAfterCreated { get; init; }
```

Property Value

[ulong](#)?

MaxTime

```
public ulong? MaxTime { get; init; }
```

Property Value

[ulong](#)?

MinBlocksSinceCreated

```
public ulong? MinBlocksSinceCreated { get; init; }
```

Property Value

[ulong](#)?

MinHeight

```
public uint? MinHeight { get; init; }
```

Property Value

[uint](#)?

MinSecsSinceCreated

```
public ulong? MinSecsSinceCreated { get; init; }
```

Property Value

ulong?

MinTime

```
public ulong? MinTime { get; init; }
```

Property Value

ulong?

Class ConditionWithVars

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

This structure is used to store parsed CLVM conditions Conditions in CLVM have either format of(opcode, var1) or(opcode, var1, var2)

```
public record ConditionWithVars : IEquatable<ConditionWithVars>
```

Inheritance

[object](#) ← ConditionWithVars

Implements

[IEquatable](#)<[ConditionWithVars](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Opcode

```
public ushort Opcode { get; init; }
```

Property Value

[ushort](#)

This structure is used to store parsed CLVM conditions Conditions in CLVM have either format of(opcode, var1) or(opcode, var1, var2)

Vars

```
public IEnumerable<string> Vars { get; init; }
```

Property Value

[IEnumerable](#)<[string](#)>

This structure is used to store parsed CLVM conditions Conditions in CLVM have either format
of(opcode, var1) or(opcode, var1, var2)

Class Config

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

Represents a chia config yaml file and its contents. Used to find the uri and ssl certs needed to connect

```
public sealed class Config
```

Inheritance

[object](#) ← Config

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

ChiaRootPath

Full path to the chia root

```
public string ChiaRootPath { get; }
```

Property Value

[string](#)

Represents a chia config yaml file and its contents. Used to find the uri and ssl certs needed to connect

Contents

The contents of the config yaml

```
public dynamic Contents { get; init; }
```

Property Value

dynamic

Represents a chia config yaml file and its contents. Used to find the uri and ssl certs needed to connect

DefaultRootPath

The OS specific default location of the chia root folder (respects CHIA_ROOT)

```
public static string DefaultRootPath { get; }
```

Property Value

[string](#)

Represents a chia config yaml file and its contents. Used to find the uri and ssl certs needed to connect

Methods

GetEndpoint(string)

Creates an [EndpointInfo](#) from the named service section

```
public EndpointInfo GetEndpoint(string serviceName)
```

Parameters

[serviceName](#) [string](#)

The setion name in the config file. Use 'daemon' for the root config that include 'self_hostname'; i.e. the local daemon

Returns

[EndpointInfo](#)

An [EndpointInfo](#) that can be used to connect to the given service's RPC interface

Open()

Opens the [Config](#) from [DefaultRootPath](#) plus 'config' and 'config.yaml'

```
public static Config Open()
```

Returns

[Config](#)

The user's chia install [Config](#) instance

Open(string)

Opens a chia config yaml file

```
public static Config Open(string fullPath)
```

Parameters

fullPath [string](#) ↗

The full filesystem path to the config file

Returns

[Config](#)

The config [Config](#) instance

Class ConnectionInfo

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

Chia's representation of a connection from node to node

```
public record ConnectionInfo : IEquatable<ConnectionInfo>
```

Inheritance

[object](#) ← ConnectionInfo

Implements

[IEquatable](#)<[ConnectionInfo](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

BytesRead

```
public uint? BytesRead { get; init; }
```

Property Value

[uint](#)?

Chia's representation of a connection from node to node

BytesWritten

```
public uint? BytesWritten { get; init; }
```

Property Value

[uint](#)?

Chia's representation of a connection from node to node

CreationDateTime

```
[JsonIgnore]  
public DateTime CreationDateTime { get; }
```

Property Value

[DateTime](#)

Chia's representation of a connection from node to node

CreationTime

```
public double CreationTime { get; init; }
```

Property Value

[double](#)

Chia's representation of a connection from node to node

IsLocal

Flag indicating whether the peer connection is local to the node

```
[JsonIgnore]  
public bool IsLocal { get; }
```

Property Value

[bool](#)

Chia's representation of a connection from node to node

LastMessageDateTime

```
[JsonIgnore]  
public DateTime LastMessageDateTime { get; }
```

Property Value

[DateTime](#)

Chia's representation of a connection from node to node

LastMessageTime

```
public double LastMessageTime { get; init; }
```

Property Value

[double](#)

Chia's representation of a connection from node to node

LocalPort

```
public int LocalPort { get; init; }
```

Property Value

[int](#)

Chia's representation of a connection from node to node

NodeId

```
public string NodeId { get; init; }
```

Property Value

[string](#) ↗

Chia's representation of a connection from node to node

PeakHash

```
public string PeakHash { get; init; }
```

Property Value

[string](#) ↗

Chia's representation of a connection from node to node

PeakHeight

```
public uint? PeakHeight { get; init; }
```

Property Value

[uint](#) ↗?

Chia's representation of a connection from node to node

PeakWeight

```
public ulong? PeakWeight { get; init; }
```

Property Value

[ulong](#) ↗?

Chia's representation of a connection from node to node

PeerHost

```
public string PeerHost { get; init; }
```

Property Value

[string](#) ↗

Chia's representation of a connection from node to node

PeerPort

```
public int PeerPort { get; init; }
```

Property Value

[int](#) ↗

Chia's representation of a connection from node to node

PeerServerPort

```
public int PeerServerPort { get; init; }
```

Property Value

[int](#) ↗

Chia's representation of a connection from node to node

Type

```
public NodeType Type { get; init; }
```

Property Value

NodeType

Chia's representation of a connection from node to node

Class CrawlerProxy

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

Proxy that communicates with the crawler

```
public sealed class CrawlerProxy : ServiceProxy
```

Inheritance

[object](#) ← [ServiceProxy](#) ← CrawlerProxy

Inherited Members

[ServiceProxy.OriginService](#) , [ServiceProxy.DestinationService](#) , [ServiceProxy.RpcClient](#) ,
[ServiceProxy.HealthZ\(CancellationToken\)](#) , [ServiceProxy.StopNode\(CancellationToken\)](#) ,
[ServiceProxy.GetConnections\(CancellationToken\)](#) , [ServiceProxy.GetRoutes\(CancellationToken\)](#) ,
[ServiceProxy.OpenConnection\(string, int, CancellationToken\)](#) ,
[ServiceProxy.CloseConnection\(string, CancellationToken\)](#) , [object.Equals\(object\)](#) ,
[object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Remarks

ctor

Constructors

CrawlerProxy(IRpcClient, string)

Proxy that communicates with the crawler

```
public CrawlerProxy(IRpcClient rpcClient, string originService)
```

Parameters

rpcClient [IRpcClient](#)

[IRpcClient](#) instance to use for rpc communication

`originService` [string](#)

[Origin](#)

Remarks

ctor

Methods

GetIPs(DateTime, int, int, CancellationToken)

Retrieves ip addresses of peers that have connected after a given time

```
public Task<(IEnumerable<string> ips, int total)> GetIPs(DateTime after, int offset = 0, int limit = 10000, CancellationToken cancellationToken = default)
```

Parameters

`after` [DateTime](#)

`offset` [int](#)

`limit` [int](#)

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<(IEnumerable<string> ips, int total)>

IP addresses

GetPeerCounts(CancellationToken)

Retrieves aggregate information about peers

```
public Task<PeerCounts> GetPeerCounts(CancellationToken cancellationToken = default)
```

Parameters

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[PeerCounts](#)>

Information about peers

Class DAOInfo

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record DAOInfo : IEquatable<DAOInfo>
```

Inheritance

[object](#) ← DAOInfo

Implements

[IEquatable](#)<[DAOInfo](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Assets

```
public IEnumerable<string> Assets { get; init; }
```

Property Value

[IEnumerable](#)<[string](#)>

CatWalletId

```
public uint CatWalletId { get; init; }
```

Property Value

[uint](#)

CurrentHeight

```
public uint CurrentHeight { get; init; }
```

Property Value

[uint](#)

CurrentTreasuryCoin

```
public Coin? CurrentTreasuryCoin { get; init; }
```

Property Value

[Coin](#)

CurrentTreasuryInnerpuz

```
public string? CurrentTreasuryInnerpuz { get; init; }
```

Property Value

[string](#)

DaoCatWalletId

```
public uint DaoCatWalletId { get; init; }
```

Property Value

[uint](#)

FilterBelowVoteAmount

we ignore proposals with fewer votes than this - defaults to 1

```
public ulong FilterBelowVoteAmount { get; init; }
```

Property Value

[ulong](#)

ParentInfo

```
public IEnumerable<IDictionary<string, LineageProof?>> ParentInfo { get; init; }
```

Property Value

[IEnumerable](#)<[IDictionary](#)<[string](#), [LineageProof](#)>>

ProposalsList

```
public IEnumerable<ProposalInfo> ProposalsList { get; init; }
```

Property Value

[IEnumerable](#)<[ProposalInfo](#)>

SingletonBlockHeight

the block height that the current treasury singleton was created in

```
public uint SingletonBlockHeight { get; init; }
```

Property Value

[uint](#)

TreasuryId

```
public string TreasuryId { get; init; }
```

Property Value

[string](#) ↗

Class DAORules

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record DAORules : IEquatable<DAORules>
```

Inheritance

[object](#) ← DAORules

Implements

[IEquatable](#)<[DAORules](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

AttendanceRequired

```
public ulong AttendanceRequired { get; init; }
```

Property Value

[ulong](#)

OracleSpendDelay

```
public ulong OracleSpendDelay { get; init; }
```

Property Value

[ulong](#)

PassPercentage

```
public ulong PassPercentage { get; init; }
```

Property Value

[ulong](#) ↗

ProposalMinimumAmount

```
public ulong ProposalMinimumAmount { get; init; }
```

Property Value

[ulong](#) ↗

ProposalTimelock

```
public ulong ProposalTimelock { get; init; }
```

Property Value

[ulong](#) ↗

SelfDestructLength

```
public ulong SelfDestructLength { get; init; }
```

Property Value

[ulong](#) ↗

SoftCloseLength

```
public ulong SoftCloseLength { get; init; }
```

Property Value

[ulong](#) ↗

Class DAOWallet

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

Wraps a DAO Wallet

```
public sealed class DAOWallet : Wallet
```

Inheritance

[object](#) ← [Wallet](#) ← DAOWallet

Inherited Members

[Wallet.WalletId](#) , [Wallet.WalletProxy](#) , [Wallet.GetWalletInfo\(CancellationToken\)](#) ,
[Wallet.GetBalance\(CancellationToken\)](#) ,
[Wallet.SelectCoins\(ulong, IEnumerable<Coin>, IEnumerable<ulong>, ulong?, ulong?, CancellationToken\)](#) ,
'
[Wallet.GetTransactions\(string, TransactionTypeFilter, bool, string, uint, uint, bool?, CancellationToken\)](#) ,
[Wallet.GetSpendableCoins\(ulong?, ulong?, IEnumerable<ulong>, IEnumerable<Coin>,
IEnumerable<string>, CancellationToken\)](#) ,
[Wallet.GetNextAddress\(bool, CancellationToken\)](#) ,
[Wallet.GetTransactionCount\(TransactionTypeFilter, CancellationToken\)](#) ,
[Wallet.DeleteUnconfirmedTransactions\(CancellationToken\)](#) ,
[Wallet.SendTransaction\(string, ulong, IEnumerable<string>, IEnumerable<ulong>, IEnumerable<string>,
ulong?, ulong?, bool, ulong, CancellationToken\)](#) ,
[Wallet.SendTransactionMulti\(IEnumerable<Coin>, IEnumerable<Coin>, ulong, CancellationToken\)](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Remarks

ctor

Constructors

DAOWallet(uint, WalletProxy)

Wraps a DAO Wallet

```
public DAOWallet(uint walletId, WalletProxy walletProxy)
```

Parameters

walletId [uint](#)

The wallet_id to wrap

walletProxy [WalletProxy](#)

Wallet RPC proxy to use for communication

Remarks

ctor

Methods

AddFundsToTreasury(ulong, uint, ulong, CancellationToken)

Adds funds to a DAO's treasury.

```
public Task<(string txId, TransactionRecord tx)> AddFundsToTreasury(ulong amount, uint fundingWalletId, ulong fee = 0, CancellationToken cancellationToken = default)
```

Parameters

amount [ulong](#)

fundingWalletId [uint](#)

fee [ulong](#)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<(string txId, TransactionRecord tx)>

AdjustFilterLevel(ulong, CancellationToken)

Adjusts the DAO filter level.

```
public Task<DAOInfo> AdjustFilterLevel(ulong filterLevel, CancellationToken  
cancellationToken = default)
```

Parameters

`filterLevel` [ulong](#)

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[DAOInfo](#)>

[DAOInfo](#)

CloseProposal(string, string, string, ulong, CancellationToken)

Closes a DAO proposal.

```
public Task<(string txId, TransactionRecord tx)> CloseProposal(string selfDestruct, string  
genesisId, string proposalId, ulong fee = 0, CancellationToken cancellationToken = default)
```

Parameters

`selfDestruct` [string](#)

`genesisId` [string](#)

`proposalId` [string](#)

`fee` [ulong](#)

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<(string [txId](#), TransactionRecord [tx](#))>

CreateProposal(string, CancellationToken)

Creates a DAO proposal.

```
public Task<(string proposalId, string txId, TransactionRecord tx)> CreateProposal(string  
proposalType, CancellationToken cancellationToken = default)
```

Parameters

proposalType [string](#)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<(string [proposalId](#), string [txId](#), TransactionRecord [tx](#))>

ExitLockup(IEnumerable<object>, ulong, CancellationToken)

Exits the DAO lockup period.

```
public Task<(string txId, TransactionRecord tx)> ExitLockup(IEnumerable<object> coins, ulong  
fee = 0, CancellationToken cancellationToken = default)
```

Parameters

coins [IEnumerable](#)<[object](#)>

fee [ulong](#)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<(string [txId](#), TransactionRecord [tx](#))>

FreeCoinsFromFinishedProposals(ulong, CancellationToken)

Frees coins from proposals that are finished.

```
public Task<(string txId, TransactionRecord tx)> FreeCoinsFromFinishedProposals(ulong fee = 0, CancellationToken cancellationToken = default)
```

Parameters

fee [ulong](#)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<(string [txId](#), TransactionRecord [tx](#))>

GetProposalState(string, CancellationToken)

Use this to figure out whether a proposal has passed or failed and whether it can be closed Given a proposal_id:

- if required yes votes are recorded then proposal passed.
- if timelock and attendance are met then proposal can close Returns a dict of passed and closable bools, and the remaining votes/blocks needed

Note that a proposal can be in a passed and closable state now, but become failed if a large number of 'no' votes are received before the soft close is reached.

```
public Task<ProposalState> GetProposalState(string proposalId, CancellationToken cancellationToken = default)
```

Parameters

`proposalId` [string](#)

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[ProposalState](#)>

[ProposalState](#)

GetProposals(bool, CancellationToken)

Get all proposals for a given dao wallet.

```
public Task<(IEnumerable<ProposalInfo> proposals, ulong proposalTimelock, ulong softCloseLength)> GetProposals(bool includeClosed = true, CancellationToken cancellationToken = default)
```

Parameters

`includeClosed` [bool](#)

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <([IEnumerable](#) <[ProposalInfo](#)> `proposals`, [ulong](#) `proposalTimelock`, [ulong](#) `softCloseLength`)>

GetRules(CancellationToken)

Retrieves the rules of a DAO wallet.

```
public Task<DAORules> GetRules(CancellationToken cancellationToken = default)
```

Parameters

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[DAORules](#)>

[DAORules](#)

GetTreasuryBalance(CancellationToken)

Retrieves the balance of a DAO's treasury.

```
public Task<IDictionary<string, BigInteger>> GetTreasuryBalance(CancellationToken  
cancellationToken = default)
```

Parameters

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[IDictionary](#)<[string](#), [BigInteger](#)>>

GetTreasuryId(CancellationToken)

Retrieves the treasury id of a DAO wallet.

```
public Task<string> GetTreasuryId(CancellationToken cancellationToken = default)
```

Parameters

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[string](#)>

[string](#)

ParseProposal(string, CancellationToken)

Parses a DAO proposal.

```
public Task<IDictionary<string, object>> ParseProposal(string proposalId, CancellationToken cancellationToken = default)
```

Parameters

proposalId [string](#)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[IDictionary](#)<[string](#), [object](#)>>

Dictionary

SendToLockup(ulong, ulong, CancellationToken)

Sends the DAO to lockup.

```
public Task<(string txId, TransactionRecord tx)> SendToLockup(ulong amount, ulong fee = 0, CancellationToken cancellationToken = default)
```

Parameters

amount [ulong](#)

`fee` `ulong`

`cancellationToken` `CancellationToken`

A token to allow the call to be cancelled

Returns

`Task`<(`string` `txId`, `TransactionRecord` `tx`)>

Validate(CancellationToken)

Validates that `WalletId` is a `DAO`

```
public override Task Validate(CancellationToken cancellationToken = default)
```

Parameters

`cancellationToken` `CancellationToken`

Wraps a DAO Wallet

Returns

`Task`

True if the wallet is a DAO wallet

VoteOnProposal(bool, ulong, string, ulong, CancellationToken)

Vote on a DAO proposal.

```
public Task<(<code>string txId, TransactionRecord tx</code>)> VoteOnProposal(<code>bool isYesVote, ulong voteAmount, string proposalId, ulong fee = 0, CancellationToken cancellationToken = default)
```

Parameters

`isYesVote` `bool`

`voteAmount` [ulong](#)

`proposalId` [string](#)

`fee` [ulong](#)

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <([string](#) `txId`, [TransactionRecord](#) `tx`)>

Class DIDWallet

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

Wraps a Distributed Identity Wallet

```
public sealed class DIDWallet : Wallet
```

Inheritance

[object](#) ← [Wallet](#) ← DIDWallet

Inherited Members

[Wallet.WalletId](#) , [Wallet.WalletProxy](#) , [Wallet.GetWalletInfo\(CancellationToken\)](#) ,
[Wallet.GetBalance\(CancellationToken\)](#) ,
[Wallet.SelectCoins\(ulong, IEnumerable<Coin>, IEnumerable<ulong>, ulong?, ulong?, CancellationToken\)](#) ,
'
[Wallet.GetTransactions\(string, TransactionTypeFilter, bool, string, uint, uint, bool?, CancellationToken\)](#) ,
[Wallet.GetSpendableCoins\(ulong?, ulong?, IEnumerable<ulong>, IEnumerable<Coin>, IEnumerable<string>, CancellationToken\)](#) ,
[Wallet.GetNextAddress\(bool, CancellationToken\)](#) ,
[Wallet.GetTransactionCount\(TransactionTypeFilter, CancellationToken\)](#) ,
[Wallet.DeleteUnconfirmedTransactions\(CancellationToken\)](#) ,
[Wallet.SendTransaction\(string, ulong, IEnumerable<string>, IEnumerable<ulong>, IEnumerable<string>, ulong?, ulong?, bool, ulong, CancellationToken\)](#) ,
[Wallet.SendTransactionMulti\(IEnumerable<Coin>, IEnumerable<Coin>, ulong, CancellationToken\)](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Remarks

ctor

Constructors

DIDWallet(uint, WalletProxy)

Wraps a Distributed Identity Wallet

```
public DIDWallet(uint walletId, WalletProxy walletProxy)
```

Parameters

walletId [uint](#)

The wallet_id to wrap

walletProxy [WalletProxy](#)

Wallet RPC proxy to use for communication

Remarks

ctor

Methods

CreateAttest(string, string, string, CancellationToken)

Create an attest file

```
public Task<(string MessageSpendBundle, (string Parent, string InnerPuzzleHash, ulong Amount) Info, string AttestData)> CreateAttest(string coinName, string pubkey, string puzHash, CancellationToken cancellationToken = default)
```

Parameters

coinName [string](#)

The coin name

pubkey [string](#)

The public key

puzHash [string](#)

The puzzlehash

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

```
Task<(string MessageSpendBundle, string Parent, string InnerPuzzleHash, ulong Amount, Info, string AttestData)>
```

A spendbundle and information about the attest

CreateBackupFile(CancellationToken)

Create a backup of the wallet

```
public Task<string> CreateBackupFile(CancellationToken cancellationToken = default)
```

Parameters

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

```
Task<string>
```

The backup data

GetCurrentCoinInfo(CancellationToken)

Gets information about the DID wallets current coin

```
public Task<(string MyDid, string Parent, string InnerPuzzle, ulong Amount)> GetCurrentCoinInfo(CancellationToken cancellationToken = default)
```

Parameters

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<([string](#) [MyDid](#), [string](#) [Parent](#), [string](#) [InnerPuzzle](#), [ulong](#) [Amount](#))>

The coin info

GetDid(CancellationToken)

Get the distributed identity and coin if present

```
public Task<(string MyDid, string? CoinID)> GetDid(CancellationToken cancellationToken  
= default)
```

Parameters

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<([string](#) [MyDid](#), [string](#) [CoinID](#))>

A DID and optional CoinID

GetInformationNeededForRecovery(CancellationToken)

Create an attestation

```
public Task<(string MyDID, string CoinName, string NewPuzzleHash, string PublicKey,  
IEnumerable<byte> BackUpIDs)> GetInformationNeededForRecovery(CancellationToken  
cancellationToken = default)
```

Parameters

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<(string [MyDID](#), string [CoinName](#), string [NewPuzzleHash](#), string [PublicKey](#),
[IEnumerable](#)<byte> [BackUplds](#))>

A spendbundle and information about the attest

GetMetadata(CancellationToken)

Updates the metadata

```
public Task<IDictionary<string, string>> GetMetadata(CancellationToken cancellationToken  
= default)
```

Parameters

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<IDictionary<string, string>>

The metadata

GetName(CancellationToken)

Get the wallet name

```
public Task<string> GetName(CancellationToken cancellationToken = default)
```

Parameters

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[string](#)>

The name

GetPubKey(CancellationToken)

Get the wallet pubkey

```
public Task<string> GetPubKey(CancellationToken cancellationToken = default)
```

Parameters

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[string](#)>

The pubkey

GetRecoveryList(CancellationToken)

Get the recover list

```
public Task<(IEnumerable<string> RecoverList, int NumRequired)>
GetRecoveryList(CancellationToken cancellationToken = default)
```

Parameters

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <(IEnumerable <[string](#)> ips, int total)>

The recover list and num required property of the wallet

MessageSpend(IEnumerable<string>, IEnumerable<string>, CancellationToken)

Spends a DID message.

```
public Task<SpendBundle> MessageSpend(IEnumerable<string> puzzleAnnouncements,  
IEnumerable<string> coinAnnouncements, CancellationToken cancellationToken = default)
```

Parameters

puzzleAnnouncements [IEnumerable<string>](#)

coinAnnouncements [IEnumerable<string>](#)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task<SpendBundle>](#)

[SpendBundle](#)

RecoverySpend(IEnumerable<string>, string?, string?, CancellationToken)

Recovery spend

```
public Task RecoverySpend(IEnumerable<string> attestData, string? pubkey, string?  
puzzlehash, CancellationToken cancellationToken = default)
```

Parameters

attestData [IEnumerable<string>](#)

List of attest messages. Must be >= num_of_backup_ids_needed

pubkey [string](#)

The public key

`puzzlehash` [string](#)

The puzzlehash of the spend

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

An awaitable [Task](#)

SetName(string, CancellationToken)

Sets the name

```
public Task SetName(string name, CancellationToken cancellationToken = default)
```

Parameters

`name` [string](#)

The name

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

An awaitable [Task](#)

Spend(string, CancellationToken)

Spend from the DID wallet

```
public Task Spend(string puzzlehash, CancellationToken cancellationToken = default)
```

Parameters

puzzlehash [string](#)

The puzzlehash to spend

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

An awaitable [Task](#)

Transfer(string, bool, bool?, ulong, CancellationToken)

Transfer the DID wallet to another owner

```
public Task<TransactionRecord> Transfer(string innerAddress, bool withRecoveryInfo = true,  
bool? reusePuzhash = null, ulong fee = 0, CancellationToken cancellationToken = default)
```

Parameters

innerAddress [string](#)

the address

withRecoveryInfo [bool](#)

Indiciator whether to include recovery infor

reusePuzhash [bool](#)?

fee [ulong](#)

Trasnaction fee

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) < [TransactionRecord](#) >

The backup data

UpdateMetadata(string, bool?, ulong, CancellationToken)

Updates the metadata

```
public Task<SpendBundle> UpdateMetadata(string metadata, bool? reusePuzhash = null, ulong fee = 0, CancellationToken cancellationToken = default)
```

Parameters

metadata [string](#) ↗

The name

reusePuzhash [bool](#) ↗?

fee [ulong](#) ↗

Transaction fee

cancellationToken [CancellationToken](#) ↗

A token to allow the call to be cancelled

Returns

[Task](#) < [SpendBundle](#) >

An awaitable [Task](#) ↗

UpdateRecoveryIds(IEnumerable<string>, ulong?, bool?, CancellationToken)

Updates recovery ID's

```
public Task UpdateRecoveryIds(IEnumerable<string> newList, ulong? numVerificationsRequired = null, bool? reusePuzhash = null, CancellationToken cancellationToken = default)
```

Parameters

newList [IEnumerable](#)<[string](#)>

The new ids

numVerificationsRequired [ulong](#)?

The number of verifications required

reusePuzhash [bool](#)?

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

An awaitable [Task](#)

UpdateRecoveryIds(IEnumerable<string>, ulong, CancellationToken)

Updates recovery ID's

```
public Task UpdateRecoveryIds(IEnumerable<string> newList, ulong numVerificationsRequired, CancellationToken cancellationToken = default)
```

Parameters

newList [IEnumerable](#)<[string](#)>

The new ids

numVerificationsRequired [ulong](#)

The number of verifications required

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

An awaitable [Task](#)

Validate(CancellationToken)

Validates that [WalletId](#) is a [DISTRIBUTED_ID](#)

```
public override Task Validate(CancellationToken cancellationToken = default)
```

Parameters

cancellationToken [CancellationToken](#)

Wraps a Distributed Identity Wallet

Returns

[Task](#)

True if the wallet is a DID wallet

Class DaemonProxy

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

[WebSocketRpcClient](#) for the daemon interface. The daemon can be used to proxy messages to and from other chia services as well as controlling the [PlotterProxy](#) and having it's own procedures

```
public sealed class DaemonProxy : ServiceProxy
```

Inheritance

[object](#) ← [ServiceProxy](#) ← DaemonProxy

Inherited Members

[ServiceProxy.OriginService](#) , [ServiceProxy.DestinationService](#) , [ServiceProxy.RpcClient](#) ,
[ServiceProxy.HealthZ\(CancellationToken\)](#) , [ServiceProxy.StopNode\(CancellationToken\)](#) ,
[ServiceProxy.GetConnections\(CancellationToken\)](#) , [ServiceProxy.GetRoutes\(CancellationToken\)](#) ,
[ServiceProxy.OpenConnection\(string, int, CancellationToken\)](#) ,
[ServiceProxy.CloseConnection\(string, CancellationToken\)](#) , [object.Equals\(object\)](#) ,
[object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Remarks

ctor

Constructors

DaemonProxy(WebSocketRpcClient, string)

[WebSocketRpcClient](#) for the daemon interface. The daemon can be used to proxy messages to and from other chia services as well as controlling the [PlotterProxy](#) and having it's own procedures

```
public DaemonProxy(WebSocketRpcClient rpcClient, string originService)
```

Parameters

rpcClient [WebSocketRpcClient](#)

[IRpcClient](#) instance to use for rpc communication

originService [string](#)

[Origin](#)

Remarks

ctor

Methods

AddPrivateKey(string, string, CancellationToken)

Adds a private key to the keychain, with the given entropy and passphrase. The keychain itself will store the public key, and the entropy bytes, but not the passphrase.

```
public Task<uint> AddPrivateKey(string mnemonic, string passphrase, CancellationToken  
cancellationToken = default)
```

Parameters

mnemonic [string](#)

Mnemonic entropy of the key

passphrase [string](#)

Keyring passphrase

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[uint](#)>

Awaitable [Task](#)

CheckKeys(string, CancellationToken)

Checks the keys

```
public Task CheckKeys(string rootPath, CancellationToken cancellationToken = default)
```

Parameters

rootPath [string](#)

The config root path

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

Awaitable [Task](#)

Remarks

This seems to send the daemon out to lunch

CreateProxyFrom<T>()

Create a new derived [ServiceProxy](#) instance sharing this daemon's [RpcClient](#)

```
public T CreateProxyFrom<T>() where T : ServiceProxy
```

Returns

T

The [ServiceProxy](#)

Type Parameters

T

The type of [ServiceProxy](#) to create

Remarks

This only works for daemons because they can forward messages to other services through their [Web SocketRpcClient](#)

DeleteAllKeys(CancellationToken)

Deletes all keys from the keychain

```
public Task DeleteAllKeys(CancellationToken cancellationToken = default)
```

Parameters

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

Awaitable [Task](#)

DeleteKeyByFingerprint(uint, CancellationToken)

Deletes all keys which have the given public key fingerprint

```
public Task DeleteKeyByFingerprint(uint fingerprint, CancellationToken cancellationToken = default)
```

Parameters

`fingerprint` [uint](#)

The fingerprint

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

Awaitable [Task](#)

DeleteLabel(uint, CancellationToken)

Removes the label assigned to the key with the given fingerprint.

```
public Task DeleteLabel(uint fingerprint, CancellationToken cancellationToken = default)
```

Parameters

fingerprint [uint](#)

The fingerprint

cancellationToken [CancellationToken](#)

Returns

[Task](#)

Exit(CancellationToken)

Tells the daemon at the RPC endpoint to exit.

```
public Task Exit(CancellationToken cancellationToken = default)
```

Parameters

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

Awaitable [Task](#)

Remarks

There isn't a way to start the daemon remotely via RPC, so take care that you have access to the RPC host if needed

GetAllPrivateKeys(CancellationToken)

Returns all private keys as a tuple of key, and entropy bytes (i.e. mnemonic) for each key.

```
public Task<IEnumerable<PrivateKeyData>> GetAllPrivateKeys(CancellationToken  
cancellationToken = default)
```

Parameters

[cancellationToken](#) [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[IEnumerable](#)<[PrivateKeyData](#)>>

All of the [PrivateKeys](#)

GetFirstPrivateKey(CancellationToken)

Returns the first private key

```
public Task<PrivateKeyData> GetFirstPrivateKey(CancellationToken cancellationToken  
= default)
```

Parameters

[cancellationToken](#) [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[PrivateKeyData](#)>

The first [PrivateKey](#).

GetKey(uint, bool, CancellationToken)

Locates and returns KeyData matching the provided fingerprint

```
public Task<KeyData> GetKey(uint fingerprint, bool includeSecrets = false, CancellationToken cancellationToken = default)
```

Parameters

fingerprint [uint](#)

The fingerprint

includeSecrets [bool](#)

Include secrets

cancellationToken [CancellationToken](#)

Returns

[Task](#) <[KeyData](#)>

GetKeyForFingerprint(uint, CancellationToken)

Gets the private key associated with the given fingerprint

```
public Task<PrivateKeyData> GetKeyForFingerprint(uint fingerprint, CancellationToken cancellationToken = default)
```

Parameters

fingerprint [uint](#)

The fingerprint

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[PrivateKeyData](#)>

The [PrivateKey](#)

GetKeyringStatus(CancellationToken)

Retrieves the status of the keyring

```
public Task<KeyringStatus> GetKeyringStatus(CancellationToken cancellationToken = default)
```

Parameters

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[KeyringStatus](#)>

Awaitable [Task](#)

GetKeys(uint, bool, CancellationToken)

Returns the KeyData of all keys which can be retrieved

```
public Task<IEnumerable<KeyData>> GetKeys(uint fingerprint, bool includeSecrets = false, CancellationToken cancellationToken = default)
```

Parameters

fingerprint [uint](#)

The fingerprint

`includeSecrets bool`

Include secrets

`cancellationToken CancellationToken`

Returns

`Task<IEnumerable<KeyData>>`

GetStatus(CancellationToken)

Get whether the genesis block has been initialized

```
public Task<bool> GetStatus(CancellationToken cancellationToken = default)
```

Parameters

`cancellationToken CancellationToken`

A token to allow the call to be cancelled

Returns

`Task<bool>`

Boolean indicator

GetVersion(CancellationToken)

Get the installed version of chia at the endpoint

```
public Task<string> GetVersion(CancellationToken cancellationToken = default)
```

Parameters

`cancellationToken CancellationToken`

A token to allow the call to be cancelled

Returns

[Task](#)<[string](#)>

The chia version as a string

GetWalletAddresses(IEnumerable<uint>?, bool, uint, uint, CancellationToken)

Returns the list of addresses.

```
public Task<IDictionary<uint, IEnumerable<WalletAddress>>>
GetWalletAddresses(IEnumerable<uint>? fingerprints = null, bool nonObserverDerivation =
false, uint index = 0, uint count = 1, CancellationToken cancellationToken = default)
```

Parameters

fingerprints [IEnumerable](#)<[uint](#)>

nonObserverDerivation [bool](#)

index [uint](#)

count [uint](#)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[IDictionary](#)<[uint](#), [IEnumerable](#)<[WalletAddress](#)>>>

A dictionary of fingerprints and [WalletAddress](#)

IsKeyringLocked(CancellationToken)

Determine if the keyring is locked

```
public Task<bool> IsKeyringLocked(CancellationToken cancellationToken = default)
```

Parameters

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <bool>

Boolean indicator as to whether the keyring is locked

IsRunning(string, CancellationToken)

Determines if the named service is running.

```
public Task<bool> IsRunning(string service, CancellationToken cancellationToken = default)
```

Parameters

service [string](#)

The service name

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <bool>

Boolean indicator as to whether the service is running

Ping(CancellationToken)

Sends ping message to the service

```
public Task Ping(CancellationToken cancellationToken = default)
```

Parameters

cancellationToken [CancellationToken ↗](#)

A token to allow the call to be cancelled

Returns

[Task ↗](#)

Awaitable [Task ↗](#)

RegisterService(string, CancellationToken)

Registers this daemon to receive messages. This is needed to receive responses from services other than the daemon. This is not a [ServiceNames](#) but usually the name of the consumer application such as 'wallet_ui'

```
public Task RegisterService(string service, CancellationToken cancellationToken = default)
```

Parameters

service [string ↗](#)

The name to register

cancellationToken [CancellationToken ↗](#)

A token to allow the call to be cancelled

Returns

[Task ↗](#)

Awaitable [Task ↗](#)

RegisterService(CancellationToken)

Registers this websocket to receive messages using [OriginService](#) This is needed to receive responses from services other than the daemon.

```
public Task RegisterService(CancellationToken cancellationToken = default)
```

Parameters

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

Awaitable [Task](#)

RemoveKeyringPassphrase(string, CancellationToken)

Remove the key ring passphrase

```
public Task RemoveKeyringPassphrase(string currentPassphrase, CancellationToken cancellationToken = default)
```

Parameters

`currentPassphrase` [string](#)

The current keyring passphrase

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

Awaitable [Task](#)

RunningServices(CancellationToken)

Get the list of running services

```
public Task<IEnumerable<string>> RunningServices(CancellationToken cancellationToken = default)
```

Parameters

`cancellationToken CancellationToken`

A token to allow the call to be cancelled

Returns

`Task<IEnumerable<string>>`

A list of services

SetKeyringPassphrase(string, string, string, bool, CancellationToken)

Update the key ring passphrase

```
public Task SetKeyringPassphrase(string currentPassphrase, string newPassphrase, string passphraseHint, bool savePassphrase = false, CancellationToken cancellationToken = default)
```

Parameters

`currentPassphrase string`

The current keyring passphrase

`newPassphrase string`

The new keyring passphrase

`passphraseHint string`

A passphrase hint

`savePassphrase` [bool](#)

Should the passphrase be saved

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

Awaitable [Task](#)

SetLabel(uint, string, CancellationToken)

Assigns the given label to the first key with the given fingerprint.

```
public Task SetLabel(uint fingerprint, string label, CancellationToken cancellationToken  
= default)
```

Parameters

`fingerprint` [uint](#)

The fingerprint

`label` [string](#)

The label

`cancellationToken` [CancellationToken](#)

Returns

[Task](#)

StartService(string, CancellationToken)

Starts the named service.

```
public Task StartService(string service, CancellationToken cancellationToken = default)
```

Parameters

service [string](#)

The [ServiceNames](#) of the service

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

Awaitable [Task](#)

StopService(string, CancellationToken)

Stops the named service

```
public Task StopService(string service, CancellationToken cancellationToken = default)
```

Parameters

service [string](#)

The [ServiceNames](#) of the service

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

Awaitable [Task](#)

UnlockKeyring(string, CancellationToken)

Unlock the keyring

```
public Task UnlockKeyring(string key, CancellationToken cancellationToken = default)
```

Parameters

key [string](#)

Keyring passphrase

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

Awaitable [Task](#)

ValidateKeyringPassphrase(string, CancellationToken)

Test the validity of a passphrase

```
public Task ValidateKeyringPassphrase(string key, CancellationToken cancellationToken = default)
```

Parameters

key [string](#)

Keyring passphrase

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task ↗](#)

Awaitable [Task ↗](#)

Class DataLayerOffer

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record DataLayerOffer : IEquatable<DataLayerOffer>
```

Inheritance

[object](#) ← DataLayerOffer

Implements

[IEquatable](#)<[DataLayerOffer](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Maker

```
public StoreProofs Maker { get; init; }
```

Property Value

[StoreProofs](#)

Offer

```
public string Offer { get; init; }
```

Property Value

[string](#)

Taker

```
public OfferStore Taker { get; init; }
```

Property Value

[OfferStore](#)

TradeId

```
public string TradeId { get; init; }
```

Property Value

[string](#) ↗

Class DataLayerProxy

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

Proxy that communicates with the Data Layer

```
public sealed class DataLayerProxy : ServiceProxy
```

Inheritance

[object](#) ← [ServiceProxy](#) ← DataLayerProxy

Inherited Members

[ServiceProxy.OriginService](#) , [ServiceProxy.DestinationService](#) , [ServiceProxy.RpcClient](#) ,
[ServiceProxy.HealthZ\(CancellationToken\)](#) , [ServiceProxy.StopNode\(CancellationToken\)](#) ,
[ServiceProxy.GetConnections\(CancellationToken\)](#) , [ServiceProxy.GetRoutes\(CancellationToken\)](#) ,
[ServiceProxy.OpenConnection\(string, int, CancellationToken\)](#) ,
[ServiceProxy.CloseConnection\(string, CancellationToken\)](#) , [object.Equals\(object\)](#) ,
[object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Remarks

ctor

Constructors

DataLayerProxy(IRpcClient, string)

Proxy that communicates with the Data Layer

```
public DataLayerProxy(IRpcClient rpcClient, string originService)
```

Parameters

rpcClient [IRpcClient](#)

[IRpcClient](#) instance to use for rpc communication

`originService` [string](#)

[Origin](#)

Remarks

ctor

Methods

`AddMirror(string, ulong, IEnumerable<string>, ulong, CancellationToken)`

Adds a mirror

```
public Task AddMirror(string id, ulong amount, IEnumerable<string> urls, ulong fee = 0, CancellationToken cancellationToken = default)
```

Parameters

`id` [string](#)

Mirror id

`amount` [ulong](#)

The Amount

`urls` [IEnumerable](#)<[string](#)>

List of mirror urls

`fee` [ulong](#)

Fee amount (in units of mojos)

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

An awaitable Task

AddMissingFiles(string[], string, bool, CancellationToken)

Adds missing files

```
public Task AddMissingFiles(string[] ids, string foldername, bool overwrite = false,  
CancellationToken cancellationToken = default)
```

Parameters

ids [string](#)[]

List of file id's

foldername [string](#)

The folder name

overwrite [bool](#)

Indicator whetehr to overwrite files

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

An awaitable Task

BatchUpdate(string, IDictionary<string, string>, ulong, CancellationToken)

Applies a batch of updates.

```
public Task<string> BatchUpdate(string id, IDictionary<string, string> changeList, ulong fee = 0, CancellationToken cancellationToken = default)
```

Parameters

id [string](#)

Id

changeList [IDictionary](#)<[string](#), [string](#)>

Name value pairs of changes

fee [ulong](#)

Fee amount (in units of mojos)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[string](#)>

Transaction id

CancelOffer(string, bool, ulong, CancellationToken)

Cancels an offer using a transaction

```
public Task CancelOffer(string tradeId, bool secure = false, ulong fee = 0, CancellationToken cancellationToken = default)
```

Parameters

tradeId [string](#)

The trade id of the offer

secure [bool](#)

This will create a transaction that includes coins that were offered

fee [ulong](#)

Transaction fee

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

An awaitable Task

CheckPlugins(CancellationToken)

Checks the status of plugins.

```
public Task<PluginStatus> CheckPlugins(CancellationToken cancellationToken = default)
```

Parameters

cancellationToken [CancellationToken](#)

Proxy that communicates with the Data Layer

Returns

[Task](#) <[PluginStatus](#)>

[PluginStatus](#)

ClearPendingRoots(string, CancellationToken)

Clears pending roots.

```
public Task<Root> ClearPendingRoots(string storeId, CancellationToken cancellationToken = default)
```

Parameters

`storeId` [string](#)

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[Root](#)>

[Root](#)

CreateDataStore(ulong, CancellationToken)

Creates a data store.

```
public Task<(string id, IEnumerable<TransactionRecord> txs)> CreateDataStore(ulong fee = 0,  
CancellationToken cancellationToken = default)
```

Parameters

`fee` [ulong](#)

Fee amount (in units of mojos)

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <(string id, IEnumerable<TransactionRecord> txs)>

The tree id and list of transactions

DeleteKey(string, string, ulong, CancellationToken)

Deletes a data store.

```
public Task<string> DeleteKey(string key, string id, ulong fee = 0, CancellationToken cancellationToken = default)
```

Parameters

key [string](#)

Row key

id [string](#)

Row id

fee [ulong](#)

Fee amount (in units of mojos)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[string](#)>

Transaction id

DeleteMirror(string, ulong, CancellationToken)

Deletes a mirror.

```
public Task DeleteMirror(string coinId, ulong fee = 0, CancellationToken cancellationToken = default)
```

Parameters

coinId [string](#)

Mirror coin id

fee [ulong](#)

Fee amount (in units of mojos)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

GetAncestors(string, string, CancellationToken)

Gets the list of ancestors for a given id/hash pair.

```
public Task<IEnumerable<InternalNode>> GetAncestors(string id, string hash,  
CancellationToken cancellationToken = default)
```

Parameters

id [string](#)

Id

hash [string](#)

Hash

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[IEnumerable](#)<[InternalNode](#)>>

GetKVDiff(string, string, string, CancellationToken)

Get kv diff between two root hashes.

```
public Task<KVDiff> GetKVDiff(string id, string hash1, string hash2, CancellationToken
```

```
cancellationToken = default)
```

Parameters

id [string](#)

Id

hash1 [string](#)

First Hash

hash2 [string](#)

Second Hash

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[KVDiff](#)>

[KVDiff](#)

GetKeys(string, string?, CancellationToken)

Gets the list of ancestors for a given id/hash pair.

```
public Task<IEnumerable<string>> GetKeys(string id, string? rootHash, CancellationToken  
cancellationToken = default)
```

Parameters

id [string](#)

Id

rootHash [string](#)

Root Hash

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <IEnumerable <[string](#)>>

GetKeysValues(string, string, CancellationToken)

Get the keys and values for a given id/root_hash pair.

```
public Task<IEnumerable<TerminalNode>> GetKeysValues(string id, string rootHash,  
CancellationToken cancellationToken = default)
```

Parameters

`id` [string](#)

Id

`rootHash` [string](#)

Root Hash

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <IEnumerable <[TerminalNode](#)>>

GetLocalRoot(string, CancellationToken)

Gets hash of latest tree root saved in our local datastore.

```
public Task<KVDiff> GetLocalRoot(string id, CancellationToken cancellationToken = default)
```

Parameters

id [string](#)

Id

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[KVDiff](#)>

A hash

GetMirrors(string, CancellationToken)

Gets the mirrors for a given store id.

```
public Task<IEnumerable<Mirror>> GetMirrors(string id, CancellationToken cancellationToken  
= default)
```

Parameters

id [string](#)

Store Id

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[IEnumerable](#) <[Mirror](#)>>

A list of [Mirror](#)

GetOwnedStores(CancellationToken)

Gets the list of owned store ids.

```
public Task<IEnumerable<string>> GetOwnedStores(CancellationToken cancellationToken = default)
```

Parameters

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<IEnumerable<string>>

A list of [Mirror](#)

GetRoot(string, CancellationToken)

Gets hash of latest tree root.

```
public Task<RootHash> GetRoot(string id, CancellationToken cancellationToken = default)
```

Parameters

`id` [string](#)

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<RootHash>

A [Root](#)

GetRootHistory(string, CancellationToken)

Get history of state hashes for a store.

```
public Task<IEnumerable<RootHistory>> GetRootHistory(string id, CancellationToken cancellationToken = default)
```

Parameters

id [string](#)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[IEnumerable](#) <[RootHistory](#)>>

A list of [RootHistory](#)

GetRoots(IEnumerable<string>, CancellationToken)

Gets state hashes for a list of roots

```
public Task<IEnumerable<RootHash>> GetRoots(IEnumerable<string> ids, CancellationToken cancellationToken = default)
```

Parameters

ids [IEnumerable](#) <[string](#)>

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[IEnumerable](#) <[RootHash](#)>>

A list of [RootHash](#)

GetSyncStatus(string, CancellationToken)

Gets the sync status of a store.

```
public Task<DataLayerSyncStatus> GetSyncStatus(string id, CancellationToken  
cancellationToken = default)
```

Parameters

id [string](#)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[DataLayerSyncStatus](#)>

A list of [DataLayerSyncStatus](#)

GetValue(string, string, string?, CancellationToken)

Get the value for a given id/key pair.

```
public Task<string> GetValue(string id, string key, string? rootHash, CancellationToken  
cancellationToken = default)
```

Parameters

id [string](#)

key [string](#)

rootHash [string](#)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[string](#)>

[string](#)

Insert(string, string, ulong, CancellationToken)

Adds a list of clvm objects as bytes to add to table.

```
public Task<string> Insert(string id, string value, ulong fee = 0, CancellationToken cancellationToken = default)
```

Parameters

[id](#) [string](#)

[value](#) [string](#)

[fee](#) [ulong](#)

Fee (in units of mojos)

[cancellationToken](#) [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[string](#)>

[string](#)

MakeOffer(IEnumerable<OfferStore>, IEnumerable<OfferStore>, ulong, CancellationToken)

Makes an offer.

```
public Task<DataLayerOffer> MakeOffer(IEnumerable<OfferStore> maker, IEnumerable<OfferStore> taker, ulong fee = 0, CancellationToken cancellationToken = default)
```

Parameters

maker [IEnumerable](#)<[OfferStore](#)>

taker [IEnumerable](#)<[OfferStore](#)>

fee [ulong](#)

Fee (in units of mojos)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[DataLayerOffer](#)>

[DataLayerOffer](#)

RemoveSubscriptions(string, IEnumerable<string>, CancellationToken)

Removes subscriptions for the given id.

```
public Task RemoveSubscriptions(string id, IEnumerable<string> urls, CancellationToken cancellationToken = default)
```

Parameters

id [string](#)

urls [IEnumerable](#)<[string](#)>

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

An awaitable Task

Subscribe(string, IEnumerable<string>, CancellationToken)

Subscribe to singleton.

```
public Task Subscribe(string id, IEnumerable<string> urls, CancellationToken cancellationToken = default)
```

Parameters

id [string](#)

urls [IEnumerable](#)<[string](#)>

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

An awaitable Task

Subscriptions(CancellationToken)

List current subscriptions.

```
public Task<IEnumerable<string>> Subscriptions(CancellationToken cancellationToken = default)
```

Parameters

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[IEnumerable](#)<[string](#)>>

A list of [string](#)

TakeOffer(object, ulong, CancellationToken)

Takes an offer.

```
public Task<string> TakeOffer(object offer, ulong fee = 0, CancellationToken  
cancellationToken = default)
```

Parameters

offer [object](#)

fee [ulong](#)

Fee (in units of mojos)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[string](#)>

[string](#)

Unsubscribe(string, bool, CancellationToken)

Unsubscribe from singleton.

```
public Task Unsubscribe(string id, bool retain = false, CancellationToken cancellationToken  
= default)
```

Parameters

id [string](#)

retain [bool](#)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

An awaitable Task

VerifyOffer(DataLayerOffer, ulong, CancellationToken)

Verifies an offer.

```
public Task<(bool Valid, ulong Fee)> VerifyOffer(DataLayerOffer offer, ulong fee = 0,  
CancellationToken cancellationToken = default)
```

Parameters

offer [DataLayerOffer](#)

fee [ulong](#)

Fee (in units of mojos)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<(bool Valid, ulong Fee)>

boolean valid flag and fee amount

WalletLogIn(uint, CancellationToken)

Sets a key to active.

```
public Task WalletLogIn(uint fingerprint, CancellationToken cancellationToken = default)
```

Parameters

fingerprint [uint](#)

The fingerprint

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

An awaitable task

Class DataLayerSyncStatus

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record DataLayerSyncStatus : IEquatable<DataLayerSyncStatus>
```

Inheritance

[object](#) ← DataLayerSyncStatus

Implements

[IEquatable](#)<[DataLayerSyncStatus](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Generation

```
public uint Generation { get; init; }
```

Property Value

[uint](#)

RootHash

```
public string RootHash { get; init; }
```

Property Value

[string](#)

TargetGeneration

```
public int TargetGeneration { get; init; }
```

Property Value

[int ↗](#)

TargetRootHash

```
public string TargetRootHash { get; init; }
```

Property Value

[string ↗](#)

Class DataLayerWallet

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

Wraps a Data Layer Wallet

```
public sealed class DataLayerWallet : Wallet
```

Inheritance

[object](#) ← [Wallet](#) ← DataLayerWallet

Inherited Members

[Wallet.WalletId](#) , [Wallet.WalletProxy](#) , [Wallet.GetWalletInfo\(CancellationToken\)](#) ,
[Wallet.GetBalance\(CancellationToken\)](#) ,
[Wallet.SelectCoins\(ulong, IEnumerable<Coin>, IEnumerable<ulong>, ulong?, ulong?, CancellationToken\)](#) ,
,
[Wallet.GetTransactions\(string, TransactionTypeFilter, bool, string, uint, uint, bool?, CancellationToken\)](#) ,
[Wallet.GetSpendableCoins\(ulong?, ulong?, IEnumerable<ulong>, IEnumerable<Coin>,
IEnumerable<string>, CancellationToken\)](#) ,
[Wallet.GetNextAddress\(bool, CancellationToken\)](#) ,
[Wallet.GetTransactionCount\(TransactionTypeFilter, CancellationToken\)](#) ,
[Wallet.DeleteUnconfirmedTransactions\(CancellationToken\)](#) ,
[Wallet.SendTransaction\(string, ulong, IEnumerable<string>, IEnumerable<ulong>, IEnumerable<string>,
ulong?, ulong?, bool, ulong, CancellationToken\)](#) ,
[Wallet.SendTransactionMulti\(IEnumerable<Coin>, IEnumerable<Coin>, ulong, CancellationToken\)](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Remarks

ctor

Constructors

DataLayerWallet(uint, WalletProxy)

Wraps a Data Layer Wallet

```
public DataLayerWallet(uint walletId, WalletProxy walletProxy)
```

Parameters

walletId [uint](#)

The wallet_id to wrap

walletProxy [WalletProxy](#)

Wallet RPC proxy to use for communication

Remarks

ctor

Methods

DeleteMirror(string, ulong, CancellationToken)

Remove an existing mirror for a specific singleton.

```
public Task<IEnumerable<TransactionRecord>> DeleteMirror(string coinId, ulong fee = 0,  
CancellationToken cancellationToken = default)
```

Parameters

coinId [string](#)

fee [ulong](#)

Fee (in units of mojos)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[IEnumerable](#)<[TransactionRecord](#)>>

A list of [TransactionRecord](#)

GetMirrors(string, CancellationToken)

Get all of the mirrors for a specific singleton.

```
public Task<IEnumerable<Mirror>> GetMirrors(string launcherId, CancellationToken  
cancellationToken = default)
```

Parameters

launcherId [string](#)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[IEnumerable](#)<[Mirror](#)>>

A list of [Mirror](#)

History(string, uint?, uint?, uint?, CancellationToken)

Get the singleton record for the latest singleton of a launcher ID.

```
public Task<IEnumerable<SingletonRecord>> History(string launcherId, uint? minGeneration =  
null, uint? maxGeneration = null, uint? numResults = null, CancellationToken  
cancellationToken = default)
```

Parameters

launcherId [string](#)

minGeneration [uint](#)?

maxGeneration [uint](#)?

numResults [uint](#)?

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <IEnumerable <[SingletonRecord](#)>>

A list of [SingletonRecord](#)

LatestSingleton(string, string, CancellationToken)

Get the singleton records that contain the specified root.

```
public Task<LineageProof> LatestSingleton(string root, string launcherId, CancellationToken cancellationToken = default)
```

Parameters

`root` [string](#)

`launcherId` [string](#)

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[LineageProof](#)>

[LineageProof](#)

NewMirror(string, ulong, IEnumerable<string>, ulong, CancellationToken)

Add a new on chain message for a specific singleton.

```
public Task<IEnumerable<TransactionRecord>> NewMirror(string launcherId, ulong amount, IEnumerable<string> urls, ulong fee = 0, CancellationToken cancellationToken = default)
```

Parameters

`launcherId` [string](#)

`amount` [ulong](#)

`urls` [IEnumerable](#) <[string](#)>

`fee` [ulong](#)

Fee (in units of mojos)

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[IEnumerable](#) <[TransactionRecord](#)>>

A list of [TransactionRecord](#)

OwnedSingletons(CancellationToken)

Get all owned singleton records.

```
public Task<IEnumerable<SingletonRecord>> OwnedSingletons(CancellationToken  
cancellationToken = default)
```

Parameters

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[IEnumerable](#) <[SingletonRecord](#)>>

A list of [SingletonRecord](#)

SingletonsByRoot(string, string, CancellationToken)

Get the singleton records that contain the specified root.

```
public Task<IEnumerable<SingletonRecord>> SingletonsByRoot(string root, string launcherId, CancellationToken cancellationToken = default)
```

Parameters

`root` [string](#)

`launcherId` [string](#)

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[IEnumerable](#)<[SingletonRecord](#)>>

A list of [SingletonRecord](#)

StopTracking(string, CancellationToken)

Stop tracking the data layer wallets.

```
public Task StopTracking(string launcherId, CancellationToken cancellationToken = default)
```

Parameters

`launcherId` [string](#)

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

An awaitable Task

TrackNew(string, CancellationToken)

Track the new data layer wallet

```
public Task TrackNew(string launcherId, CancellationToken cancellationToken = default)
```

Parameters

`launcherId` [string](#)

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

An awaitable Task

UpdateMultiple(IEnumerable<SingletonInfo>, CancellationToken)

Update multiple singletons with new merkle roots

```
public Task<IEnumerable<TransactionRecord>> UpdateMultiple(IEnumerable<SingletonInfo> updates, CancellationToken cancellationToken = default)
```

Parameters

`updates` [IEnumerable](#)<[SingletonInfo](#)>

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <IEnumerable<[TransactionRecord](#)>>

A list of [TransactionRecord](#)

UpdateRoot(string, string, ulong, CancellationToken)

Update a data layer root.

```
public Task<TransactionRecord> UpdateRoot(string newRoot, string launcherId, ulong fee = 0,  
CancellationToken cancellationToken = default)
```

Parameters

newRoot [string](#)

launcherId [string](#)

fee [ulong](#)

Fee (in units of mojos)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[TransactionRecord](#)>

[TransactionRecord](#)

Validate(CancellationToken)

Validates that [WalletId](#) is a [DATA LAYER](#)

```
public override Task Validate(CancellationToken cancellationToken = default)
```

Parameters

cancellationToken [CancellationToken](#)

Wraps a Data Layer Wallet

Returns

[Task ↗](#)

True if the wallet is a Data Layer wallet

Class DidInfo

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record DidInfo : IEquatable<DidInfo>
```

Inheritance

[object](#) ← DidInfo

Implements

[IEquatable](#) <[DidInfo](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

DidId

```
public string DidId { get; init; }
```

Property Value

[string](#)

FullPuzzle

```
public string FullPuzzle { get; init; }
```

Property Value

[string](#)

Hints

```
public IEnumerable<string> Hints { get; init; }
```

Property Value

[IEnumerable](#)<[string](#)>

LastestCoin

```
public string LastestCoin { get; init; }
```

Property Value

[string](#)

LauncherId

```
public string LauncherId { get; init; }
```

Property Value

[string](#)

Metadata

```
public IDictionary<string, string> Metadata { get; init; }
```

Property Value

[IDictionary](#)<[string](#), [string](#)>

NumVerifications

```
public int NumVerifications { get; init; }
```

Property Value

[int ↗](#)

P2Address

```
public string P2Address { get; init; }
```

Property Value

[string ↗](#)

PublicKey

```
public string PublicKey { get; init; }
```

Property Value

[string ↗](#)

RecoveryListHash

```
public string RecoveryListHash { get; init; }
```

Property Value

[string ↗](#)

Solution

```
public IEnumerable<object> Solution { get; init; }
```

Property Value

[IEnumerable](#) <[object](#)>

Class EndOfSubSlotBundle

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record EndOfSubSlotBundle : IEquatable<EndOfSubSlotBundle>
```

Inheritance

[object](#) ← EndOfSubSlotBundle

Implements

[IEquatable](#) <[EndOfSubSlotBundle](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

ChallengeChain

```
public ChallengeChainSubSlot ChallengeChain { get; init; }
```

Property Value

[ChallengeChainSubSlot](#)

InfusedChallengeChain

```
public InfusedChallengeChainSubSlot? InfusedChallengeChain { get; init; }
```

Property Value

[InfusedChallengeChainSubSlot](#)

Proofs

```
public SubSlotProofs Proofs { get; init; }
```

Property Value

[SubSlotProofs](#)

RewardChain

```
public RewardChainSubSlot RewardChain { get; init; }
```

Property Value

[RewardChainSubSlot](#)

Class EndpointInfo

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

Information about how to connect and authenticate with the RPC endpoint

```
public record EndpointInfo : IEquatable<EndpointInfo>
```

Inheritance

[object](#) ← EndpointInfo

Implements

[IEquatable](#) <[EndpointInfo](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Remarks

Using the [CertPath](#)/[KeyPath](#) vs [Cert](#)/[Key](#) are independent of each other

Properties

Cert

The loaded cert as base 64 encoded blob

```
public string Cert { get; init; }
```

Property Value

[string](#)

Information about how to connect and authenticate with the RPC endpoint

CertPath

The full file system path to the public certificate used to authenticate with the endpoint (.crt)

```
public string CertPath { get; init; }
```

Property Value

[string](#)

Information about how to connect and authenticate with the RPC endpoint

Key

The loaded key as base 64 encoded blob

```
public string Key { get; init; }
```

Property Value

[string](#)

Information about how to connect and authenticate with the RPC endpoint

KeyPath

The full file system path to the base64 encoded RSA private key to authenticate with the endpoint (.key)

```
public string KeyPath { get; init; }
```

Property Value

[string](#)

Information about how to connect and authenticate with the RPC endpoint

Uri

The [Uri](#) of the RPC endpoint

```
public Uri Uri { get; init; }
```

Property Value

[Uri](#)

Information about how to connect and authenticate with the RPC endpoint

Methods

GetCert()

```
public X509Certificate2Collection GetCert()
```

Returns

[X509Certificate2Collection](#)

Information about how to connect and authenticate with the RPC endpoint

Class ErrorResponse

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

Response in error case for all endpoints of the pool protocol

```
public record ErrorResponse : IEquatable<ErrorResponse>
```

Inheritance

[object](#) ← ErrorResponse

Implements

[IEquatable](#) <[ErrorResponse](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

ErrorCode

```
public ushort ErrorCode { get; init; }
```

Property Value

[ushort](#)

Response in error case for all endpoints of the pool protocol

ErrorMessage

```
public string? ErrorMessage { get; init; }
```

Property Value

[string](#) ↗

Response in error case for all endpoints of the pool protocol

Class Extensions

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

Helper extension methods

```
public static class Extensions
```

Inheritance

[object](#) ← Extensions

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Fields

OneTrillion

There are 1 trillion mojo in a chia

```
public const decimal OneTrillion = 1000000000000
```

Field Value

[decimal](#)

Helper extension methods

Methods

AsChia(ulong)

Formats a value expressed in mojo to chia

```
public static string AsChia(this ulong mojo)
```

Parameters

mojo [ulong](#)

The amount of mojo

Returns

[string](#)

Formatted string expressed in a unit of chia

AsChia(ulong, IFormatProvider?)

Formats a value expressed in mojo to chia

```
public static string AsChia(this ulong mojo, IFormatProvider? provider)
```

Parameters

mojo [ulong](#)

The amount of mojo

provider [IFormatProvider](#)

An object that supplies culture-specific formatting information.

Returns

[string](#)

Formatted string expressed in a unit of chia

AsChia(ulong, string?)

Formats a value expressed in mojo to chia

```
public static string AsChia(this ulong mojo, string? format)
```

Parameters

mojo [ulong](#)

The amount of mojo

format [string](#)

A numeric format string.

Returns

[string](#)

Formatted string expressed in a unit of chia

AsChia(ulong, string?, IFormatProvider?)

Formats a value expressed in mojo to chia

```
public static string AsChia(this ulong mojo, string? format, IFormatProvider? provider)
```

Parameters

mojo [ulong](#)

The amount of mojo

format [string](#)

A numeric format string.

provider [IFormatProvider](#)

An object that supplies culture-specific formatting information.

Returns

[string](#)

Formatted string expressed in a unit of chia

ToBytesString(double, string)

Format a number of bytes in human readable format

```
public static string ToBytesString(this double byteCount, string format = "N3")
```

Parameters

byteCount [double](#)

The number of bytes

format [string](#)

Return string culture format

Returns

[string](#)

A human readable string

ToBytesString(int, string)

Format a number of bytes in human readable format

```
public static string ToBytesString(this int byteCount, string format = "N3")
```

Parameters

byteCount [int](#)

The number of bytes

format [string](#)

Return string culture format

Returns

[string](#)

A human readable string

ToBytesString(long, string)

Format a number of bytes in human readable format

```
public static string ToBytesString(this long byteCount, string format = "N3")
```

Parameters

[byteCount](#) [long](#)

The number of bytes

[format](#) [string](#)

Return string culture format

Returns

[string](#)

A human readable string

ToBytesString(BigInteger, string)

Format a number of bytes in human readable format

```
public static string ToBytesString(this BigInteger byteCount, string format = "N3")
```

Parameters

[byteCount](#) [BigInteger](#)

The number of bytes

`format` [string](#)

Return string culture format

Returns

[string](#)

A human readable string

Remarks

Adapted from <https://stackoverflow.com/questions/281640/how-do-i-get-a-human-readable-file-size-in-bytes-abbreviation-using-net>

ToBytesString(UInt128, string)

Format a number of bytes in human readable format

```
public static string ToBytesString(this UInt128 byteCount, string format = "N3")
```

Parameters

`byteCount` [UInt128](#)

The number of bytes

`format` [string](#)

Return string culture format

Returns

[string](#)

A human readable string

Remarks

Adapted from <https://stackoverflow.com/questions/281640/how-do-i-get-a-human-readable-file-size-in-bytes-abbreviation-using-net>

ToBytesString(uint, string)

Format a number of bytes in human readable format

```
public static string ToBytesString(this uint byteCount, string format = "N3")
```

Parameters

byteCount [uint](#)

The number of bytes

format [string](#)

Return string culture format

Returns

[string](#)

A human readable string

ToBytesString(ulong, string)

Format a number of bytes in human readable format

```
public static string ToBytesString(this ulong byteCount, string format = "N3")
```

Parameters

byteCount [ulong](#)

The number of bytes

format [string](#)

Return string culture format

Returns

[string](#)

A human readable string

Remarks

Adapted from <https://stackoverflow.com/questions/281640/how-do-i-get-a-human-readable-file-size-in-bytes-abbreviation-using-net>

ToChia(ulong)

Converts an amount of mojo to the same amount in chia, converting from [ulong](#) to [decimal](#)

```
public static decimal ToChia(this ulong mojo)
```

Parameters

[mojo](#) [ulong](#)

The amount of mojo

Returns

[decimal](#)

The amount of chia

ToMojo(decimal)

Converts an amount of chia to the same amount in mojo, converting from [decimal](#) to [ulong](#)

```
public static ulong ToMojo(this decimal chia)
```

Parameters

[chia](#) [decimal](#)

The amount of chia

Returns

[ulong](#) ↗

The amount of mojo

Class FarmerProxy

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

Proxy that communicates with the farmer

```
public sealed class FarmerProxy : ServiceProxy
```

Inheritance

[object](#) ← [ServiceProxy](#) ← FarmerProxy

Inherited Members

[ServiceProxy.OriginService](#) , [ServiceProxy.DestinationService](#) , [ServiceProxy.RpcClient](#) ,
[ServiceProxy.HealthZ\(CancellationToken\)](#) , [ServiceProxy.StopNode\(CancellationToken\)](#) ,
[ServiceProxy.GetConnections\(CancellationToken\)](#) , [ServiceProxy.GetRoutes\(CancellationToken\)](#) ,
[ServiceProxy.OpenConnection\(string, int, CancellationToken\)](#) ,
[ServiceProxy.CloseConnection\(string, CancellationToken\)](#) , [object.Equals\(object\)](#) ,
[object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Remarks

ctor

Constructors

FarmerProxy(IRpcClient, string)

Proxy that communicates with the farmer

```
public FarmerProxy(IRpcClient rpcClient, string originService)
```

Parameters

rpcClient [IRpcClient](#)

[IRpcClient](#) instance to use for rpc communication

`originService` [string](#)

[Origin](#)

Remarks

ctor

Methods

GetHarvesterPlotsDuplicates(PlotPathrequestData, CancellationToken)

Get a paginated list of duplicate plots

```
public Task<PaginatedPlotRequest> GetHarvesterPlotsDuplicates(PlotPathrequestData  
requestData, CancellationToken cancellationToken = default)
```

Parameters

`requestData` [PlotPathrequestData](#)

Info about the request

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[PaginatedPlotRequest](#)>

A page of duplicate plots

GetHarvesterPlotsInvalid(PlotPathrequestData, CancellationToken)

Get a paginated list of invalid plots

Info about the request

```
public Task<PaginatedPlotRequest> GetHarvesterPlotsInvalid(PlotPathRequestData requestData,  
CancellationToken cancellationToken = default)
```

Parameters

requestData [PlotPathRequestData](#)

Proxy that communicates with the farmer

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[PaginatedPlotRequest](#)>

A page of invalid plots

GetHarvesterPlotsKeysMissing(PlotPathRequestData, CancellationToken)

Get a paginated list of plots with missing keys

```
public Task<PaginatedPlotRequest> GetHarvesterPlotsKeysMissing(PlotPathRequestData  
requestData, CancellationToken cancellationToken = default)
```

Parameters

requestData [PlotPathRequestData](#)

Info about the request

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[PaginatedPlotRequest](#)>

A page of plots with missing keys

GetHarvesterPlotsValid(PlotInforequestData, CancellationToken)

Get a paginated list of valid plots

```
public Task<PaginatedPlotRequest> GetHarvesterPlotsValid(PlotInforequestData requestData,  
CancellationToken cancellationToken = default)
```

Parameters

requestData [PlotInforequestData](#)

Info about the request

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[PaginatedPlotRequest](#)>

A page of valid plots

GetHarvesters(CancellationToken)

Get the list of harvesters

```
public Task<IEnumerable<HarvesterInfo>> GetHarvesters(CancellationToken cancellationToken  
= default)
```

Parameters

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <IEnumerable<[HarvesterInfo](#)>>

A list of harvesters

GetHarvestersSummary(CancellationToken)

Get a summary of harvesters

```
public Task<IEnumerable<HarvesterSummary>> GetHarvestersSummary(CancellationToken  
cancellationToken = default)
```

Parameters

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <IEnumerable<[HarvesterSummary](#)>>

A list of harvesters

GetPoolLoginLink(string, CancellationToken)

Get's the pool login link, if any

```
public Task<string> GetPoolLoginLink(string launcherID, CancellationToken cancellationToken  
= default)
```

Parameters

`launcherID` [string](#)

The id of the pool launcher

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[string](#)>

The link

GetPoolState(CancellationToken)

Get's the state of the pool

```
public Task<IEnumerable<PoolStateInfo>> GetPoolState(CancellationToken cancellationToken  
= default)
```

Parameters

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[IEnumerable](#) <[PoolStateInfo](#)>>

A list of pool states

GetRewardTargets(int, CancellationToken)

Get the farm and pool reward targets

```
public Task<(<string FarmerTarget, string PoolTarget>) GetRewardTargets(int maxPhToSearch,  
CancellationToken cancellationToken = default)
```

Parameters

maxPhToSearch [int](#)

The max number of puzzle hashes to search

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<(string [MyDid](#), string [CoinID](#))>

the farm and pool reward targets

GetRewardTargets(CancellationToken)

Get the farm and pool reward targets

```
public Task<(string FarmerTarget, string PoolTarget)> GetRewardTargets(CancellationToken  
cancellationToken = default)
```

Parameters

[cancellationToken](#) [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<(string [MyDid](#), string [CoinID](#))>

the farm and pool reward targets

GetRewardTargetsIncludingPrivateKey(int, CancellationToken)

Get the farm and pool reward targets, including private keys in the search

```
public Task<(string FarmerTarget, string PoolTarget, bool HaveFarmerSk, bool HavePoolSk)>  
GetRewardTargetsIncludingPrivateKey(int maxPhToSearch = 500, CancellationToken  
cancellationToken = default)
```

Parameters

[maxPhToSearch](#) [int](#)

The max number of puzzle hashes to search

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <([string](#) [FarmerTarget](#), [string](#) [PoolTarget](#), [bool](#) [HaveFarmerSk](#), [bool](#) [HavePoolSk](#))>

The farm and pool reward targets plus indicator if private keys are present

GetRewardTargetsIncludingPrivateKey(CancellationToken)

Get the farm and pool reward targets, including private keys in the search

```
public Task<(string FarmerTarget, string PoolTarget, bool HaveFarmerSk, bool HavePoolSk)>
GetRewardTargetsIncludingPrivateKey(CancellationToken cancellationToken = default)
```

Parameters

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <([string](#) [FarmerTarget](#), [string](#) [PoolTarget](#), [bool](#) [HaveFarmerSk](#), [bool](#) [HavePoolSk](#))>

The farm and pool reward targets

GetSignagePoint(string, CancellationToken)

Get's a signage point by hash

```
public Task<(IEnumerable<(string SpHash, ProofOfSpace ProofOfSpace)> Proofs,
FarmerSignagePoint SignagePoint)> GetSignagePoint(string spHash, CancellationToken
cancellationToken = default)
```

Parameters

`spHash` [string](#)

signage point hash

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <([IEnumerable](#) <([string](#) `SpHash`, [ProofOfSpace](#) `ProofOfSpace`)> [Proofs](#), [FarmerSignagePoint](#) `SignagePoint`)>

a signage point and proofs of space

GetSignagePoints(CancellationToken)

Get signage points

```
public Task<IEnumerable<(IEnumerable<(string SpHash, ProofOfSpace ProofOfSpace)> Proofs,
FarmerSignagePoint SignagePoint)>> GetSignagePoints(CancellationToken cancellationToken
= default)
```

Parameters

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[IEnumerable](#) <([IEnumerable](#) <([string](#) `SpHash`, [ProofOfSpace](#) `ProofOfSpace`)> [Proofs](#),
[FarmerSignagePoint](#) `SignagePoint`)>>

List of signage points

SetPayoutInstructions(string, string, CancellationToken)

Set's a pool's payout instructions

```
public Task SetPayoutInstructions(string launcherID, string payoutInstructions,  
CancellationToken cancellationToken = default)
```

Parameters

launcherID [string](#)

The id of the pool launcher

payoutInstructions [string](#)

The instructions

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

An awaitable [Task](#)

SetRewardTargets(string, string, CancellationToken)

Sets the farm and pool targets for the farmer

```
public Task SetRewardTargets(string farmerTarget, string poolTarget, CancellationToken  
cancellationToken = default)
```

Parameters

farmerTarget [string](#)

Farmer target

poolTarget [string](#)

Pool target

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

An awaitable [Task](#)

Class FarmerRewards

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record FarmerRewards : IEquatable<FarmerRewards>
```

Inheritance

[object](#) ← FarmerRewards

Implements

[IEquatable](#) <[FarmerRewards](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

BlocksWon

```
public uint BlocksWon { get; init; }
```

Property Value

[uint](#)

FarmedAmount

```
public ulong FarmedAmount { get; init; }
```

Property Value

[ulong](#)

FarmerRewardAmount

```
public ulong FarmerRewardAmount { get; init; }
```

Property Value

[ulong](#) ↗

FeeAmount

```
public ulong FeeAmount { get; init; }
```

Property Value

[ulong](#) ↗

LastHeightFarmed

```
public ulong LastHeightFarmed { get; init; }
```

Property Value

[ulong](#) ↗

LastTimeFarmed

```
public ulong LastTimeFarmed { get; init; }
```

Property Value

[ulong](#) ↗

NextFarmerUpdateDateTime

```
[JsonIgnore]  
public DateTime NextFarmerUpdateDateTime { get; }
```

Property Value

[DateTime](#)

PoolRewardAmount

```
public ulong PoolRewardAmount { get; init; }
```

Property Value

[ulong](#)

Class FarmerSignagePoint

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

This type doesn't exist in the chia code but is generated and passed around as a dicitonary (not to be ocnfused with [SignagePoint](#))

```
public record FarmerSignagePoint : IEquatable<FarmerSignagePoint>
```

Inheritance

[object](#) ← FarmerSignagePoint

Implements

[IEquatable](#)<[FarmerSignagePoint](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

ChallengeChainSp

```
public string ChallengeChainSp { get; init; }
```

Property Value

[string](#)

This type doesn't exist in the chia code but is generated and passed around as a dicitonary (not to be ocnfused with)

ChallengeHash

```
public string ChallengeHash { get; init; }
```

Property Value

[string](#) ↗

This type doesn't exist in the chia code but is generated and passed around as a dicitonary (not to be ocnfused with)

Difficulty

```
public ulong Difficulty { get; init; }
```

Property Value

[ulong](#) ↗

This type doesn't exist in the chia code but is generated and passed around as a dicitonary (not to be ocnfused with)

RewardChainSp

```
public string RewardChainSp { get; init; }
```

Property Value

[string](#) ↗

This type doesn't exist in the chia code but is generated and passed around as a dicitonary (not to be ocnfused with)

SignagePointIndex

```
public byte SignagePointIndex { get; init; }
```

Property Value

[byte](#) ↗

This type doesn't exist in the chia code but is generated and passed around as a dicitonary (not to be ocnfused with)

SubSlotIter

```
public ulong SubSlotIter { get; init; }
```

Property Value

[ulong](#)

This type doesn't exist in the chia code but is generated and passed around as a dicitonary (not to be ocnfused with)

Class FilterItem

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record FilterItem : IEquatable<FilterItem>
```

Inheritance

[object](#) ← FilterItem

Implements

[IEquatable](#) <[FilterItem](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Key

```
public string Key { get; init; }
```

Property Value

[string](#)

Value

```
public string? Value { get; init; }
```

Property Value

[string](#)

Enum FilterMode

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public enum FilterMode : byte
```

Fields

Exclude = 2

Include = 1

Class Foliage

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

The entire foliage block, containing signature and the unsigned back pointer The hash of this is the "header hash". Note that for unfinished blocks, the prev_block_hash is the prev from the signage point, and can be replaced with a more recent block

```
public record Foliage : IEquatable<Foliage>
```

Inheritance

[object](#) ← Foliage

Implements

[IEquatable](#)<[Foliage](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

FoliageBlockData

```
public FoliageBlockData FoliageBlockData { get; init; }
```

Property Value

[FoliageBlockData](#)

The entire foliage block, containing signature and the unsigned back pointer The hash of this is the "header hash". Note that for unfinished blocks, the prev_block_hash is the prev from the signage point, and can be replaced with a more recent block

FoliageBlockDataSignature

```
public string FoliageBlockDataSignature { get; init; }
```

Property Value

[string](#) ↗

The entire foliage block, containing signature and the unsigned back pointer The hash of this is the "header hash". Note that for unfinished blocks, the prev_block_hash Is the prev from the signage point, and can be replaced with a more recent block

FoliageTransactionBlockHash

```
public string? FoliageTransactionBlockHash { get; init; }
```

Property Value

[string](#) ↗

The entire foliage block, containing signature and the unsigned back pointer The hash of this is the "header hash". Note that for unfinished blocks, the prev_block_hash Is the prev from the signage point, and can be replaced with a more recent block

FoliageTransactionBlockSignature

```
public string? FoliageTransactionBlockSignature { get; init; }
```

Property Value

[string](#) ↗

The entire foliage block, containing signature and the unsigned back pointer The hash of this is the "header hash". Note that for unfinished blocks, the prev_block_hash Is the prev from the signage point, and can be replaced with a more recent block

PrevBlockHash

```
public string PrevBlockHash { get; init; }
```

Property Value

[string ↗](#)

The entire foliage block, containing signature and the unsigned back pointer The hash of this is the "header hash". Note that for unfinished blocks, the prev_block_hash Is the prev from the signage point, and can be replaced with a more recent block

RewardBlockHash

```
public string RewardBlockHash { get; init; }
```

Property Value

[string ↗](#)

The entire foliage block, containing signature and the unsigned back pointer The hash of this is the "header hash". Note that for unfinished blocks, the prev_block_hash Is the prev from the signage point, and can be replaced with a more recent block

Class FoliageBlockData

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

Part of the block that is signed by the plot key

```
public record FoliageBlockData : IEquatable<FoliageBlockData>
```

Inheritance

[object](#) ← FoliageBlockData

Implements

[IEquatable](#) <[FoliageBlockData](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

ExtensionData

Used for future updates. Can be any 32 byte value initially

```
public string ExtensionData { get; init; }
```

Property Value

[string](#)

Part of the block that is signed by the plot key

FarmerRewardPuzzleHash

```
public string FarmerRewardPuzzleHash { get; init; }
```

Property Value

[string](#) ↗

Part of the block that is signed by the plot key

PoolSignature

Iff ProofOfSpace has a pool pk

```
public string? PoolSignature { get; init; }
```

Property Value

[string](#) ↗

Part of the block that is signed by the plot key

PoolTarget

```
public PoolTarget PoolTarget { get; init; }
```

Property Value

[PoolTarget](#)

Part of the block that is signed by the plot key

UnfinishedRewardBlockHash

```
public string UnfinishedRewardBlockHash { get; init; }
```

Property Value

[string](#) ↗

Part of the block that is signed by the plot key

Class FoliageTransactionBlock

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

Information that goes along with each transaction block that is relevant for light clients

```
public record FoliageTransactionBlock : IEquatable<FoliageTransactionBlock>
```

Inheritance

[object](#) ← FoliageTransactionBlock

Implements

[IEquatable](#) <[FoliageTransactionBlock](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

AdditionsRoot

```
public string AdditionsRoot { get; init; }
```

Property Value

[string](#)

Information that goes along with each transaction block that is relevant for light clients

DateTimestamp

```
[JsonIgnore]  
public DateTime DateTimestamp { get; }
```

Property Value

[DateTime](#)

Information that goes along with each transaction block that is relevant for light clients

FilterHash

```
public string FilterHash { get; init; }
```

Property Value

[string](#)

Information that goes along with each transaction block that is relevant for light clients

PrevTransactionBlockHash

```
public string PrevTransactionBlockHash { get; init; }
```

Property Value

[string](#)

Information that goes along with each transaction block that is relevant for light clients

RemovalsRoot

```
public string RemovalsRoot { get; init; }
```

Property Value

[string](#)

Information that goes along with each transaction block that is relevant for light clients

Timestamp

```
public ulong Timestamp { get; init; }
```

Property Value

[ulong](#) ↗

Information that goes along with each transaction block that is relevant for light clients

TransactionsInfoHash

```
public string TransactionsInfoHash { get; init; }
```

Property Value

[string](#) ↗

Information that goes along with each transaction block that is relevant for light clients

Class FullBlock

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

All the information required to validate a block

```
public record FullBlock : IEquatable<FullBlock>
```

Inheritance

[object](#) ← FullBlock

Implements

[IEquatable](#)<[FullBlock](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

ChallengeChainIpProof

```
public VDFProof ChallengeChainIpProof { get; init; }
```

Property Value

[VDFProof](#)

All the information required to validate a block

ChallengeChainSpProof

If not first sp in sub-slot

```
public VDFProof? ChallengeChainSpProof { get; init; }
```

Property Value

VDFProof

All the information required to validate a block

FinishedSubSlots

If first sb

```
public IEnumerable<EndOfSubSlotBundle> FinishedSubSlots { get; init; }
```

Property Value

IEnumerable<EndOfSubSlotBundle>

All the information required to validate a block

Foliage

Reward chain foliage data

```
public Foliage Foliage { get; init; }
```

Property Value

Foliage

All the information required to validate a block

FoliageTransactionBlock

Reward chain foliage data (tx block)

```
public FoliageTransactionBlock? FoliageTransactionBlock { get; init; }
```

Property Value

FoliageTransactionBlock

All the information required to validate a block

HeaderHash

```
public string HeaderHash { get; init; }
```

Property Value

[string](#) ↗

All the information required to validate a block

InfusedChallengeChainIpProof

Iff deficit < 4

```
public VDFProof? InfusedChallengeChainIpProof { get; init; }
```

Property Value

[VDFProof](#)

All the information required to validate a block

IsTransactionBlock

Is this block from a transaction

```
[JsonIgnore]  
public bool IsTransactionBlock { get; }
```

Property Value

[bool](#) ↗

All the information required to validate a block

RewardChainBlock

Reward chain trunk data

```
public RewardChainBlock RewardChainBlock { get; init; }
```

Property Value

[RewardChainBlock](#)

All the information required to validate a block

RewardChainIpProof

```
public VDFProof RewardChainIpProof { get; init; }
```

Property Value

[VDFProof](#)

All the information required to validate a block

RewardChainSpProof

If not first sp in sub-slot

```
public VDFProof? RewardChainSpProof { get; init; }
```

Property Value

[VDFProof](#)

All the information required to validate a block

TransactionsGenerator

Program that generates transactions

```
public string? TransactionsGenerator { get; init; }
```

Property Value

[string](#)

All the information required to validate a block

TransactionsGeneratorRefList

List of block heights of previous generators referenced in this block

```
public IEnumerable<uint> TransactionsGeneratorRefList { get; init; }
```

Property Value

[IEnumerable](#) <[uint](#)>

All the information required to validate a block

TransactionsInfo

Reward chain foliage data (tx block additional)

```
public TransactionsInfo? TransactionsInfo { get; init; }
```

Property Value

[TransactionsInfo](#)

All the information required to validate a block

Class FullNodeProxy

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

Proxy that communicates with the full node

```
public sealed class FullNodeProxy : ServiceProxy
```

Inheritance

[object](#) ← [ServiceProxy](#) ← FullNodeProxy

Inherited Members

[ServiceProxy.OriginService](#) , [ServiceProxy.DestinationService](#) , [ServiceProxy.RpcClient](#) ,
[ServiceProxy.HealthZ\(CancellationToken\)](#) , [ServiceProxy.StopNode\(CancellationToken\)](#) ,
[ServiceProxy.GetConnections\(CancellationToken\)](#) , [ServiceProxy.GetRoutes\(CancellationToken\)](#) ,
[ServiceProxy.OpenConnection\(string, int, CancellationToken\)](#) ,
[ServiceProxy.CloseConnection\(string, CancellationToken\)](#) , [object.Equals\(object\)](#) ,
[object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Remarks

ctor

Constructors

FullNodeProxy(IRpcClient, string)

Proxy that communicates with the full node

```
public FullNodeProxy(IRpcClient rpcClient, string originService)
```

Parameters

rpcClient [IRpcClient](#)

[IRpcClient](#) instance to use for rpc communication

`originService` [string](#)

[Origin](#)

Remarks

ctor

Methods

GetAdditionsAndRemovals(string, CancellationToken)

Retrieves the additions and removals (state transitions) for a certain block. Returns coin records for each addition and removal.

```
public Task<(IEnumerable<CoinRecord> Additions, IEnumerable<CoinRecord> Removals)>
GetAdditionsAndRemovals(string headerhash, CancellationToken cancellationToken = default)
```

Parameters

`headerhash` [string](#)

The header hash

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<(IEnumerable<CoinRecord> [Additions](#), IEnumerable<CoinRecord> [Removals](#))>

A list of additions and a list of removals

GetAllMemppoolTxIds(CancellationToken)

Returns a list of all transaction IDs (spend bundle hashes) in the mempool.

```
public Task<IEnumerable<string>> GetAllMemppoolTxIds(CancellationToken cancellationToken
= default)
```

Parameters

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[IEnumerable](#) <[string](#)>>

a list of tx_ids

GetAllMempoolItems(CancellationToken)

Returns all items in the mempool.

```
public Task<IDictionary<string, MempoolItem>> GetAllMempoolItems(CancellationToken  
cancellationToken = default)
```

Parameters

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[IDictionary](#) <[string](#), [MempoolItem](#)>>

A dictionary of mempool items

GetAverageBlockTime(CancellationToken)

Estimates the average time it is taking to process the last 500 blocks

```
public Task<TimeSpan> GetAverageBlockTime(CancellationToken cancellationToken = default)
```

Parameters

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[TimeSpan](#)>

The [TimeSpan](#) estimation

GetBlock(string, CancellationToken)

Get a block by a header hash

```
public Task<FullBlock> GetBlock(string headerhash, CancellationToken cancellationToken  
= default)
```

Parameters

headerhash [string](#)

The header hash

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[FullBlock](#)>

The [FullBlock](#)

GetBlockCountMetrics(CancellationToken)

Retrieves aggregate information about blocks.

```
public Task<BlockCountMetrics> GetBlockCountMetrics(CancellationToken cancellationToken  
= default)
```

Parameters

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[BlockCountMetrics](#)>

The [BlockCountMetrics](#)

GetBlockRecord(string, CancellationToken)

Get a block record by a header hash

```
public Task<BlockRecord> GetBlockRecord(string headerhash, CancellationToken  
cancellationToken = default)
```

Parameters

`headerhash` [string](#)

The header hash

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[BlockRecord](#)>

The [BlockRecord](#)

GetBlockRecordByHeight(uint, CancellationToken)

Retrieves a block record by height (assuming the height is less than or equal peak height)

```
public Task<BlockRecord> GetBlockRecordByHeight(uint height, CancellationToken
```

```
cancellationToken = default)
```

Parameters

height [uint](#)

the height to get

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[BlockRecord](#)>

The [BlockRecord](#)

GetBlockRecords(uint, uint, CancellationToken)

Retrieves block records in a range

```
public Task<IEnumerable<BlockRecord>> GetBlockRecords(uint start, uint end,  
CancellationToken cancellationToken = default)
```

Parameters

start [uint](#)

Start height

end [uint](#)

End Height - non-inclusive

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[IEnumerable](#)<[BlockRecord](#)>>

list of [BlockRecords](#)

GetBlockSpends(string, CancellationToken)

Retrieves every coin that was spent in a block

```
public Task<IEnumerable<CoinSpend>> GetBlockSpends(string headerhash, CancellationToken cancellationToken = default)
```

Parameters

headerhash [string](#)

The block's header_hash

cancellationToken [CancellationToken](#)

Returns

[Task](#)<IEnumerable<CoinSpend>>

Exceptions

[ArgumentNullException](#)

GetBlockSpendsWithConditions(string, CancellationToken)

Retrieves the spends in the given block, including its conditions.

```
public Task<IEnumerable<BlockSpendWithConditions>> GetBlockSpendsWithConditions(string headerHash, CancellationToken cancellationToken = default)
```

Parameters

headerHash [string](#)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <IEnumerable<[BlockSpendWithConditions](#)>>

A list of [BlockSpendWithConditions](#)

GetBlockchainState(CancellationToken)

Returns a summary of the node's view of the blockchain.

```
public Task<BlockchainState> GetBlockchainState(CancellationToken cancellationToken  
= default)
```

Parameters

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[BlockchainState](#)>

The [BlockchainState](#)

GetBlocks(uint, uint, bool?, bool?, CancellationToken)

Get the blocks between a start and end height

```
public Task<IEnumerable<FullBlock>> GetBlocks(uint start, uint end, bool? excludeHeaderhash  
= null, bool? excludeReorged = null, CancellationToken cancellationToken = default)
```

Parameters

start [uint](#)

Start height

end [uint](#)

End Height - non-inclusive

excludeHeaderhash [bool](#)?

Flag indicating whether to include the header hash in the result or not

excludeReorged [bool](#)?

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[IEnumerable](#) <[FullBlock](#)>>

A list of [FullBlocks](#)

GetCoinRecordByName(string, CancellationToken)

Retrieves a coin record by its name/id.

```
public Task<CoinRecord> GetCoinRecordByName(string name, CancellationToken cancellationToken  
= default)
```

Parameters

name [string](#)

The coin name

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[CoinRecord](#)>

A [CoinRecord](#)

GetCoinRecordsByHint(string, bool, uint?, uint?, CancellationToken)

Retrieves a coin record by hint

```
public Task<IEnumerable<CoinRecord>> GetCoinRecordsByHint(string hint, bool includeSpentCoins, uint? startHeight = null, uint? endHeight = null, CancellationToken cancellationToken = default)
```

Parameters

hint [string](#)

The hint

includeSpentCoins [bool](#)

whether to include spent coins too, instead of just unspent

startHeight [uint](#)?

confirmation start height for search

endHeight [uint](#)?

confirmation end height for search

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[IEnumerable](#) <[CoinRecord](#)>>

A list of [CoinRecord](#)s

GetCoinRecordsByNames(IEnumerable<string>, bool, uint?, uint?, CancellationToken)

Retrieves the coins for given coin IDs

```
public Task<IEnumerable<CoinRecord>> GetCoinRecordsByNames(IEnumerable<string> names, bool includeSpentCoins, uint? startHeight = null, uint? endHeight = null, CancellationToken cancellationToken = default)
```

Parameters

names [IEnumerable<string>](#)

The coin names

includeSpentCoins [bool](#)

Flag indicating whether to include spent coins or not

startHeight [uint](#)?

confirmation start height for search

endHeight [uint](#)?

confirmation end height for search

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task<IEnumerable<CoinRecord>>](#)

A list of [CoinRecords](#)

GetCoinRecordsByParentIds(IEnumerable<string>, bool, uint?, uint?, CancellationToken)

Retrieves the coins for a given list of parent ids

```
public Task<IEnumerable<CoinRecord>> GetCoinRecordsByParentIds(IEnumerable<string> parentIds, bool includeSpentCoins, uint? startHeight = null, uint? endHeight = null, CancellationToken cancellationToken = default)
```

Parameters

`parentIds` [IEnumerable](#)<[string](#)>

The list of parent ids hashes

`includeSpentCoins` [bool](#)

whether to include spent coins too, instead of just unspent

`startHeight` [uint](#)?

confirmation start height for search

`endHeight` [uint](#)?

confirmation end height for search

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[IEnumerable](#)<[CoinRecord](#)>>

A list of [CoinRecord](#)s

GetCoinRecordsByPuzzleHash(string, bool, uint?, uint?, CancellationToken)

Retrieves the coins for a given puzzlehash

```
public Task<IEnumerable<CoinRecord>> GetCoinRecordsByPuzzleHash(string puzzlehash, bool includeSpentCoins, uint? startHeight, uint? endHeight, CancellationToken cancellationToken = default)
```

Parameters

`puzzlehash` [string](#)

The puzzle hash

`includeSpentCoins` [bool](#)

whether to include spent coins too, instead of just unspent

`startHeight` [uint](#)?

confirmation start height for search

`endHeight` [uint](#)?

confirmation end height for search

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[IEnumerable](#) <[CoinRecord](#)>>

A list of [CoinRecords](#)

GetCoinRecordsByPuzzleHashes(IEnumerable<string>, bool, uint?, uint?, CancellationToken)

Retrieves the coins for a given list of puzzlehashes, by default returns unspent coins.

```
public Task<IEnumerable<CoinRecord>> GetCoinRecordsByPuzzleHashes(IEnumerable<string> puzzlehashes, bool includeSpentCoins, uint? startHeight = null, uint? endHeight = null, CancellationToken cancellationToken = default)
```

Parameters

`puzzlehashes` [IEnumerable](#) <[string](#)>

The list of puzzle hashes

`includeSpentCoins` [bool](#)

whether to include spent coins too, instead of just unspent

`startHeight` [uint](#)?

confirmation start height for search

endHeight [uint](#)?

confirmation end height for search

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[IEnumerable](#)<[CoinRecord](#)>>

A list of [CoinRecord](#)s

GetFeeEstimate(ulong, IEnumerable<int>, CancellationToken)

Estimate a spend fee

```
public Task<(IEnumerable<ulong> estimates, IEnumerable<int> targetTimes, float currentFeeRate, ulong mempoolSize, ulong mempoolFees, int numSpends, ulong mempoolMaxSize, bool synced, uint peakHeight, ulong lastPeakTimestamp, ulong nodeTimeUtc, ulong lastBlockCost, ulong feesLastBlock, float feeRateLastBlock, uint lastTxBlockHeight)> GetFeeEstimate(ulong cost, IEnumerable<int> targetTimes, CancellationToken cancellationToken = default)
```

Parameters

cost [ulong](#)

targetTimes [IEnumerable](#)<[int](#)>

Array of target times

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<([IEnumerable](#)<[ulong](#)> **estimates**, [IEnumerable](#)<[int](#)> **targetTimes**, [float](#) **currentFeeRate**, [ulong](#) **mempoolSize**, [ulong](#) **mempoolFees**, [int](#) **numSpends**, [ulong](#) **mempoolMaxSize**

```
bool synced, uint peakHeight, ulong lastPeakTimestamp, ulong nodeTimeUtc, ulong lastBlockCost, ulong feesLastBlock, float feeRateLastBlock, uint lastTxBlockHeight) >
```

Fee estimate details

`GetFeeEstimate(SpendBundle, IEnumerable<int>, CancellationToken)`

Estimate a spend fee

```
public Task<(IEnumerable<int> Estimates, IEnumerable<int> TargetTimes, ulong CurrentFeeRate,


- ulong MempoolSize, ulong MempoolMaxSize, bool Synced, uint PeakHeight, ulong
- LastPeakTimestamp, ulong UtcTimestamp)> GetFeeEstimate(SpendBundle spendBundle,
- IEnumerable<int> targetTimes, CancellationToken cancellationToken = default)

```

Parameters

spendBundle SpendBundle

The spend bundle to esimtate

targetTimes [IEnumerable](#)[<int>](#)

Array of target times

cancellationToken CancellationToken

A token to allow the call to be cancelled

Task ↗ < (IE)

`, ulong MempoolSize, ulong MempoolMaxSize, bool Synced, uint PeakHeight, ulong LastPeakTimestamp, ulong UtcTimestamp)>`

Fee estimate details

`GetMempoolItemByTxId(string, bool, CancellationToken)`

Gets a mempool item by tx id.

```
public Task<MempoolItem> GetMemmpoolItemByTxId(string txId, bool includePending = false,  
CancellationToken cancellationToken = default)
```

Parameters

txId [string](#)

Transaction id

includePending [bool](#)

Including pending transactions

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[MempoolItem](#)>

The [MempoolItem](#)

GetMemmpoolItemsByCoinName(string, CancellationToken)

Gets a mempool item by coin name.

```
public Task<IEnumerable<MempoolItem>> GetMemmpoolItemsByCoinName(string coinName,  
CancellationToken cancellationToken = default)
```

Parameters

coinName [string](#)

Coin name

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <IEnumerable<[MempoolItem](#)>>

The [MempoolItem](#)

GetNetworkInfo(CancellationToken)

Retrieves information about the current network

```
public Task<(string NetworkName, string NetworkPrefix)> GetNetworkInfo(CancellationToken cancellationToken = default)
```

Parameters

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <(string [MyDid](#), string [CoinID](#))>

The network name and coin prefix

GetNetworkSpace(string, string, CancellationToken)

Retrieves an estimate of total space validating the chain between two block header hashes.

```
public Task<BigInteger> GetNetworkSpace(string newerBlockHeaderhash, string olderBlockHeaderhash, CancellationToken cancellationToken = default)
```

Parameters

newerBlockHeaderhash [string](#)

olderBlockHeaderhash [string](#)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[BigInteger](#)>

[BigInteger](#) of network space in bytes

GetPuzzleAndSolution(string, uint, CancellationToken)

Gets a coin solution

```
public Task<CoinSpend> GetPuzzleAndSolution(string coinId, uint height, CancellationToken cancellationToken = default)
```

Parameters

coinId [string](#)

Id/name of the coin

height [uint](#)

Block height at which the coin was spent 'spent_block_index'

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[CoinSpend](#)>

A [CoinSpend](#)

Remarks

coinId is the coin name

GetRecentEOS(string, CancellationToken)

Gets a recent end of slot

```
public Task<(EndOfSubSlotBundle Eos, double TimeReceived, bool Reverted, DateTime  
DateTimeReceived)> GetRecentEOS(string challengeHash, CancellationToken cancellationToken  
= default)
```

Parameters

challengeHash [string](#)

challenge hash

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<(EndOfSubSlotBundle Eos, double TimeReceived, bool Reverted, DateTime DateTime
Received)>

The [EndOfSubSlotBundle](#)

GetRecentSignagePoint(string, CancellationToken)

Gets a recent signage point

```
public Task<(SignagePoint SignagePoint, double TimeReceived, bool Reverted, DateTime  
DateTimeReceived)> GetRecentSignagePoint(string spHash, CancellationToken cancellationToken  
= default)
```

Parameters

spHash [string](#)

signage point hash

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<([SignagePoint](#) [SignagePoint](#), [double](#) [TimeReceived](#), [bool](#) [Reverted](#), [DateTime](#) [Date](#) [TimeReceived](#))>

The [SignagePoint](#)

GetUnfinishedBlockHeaders(CancellationToken)

Get unfinished block headers

```
public Task<IEnumerable<UnfinishedHeaderBlock>> GetUnfinishedBlockHeaders(CancellationToken cancellationToken = default)
```

Parameters

[cancellationToken](#) [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[IEnumerable](#)<[UnfinishedHeaderBlock](#)>>

A list of [UnfinishedHeaderBlock](#)s

PushTx(SpendBundle, CancellationToken)

Pushes a transaction / spend bundle to the mempool and blockchain. Returns whether the spend bundle was successfully included into the mempool

```
public Task<bool> PushTx(SpendBundle spendBundle, CancellationToken cancellationToken = default)
```

Parameters

[spendBundle](#) [SpendBundle](#)

[cancellationToken](#) [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <bool>

Indicator of whether the spend bundle was successfully included in the mempool

WaitForSync(int, CancellationToken)

Will wait until [GetBlockchainState\(CancellationToken\)](#) indicates that the full node has synced or until the cancellation token is canceled

```
public Task WaitForSync(int millisecondsDelay = 10000, CancellationToken cancellationToken  
= default)
```

Parameters

`millisecondsDelay` [int](#)

The number of milliseconds to wait each time before checking sync status

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

An awaitable [Task](#)

Exceptions

[TaskCanceledException](#)

When cancellation token expires or is cancelled

Class FungibleAsset

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record FungibleAsset : IEquatable<FungibleAsset>
```

Inheritance

[object](#) ← FungibleAsset

Implements

[IEquatable](#) <[FungibleAsset](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Amount

```
public ulong Amount { get; init; }
```

Property Value

[ulong](#)

Asset

```
public string Asset { get; init; }
```

Property Value

[string](#)

Class HarvesterConfig

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record HarvesterConfig : IEquatable<HarvesterConfig>
```

Inheritance

[object](#) ← HarvesterConfig

Implements

[IEquatable](#)<[HarvesterConfig](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

DecompressorThreadCount

```
public int DecompressorThreadCount { get; init; }
```

Property Value

[int](#)

DisableCpuAffinity

```
public bool DisableCpuAffinity { get; init; }
```

Property Value

[bool](#)

EnforceGpuIndex

```
public bool EnforceGpuIndex { get; init; }
```

Property Value

[bool](#) ↗

GpuIndex

```
public int GpuIndex { get; init; }
```

Property Value

[int](#) ↗

ParallelDecompressorCount

```
public int ParallelDecompressorCount { get; init; }
```

Property Value

[int](#) ↗

RecursivePlotScan

```
public bool RecursivePlotScan { get; init; }
```

Property Value

[bool](#) ↗

RefreshParameterIntervalSeconds

```
public uint RefreshParameterIntervalSeconds { get; init; }
```

Property Value

[uint](#)

UseGpuHarvesting

```
public bool UseGpuHarvesting { get; init; }
```

Property Value

[bool](#)

Class HarvesterConnection

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record HarvesterConnection : IEquatable<HarvesterConnection>
```

Inheritance

[object](#) ← HarvesterConnection

Implements

[IEquatable](#)<[HarvesterConnection](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Host

```
public string Host { get; init; }
```

Property Value

[string](#)

IsLocal

Flag indicating whether the harvester is local to the node

```
[JsonIgnore]  
public bool IsLocal { get; }
```

Property Value

[bool](#) ↗

NodeId

```
public string NodeId { get; init; }
```

Property Value

[string](#) ↗

Port

```
public int Port { get; init; }
```

Property Value

[int](#) ↗

Class HarvesterInfo

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record HarvesterInfo : IEquatable<HarvesterInfo>
```

Inheritance

[object](#) ← HarvesterInfo

Implements

[IEquatable](#)<[HarvesterInfo](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Connection

```
public HarvesterConnection Connection { get; init; }
```

Property Value

[HarvesterConnection](#)

Duplicates

```
public IEnumerable<string> Duplicates { get; init; }
```

Property Value

[IEnumerable](#)<[string](#)>

FailedToOpenFileNames

```
public IEnumerable<string> FailedToOpenFileNames { get; init; }
```

Property Value

[IEnumerable](#)<[string](#)>

LastSyncDateTime

```
[JsonIgnore]
public DateTime LastSyncDateTime { get; }
```

Property Value

[DateTime](#)

LastSyncTime

```
public double LastSyncTime { get; init; }
```

Property Value

[double](#)

NoKeyFilenames

```
public IEnumerable<string> NoKeyFilenames { get; init; }
```

Property Value

[IEnumerable](#)<[string](#)>

Plots

```
public IEnumerable<PlotInfo> Plots { get; init; }
```

Property Value

[IEnumerable](#) <[PlotInfo](#)>

Syncing

```
public HarvesterSync Syncing { get; init; }
```

Property Value

[HarvesterSync](#)

TotalPlotSize

```
public long TotalPlotSize { get; init; }
```

Property Value

[long](#)

Class HarvesterProxy

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

Proxy that communicates with the harvester

```
public sealed class HarvesterProxy : ServiceProxy
```

Inheritance

[object](#) ← [ServiceProxy](#) ← HarvesterProxy

Inherited Members

[ServiceProxy.OriginService](#) , [ServiceProxy.DestinationService](#) , [ServiceProxy.RpcClient](#) ,
[ServiceProxy.HealthZ\(CancellationToken\)](#) , [ServiceProxy.StopNode\(CancellationToken\)](#) ,
[ServiceProxy.GetConnections\(CancellationToken\)](#) , [ServiceProxy.GetRoutes\(CancellationToken\)](#) ,
[ServiceProxy.OpenConnection\(string, int, CancellationToken\)](#) ,
[ServiceProxy.CloseConnection\(string, CancellationToken\)](#) , [object.Equals\(object\)](#) ,
[object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Remarks

ctor

Constructors

HarvesterProxy(IRpcClient, string)

Proxy that communicates with the harvester

```
public HarvesterProxy(IRpcClient rpcClient, string originService)
```

Parameters

rpcClient [IRpcClient](#)

[IRpcClient](#) instance to use for rpc communication

`originService` [string](#)

[Origin](#)

Remarks

ctor

Methods

AddPlotDirectory(string, CancellationToken)

Add a plot directory to the harvester configuration

```
public Task AddPlotDirectory(string dirname, CancellationToken cancellationToken = default)
```

Parameters

`dirname` [string](#)

The plot directory to add

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

An awaitable [Task](#)

DeletePlot(string, CancellationToken)

Permanently delete a plot file

```
public Task DeletePlot(string filename, CancellationToken cancellationToken = default)
```

Parameters

`filename` [string](#)

the file name of the plot

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

An awaitable [Task](#)

Remarks

Calling this DELETES the plot file. Proceed with caution.

GetHarvesterConfig(CancellationToken)

Gets harvester configuration.

```
public Task<HarvesterConfig> GetHarvesterConfig(CancellationToken cancellationToken  
= default)
```

Parameters

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[HarvesterConfig](#)>

[HarvesterConfig](#)

GetPlotDirectories(CancellationToken)

Get the list of plot directories from the harvester configuration

```
public Task<IEnumerable<string>> GetPlotDirectories(CancellationToken cancellationToken = default)
```

Parameters

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<IEnumerable<string>>

List of directories

GetPlots(CancellationToken)

Get the list of plot files

```
public Task<(IEnumerable<string> FailedToOpenFilenames, IEnumerable<string> NotFoundFileNames, IEnumerable<PlotInfo> Plots)> GetPlots(CancellationToken cancellationToken = default)
```

Parameters

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<(IEnumerable<string> FailedToOpenFilenames, IEnumerable<string> NotFoundFileNames, IEnumerable<PlotInfo> Plots)>

A list of plots

RefreshPlots(CancellationToken)

Refresh the list of plots

```
public Task RefreshPlots(CancellationToken cancellationToken = default)
```

Parameters

cancellationToken [CancellationToken ↗](#)

A token to allow the call to be cancelled

Returns

[Task ↗](#)

An awaitable [Task ↗](#)

RemovePlotDirectory(string, CancellationToken)

Removes a plot directory from the harveser configuration

```
public Task RemovePlotDirectory(string dirname, CancellationToken cancellationToken
= default)
```

Parameters

dirname [string ↗](#)

The plot directory to remove

cancellationToken [CancellationToken ↗](#)

A token to allow the call to be cancelled

Returns

[Task ↗](#)

An awaitable [Task ↗](#)

UpdateHarvesterConfig(bool?, int?, bool?, bool?, int?, int?, bool?, uint?, CancellationToken)

Sets harvester configuration.

```
public Task UpdateHarvesterConfig(bool? useGpuHarvesting = null, int? gpuIndex = null, bool?  
enforceGpuIndex = null, bool? disableCpuAffinity = null, int? parallelDecompressorCount =  
null, int? decompressorThreadCount = null, bool? recursivePlotScan = null, uint?  
refreshParameterIntervalSeconds = null, CancellationToken cancellationToken = default)
```

Parameters

useGpuHarvesting [bool](#)?

gpuIndex [int](#)?

enforceGpuIndex [bool](#)?

disableCpuAffinity [bool](#)?

parallelDecompressorCount [int](#)?

decompressorThreadCount [int](#)?

recursivePlotScan [bool](#)?

refreshParameterIntervalSeconds [uint](#)?

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

An awaitable task>

Class HarvesterSummary

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record HarvesterSummary : IEquatable<HarvesterSummary>
```

Inheritance

[object](#) ← HarvesterSummary

Implements

[IEquatable](#)<[HarvesterSummary](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Connection

```
public HarvesterConnection Connection { get; init; }
```

Property Value

[HarvesterConnection](#)

Duplicates

```
public int Duplicates { get; init; }
```

Property Value

[int](#)

FailedToOpenFileNames

```
public int FailedToOpenFileNames { get; init; }
```

Property Value

[int ↗](#)

LastSyncDateTime

```
[JsonIgnore]  
public DateTime LastSyncDateTime { get; }
```

Property Value

[DateTime ↗](#)

LastSyncTime

```
public double LastSyncTime { get; init; }
```

Property Value

[double ↗](#)

NoKeyFilenames

```
public int NoKeyFilenames { get; init; }
```

Property Value

[int ↗](#)

NotFoundFileNames

```
public int NotFoundFileNames { get; init; }
```

Property Value

[int](#) ↗

Plots

```
public int Plots { get; init; }
```

Property Value

[int](#) ↗

Syncing

```
public HarvesterSync Syncing { get; init; }
```

Property Value

[HarvesterSync](#)

TotalPlotSize

```
public long TotalPlotSize { get; init; }
```

Property Value

[long](#) ↗

Class HarvesterSync

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record HarvesterSync : IEquatable<HarvesterSync>
```

Inheritance

[object](#) ← HarvesterSync

Implements

[IEquatable](#) <[HarvesterSync](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Initial

```
public bool Initial { get; init; }
```

Property Value

[bool](#)

PlotFilesProcessed

```
public uint PlotFilesProcessed { get; init; }
```

Property Value

[uint](#)

PlotFilesTotal

```
public uint PlotFilesTotal { get; init; }
```

Property Value

[uint](#)

Class HashFilter

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record HashFilter : IEquatable<HashFilter>
```

Inheritance

[object](#) ← HashFilter

Implements

[IEquatable](#)<[HashFilter](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Mode

```
public FilterMode Mode { get; init; }
```

Property Value

[FilterMode](#)

Values

```
public IEnumerable<string> Values { get; init; }
```

Property Value

[IEnumerable](#)<[string](#)>

Class HttpRpcClient

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

Class that handles core communication with the rpc endpoint using http(s)

```
public class HttpRpcClient : IRpcClient, IDisposable
```

Inheritance

[object](#) ← HttpRpcClient

Implements

[IRpcClient](#), [IDisposable](#)

Inherited Members

[object.Equals\(object\)](#), [object.Equals\(object, object\)](#), [object.GetHashCode\(\)](#), [object.GetType\(\)](#),
[object.MemberwiseClone\(\)](#), [object.ReferenceEquals\(object, object\)](#), [object.ToString\(\)](#)

Constructors

HttpRpcClient(EndpointInfo)

ctor

```
public HttpRpcClient(EndpointInfo endpoint)
```

Parameters

endpoint [EndpointInfo](#)

Details of the service endpoint

HttpRpcClient(EndpointInfo, DelegatingHandler)

ctor

```
public HttpRpcClient(EndpointInfo endpoint, DelegatingHandler delegatingHandler)
```

Parameters

endpoint [EndpointInfo](#)

Details of the service endpoint

delegatingHandler [DelegatingHandler](#)

A handler created elsewhere that may have things like resiliency chains

HttpRpcClient(EndpointInfo, HttpClient)

ctor

```
public HttpRpcClient(EndpointInfo endpoint, HttpClient httpClient)
```

Parameters

endpoint [EndpointInfo](#)

Details of the service endpoint

httpClient [HttpClient](#)

A fully configured client, including ssl certs, intended for use with IHttpClientFactory

Properties

Endpoint

Details of the RPC service endpoint

```
public EndpointInfo Endpoint { get; init; }
```

Property Value

[EndpointInfo](#)

Class that handles core communication with the rpc endpoint using http(s)

Methods

Dispose()

[Dispose\(\)](#)

```
public void Dispose()
```

Dispose(bool)

Called when the instance is being disposed or finalized

```
protected virtual void Dispose(bool disposing)
```

Parameters

disposing [bool](#)

Invoke from [Dispose\(\)](#)

PostMessage(Message, CancellationToken)

Posts a [Message](#) to the [Endpoint](#) but does not wait for a response

```
public Task PostMessage(Message message, CancellationToken cancellationToken = default)
```

Parameters

message [Message](#)

The message to send

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

Awaitable [Task](#)

Remarks

Awaiting this method waits for the message to be sent only. It doesn't await a response.

SendMessage(Message, CancellationToken)

Sends a [Message](#) to the endpoint and waits for a response

```
public Task<dynamic> SendMessage(Message message, CancellationToken cancellationToken  
= default)
```

Parameters

message [Message](#)

The message to send

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<dynamic>

The response message

Remarks

Awaiting this method will block until a response is received from the rpc endpoint or the A token to allow the call to be cancelled is cancelled

Exceptions

[ResponseException](#)

Throws when [`IsSuccessfulResponse`](#) is False

Interface IRpcClient

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

Interface representing rpc communication

```
public interface IRpcClient : IDisposable
```

Inherited Members

[IDisposable.Dispose\(\)](#)

Properties

Endpoint

Details of the RPC service endpoint

```
EndpointInfo Endpoint { get; init; }
```

Property Value

[EndpointInfo](#)

Interface representing rpc communication

Methods

PostMessage(Message, CancellationToken)

Posts a [Message](#) to the [Endpoint](#) but does not wait for a response

```
Task PostMessage(Message message, CancellationToken cancellationToken = default)
```

Parameters

`message` [Message](#)

The message to post

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

`Awaitable` [Task](#)

Remarks

Awaiting this method waits for the message to be sent only. It doesn't await a response.

SendMessage(Message, CancellationToken)

Sends a [Message](#) to the [Endpoint](#) and waits for a response

```
Task<dynamic> SendMessage(Message message, CancellationToken cancellationToken = default)
```

Parameters

`message` [Message](#)

The message to send

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<dynamic>

The response message

Remarks

Awaiting this method will block until a response is received from the [Endpoint](#) or the A token to allow the call to be cancelled is cancelled

Exceptions

[ResponseException](#)

Throws when [IsSuccessfulResponse](#) is False

Class InfusedChallengeChainSubSlot

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record InfusedChallengeChainSubSlot : IEquatable<InfusedChallengeChainSubSlot>
```

Inheritance

[object](#) ← InfusedChallengeChainSubSlot

Implements

[IEquatable](#)<[InfusedChallengeChainSubSlot](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

InfusedChallengeChainEndOfSlotVdf

```
public VDFInfo InfusedChallengeChainEndOfSlotVdf { get; init; }
```

Property Value

[VDFInfo](#)

Class InternalNode

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record InternalNode : IEquatable<InternalNode>
```

Inheritance

[object](#) ← InternalNode

Implements

[IEquatable](#)<[InternalNode](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Atom

```
public string Atom { get; init; }
```

Property Value

[string](#)

Hash

```
public string Hash { get; init; }
```

Property Value

[string](#)

LeftHash

```
public string LeftHash { get; init; }
```

Property Value

[string](#)

Pair

```
public (string, string) Pair { get; init; }
```

Property Value

[\(string\)](#), [\(string\)](#)

RightHash

```
public string RightHash { get; init; }
```

Property Value

[string](#)

Enum KSize

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

Valid plot sizes <https://github.com/Chia-Network/chia-blockchain/wiki/k-sizes>

```
public enum KSize
```

Fields

K25 = 25

Valid for testing only - [OverrideK](#) must be true in order to use

K32 = 32

K33 = 33

K34 = 34

K35 = 35

Class KVDiff

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record KVDiff : IEquatable<KVDiff>
```

Inheritance

[object](#) ← KVDiff

Implements

[IEquatable](#) <[KVDiff](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Key

```
public string Key { get; init; }
```

Property Value

[string](#)

Type

```
public string Type { get; init; }
```

Property Value

[string](#)

Value

```
public string Value { get; init; }
```

Property Value

[string](#) ↗

Class KeyData

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record KeyData : IEquatable<KeyData>
```

Inheritance

[object](#) ← KeyData

Implements

[IEquatable](#) <[KeyData](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Fingerprint

```
public uint Fingerprint { get; init; }
```

Property Value

[uint](#)

Label

```
public string? Label { get; init; }
```

Property Value

[string](#)

PublicKey

```
public string PublicKey { get; init; }
```

Property Value

[string](#) ↗

Secrets

```
public KeyDataSecrets? Secrets { get; init; }
```

Property Value

[KeyDataSecrets](#)

Class KeyDataSecrets

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record KeyDataSecrets : IEquatable<KeyDataSecrets>
```

Inheritance

[object](#) ← KeyDataSecrets

Implements

[IEquatable](#)<[KeyDataSecrets](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Bytes

```
public string Bytes { get; init; }
```

Property Value

[string](#)

Mnemonic

```
public IEnumerable<string> Mnemonic { get; init; }
```

Property Value

[IEnumerable](#)<[string](#)>

PrivateKey

```
public PrivateKey PrivateKey { get; init; }
```

Property Value

[PrivateKey](#)

Class KeyringStatus

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record KeyringStatus : IEquatable<KeyringStatus>
```

Inheritance

[object](#) ← KeyringStatus

Implements

[IEquatable](#) <[KeyringStatus](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

CanRemoveLegacyKeys

```
public bool CanRemoveLegacyKeys { get; init; }
```

Property Value

[bool](#)

CanSavePassphrase

```
public bool CanSavePassphrase { get; init; }
```

Property Value

[bool](#)

CanSetPassphraseHint

```
public bool CanSetPassphraseHint { get; init; }
```

Property Value

[bool](#) ↗

IsKeyringLocked

```
public bool IsKeyringLocked { get; init; }
```

Property Value

[bool](#) ↗

NeedsMigration

```
public bool NeedsMigration { get; init; }
```

Property Value

[bool](#) ↗

PassphraseHint

```
public string PassphraseHint { get; init; }
```

Property Value

[string](#) ↗

PassphraseRequirements

```
public PassphraseRequirements PassphraseRequirements { get; init; }
```

Property Value

[PassphraseRequirements](#)

PassphreaseSupportEnabled

```
public bool PassphreaseSupportEnabled { get; init; }
```

Property Value

[bool](#)

UserPassphrasesSet

```
public bool UserPassphraseIsSet { get; init; }
```

Property Value

[bool](#)

Class Layer

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record Layer : IEquatable<Layer>
```

Inheritance

[object](#) ← Layer

Implements

[IEquatable](#)<[Layer](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

CombinedHash

```
public string CombinedHash { get; init; }
```

Property Value

[string](#)

OtherHash

```
public string OtherHash { get; init; }
```

Property Value

[string](#)

OtherHashSide

```
public Side OtherHashSide { get; init; }
```

Property Value

[Side](#)

Class LineageProof

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record LineageProof : IEquatable<LineageProof>
```

Inheritance

[object](#) ← LineageProof

Implements

[IEquatable](#)<[LineageProof](#)>

Derived

[VCLineageProof](#)

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Amount

```
public ulong? Amount { get; init; }
```

Property Value

[ulong](#)?

InnerPuzzleHash

```
public string? InnerPuzzleHash { get; init; }
```

Property Value

[string](#) ↗

ParentName

```
public string? ParentName { get; init; }
```

Property Value

[string](#) ↗

Enum MempoolInclusionStatus

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public enum MempoolInclusionStatus : byte
```

Fields

FAILED = 3

Transaction was invalid and dropped

PENDING = 2

Transaction not yet added to mempool

SUCCESS = 1

Transaction added to mempool

Class MempoolItem

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record MempoolItem : IEquatable<MempoolItem>
```

Inheritance

[object](#) ← MempoolItem

Implements

[IEquatable](#) <[MempoolItem](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Additions

```
public IEnumerable<Coin> Additions { get; init; }
```

Property Value

[IEnumerable](#) <[Coin](#)>

Cost

```
public ulong Cost { get; init; }
```

Property Value

[ulong](#)

Fee

```
public ulong Fee { get; init; }
```

Property Value

[ulong](#) ↗

NPCResult

```
public NPCResult NPCResult { get; init; }
```

Property Value

[NPCResult](#)

Program

```
public string Program { get; init; }
```

Property Value

[string](#) ↗

Removals

```
public IEnumerable<Coin> Removals { get; init; }
```

Property Value

[IEnumerable](#) ↗ <[Coin](#)>

SpendBudndleName

```
public string SpendBudndleName { get; init; }
```

Property Value

[string](#) ↗

SpendBundle

```
public SpendBundle SpendBundle { get; init; }
```

Property Value

[SpendBundle](#)

Class MempoolMinFees

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record MempoolMinFees : IEquatable<MempoolMinFees>
```

Inheritance

[object](#) ← MempoolMinFees

Implements

[IEquatable](#) <[MempoolMinFees](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Cost5000000

```
public double Cost5000000 { get; init; }
```

Property Value

[double](#)

Class Message

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

The messaging data structure for request and response exchange with the RPC endpoint

```
public record Message : IEquatable<Message>
```

Inheritance

[object](#) ← Message

Implements

[IEquatable](#) <[Message](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Ack

Indication whether message is an acknowledgement (i.e response)

```
public bool Ack { get; init; }
```

Property Value

[bool](#)

The messaging data structure for request and response exchange with the RPC endpoint

Command

The command to be processed by the endpoint service

```
public string Command { get; init; }
```

Property Value

[string](#) ↗

The messaging data structure for request and response exchange with the RPC endpoint

Data

Data to go along with the command

```
public dynamic? Data { get; init; }
```

Property Value

dynamic

The messaging data structure for request and response exchange with the RPC endpoint

Destination

The name of the destination service

```
public string Destination { get; init; }
```

Property Value

[string](#) ↗

The messaging data structure for request and response exchange with the RPC endpoint

IsSuccessfulResponse

Inidicates whether this is a response ([Ack](#) is true) and the success flag is also true

```
[JsonIgnore]  
public bool IsSuccessfulResponse { get; }
```

Property Value

[bool](#)

The messaging data structure for request and response exchange with the RPC endpoint

Origin

The name of the origin service

```
public string Origin { get; init; }
```

Property Value

[string](#)

The messaging data structure for request and response exchange with the RPC endpoint

RequestId

Unique correlation id of the message. This will round trip to the RPC server and back in its response

```
public string RequestId { get; init; }
```

Property Value

[string](#)

The messaging data structure for request and response exchange with the RPC endpoint

Methods

Create(string, object?, string, string)

Construct a new instance of a [Message](#)

```
public static Message Create(string command, object? data, string destination,  
string origin)
```

Parameters

command [string](#)

[Command](#)

data [object](#)

[Data](#)

destination [string](#)

[Destination](#)

origin [string](#)

[Origin](#)

Returns

[Message](#)

A populated [Message](#)

Remarks

Ensure that [Data](#) and [RequestId](#) are set appropriately

Class Mirror

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record Mirror : IEquatable<Mirror>
```

Inheritance

[object](#) ← Mirror

Implements

[IEquatable](#) <[Mirror](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Amount

```
public ulong Amount { get; init; }
```

Property Value

[ulong](#)

CoinId

```
public string CoinId { get; init; }
```

Property Value

[string](#)

LauncherId

```
public string LauncherId { get; init; }
```

Property Value

[string](#)

Ours

```
public bool Ours { get; init; }
```

Property Value

[bool](#)

Urls

```
public IEnumerable<string>Urls { get; init; }
```

Property Value

[IEnumerable](#)<[string](#)>

Class NFTBulkMintingInfo

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

Info for minting NFTs in bulk

```
public record NFTBulkMintingInfo : IEquatable<NFTBulkMintingInfo>
```

Inheritance

[object](#) ← NFTBulkMintingInfo

Implements

[IEquatable](#)<[NFTBulkMintingInfo](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

DidCoin

```
public Coin? DidCoin { get; init; }
```

Property Value

[Coin](#)

Info for minting NFTs in bulk

DidLineageParentHex

```
public string? DidLineageParentHex { get; init; }
```

Property Value

[string](#)

Info for minting NFTs in bulk

MetadataList

```
public IEnumerable<NftMintEntry> MetadataList { get; init; }
```

Property Value

[IEnumerable](#)<[NftMintEntry](#)>

Info for minting NFTs in bulk

MintFromDid

```
public bool MintFromDid { get; init; }
```

Property Value

[bool](#)

Info for minting NFTs in bulk

MintNumberStart

The starting point for mint number used in intermediate launcher puzzle

```
public int MintNumberStart { get; init; }
```

Property Value

[int](#)

Info for minting NFTs in bulk

MintTotal

The total number of NFTs being minted

```
public int? MintTotal { get; init; }
```

Property Value

[int](#)?

Info for minting NFTs in bulk

NewInnerpuzhash

```
public string? NewInnerpuzhash { get; init; }
```

Property Value

[string](#)?

Info for minting NFTs in bulk

NewP2Puzhash

```
public string? NewP2Puzhash { get; init; }
```

Property Value

[string](#)?

Info for minting NFTs in bulk

RoyaltyAddress

```
public string? RoyaltyAddress { get; init; }
```

Property Value

[string](#)

Info for minting NFTs in bulk

RoyaltyPercentage

```
public ushort? RoyaltyPercentage { get; init; }
```

Property Value

[ushort](#)?

Info for minting NFTs in bulk

TargetList

a list of targets for transferring minted NFTs (aka airdrop)

```
public IEnumerable<string>? TargetList { get; init; }
```

Property Value

[IEnumerable](#)<[string](#)>

Info for minting NFTs in bulk

XchChangeTarget

For use with bulk minting, so we can specify the puzzle hash that the change from the funding transaction goes to.

```
public string? XchChangeTarget { get; init; }
```

Property Value

[string](#)

Info for minting NFTs in bulk

XchCoins

For use with bulk minting to provide the coin used for funding the minting spend. This coin can be one that will be created in the future

```
public IEnumerable<Coin>? XchCoins { get; init; }
```

Property Value

[IEnumerable](#) <[Coin](#)>

Info for minting NFTs in bulk

Class NFTInfo

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

NFT Info for displaying NFT on the UI

```
public record NFTInfo : IEquatable<NFTInfo>
```

Inheritance

[object](#) ← NFTInfo

Implements

[IEquatable](#)<[NFTInfo](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

ChainInfo

Information saved on the chain in hex

```
public string ChainInfo { get; init; }
```

Property Value

[string](#)

NFT Info for displaying NFT on the UI

DataHash

Hash of the content

```
public string DataHash { get; init; }
```

Property Value

[string](#)

NFT Info for displaying NFT on the UI

DataUris

A list of content URIs

```
public IEnumerable<string> DataUris { get; init; }
```

Property Value

[IEnumerable](#)<[string](#)>

NFT Info for displaying NFT on the UI

EditionNumber

Number of the current NFT in the edition

```
public ulong EditionNumber { get; init; }
```

Property Value

[ulong](#)

NFT Info for displaying NFT on the UI

EditionTotal

How many NFTs in the current edition

```
public ulong EditionTotal { get; init; }
```

Property Value

[ulong](#) ↗

NFT Info for displaying NFT on the UI

LauncherId

Launcher coin ID

```
public string LauncherId { get; init; }
```

Property Value

[string](#) ↗

NFT Info for displaying NFT on the UI

LauncherPuzhash

Puzzle hash of the singleton launcher in hex

```
public string LauncherPuzhash { get; init; }
```

Property Value

[string](#) ↗

NFT Info for displaying NFT on the UI

LicenseHash

Hash of the license

```
public string LicenseHash { get; init; }
```

Property Value

[string](#)

NFT Info for displaying NFT on the UI

LicenseUris

A list of license URLs

```
public IEnumerable<string> LicenseUris { get; init; }
```

Property Value

[IEnumerable](#) <[string](#)>

NFT Info for displaying NFT on the UI

MetadataHash

Hash of the metadata

```
public string MetadataHash { get; init; }
```

Property Value

[string](#)

NFT Info for displaying NFT on the UI

MetadataUris

A list of metadata URLs

```
public IEnumerable<string> MetadataUris { get; init; }
```

Property Value

[IEnumerable](#)<[string](#)>

NFT Info for displaying NFT on the UI

MintHeight

Block height of the NFT minting

```
public uint MintHeight { get; init; }
```

Property Value

[uint](#)

NFT Info for displaying NFT on the UI

MinterDID

DID of the NFT minter

```
public string? MinterDID { get; init; }
```

Property Value

[string](#)

NFT Info for displaying NFT on the UI

NFTCoinID

Current NFT coin ID

```
public string NFTCoinID { get; init; }
```

Property Value

[string](#) ↗

NFT Info for displaying NFT on the UI

OffChainMetadata

Serialized off-chain metadata

```
public string? OffChainMetadata { get; init; }
```

Property Value

[string](#) ↗

NFT Info for displaying NFT on the UI

OwnerDID

Owner DID

```
public string? OwnerDID { get; init; }
```

Property Value

[string](#) ↗

NFT Info for displaying NFT on the UI

P2Address

The innermost puzzle hash of the NFT

```
public string P2Address { get; init; }
```

Property Value

[string](#) ↗

NFT Info for displaying NFT on the UI

PendingTransaction

Indicate if the NFT is pending for a transaction

```
public bool PendingTransaction { get; init; }
```

Property Value

[bool](#) ↗

NFT Info for displaying NFT on the UI

RoyaltyPercentage

Percentage of the transaction fee paid to the author, e.g. 1000 = 1%

```
public ushort? RoyaltyPercentage { get; init; }
```

Property Value

[ushort](#) ↗?

NFT Info for displaying NFT on the UI

RoyaltyPuzzleHash

uzzle hash where royalty will be sent to

```
public string? RoyaltyPuzzleHash { get; init; }
```

Property Value

[string](#) ↗

NFT Info for displaying NFT on the UI

SupportsDID

If the inner puzzle supports DID

```
public bool SupportsDID { get; init; }
```

Property Value

[bool](#) ↗

NFT Info for displaying NFT on the UI

UpdaterPuzhash

Puzzle hash of the metadata updater in hex

```
public string UpdaterPuzhash { get; init; }
```

Property Value

[string](#) ↗

NFT Info for displaying NFT on the UI

Class NFTMintingInfo

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

Info for minting an NFT

```
public record NFTMintingInfo : NftMintEntry, IEquatable<NftMintEntry>,  
IEquatable<NFTMintingInfo>
```

Inheritance

[object](#) ← [NftMintEntry](#) ← NFTMintingInfo

Implements

[IEquatable](#)<[NftMintEntry](#)>, [IEquatable](#)<[NFTMintingInfo](#)>

Inherited Members

[NftMintEntry.RoyaltyPercentage](#), [NftMintEntry.Uris](#), [NftMintEntry.Hash](#), [NftMintEntry.MetaUris](#),
[NftMintEntry.MetaHash](#), [NftMintEntry.LicenseUris](#), [NftMintEntry.LicenseHash](#),
[NftMintEntry.EditionTotal](#), [NftMintEntry.EditionNumber](#), [object.Equals\(object\)](#),
[object.Equals\(object, object\)](#), [object.GetHashCode\(\)](#), [object.GetType\(\)](#),
[object.MemberwiseClone\(\)](#), [object.ReferenceEquals\(object, object\)](#), [object.ToString\(\)](#)

Properties

DidId

```
public string? DidId { get; init; }
```

Property Value

[string](#)

Info for minting an NFT

RoyaltyAddress

```
public string? RoyaltyAddress { get; init; }
```

Property Value

[string](#) ↗

Info for minting an NFT

TargetAddress

```
public string? TargetAddress { get; init; }
```

Property Value

[string](#) ↗

Info for minting an NFT

Class NFTWallet

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

Wraps an NFT wallet

```
public sealed class NFTWallet : Wallet
```

Inheritance

[object](#) ← [Wallet](#) ← NFTWallet

Inherited Members

[Wallet.WalletId](#) , [Wallet.WalletProxy](#) , [Wallet.GetWalletInfo\(CancellationToken\)](#) ,
[Wallet.GetBalance\(CancellationToken\)](#) ,
[Wallet.SelectCoins\(ulong, IEnumerable<Coin>, IEnumerable<ulong>, ulong?, ulong?, CancellationToken\)](#) ,
,
[Wallet.GetTransactions\(string, TransactionTypeFilter, bool, string, uint, uint, bool?, CancellationToken\)](#) ,
[Wallet.GetSpendableCoins\(ulong?, ulong?, IEnumerable<ulong>, IEnumerable<Coin>,
IEnumerable<string>, CancellationToken\)](#) ,
[Wallet.GetNextAddress\(bool, CancellationToken\)](#) ,
[Wallet.GetTransactionCount\(TransactionTypeFilter, CancellationToken\)](#) ,
[Wallet.DeleteUnconfirmedTransactions\(CancellationToken\)](#) ,
[Wallet.SendTransaction\(string, ulong, IEnumerable<string>, IEnumerable<ulong>, IEnumerable<string>,
ulong?, ulong?, bool, ulong, CancellationToken\)](#) ,
[Wallet.SendTransactionMulti\(IEnumerable<Coin>, IEnumerable<Coin>, ulong, CancellationToken\)](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Remarks

ctor

Constructors

NFTWallet(uint, WalletProxy)

Wraps an NFT wallet

```
public NFTWallet(uint walletId, WalletProxy walletProxy)
```

Parameters

walletId [uint](#)

The wallet_id to wrap

walletProxy [WalletProxy](#)

Wallet RPC proxy to use for communication

Remarks

ctor

Methods

AddUri(string, string, string, bool?, ulong, CancellationToken)

Adds an Uri to an NFT

```
public Task<SpendBundle> AddUri(string uri, string key, string nftCoinId, bool? reusePuzhash  
= null, ulong fee = 0, CancellationToken cancellationToken = default)
```

Parameters

uri [string](#)

The uri

key [string](#)

The type of uri:

- u Uri for the NFT data
- mu Uri for NFT metadata
- lu Uri for the NFT license

nftCoinId [string](#)

The nft coin id

reusePuzhash [bool](#)?

fee [ulong](#)

Transaction fee

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[SpendBundle](#)>

An [SpendBundle](#)

GetDid(CancellationToken)

Gets the DID

```
public Task<string> GetDid(CancellationToken cancellationToken = default)
```

Parameters

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[string](#)>

The Did

GetNFTs(int, int, bool, CancellationToken)

Gets NFTs from a wallet

```
public Task<IEnumerable<NFTInfo>> GetNFTs(int startIndex = 0, int num = 0, bool ignoreSizeLimit = false, CancellationToken cancellationToken = default)
```

Parameters

startIndex [int](#)

num [int](#)

ignoreSizeLimit [bool](#)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[IEnumerable](#)<[NFTInfo](#)>>

A list of [NFTInfo](#)

MintNFT(NFTMintingInfo, bool?, ulong, CancellationToken)

Mints an NFT

```
public Task<(SpendBundle SpendBundle, string NftId)> MintNFT(NFTMintingInfo info, bool? reusePuzhash = null, ulong fee = 0, CancellationToken cancellationToken = default)
```

Parameters

info [NFTMintingInfo](#)

Info about the NFT to be minted

reusePuzhash [bool](#)?

fee [ulong](#)

Transaction fee

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<([SpendBundle](#) [SpendBundle](#), [string](#) [NftId](#))>

A [SpendBundle](#)

NftCountNfts(CancellationToken)

Retrieves the number of NFTs in a wallet.

```
public Task<int> NftCountNfts(CancellationToken cancellationToken = default)
```

Parameters

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[int](#)>

The number of NFTs in the wallet

NftMintBulk(NFTBulkMintingInfo, bool?, ulong, CancellationToken)

Mints an set NFTs in bulk

```
public Task<(SpendBundle SpendBundle, IEnumerable<string> NftIdList)>
NftMintBulk(NFTBulkMintingInfo info, bool? reusePuzhash = null, ulong fee = 0,
CancellationToken cancellationToken = default)
```

Parameters

info [NFTBulkMintingInfo](#)

A list of dicts containing the metadata for each NFT to be minted

`reusePuzhash` [bool](#)?

`fee` [ulong](#)

Transaction fee

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <([SpendBundle](#) [SpendBundle](#), [IEnumerable](#) <[string](#)> [NftIdList](#))>

[SpendBundle](#) and a list of [string](#)

SetDID(string, string, bool?, ulong, CancellationToken)

Sets the DID for an NFT

```
public Task<SpendBundle> SetDID(string didId, string nftCoinId, bool? reusePuzhash = null,  
        ulong fee = 0, CancellationToken cancellationToken = default)
```

Parameters

`didId` [string](#)

The DID ID

`nftCoinId` [string](#)

The coin id for the nft

`reusePuzhash` [bool](#)?

`fee` [ulong](#)

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[SpendBundle](#)>

A [SpendBundle](#)

SetStatus(string, bool, CancellationToken)

Sets the status of an NFT

```
public Task SetStatus(string coinId, bool inTransaction = true, CancellationToken cancellationToken = default)
```

Parameters

coinId [string](#)

The coin ID

inTransaction [bool](#)

In transaction idicator

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

An awaitable [Task](#)

Transfer(string, string, bool?, ulong, CancellationToken)

Sets the status of an NFT

```
public Task<SpendBundle> Transfer(string targetAddress, string coinId, bool? reusePuzhash = null, ulong fee = 0, CancellationToken cancellationToken = default)
```

Parameters

`targetAddress` [string](#)

The target address

`coinId` [string](#)

The coin ID

`reusePuzhash` [bool](#)?

`fee` [ulong](#)

Transaction fee

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[SpendBundle](#)>

A [SpendBundle](#)

Validate(CancellationToken)

Validates that [WalletId](#) is a [NFT](#)

```
public override Task Validate(CancellationToken cancellationToken = default)
```

Parameters

`cancellationToken` [CancellationToken](#)

Wraps an NFT wallet

Returns

[Task](#)

True if the wallet is an NFT wallet

Class NPC

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record NPC : IEquatable<NPC>
```

Inheritance

[object](#) ← NPC

Implements

[IEquatable](#)<[NPC](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

CoinName

```
public string CoinName { get; init; }
```

Property Value

[string](#)

Conditions

```
public IEnumerable<Condition> Conditions { get; init; }
```

Property Value

[IEnumerable](#)<[Condition](#)>

PuzzleHash

```
public string PuzzleHash { get; init; }
```

Property Value

[string](#) ↗

Class NPCResult

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record NPCResult : IEquatable<NPCResult>
```

Inheritance

[object](#) ← NPCResult

Implements

[IEquatable](#)<[NPCResult](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

ClvmCost

CLVM cost only, cost of conditions and tx size is not included

```
public ulong ClvmCost { get; init; }
```

Property Value

[ulong](#)

Error

```
public ushort? Error { get; init; }
```

Property Value

[ushort](#)?

NpcList

```
public IEnumerable<NPC> NpcList { get; init; }
```

Property Value

[IEnumerable](#) <[NPC](#)>

Class NftCoinInfo

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record NftCoinInfo : IEquatable<NftCoinInfo>
```

Inheritance

[object](#) ← NftCoinInfo

Implements

[IEquatable](#)<[NftCoinInfo](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

NftCoinId

```
public string NftCoinId { get; init; }
```

Property Value

[string](#)

WalletId

```
public int WalletId { get; init; }
```

Property Value

[int](#)

Class NftMintEntry

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record NftMintEntry : IEquatable<NftMintEntry>
```

Inheritance

[object](#) ← NftMintEntry

Implements

[IEquatable](#) <[NftMintEntry](#)>

Derived

[NFTMintingInfo](#)

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

EditionNumber

```
public ulong EditionNumber { get; init; }
```

Property Value

[ulong](#)

EditionTotal

```
public ulong EditionTotal { get; init; }
```

Property Value

[ulong](#)

Hash

```
public string Hash { get; init; }
```

Property Value

[string](#)

LicenseHash

```
public string? LicenseHash { get; init; }
```

Property Value

[string](#)

LicenseUris

```
public IEnumerable<string> LicenseUris { get; init; }
```

Property Value

[IEnumerable](#)<[string](#)>

MetaHash

```
public string? MetaHash { get; init; }
```

Property Value

[string](#)

MetaUris

```
public IEnumerable<string> MetaUris { get; init; }
```

Property Value

[IEnumerable](#)<[string](#)>

RoyaltyPercentage

```
public ushort RoyaltyPercentage { get; init; }
```

Property Value

[ushort](#)

Uris

```
public IEnumerable<string> Uris { get; init; }
```

Property Value

[IEnumerable](#)<[string](#)>

Enum NodeType

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public enum NodeType
```

Fields

DATA_LAYER = 7

FARMER = 3

FULL_NODE = 1

HARVESTER = 2

INTRODUCER = 5

TIMELORD = 4

WALLET = 6

Class OfferRecord

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

The in memory representation of an offer and its record of trade

```
public record OfferRecord : IEquatable<OfferRecord>
```

Inheritance

[object](#) ← OfferRecord

Implements

[IEquatable](#) <[OfferRecord](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Offer

The bech32 encoded value of the offer

```
public string Offer { get; init; }
```

Property Value

[string](#)

The in memory representation of an offer and its record of trade

TradeRecord

Trade record associated with the offer

```
public TradeRecord TradeRecord { get; init; }
```

Property Value

[TradeRecord](#)

The in memory representation of an offer and its record of trade

Class OfferStore

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record OfferStore : IEquatable<OfferStore>
```

Inheritance

[object](#) ← OfferStore

Implements

[IEquatable](#) <[OfferStore](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Inclusions

```
public IEnumerable<KeyValuePair<string, string>> Inclusions { get; init; }
```

Property Value

[IEnumerable](#) <[KeyValuePair](#) <[string](#), [string](#)>>

StoreID

```
public string StoreID { get; init; }
```

Property Value

[string](#)

Class OfferSummary

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record OfferSummary : IEquatable<OfferSummary>
```

Inheritance

[object](#) ← OfferSummary

Implements

[IEquatable](#) <[OfferSummary](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Fees

```
public ulong Fees { get; init; }
```

Property Value

[ulong](#)

Id

```
public string Id { get; init; }
```

Property Value

[string](#)

Infos

```
public IDictionary<string, object> Infos { get; init; }
```

Property Value

[IDictionary](#)<[string](#), [object](#)>

Offered

```
public IDictionary<string, ulong> Offered { get; init; }
```

Property Value

[IDictionary](#)<[string](#), [ulong](#)>

Requested

```
public IDictionary<string, ulong> Requested { get; init; }
```

Property Value

[IDictionary](#)<[string](#), [ulong](#)>

Class PaginatedPlotRequest

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record PaginatedPlotRequest : IEquatable<PaginatedPlotRequest>
```

Inheritance

[object](#) ← PaginatedPlotRequest

Implements

[IEquatable](#)<[PaginatedPlotRequest](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

NodeId

```
public string NodeId { get; init; }
```

Property Value

[string](#)

Page

```
public int Page { get; init; }
```

Property Value

[int](#)

PageCount

```
public int PageCount { get; init; }
```

Property Value

[int↗](#)

Plots

```
public IEnumerable<PlotInfo> Plots { get; init; }
```

Property Value

[IEnumerable↗ <PlotInfo>](#)

TotalCount

```
public int TotalCount { get; init; }
```

Property Value

[int↗](#)

Class PassphraseRequirements

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record PassphraseRequirements : IEquatable<PassphraseRequirements>
```

Inheritance

[object](#) ← PassphraseRequirements

Implements

[IEquatable](#) <[PassphraseRequirements](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

IsOptional

```
public bool IsOptional { get; init; }
```

Property Value

[bool](#)

MinLength

```
public int MinLength { get; init; }
```

Property Value

[int](#)

Class PeerCounts

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record PeerCounts : IEquatable<PeerCounts>
```

Inheritance

[object](#) ← PeerCounts

Implements

[IEquatable](#)<[PeerCounts](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Ipv4Last5Days

```
[JsonProperty("ipv4_last_5_days")]
public int Ipv4Last5Days { get; init; }
```

Property Value

[int](#)

Ipv6Last5Days

```
[JsonProperty("ipv6_last_5_days")]
public int Ipv6Last5Days { get; init; }
```

Property Value

[int](#)

ReliableNodes

```
public int ReliableNodes { get; init; }
```

Property Value

[int](#)

TotalLast5Days

```
[JsonProperty("total_last_5_days")]
public int TotalLast5Days { get; init; }
```

Property Value

[int](#)

Versions

```
public IDictionary<string, int> Versions { get; init; }
```

Property Value

[IDictionary](#) <[string](#), [int](#)>

Class PlotInfo

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

Info about a plot file

```
public record PlotInfo : IEquatable<PlotInfo>
```

Inheritance

[object](#) ← PlotInfo

Implements

[IEquatable](#) <[PlotInfo](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

DateTimeModified

```
[JsonIgnore]  
public DateTime DateTimeModified { get; }
```

Property Value

[DateTime](#)

Info about a plot file

FileSize

```
public ulong FileSize { get; init; }
```

PropertyValue

[ulong](#) ↗

Info about a plot file

Filename

```
public string Filename { get; init; }
```

PropertyValue

[string](#) ↗

Info about a plot file

PlotId

```
public string PlotId { get; init; }
```

PropertyValue

[string](#) ↗

Info about a plot file

PlotPublicKey

```
public string? PlotPublicKey { get; init; }
```

PropertyValue

[string](#) ↗

Info about a plot file

PoolContractPuzzleHash

```
public string PoolContractPuzzleHash { get; init; }
```

Property Value

[string](#)

Info about a plot file

PoolPublicKey

```
public string? PoolPublicKey { get; init; }
```

Property Value

[string](#)

Info about a plot file

Size

```
public KSize Size { get; init; }
```

Property Value

[KSize](#)

Info about a plot file

TimeModified

```
public double TimeModified { get; init; }
```

Property Value

double ↗

Info about a plot file

Class PlotInfoRequestData

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record PlotInforequestData : IEquatable<PlotInforequestData>
```

Inheritance

[object](#) ← PlotInforequestData

Implements

[IEquatable](#) <[PlotInforequestData](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Filter

```
public IEnumerable<FilterItem> Filter { get; init; }
```

Property Value

[IEnumerable](#) <[FilterItem](#)>

NodeId

```
public string NodeId { get; init; }
```

Property Value

[string](#)

Page

```
public int Page { get; init; }
```

Property Value

[int ↗](#)

PageSize

```
public int PageSize { get; init; }
```

Property Value

[int ↗](#)

Reverse

```
public bool Reverse { get; init; }
```

Property Value

[bool ↗](#)

SortKey

```
public string SortKey { get; init; }
```

Property Value

[string ↗](#)

Class PlotPathrequestData

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record PlotPathrequestData : IEquatable<PlotPathrequestData>
```

Inheritance

[object](#) ← PlotPathrequestData

Implements

[IEquatable](#)<[PlotPathrequestData](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Filter

```
public IEnumerable<string> Filter { get; init; }
```

Property Value

[IEnumerable](#)<[string](#)>

NodeId

```
public string NodeId { get; init; }
```

Property Value

[string](#)

Page

```
public int Page { get; init; }
```

Property Value

[int ↗](#)

PageSize

```
public int PageSize { get; init; }
```

Property Value

[int ↗](#)

Reverse

```
public bool Reverse { get; init; }
```

Property Value

[bool ↗](#)

Enum PlotState

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public enum PlotState
```

Fields

FINISHED = 3

REMOVING = 2

RUNNING = 1

SUBMITTED = 0

Class PlotterConfig

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

Configuration settings for the plotter. (equivalent to chia plots create command line args)

<https://github.com/Chia-Network/chia-blockchain/wiki/CLI-Commands-Reference>

```
public record PlotterConfig : IEquatable<PlotterConfig>
```

Inheritance

[object](#) ← PlotterConfig

Implements

[IEquatable](#) <[PlotterConfig](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

AltFingerprint

This is the key Fingerprint used to select both the Farmer and Pool Public Keys to use. Utilize this when you want to select one key out of several in your keychain.

```
[JsonProperty("a")]
public uint? AltFingerprint { get; init; }
```

Property Value

[uint](#)?

Configuration settings for the plotter. (equivalent to chia plots create command line args)

<https://github.com/Chia-Network/chia-blockchain/wiki/CLI-Commands-Reference>

Buckets

More buckets require less RAM but more random seeks to disk. With spinning disks you want less buckets and with NVMe more buckets. There is no significant benefit from using smaller buckets - just use 128.

```
[JsonProperty("u")]
public int Buckets { get; init; }
```

Property Value

[int ↗](#)

Defaults to 128

Buffer

Define memory/RAM usage. Default is 4608 (4.6 GiB). More RAM will marginally increase speed of plot creation. Please bear in mind that this is what is allocated to the plotting algorithm alone. Code, container, libraries etc. will require additional RAM from your system.

```
[JsonProperty("b")]
public int Buffer { get; init; }
```

Property Value

[int ↗](#)

Defaults to 4608

Delay

The number of seconds to delay before beginning the plotting

```
[JsonProperty("delay")]
public int Delay { get; init; }
```

Property Value

[int](#)

Defaults to 0

DestinationDir

Define the final location for plot(s). Of course, -d should have enough free space as the final size of the plot. This directory is automatically added to your ~/.chia/VERSION/config/config.yaml file.

```
[JsonProperty("d")]
public string DestinationDir { get; init; }
```

Property Value

[string](#)

No default - must be set

ExcludeFinalDir

Skips adding [final dir] to harvester for farming.

```
[JsonProperty("x")]
public bool ExcludeFinalDir { get; init; }
```

Property Value

[bool](#)

Defaults to false

FarmerPublicKey

This is your "Farmer Public Key". Utilise this when you want to create plots on other machines for which you do not want to give full chia account access

```
[JsonProperty("f")]

```

```
public string? FarmerPublicKey { get; init; }
```

Property Value

[string](#)

Configuration settings for the plotter. (equivalent to chia plots create command line args)

<https://github.com/Chia-Network/chia-blockchain/wiki/CLI-Commands-Reference>

Memo

Debug purposes only

```
[JsonProperty("memo")]
public string? Memo { get; init; }
```

Property Value

[string](#)

Configuration settings for the plotter. (equivalent to chia plots create command line args)

<https://github.com/Chia-Network/chia-blockchain/wiki/CLI-Commands-Reference>

NoBitField

Setting to true will disable the bitfield plotting algorithm, and revert back to the older b17 plotting style.
After 1.0.4 it's better to use bitfield for most cases

```
[JsonProperty("e")]
public bool NoBitField { get; init; }
```

Property Value

[bool](#)

Defaults to false

NumThreads

The number of threads to devote to each plot

```
[JsonProperty("r")]
public int NumThreads { get; init; }
```

Property Value

[int](#)

Defaults to 2

Number

The number of plots that will be made, in sequence.

```
[JsonProperty("n")]
public int Number { get; init; }
```

Property Value

[int](#)

Defaults to 1

OverrideK

Only needed when [Size](#) is set to [K25](#)

```
[JsonProperty("overrideK")]
public bool? OverrideK { get; init; }
```

Property Value

[bool](#)

Defaults to false.

Parallel

```
[JsonProperty("parallel")]
public bool Parallel { get; init; }
```

Property Value

[bool](#) ↗

Defaults to false

PoolContractAddress

```
[JsonProperty("c")]
public string? PoolContractAddress { get; init; }
```

Property Value

[string](#) ↗

Configuration settings for the plotter. (equivalent to chia plots create command line args)
<https://github.com/Chia-Network/chia-blockchain/wiki/CLI-Commands-Reference>

PoolPublicKey

This is your "Pool Public Key". Utilise this when you want to create plots on other machines for which you do not want to give full chia account access.

```
[JsonProperty("p")]
public string? PoolPublicKey { get; init; }
```

Property Value

[string](#) ↗

Configuration settings for the plotter. (equivalent to chia plots create command line args)
<https://github.com/Chia-Network/chia-blockchain/wiki/CLI-Commands-Reference>

Queue

```
[JsonProperty("queue")]
public string Queue { get; init; }
```

Property Value

[string](#) ↗

Defaults to "default"

Size

Define the size of the plot(s).

```
[JsonProperty("k")]
public KSize Size { get; init; }
```

Property Value

[KSize](#)

Defaults to [K32](#)

TempDir

Define the temporary directory for plot creation. This is where Plotting Phase 1 (Forward Propagation) and Phase 2 (Backpropagation) both occur. The -t dir requires the largest working space: normally about 2.5 times the size of the final plot.

```
[JsonProperty("t")]
public string TempDir { get; init; }
```

Property Value

[string](#) ↗

No default - must be set

TempDir2

Define a secondary temporary directory for plot creation. This is where Plotting Phase 3 (Compression) and Phase 4 (Checkpoints) occur.

```
[JsonProperty("t2")]
public string? TempDir2 { get; init; }
```

Property Value

[string](#) ↗

If not set defaults to [TempDir](#)

Class PlotterInfo

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record PlotterInfo : IEquatable<PlotterInfo>
```

Inheritance

[object](#) ← PlotterInfo

Implements

[IEquatable](#)<[PlotterInfo](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

CanInstall

```
public bool CanInstall { get; init; }
```

Property Value

[bool](#)

DisplayName

```
public string DisplayName { get; init; }
```

Property Value

[string](#)

Installed

```
public bool Installed { get; init; }
```

Property Value

[bool](#) ↗

Version

```
public string? Version { get; init; }
```

Property Value

[string](#) ↗

Class PlotterProxy

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

Class to manage plotting

```
public sealed class PlotterProxy : ServiceProxy
```

Inheritance

[object](#) ← [ServiceProxy](#) ← PlotterProxy

Inherited Members

[ServiceProxy.OriginService](#) , [ServiceProxy.DestinationService](#) , [ServiceProxy.RpcClient](#) ,
[ServiceProxy.HealthZ\(CancellationToken\)](#) , [ServiceProxy.StopNode\(CancellationToken\)](#) ,
[ServiceProxy.GetConnections\(CancellationToken\)](#) , [ServiceProxy.GetRoutes\(CancellationToken\)](#) ,
[ServiceProxy.OpenConnection\(string, int, CancellationToken\)](#) ,
[ServiceProxy.CloseConnection\(string, CancellationToken\)](#) , [object.Equals\(object\)](#) ,
[object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Remarks

ctor

Constructors

PlotterProxy(WebSocketRpcClient, string)

Class to manage plotting

```
public PlotterProxy(WebSocketRpcClient rpcClient, string originService)
```

Parameters

rpcClient [WebSocketRpcClient](#)

[IRpcClient](#) instance to use for rpc communication

`originService` [string](#)

[Origin](#)

Remarks

ctor

Methods

`GetKeysForPlotting(IEnumerable<uint>?, CancellationToken)`

Returns the list of plotting keys.

```
public Task<IDictionary<uint, PlottingKeys>> GetKeysForPlotting(IEnumerable<uint>?
    fingerprints = null, CancellationToken cancellationToken = default)
```

Parameters

`fingerprints` [IEnumerable](#)<[uint](#)>

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[IDictionary](#)<[uint](#), [PlottingKeys](#)>>

A dictionary of fingerprints and [PlottingKeys](#)

`GetPlotters(CancellationToken)`

Get info about installed and installable plotters

```
public Task<IDictionary<string, PlotterInfo>> GetPlotters(CancellationToken
    cancellationToken = default)
```

Parameters

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[IDictionary](#) <[string](#), [PlotterInfo](#)>>

Dictionary of supported plotters

RegisterPlotter(CancellationToken)

Registers this instance as a plotter and retrieves the plot queue

```
public Task<IEnumerable<QueuedPlotInfo>> RegisterPlotter(CancellationToken cancellationToken  
= default)
```

Parameters

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[IEnumerable](#) <[QueuedPlotInfo](#)>>

The list of [QueuedPlotInfos](#)

StartPlotting(PlotterConfig, CancellationToken)

Starts plotting. Returns after plot is added to the plotting queue. Does not wait for plot to finish

```
public Task<IEnumerable<string>> StartPlotting(PlotterConfig config, CancellationToken  
cancellationToken = default)
```

Parameters

`config` [PlotterConfig](#)

The config of the plot. Maps 1:1 to the chia plot create command line

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <IEnumerable <[string](#)>>>

An awaitable [Task](#)

StopPlotting(string, CancellationToken)

Stops the plot with the given id

```
public Task StopPlotting(string id, CancellationToken cancellationToken = default)
```

Parameters

id [string](#)

The id of the plot to stop. Can be found by inspecting the plot queue returned from [RegisterPlotter\(CancellationToken\)](#).

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

An awaitable [Task](#)

Class PlottingKeys

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record PlottingKeys : IEquatable<PlottingKeys>
```

Inheritance

[object](#) ← PlottingKeys

Implements

[IEquatable](#) <[PlottingKeys](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

FarmerPublicKey

```
public string FarmerPublicKey { get; init; }
```

Property Value

[string](#)

PoolPublicKey

```
public string PoolPublicKey { get; init; }
```

Property Value

[string](#)

Class PluginStatus

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record PluginStatus : IEquatable<PluginStatus>
```

Inheritance

[object](#) ← PluginStatus

Implements

[IEquatable](#)<[PluginStatus](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Downloaders

```
public IDictionary<string, IDictionary<string, object>> Downloaders { get; init; }
```

Property Value

[IDictionary](#)<[string](#), [IDictionary](#)<[string](#), [object](#)>>

Uploasders

```
public IDictionary<string, IDictionary<string, object>> Uploasders { get; init; }
```

Property Value

[IDictionary](#)<[string](#), [IDictionary](#)<[string](#), [object](#)>>

Class PoolInfo

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record PoolInfo : IEquatable<PoolInfo>
```

Inheritance

[object](#) ← PoolInfo

Implements

[IEquatable](#) <[PoolInfo](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Fields

POOL_PROTOCOL_VERSION

The current version of the pool protocol

```
public const byte POOL_PROTOCOL_VERSION = 1
```

Field Value

[byte](#)

Properties

AuthenticationTokenTimeout

```
public byte AuthenticationTokenTimeout { get; init; }
```

Property Value

[byte](#) ↗

Description

```
public string Description { get; init; }
```

Property Value

[string](#) ↗

Fee

```
public decimal Fee { get; init; }
```

Property Value

[decimal](#) ↗

LogoUri

```
public Uri LogoUri { get; init; }
```

Property Value

[Uri](#) ↗

MinimumDifficulty

```
public ulong MinimumDifficulty { get; init; }
```

Property Value

[ulong](#) ↗

Name

```
public string Name { get; init; }
```

Property Value

[string](#) ↗

ProtocolVersion

```
public byte ProtocolVersion { get; init; }
```

Property Value

[byte](#) ↗

RelativeLockHeight

```
public uint RelativeLockHeight { get; init; }
```

Property Value

[uint](#) ↗

TargetPuzzleHash

```
public string? TargetPuzzleHash { get; init; }
```

Property Value

[string](#) ↗

Class PoolPoint

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
[JsonConverter(typeof(PoolPointConverter))]  
public record PoolPoint : IEquatable<PoolPoint>
```

Inheritance

[object](#) ← PoolPoint

Implements

[IEquatable](#)<[PoolPoint](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

DateTimeFound

```
[JsonIgnore]  
public DateTime DateTimeFound { get; }
```

Property Value

[DateTime](#)

Difficulty

```
public ulong Difficulty { get; init; }
```

Property Value

[ulong](#)

TimeFound

```
public double TimeFound { get; init; }
```

Property Value

[double](#)

Enum PoolSingletonState

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

From the user's point of view, a pool group can be in these states:

- SELF_POOLING**: The singleton exists on the blockchain, and we are farming block rewards to a wallet address controlled by the user

- LEAVING_POOL**: The singleton exists, and we have entered the "escaping" state, which means we are waiting for a number of blocks = `relative_lock_height` to pass, so we can leave.

- FARMING_TO_POOL**: The singleton exists, and it is assigned to a pool.

- CLAIMING_SELF_POOLED_REWARDS**: We have submitted a transaction to sweep our self-pooled funds.

```
[JsonConverter(typeof(StringEnumConverter))]  
public enum PoolSingletonState
```

Fields

FARMING_TO_POOL = 3

The singleton exists, and it is assigned to a pool.

LEAVING_POOL = 2

The singleton exists, and we have entered the "escaping" state, which means we are waiting for a number of blocks = `relative_lock_height` to pass, so we can leave.

SELF_POOLING = 1

The singleton exists on the blockchain, and we are farming block rewards to a wallet address controlled by the user

Class PoolState

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

PoolState is a type that is serialized to the blockchain to track the state of the user's pool singleton. target_puzzle_hash is either the pool address, or the self-pooling address that pool rewards will be paid to. target_puzzle_hash is NOT the p2_singleton puzzle that block rewards are sent to. The p2_singleton address is the initial address, and the target_puzzle_hash is the final destination. relative_lock_height is zero when in SELF_POOLING state

```
public record PoolState : IEquatable<PoolState>
```

Inheritance

[object](#) ← PoolState

Implements

[IEquatable](#)<[PoolState](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

OwnerPubkey

owner_pubkey is set by the wallet, once

```
public string OwnerPubkey { get; init; }
```

Property Value

[string](#)

PoolState is a type that is serialized to the blockchain to track the state of the user's pool singleton. target_puzzle_hash is either the pool address, or the self-pooling address that pool rewards will be paid to. target_puzzle_hash is NOT the p2_singleton puzzle that block rewards are sent to. The

p2_singleton address is the initial address, and the target_puzzle_hash is the final destination.
relative_lock_height is zero when in SELF_POOLING state

PoolUrl

```
public string? PoolUrl { get; init; }
```

Property Value

[string](#)

PoolState is a type that is serialized to the blockchain to track the state of the user's pool singleton
target_puzzle_hash is either the pool address, or the self-pooling address that pool rewards will be
paid to. target_puzzle_hash is NOT the p2_singleton puzzle that block rewards are sent to. The
p2_singleton address is the initial address, and the target_puzzle_hash is the final destination.
relative_lock_height is zero when in SELF_POOLING state

RelativeLockHeight

```
public uint RelativeLockHeight { get; init; }
```

Property Value

[uint](#)

PoolState is a type that is serialized to the blockchain to track the state of the user's pool singleton
target_puzzle_hash is either the pool address, or the self-pooling address that pool rewards will be
paid to. target_puzzle_hash is NOT the p2_singleton puzzle that block rewards are sent to. The
p2_singleton address is the initial address, and the target_puzzle_hash is the final destination.
relative_lock_height is zero when in SELF_POOLING state

State

PoolSingletonState

```
public PoolSingletonState State { get; init; }
```

Property Value

PoolSingletonState

PoolState is a type that is serialized to the blockchain to track the state of the user's pool singleton target_puzzle_hash is either the pool address, or the self-pooling address that pool rewards will be paid to. target_puzzle_hash is NOT the p2_singleton puzzle that block rewards are sent to. The p2_singleton address is the initial address, and the target_puzzle_hash is the final destination. relative_lock_height is zero when in SELF_POOLING state

TargetPuzzleHash

A puzzle_hash we pay to When self-farming, this is a main wallet address When farming-to-pool, the pool sends this to the farmer during pool protocol setup

```
public string TargetPuzzleHash { get; init; }
```

Property Value

string ↗

PoolState is a type that is serialized to the blockchain to track the state of the user's pool singleton target_puzzle_hash is either the pool address, or the self-pooling address that pool rewards will be paid to. target_puzzle_hash is NOT the p2_singleton puzzle that block rewards are sent to. The p2_singleton address is the initial address, and the target_puzzle_hash is the final destination. relative_lock_height is zero when in SELF_POOLING state

Version

```
public byte Version { get; init; }
```

Property Value

byte ↗

PoolState is a type that is serialized to the blockchain to track the state of the user's pool singleton target_puzzle_hash is either the pool address, or the self-pooling address that pool rewards will be paid to. target_puzzle_hash is NOT the p2_singleton puzzle that block rewards are sent to. The

p2_singleton address is the initial address, and the target_puzzle_hash is the final destination.
relative_lock_height is zero when in SELF_POOLING state

Class PoolStateInfo

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

This type does not exist in the chia python, but is returned as a dicitonary for the UI to show pool state.
Not to be confused with [PoolState](#)

```
public record PoolStateInfo : IEquatable<PoolStateInfo>
```

Inheritance

[object](#) ← PoolStateInfo

Implements

[IEquatable](#)<[PoolStateInfo](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

AuthenticationTokenTimeout

```
public byte? AuthenticationTokenTimeout { get; init; }
```

Property Value

[byte](#)?

This type does not exist in the chia python, but is returned as a dicitonary for the UI to show pool state. Not to be confused with

CurrentDifficulty

```
public ulong? CurrentDifficulty { get; init; }
```

Property Value

[ulong](#)?

This type does not exist in the chia python, but is returned as a dicitonary for the UI to show pool state. Not to be confused with

CurrentPoints

```
public ulong CurrentPoints { get; init; }
```

Property Value

[ulong](#)

This type does not exist in the chia python, but is returned as a dicitonary for the UI to show pool state. Not to be confused with

NextFarmerUpdate

```
public double NextFarmerUpdate { get; init; }
```

Property Value

[double](#)

This type does not exist in the chia python, but is returned as a dicitonary for the UI to show pool state. Not to be confused with

NextFarmerUpdateDateTime

```
[JsonIgnore]  
public DateTime NextFarmerUpdateDateTime { get; }
```

Property Value

[DateTime](#)

This type does not exist in the chia python, but is returned as a dicitonary for the UI to show pool state. Not to be confused with

NextPoolInfoUpdate

```
public double NextPoolInfoUpdate { get; init; }
```

Property Value

[double](#)

This type does not exist in the chia python, but is returned as a dicitonary for the UI to show pool state. Not to be confused with

NextPoolInfoUpdateDateTime

```
[JsonIgnore]  
public DateTime NextPoolInfoUpdateDateTime { get; }
```

Property Value

[DateTime](#)

This type does not exist in the chia python, but is returned as a dicitonary for the UI to show pool state. Not to be confused with

PlotCount

```
public int PlotCount { get; init; }
```

Property Value

[int](#)

This type does not exist in the chia python, but is returned as a dicitonary for the UI to show pool state. Not to be confused with

PointsAcknowledged24h

```
[JsonProperty("points_acknowledged_24h")]
public IEnumerable<PoolPoint> PointsAcknowledged24h { get; init; }
```

Property Value

[IEnumerable](#) <[PoolPoint](#)>

This type does not exist in the chia python, but is returned as a dicitonary for the UI to show pool state. Not to be confused with

PointsAcknowledgedSinceStart

```
public ulong PointsAcknowledgedSinceStart { get; init; }
```

Property Value

[ulong](#)

This type does not exist in the chia python, but is returned as a dicitonary for the UI to show pool state. Not to be confused with

PointsFound24h

```
[JsonProperty("points_found_24h")]
public IEnumerable<PoolPoint> PointsFound24h { get; init; }
```

Property Value

[IEnumerable](#) <[PoolPoint](#)>

This type does not exist in the chia python, but is returned as a dicitonary for the UI to show pool state. Not to be confused with

PointsFoundSinceStart

```
public ulong PointsFoundSinceStart { get; init; }
```

Property Value

[ulong](#) ↗

This type does not exist in the chia python, but is returned as a dicitonary for the UI to show pool state. Not to be confused with

PoolConfig

```
public PoolWalletConfig PoolConfig { get; init; }
```

Property Value

[PoolWalletConfig](#)

This type does not exist in the chia python, but is returned as a dicitonary for the UI to show pool state. Not to be confused with

PoolErrors24h

```
public IEnumerable<ErrorResponse> PoolErrors24h { get; init; }
```

Property Value

[IEnumerable](#) ↗ <[ErrorResponse](#)>

This type does not exist in the chia python, but is returned as a dicitonary for the UI to show pool state. Not to be confused with

Class PoolTarget

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record PoolTarget : IEquatable<PoolTarget>
```

Inheritance

[object](#) ← PoolTarget

Implements

[IEquatable](#) <[PoolTarget](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

MaxHeight

A max height of 0 means it is valid forever

```
public uint MaxHeight { get; init; }
```

Property Value

[uint](#)

PuzzleHash

```
public string PuzzleHash { get; init; }
```

Property Value

[string](#)

Class PoolWallet

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

Wraps a Pool Wallet

```
public sealed class PoolWallet : Wallet
```

Inheritance

[object](#) ← [Wallet](#) ← PoolWallet

Inherited Members

[Wallet.WalletId](#) , [Wallet.WalletProxy](#) , [Wallet.GetWalletInfo\(CancellationToken\)](#) ,
[Wallet.GetBalance\(CancellationToken\)](#) ,
[Wallet.SelectCoins\(ulong, IEnumerable<Coin>, IEnumerable<ulong>, ulong?, ulong?, CancellationToken\)](#) ,
'
[Wallet.GetTransactions\(string, TransactionTypeFilter, bool, string, uint, uint, bool?, CancellationToken\)](#) ,
[Wallet.GetSpendableCoins\(ulong?, ulong?, IEnumerable<ulong>, IEnumerable<Coin>,
IEnumerable<string>, CancellationToken\)](#) ,
[Wallet.GetNextAddress\(bool, CancellationToken\)](#) ,
[Wallet.GetTransactionCount\(TransactionTypeFilter, CancellationToken\)](#) ,
[Wallet.DeleteUnconfirmedTransactions\(CancellationToken\)](#) ,
[Wallet.SendTransaction\(string, ulong, IEnumerable<string>, IEnumerable<ulong>, IEnumerable<string>,
ulong?, ulong?, bool, ulong, CancellationToken\)](#) ,
[Wallet.SendTransactionMulti\(IEnumerable<Coin>, IEnumerable<Coin>, ulong, CancellationToken\)](#) ,
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Remarks

ctor

Constructors

PoolWallet(uint, WalletProxy)

Wraps a Pool Wallet

```
public PoolWallet(uint walletId, WalletProxy walletProxy)
```

Parameters

walletId [uint](#)

The wallet_id to wrap

walletProxy [WalletProxy](#)

Wallet RPC proxy to use for communication

Remarks

ctor

Methods

AbsorbRewards(ulong, CancellationToken)

Perform a sweep of the p2_singleton rewards controlled by the pool wallet singleton

```
public Task<(PoolWalletInfo State, TransactionRecord Transaction)> AbsorbRewards(ulong fee = 0, CancellationToken cancellationToken = default)
```

Parameters

fee [ulong](#)

Transaction fee (in units of mojos)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<(PoolWalletInfo State, TransactionRecord Transaction)>

Wallet state and transaction

JoinPool(string, string, uint, CancellationToken)

Join the wallet to a pool

```
public Task<TransactionRecord> JoinPool(string targetPuzzlehash, string poolUrl, uint relativeLockHeight, CancellationToken cancellationToken = default)
```

Parameters

targetPuzzlehash [string](#)

Puzzle hash

poolUrl [string](#)

Url of the pool to join

relativeLockHeight [uint](#)

Relative lock height

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[TransactionRecord](#)>

The resulting [TransactionRecord](#)

Remarks

See [GetPoolInfo\(Uri, CancellationToken\)](#)

SelfPool(CancellationToken)

Leaving a pool requires two state transitions. First we transition to PoolSingletonState.LEAVING_POOL
Then we transition to FARMING_TO_POOL or SELF_POOLING

```
public Task<TransactionRecord> SelfPool(CancellationToken cancellationToken = default)
```

Parameters

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[TransactionRecord](#)>

The resulting [TransactionRecord](#)

Status(CancellationToken)

Perform a sweep of the p2_singleton rewards controlled by the pool wallet singleton

```
public Task<(PoolWalletInfo State, IEnumerable<TransactionRecord> UnconfirmedTransactions)>
Status(CancellationToken cancellationToken = default)
```

Parameters

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<([PoolWalletInfo](#) [State](#), [IEnumerable](#)<[TransactionRecord](#)> [UnconfirmedTransactions](#))>

Wallet state and list of unconfirmed transactions

Validate(CancellationToken)

Validates that [WalletId](#) is a [POOLING WALLET](#)

```
public override Task Validate(CancellationToken cancellationToken = default)
```

Parameters

`cancellationToken` [CancellationToken](#)

Wraps a Pool Wallet

Returns

[Task](#)

True if the wallet is a pooling wallet

Class PoolWalletConfig

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

This is what goes into the user's config file, to communicate between the wallet and the farmer processes.

```
public record PoolWalletConfig : IEquatable<PoolWalletConfig>
```

Inheritance

[object](#) ← PoolWalletConfig

Implements

[IEquatable](#)<[PoolWalletConfig](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

AuthenticationPublicKey

```
public string AuthenticationPublicKey { get; init; }
```

Property Value

[string](#)

This is what goes into the user's config file, to communicate between the wallet and the farmer processes.

LauncherId

```
public string LauncherId { get; init; }
```

PropertyValue

[string ↗](#)

This is what goes into the user's config file, to communicate between the wallet and the farmer processes.

OwnerPublicKey

```
public string OwnerPublicKey { get; init; }
```

PropertyValue

[string ↗](#)

This is what goes into the user's config file, to communicate between the wallet and the farmer processes.

P2SingletonPuzzleHash

```
public string P2SingletonPuzzleHash { get; init; }
```

PropertyValue

[string ↗](#)

This is what goes into the user's config file, to communicate between the wallet and the farmer processes.

PayoutInstructions

```
public string PayoutInstructions { get; init; }
```

PropertyValue

[string ↗](#)

This is what goes into the user's config file, to communicate between the wallet and the farmer processes.

PoolUrl

```
public string PoolUrl { get; init; }
```

Property Value

[string ↗](#)

This is what goes into the user's config file, to communicate between the wallet and the farmer processes.

TargetPuzzleHash

```
public string TargetPuzzleHash { get; init; }
```

Property Value

[string ↗](#)

This is what goes into the user's config file, to communicate between the wallet and the farmer processes.

Class PoolWalletInfo

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

Internal Pool Wallet state, not destined for the blockchain. This can be completely derived with the Singleton's CoinSpends list, or with the information from the WalletPoolStore.

```
public record PoolWalletInfo : IEquatable<PoolWalletInfo>
```

Inheritance

[object](#) ← PoolWalletInfo

Implements

[IEquatable](#) <[PoolWalletInfo](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Current

```
public PoolState Current { get; init; }
```

Property Value

[PoolState](#)

Internal Pool Wallet state, not destined for the blockchain. This can be completely derived with the Singleton's CoinSpends list, or with the information from the WalletPoolStore.

CurrentInner

Inner puzzle in current singleton, not revealed yet

```
public string CurrentInner { get; init; }
```

Property Value

[string](#) ↗

Internal Pool Wallet state, not destined for the blockchain. This can be completely derived with the Singleton's CoinSpends list, or with the information from the WalletPoolStore.

LauncherCoin

```
public Coin LauncherCoin { get; init; }
```

Property Value

[Coin](#)

Internal Pool Wallet state, not destined for the blockchain. This can be completely derived with the Singleton's CoinSpends list, or with the information from the WalletPoolStore.

LauncherId

```
public string LauncherId { get; init; }
```

Property Value

[string](#) ↗

Internal Pool Wallet state, not destined for the blockchain. This can be completely derived with the Singleton's CoinSpends list, or with the information from the WalletPoolStore.

P2SingletonPuzzleHash

```
public string P2SingletonPuzzleHash { get; init; }
```

Property Value

[string](#) ↗

Internal Pool Wallet state, not destined for the blockchain. This can be completely derived with the Singleton's CoinSpends list, or with the information from the WalletPoolStore.

SingletonBlockHeight

Block height that current PoolState is from

```
public uint SingletonBlockHeight { get; init; }
```

Property Value

[uint](#) ↗

Internal Pool Wallet state, not destined for the blockchain. This can be completely derived with the Singleton's CoinSpends list, or with the information from the WalletPoolStore.

Target

```
public PoolState? Target { get; init; }
```

Property Value

[PoolState](#)

Internal Pool Wallet state, not destined for the blockchain. This can be completely derived with the Singleton's CoinSpends list, or with the information from the WalletPoolStore.

TipSingletonCoinId

```
public string TipSingletonCoinId { get; init; }
```

Property Value

string ↴

Internal Pool Wallet state, not destined for the blockchain. This can be completely derived with the Singleton's CoinSpends list, or with the information from the WalletPoolStore.

Class PrivateKey

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record PrivateKey : IEquatable<PrivateKey>
```

Inheritance

[object](#) ← PrivateKey

Implements

[IEquatable](#)<[PrivateKey](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

FarmerPk

```
public string FarmerPk { get; init; }
```

Property Value

[string](#)

Fingerprint

```
public uint Fingerprint { get; init; }
```

Property Value

[uint](#)

Pk

```
public string Pk { get; init; }
```

Property Value

[string](#) ↗

PoolPk

```
public string PoolPk { get; init; }
```

Property Value

[string](#) ↗

Seed

```
public string Seed { get; init; }
```

Property Value

[string](#) ↗

Sk

```
public string Sk { get; init; }
```

Property Value

[string](#) ↗

Class PrivateKeyData

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record PrivateKeyData : IEquatable<PrivateKeyData>
```

Inheritance

[object](#) ← PrivateKeyData

Implements

[IEquatable](#) <[PrivateKeyData](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Entropy

```
public string Entropy { get; init; }
```

Property Value

[string](#)

PK

```
public string PK { get; init; }
```

Property Value

[string](#)

Class Proof

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record Proof : IEquatable<Proof>
```

Inheritance

[object](#) ← Proof

Implements

[IEquatable](#)<Proof>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Key

```
public string Key { get; init; }
```

Property Value

[string](#)

Layers

```
public IEnumerable<Layer> Layers { get; init; }
```

Property Value

[IEnumerable](#)<Layer>

NodeHash

```
public string NodeHash { get; init; }
```

Property Value

[string](#) ↗

Value

```
public string Value { get; init; }
```

Property Value

[string](#) ↗

Class ProofOfSpace

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record ProofOfSpace : IEquatable<ProofOfSpace>
```

Inheritance

[object](#) ← ProofOfSpace

Implements

[IEquatable](#) <[ProofOfSpace](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Challenge

```
public string Challenge { get; init; }
```

Property Value

[string](#)

PlotPublicKey

```
public string PlotPublicKey { get; init; }
```

Property Value

[string](#)

PoolContractPuzzleHash

```
public string? PoolContractPuzzleHash { get; init; }
```

Property Value

[string](#) ↗

Proof

```
public string Proof { get; init; }
```

Property Value

[string](#) ↗

PublicPoolKey

Only one of these two should be present

```
public string? PublicPoolKey { get; init; }
```

Property Value

[string](#) ↗

Size

```
public KSize Size { get; init; }
```

Property Value

[KSize](#)

Class ProposallInfo

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record ProposallInfo : IEquatable<ProposallInfo>
```

Inheritance

[object](#) ← ProposallInfo

Implements

[IEquatable](#) <[ProposallInfo](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

AmountVoted

```
public ulong AmountVoted { get; init; }
```

Property Value

[ulong](#)

Closed

```
public bool? Closed { get; init; }
```

Property Value

[bool](#)?

CurrentCoin

```
public Coin CurrentCoin { get; init; }
```

Property Value

[Coin](#)

CurrentInnerpuz

```
public string? CurrentInnerpuz { get; init; }
```

Property Value

[string](#) ↗

InnerPuzzle

```
public string InnerPuzzle { get; init; }
```

Property Value

[string](#) ↗

Passed

```
public bool? Passed { get; init; }
```

Property Value

[bool](#) ↗?

ProposalId

this is launcher_id

```
public string ProposalId { get; init; }
```

Property Value

[string](#)

SingletonBlockHeight

Block height that current proposal singleton coin was created in

```
public uint SingletonBlockHeight { get; init; }
```

Property Value

[uint](#)

TimeCoin

if this is null then the proposal has finished

```
public Coin? TimeCoin { get; init; }
```

Property Value

[Coin](#)

YesVotes

```
public ulong YesVotes { get; init; }
```

Property Value

[ulong](#)

Class ProposalState

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record ProposalState : IEquatable<ProposalState>
```

Inheritance

[object](#) ← ProposalState

Implements

[IEquatable](#) <[ProposalState](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

BlocksNeeded

```
public int BlocksNeeded { get; init; }
```

Property Value

[int](#)

Closeable

```
public bool Closeable { get; init; }
```

Property Value

[bool](#)

Closed

```
public bool? Closed { get; init; }
```

Property Value

bool?

Passed

```
public bool Passed { get; init; }
```

Property Value

bool

TotalVotesNeeded

```
public int TotalVotesNeeded { get; init; }
```

Property Value

int

YesVotesNeeded

```
public int YesVotesNeeded { get; init; }
```

Property Value

int

Class PuzzleAnnouncement

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record PuzzleAnnouncement : IEquatable<PuzzleAnnouncement>
```

Inheritance

[object](#) ← PuzzleAnnouncement

Implements

[IEquatable](#)<[PuzzleAnnouncement](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Message

```
public string Message { get; init; }
```

Property Value

[string](#)

MorphBytes

```
public string? MorphBytes { get; init; }
```

Property Value

[string](#)

PuzzleHash

```
public string PuzzleHash { get; init; }
```

Property Value

[string](#) ↗

Class QueuedPlotInfo

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

An entry on the plotter queue

```
public record QueuedPlotInfo : IEquatable<QueuedPlotInfo>
```

Inheritance

[object](#) ← QueuedPlotInfo

Implements

[IEquatable](#)<[QueuedPlotInfo](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Delay

```
public int Delay { get; init; }
```

Property Value

[int](#)

An entry on the plotter queue

Deleted

```
public bool Deleted { get; init; }
```

Property Value

bool ↗

An entry on the plotter queue

Error

```
public string Error { get; init; }
```

Property Value

string ↗

An entry on the plotter queue

Id

```
public string Id { get; init; }
```

Property Value

string ↗

An entry on the plotter queue

Log

```
public string Log { get; init; }
```

Property Value

string ↗

An entry on the plotter queue

LogNew

```
public string LogNew { get; init; }
```

Property Value

[string](#) ↗

An entry on the plotter queue

Parallel

```
public bool Parallel { get; init; }
```

Property Value

[bool](#) ↗

An entry on the plotter queue

PlotState

```
[JsonIgnore]  
public PlotState PlotState { get; }
```

Property Value

[PlotState](#)

An entry on the plotter queue

Queue

```
public string Queue { get; init; }
```

Property Value

[string](#) ↗

An entry on the plotter queue

Size

```
public KSize Size { get; init; }
```

Property Value

[KSize](#)

An entry on the plotter queue

State

```
public string State { get; init; }
```

Property Value

[string](#) ↗

An entry on the plotter queue

Class ResponseException

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

Exception thrown when the RPC endpoint returns a response [Message](#) but Data.success is false or there is a communication error on the websocket or http channel

```
public sealed class ResponseException : Exception, ISerializable
```

Inheritance

[object](#) ← [Exception](#) ← ResponseException

Implements

[ISerializable](#)

Inherited Members

[Exception.GetBaseException\(\)](#) , [Exception.GetType\(\)](#) , [Exception.ToString\(\)](#) , [Exception.Data](#) ,
[Exception.HelpLink](#) , [Exception.HResult](#) , [Exception.InnerException](#) , [Exception.Message](#) ,
[Exception.Source](#) , [Exception.StackTrace](#) , [Exception.TargetSite](#) , [object.Equals\(object\)](#) ,
[object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

Remarks

ctor

Constructors

ResponseException(Message)

ctor

```
public ResponseException(Message request)
```

Parameters

request [Message](#)

The request sent to the service

ResponseException(Message, string)

ctor

```
public ResponseException(Message request, string message)
```

Parameters

request [Message](#)

The request sent to the service

message [string](#)

[Message](#)

ResponseException(Message, string, Exception?)

Exception thrown when the RPC endpoint returns a response [Message](#) but Data.success is false or there is a communication error on the websocket or http channel

```
public ResponseException(Message request, string message, Exception? innerException)
```

Parameters

request [Message](#)

The request sent to the service

message [string](#)

[Message](#)

innerException [Exception](#)

[InnerException](#)

Remarks

ctor

Properties

Request

The request sent to the service

```
public Message Request { get; init; }
```

Property Value

[Message](#)

Exception thrown when the RPC endpoint returns a response but Data.success is false or there is a communication error on the websocket or http channel

Class RewardChainBlock

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record RewardChainBlock : IEquatable<RewardChainBlock>
```

Inheritance

[object](#) ← RewardChainBlock

Implements

[IEquatable](#)<[RewardChainBlock](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

ChallengeChainIpVdf

```
public VDFInfo ChallengeChainIpVdf { get; init; }
```

Property Value

[VDFInfo](#)

ChallengeChainSpSignature

```
public string ChallengeChainSpSignature { get; init; }
```

Property Value

[string](#)

ChallengeChainSpVdf

Not present for first sp in slot

```
public VDFInfo? ChallengeChainSpVdf { get; init; }
```

Property Value

[VDFInfo](#)

Height

```
public uint Height { get; init; }
```

Property Value

[uint](#)

InfusedChallengeChainIpVdf

Iff deficit < 16

```
public VDFInfo? InfusedChallengeChainIpVdf { get; init; }
```

Property Value

[VDFInfo](#)

IsTransactionBlock

```
public bool IsTransactionBlock { get; init; }
```

Property Value

[bool](#)

PosSsCcChallengeHash

```
public string PosSsCcChallengeHash { get; init; }
```

Property Value

[string](#)

ProofOfSpace

```
public ProofOfSpace ProofOfSpace { get; init; }
```

Property Value

[ProofOfSpace](#)

RewardChainIpVdf

```
public VDFInfo RewardChainIpVdf { get; init; }
```

Property Value

[VDFInfo](#)

RewardChainSpSignature

```
public string RewardChainSpSignature { get; init; }
```

Property Value

[string](#)

RewardChainSpVdf

Not present for first sp in slot

```
public VDFInfo? RewardChainSpVdf { get; init; }
```

Property Value

[VDFInfo](#)

SignagePointIndex

```
public byte SignagePointIndex { get; init; }
```

Property Value

[byte](#)

TotalIters

```
public BigInteger TotalIters { get; init; }
```

Property Value

[BigInteger](#)

Weight

```
public BigInteger Weight { get; init; }
```

Property Value

[BigInteger](#)

Class RewardChainBlockUnfinished

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record RewardChainBlockUnfinished : IEquatable<RewardChainBlockUnfinished>
```

Inheritance

[object](#) ← RewardChainBlockUnfinished

Implements

[IEquatable](#)<[RewardChainBlockUnfinished](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

ChallengeChainSpSignature

```
public string ChallengeChainSpSignature { get; init; }
```

Property Value

[string](#)

ChallengeChainSpVdf

Not present for first sp in slot

```
public VDFInfo? ChallengeChainSpVdf { get; init; }
```

Property Value

[VDFInfo](#)

PosSsCcChallengeHash

```
public string PosSsCcChallengeHash { get; init; }
```

Property Value

[string](#) ↗

ProofOfSpace

```
public ProofOfSpace ProofOfSpace { get; init; }
```

Property Value

[ProofOfSpace](#)

RewardChainSpSignature

```
public string RewardChainSpSignature { get; init; }
```

Property Value

[string](#) ↗

RewardChainSpVdf

Not present for first sp in slot

```
public VDFInfo? RewardChainSpVdf { get; init; }
```

Property Value

[VDFInfo](#)

SignagePointIndex

```
public byte SignagePointIndex { get; init; }
```

Property Value

[byte](#) ↗

TotalIter

```
public BigInteger TotalIter { get; init; }
```

Property Value

[BigInteger](#) ↗

Class RewardChainSubSlot

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record RewardChainSubSlot : IEquatable<RewardChainSubSlot>
```

Inheritance

[object](#) ← RewardChainSubSlot

Implements

[IEquatable](#) <[RewardChainSubSlot](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

ChallengeChainSubSlotHash

```
public string ChallengeChainSubSlotHash { get; init; }
```

Property Value

[string](#)

Deficit

16 or less. usually zero

```
public byte Deficit { get; init; }
```

Property Value

[byte](#)

EndOfSlotVdf

```
public VDFInfo EndOfSlotVdf { get; init; }
```

Property Value

[VDFInfo](#)

InfusedChallengeChainSubSlotHash

```
public string? InfusedChallengeChainSubSlotHash { get; init; }
```

Property Value

[string](#) ↗

Class Root

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record Root : IEquatable<Root>
```

Inheritance

[object](#) ← Root

Implements

[IEquatable](#)<Root>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Generation

```
public uint Generation { get; init; }
```

Property Value

[uint](#)

NodeHash

```
public string? NodeHash { get; init; }
```

Property Value

[string](#)

Status

```
public Status Status { get; init; }
```

Property Value

[Status](#)

TreeId

```
public string TreeId { get; init; }
```

Property Value

[string](#) ↗

Class RootHash

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record RootHash : IEquatable<RootHash>
```

Inheritance

[object](#) ← RootHash

Implements

[IEquatable](#) <[RootHash](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Confirmed

```
public bool Confirmed { get; init; }
```

Property Value

[bool](#)

DateTimestamp

```
[JsonIgnore]  
public DateTime DateTimestamp { get; }
```

Property Value

[DateTime](#)

Hash

```
public string Hash { get; init; }
```

Property Value

[string](#) ↗

Id

```
public string? Id { get; init; }
```

Property Value

[string](#) ↗

Timestamp

```
public ulong Timestamp { get; init; }
```

Property Value

[ulong](#) ↗

Class RootHistory

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record RootHistory : IEquatable<RootHistory>
```

Inheritance

[object](#) ← RootHistory

Implements

[IEquatable](#) <[RootHistory](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Confirmed

```
public bool Confirmed { get; init; }
```

Property Value

[bool](#)

DateTimestamp

```
[JsonIgnore]  
public DateTime DateTimestamp { get; }
```

Property Value

[DateTime](#)

RootHash

```
public string RootHash { get; init; }
```

Property Value

[string](#) ↗

Timestamp

```
public ulong Timestamp { get; init; }
```

Property Value

[ulong](#) ↗

Class RoyaltyAsset

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record RoyaltyAsset : IEquatable<RoyaltyAsset>
```

Inheritance

[object](#) ← RoyaltyAsset

Implements

[IEquatable](#)<[RoyaltyAsset](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Asset

```
public string Asset { get; init; }
```

Property Value

[string](#)

RoyaltyAddress

```
public string RoyaltyAddress { get; init; }
```

Property Value

[string](#)

RoyaltyPercentage

```
public ushort RoyaltyPercentage { get; init; }
```

Property Value

[ushort](#)

Class SendPeer

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

Represents the list of peers that we sent the transaction to, whether each one included it in the mempool, and what the error message (if any) was

```
[JsonConverter(typeof(SendPeerConverter))]  
public record SendPeer : IEquatable<SendPeer>
```

Inheritance

[object](#) ← SendPeer

Implements

[IEquatable](#) <[SendPeer](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Remarks

Represented as [List\[Tuple\[str, uint8, Optional\[str\]\]\]](#) in python

Properties

ErrorMessage

```
public string? ErrorMessage { get; init; }
```

Property Value

[string](#)

Represents the list of peers that we sent the transaction to, whether each one included it in the mempool, and what the error message (if any) was

MempoolInclusionStatus

```
public MempoolInclusionStatus MempoolInclusionStatus { get; init; }
```

Property Value

[MempoolInclusionStatus](#)

Represents the list of peers that we sent the transaction to, whether each one included it in the mempool, and what the error message (if any) was

Peer

```
public string Peer { get; init; }
```

Property Value

[string](#) ↗

Represents the list of peers that we sent the transaction to, whether each one included it in the mempool, and what the error message (if any) was

Struct ServiceNames

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

The names of chia services. These are used as [Destination](#) values

```
public struct ServiceNames
```

Inherited Members

[ValueType.Equals\(object\)](#) , [ValueType.GetHashCode\(\)](#) , [ValueType.ToString\(\)](#) ,
[object.Equals\(object, object\)](#) , [object.GetType\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

Fields

Crawler

```
public const string Crawler = "chia_crawler"
```

Field Value

[string](#)

The names of chia services. These are used as values

Daemon

```
public const string Daemon = "daemon"
```

Field Value

[string](#)

The names of chia services. These are used as values

DataLayer

```
public const string DataLayer = "chia_data_layer"
```

Field Value

[string](#) ↗

The names of chia services. These are used as values

Farmer

```
public const string Farmer = "chia_farmer"
```

Field Value

[string](#) ↗

The names of chia services. These are used as values

FullNode

```
public const string FullNode = "chia_full_node"
```

Field Value

[string](#) ↗

The names of chia services. These are used as values

Harvester

```
public const string Harvester = "chia_harvester"
```

Field Value

[string](#)

The names of chia services. These are used as values

Plotter

```
public const string Plotter = "chia_plotter"
```

Field Value

[string](#)

The names of chia services. These are used as values

Simulator

```
public const string Simulator = "chia_full_node_simulator"
```

Field Value

[string](#)

The names of chia services. These are used as values

Wallet

```
public const string Wallet = "chia_wallet"
```

Field Value

[string](#)

The names of chia services. These are used as values

Class ServiceProxy

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

Base class that uses an [IRpcClient](#) to send and receive messages to services

```
public abstract class ServiceProxy
```

Inheritance

[object](#) ← ServiceProxy

Derived

[CrawlerProxy](#), [DaemonProxy](#), [DataLayerProxy](#), [FarmerProxy](#), [FullNodeProxy](#), [HarvesterProxy](#), [PlotterProxy](#),
[WalletProxy](#)

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Remarks

The lifetime of the RpcClient is not controlled by the proxy. It should be disposed outside of this class.

Constructors

ServiceProxy(IRpcClient, string, string)

ctor

```
protected ServiceProxy(IRpcClient rpcClient, string destinationService,  
string originService)
```

Parameters

rpcClient [IRpcClient](#)

[IRpcClient](#) instance to use for rpc communication

`destinationService` [string](#)

[Destination](#)

`originService` [string](#)

[Origin](#)

Properties

DestinationService

[Destination](#)

```
public string DestinationService { get; init; }
```

Property Value

[string](#)

Base class that uses an to send and receive messages to services

OriginService

The name of the service that is running. Will be used as the [Origin](#) of all messages as well as the identifier used for [RegisterService\(string, CancellationToken\)](#).

```
public string OriginService { get; init; }
```

Property Value

[string](#)

Base class that uses an to send and receive messages to services

RpcClient

The [IRpcClient](#) used for underlying RPC

```
public IRpcClient RpcClient { get; init; }
```

Property Value

[IRpcClient](#)

Base class that uses an to send and receive messages to services

Methods

CloseConnection(string, CancellationToken)

Closes a connection

```
public Task CloseConnection(string nodeId, CancellationToken cancellationToken = default)
```

Parameters

`nodeId` [string](#)

The id of the connection to close

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

An awaitable [Task](#)

GetConnections(CancellationToken)

Get the service's connections

```
public Task<IEnumerable<ConnectionInfo>> GetConnections(CancellationToken cancellationToken = default)
```

Parameters

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[IEnumerable](#) <[ConnectionInfo](#)>>

A list of [ConnectionInfo](#)s

GetRoutes(CancellationToken)

Get all endpoints of a service

```
public Task<IEnumerable<string>> GetRoutes(CancellationToken cancellationToken = default)
```

Parameters

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[IEnumerable](#) <[string](#)>>

A list of service routes

HealthZ(CancellationToken)

Sends heartbeat message to the service

```
public Task HealthZ(CancellationToken cancellationToken = default)
```

Parameters

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

Awaitable [Task](#)

Remarks

Either completes without error or throws an exception.

OpenConnection(string, int, CancellationToken)

Add a connection

```
public Task OpenConnection(string host, int port, CancellationToken cancellationToken  
= default)
```

Parameters

host [string](#)

The host name of the connection

port [int](#)

The port to use

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

An awaitable [Task](#)

StopNode(CancellationToken)

Stops the service

```
public Task StopNode(CancellationToken cancellationToken = default)
```

Parameters

cancellationToken [CancellationToken ↗](#)

A token to allow the call to be cancelled

Returns

[Task ↗](#)

Awaitable [Task ↗](#)

Enum Side

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public enum Side : byte
```

Fields

Left = 0

Right = 1

Class SignagePoint

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record SignagePoint : IEquatable<SignagePoint>
```

Inheritance

[object](#) ← SignagePoint

Implements

[IEquatable](#)<[SignagePoint](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

CcProof

```
public VDFProof? CcProof { get; init; }
```

Property Value

[VDFProof](#)

CcVdf

```
public VDFInfo? CcVdf { get; init; }
```

Property Value

[VDFInfo](#)

RcProof

```
public VDFProof? RcProof { get; init; }
```

Property Value

[VDFProof](#)

RcVdf

```
public VDFInfo? RcVdf { get; init; }
```

Property Value

[VDFInfo](#)

Class SingletonInfo

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record SingletonInfo : IEquatable<SingletonInfo>
```

Inheritance

[object](#) ← SingletonInfo

Implements

[IEquatable](#) <[SingletonInfo](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Launcher

```
public string Launcher { get; init; }
```

Property Value

[string](#)

Root

```
public string Root { get; init; }
```

Property Value

[string](#)

Class SingletonRecord

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record SingletonRecord : IEquatable<SingletonRecord>
```

Inheritance

[object](#) ← SingletonRecord

Implements

[IEquatable](#) <[SingletonRecord](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

CoinId

```
public string CoinId { get; init; }
```

Property Value

[string](#)

Confirmed

```
public bool Confirmed { get; init; }
```

Property Value

[bool](#)

ConfirmedAtHeight

```
public uint ConfirmedAtHeight { get; init; }
```

Property Value

[uint](#) ↗

DateTimestamp

```
[JsonIgnore]  
public DateTime DateTimestamp { get; }
```

Property Value

[DateTime](#) ↗

Generation

```
public uint Generation { get; init; }
```

Property Value

[uint](#) ↗

InnerPuzzleHash

```
public string InnerPuzzleHash { get; init; }
```

Property Value

[string](#) ↗

LauncherId

```
public string LauncherId { get; init; }
```

Property Value

[string](#) ↗

LineageProof

```
public LineageProof LineageProof { get; init; }
```

Property Value

[LineageProof](#)

Root

```
public string Root { get; init; }
```

Property Value

[string](#) ↗

Timestamp

```
public ulong Timestamp { get; init; }
```

Property Value

[ulong](#) ↗

Class SpendBundle

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

This is a list of coins being spent along with their solution programs, and a single aggregated signature. This is the object that most closely corresponds to a bitcoin transaction (although because of non-interactive signature aggregation, the boundaries between transactions are more flexible than in bitcoin).

```
public record SpendBundle : IEquatable<SpendBundle>
```

Inheritance

[object](#) ← SpendBundle

Implements

[IEquatable](#) <[SpendBundle](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

AggregatedSignature

```
public string AggregatedSignature { get; init; }
```

Property Value

[string](#)

This is a list of coins being spent along with their solution programs, and a single aggregated signature. This is the object that most closely corresponds to a bitcoin transaction (although because of non-interactive signature aggregation, the boundaries between transactions are more flexible than in bitcoin).

CoinSpends

```
public IEnumerable<CoinSpend> CoinSpends { get; init; }
```

Property Value

[IEnumerable](#) <[CoinSpend](#)>

This is a list of coins being spent along with their solution programs, and a single aggregated signature. This is the object that most closely corresponds to a bitcoin transaction (although because of non-interactive signature aggregation, the boundaries between transactions are more flexible than in bitcoin).

Enum Status

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public enum Status
```

Fields

COMMITTED = 2

PENDING = 1

Class StoreProofs

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record StoreProofs : IEquatable<StoreProofs>
```

Inheritance

[object](#) ← StoreProofs

Implements

[IEquatable](#) <[StoreProofs](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Proofs

```
public IEnumerable<Proof> Proofs { get; init; }
```

Property Value

[IEnumerable](#) <[Proof](#)>

StoreId

```
public string StoreId { get; init; }
```

Property Value

[string](#)

Class SubEpochSummary

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record SubEpochSummary : IEquatable<SubEpochSummary>
```

Inheritance

[object](#) ← SubEpochSummary

Implements

[IEquatable](#)<[SubEpochSummary](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

NewDifficulty

Only once per epoch (diff adjustment)

```
public ulong? NewDifficulty { get; init; }
```

Property Value

[ulong](#)?

NewSubSlotIters

Only once per epoch (diff adjustment)

```
public ulong? NewSubSlotIters { get; init; }
```

Property Value

[ulong](#)?

NumBlocksOverflow

How many more blocks than $384*(N-1)$

```
public byte NumBlocksOverflow { get; init; }
```

Property Value

[byte](#)

PrevSubepochSummaryHash

```
public string PrevSubepochSummaryHash { get; init; }
```

Property Value

[string](#)

RewardChainHash

hash of reward chain at end of last segment

```
public string RewardChainHash { get; init; }
```

Property Value

[string](#)

Class SubSlotProofs

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record SubSlotProofs : IEquatable<SubSlotProofs>
```

Inheritance

[object](#) ← SubSlotProofs

Implements

[IEquatable](#)<[SubSlotProofs](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

ChallengeChainSlotProof

```
public VDFProof ChallengeChainSlotProof { get; init; }
```

Property Value

[VDFProof](#)

InfusedChallengeChainSlotProof

```
public VDFProof? InfusedChallengeChainSlotProof { get; init; }
```

Property Value

[VDFProof](#)

RewardChainSlotProof

```
public VDFProof RewardChainSlotProof { get; init; }
```

Property Value

[VDFProof](#)

Class SyncState

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record SyncState : IEquatable<SyncState>
```

Inheritance

[object](#) ← SyncState

Implements

[IEquatable](#)<[SyncState](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

SyncMode

```
public bool SyncMode { get; init; }
```

Property Value

[bool](#)

SyncProgressHeight

```
public uint SyncProgressHeight { get; init; }
```

Property Value

[uint](#)

SyncProgressPercent

```
[JsonIgnore]  
public double SyncProgressPercent { get; }
```

Property Value

[double](#) ↗

SyncTipHeight

```
public uint SyncTipHeight { get; init; }
```

Property Value

[uint](#) ↗

Synced

```
public bool Synced { get; init; }
```

Property Value

[bool](#) ↗

Class TerminalNode

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record TerminalNode : IEquatable<TerminalNode>
```

Inheritance

[object](#) ← TerminalNode

Implements

[IEquatable](#) <[TerminalNode](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Hash

```
public string Hash { get; init; }
```

Property Value

[string](#)

Key

```
public string Key { get; init; }
```

Property Value

[string](#)

Value

```
public string Value { get; init; }
```

Property Value

[string](#) ↗

Class Token

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record Token : IEquatable<Token>
```

Inheritance

[object](#) ← Token

Implements

[IEquatable](#) <[Token](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

AssetId

```
public string AssetId { get; init; }
```

Property Value

[string](#)

FirstSeenHeight

```
public uint FirstSeenHeight { get; init; }
```

Property Value

[uint](#)

Name

```
public string Name { get; init; }
```

Property Value

[string](#) ↗

SenderPuzzleHash

```
public string SenderPuzzleHash { get; init; }
```

Property Value

[string](#) ↗

Class TradeManager

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

API wrapper for those wallet RPC methods dealing with trades and offers

```
public sealed class TradeManager
```

Inheritance

[object](#) ← TradeManager

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Constructors

TradeManager(WalletProxy)

API wrapper for those wallet RPC methods dealing with trades and offers

```
public TradeManager(WalletProxy walletProxy)
```

Parameters

walletProxy [WalletProxy](#)

API wrapper for those wallet RPC methods dealing with trades and offers

Properties

WalletProxy

```
public WalletProxy WalletProxy { get; init; }
```

Property Value

WalletProxy

API wrapper for those wallet RPC methods dealing with trades and offers

Methods

AssetIdToName(string, CancellationToken)

Get the CAT name from an asset id

```
public Task<(uint? WalletId, string Name)> AssetIdToName(string assetId, CancellationToken cancellationToken = default)
```

Parameters

assetId [string](#)

The asset id

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<(uint? WalletId, string Name)>

The wallet id and name of the CAT

CancelOffer(string, bool, ulong, CancellationToken)

Cancels an offer using a transaction

```
public Task CancelOffer(string tradeId, bool secure = false, ulong fee = 0, CancellationToken cancellationToken = default)
```

Parameters

`tradeId` [string](#)

The trade id of the offer

`secure` [bool](#)

This will create a transaction that includes coins that were offered

`fee` [ulong](#)

Transaction fee

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

An awaitable Task

CancelOffers(bool, string, bool, int, ulong, CancellationToken)

Cancels multiple offers.

```
public Task CancelOffers(bool secure, string assetId = "xch", bool cancelAll = false, int  
batchSize = 5, ulong batchFee = 0, CancellationToken cancellationToken = default)
```

Parameters

`secure` [bool](#)

`assetId` [string](#)

`cancelAll` [bool](#)

`batchSize` [int](#)

`batchFee` [ulong](#)

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

An awaitable Task

CheckOfferValidity(string, CancellationToken)

Checks the validity of an offer

```
public Task<(bool Valid, string Id)> CheckOfferValidity(string offer, CancellationToken cancellationToken = default)
```

Parameters

offer [string](#)

The bech32 encoded offer hex

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<(bool Valid, string Id)>

Indicator of the offer's validity

CreateOffer(IDictionary<string, long>, ulong?, ulong?, bool, IDictionary<string, string>?, IDictionary<string, string>?, bool?, ulong, CancellationToken)

Create an offer file from a set of id's in the form of wallet_id:amount

```
public Task<OfferRecord> CreateOffer(IDictionary<string, long> launcherIdsAndMojoAmounts, ulong? minCoinAmount = null, ulong? maxCoinAmount = null, bool validateOnly = false,
```

```
IDictionary<string, string>? driver = null, IDictionary<string, string>? solver = null,  
bool? reusePuzhash = null, ulong fee = 0, CancellationToken cancellationToken = default)
```

Parameters

`launcherIdsAndMojoAmounts` [IDictionary](#)<string, long>

The set of wallet ids and amounts (in mojo) representing the offer

`minCoinAmount` [ulong](#)?

`maxCoinAmount` [ulong](#)?

`validateOnly` [bool](#)

Only validate the offer contents. Do not create.

`driver` [IDictionary](#)<string, string>

Additional data about the puzzle

`solver` [IDictionary](#)<string, string>

`reusePuzhash` [bool](#)?

`fee` [ulong](#)

Transaction fee for offer creation

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<OfferRecord>

An awaitable [Task](#)

CreateOffer(IDictionary<uint, long>, ulong?, ulong?, bool,
IDictionary<string, string>?, IDictionary<string, string>?, bool?,
ulong, CancellationToken)

Create an offer file from a set of id's in the form of wallet_id:amount

```
public Task<OfferRecord> CreateOffer(IDictionary<uint, long> walletIdsAndMojoAmounts, ulong? minCoinAmount = null, ulong? maxCoinAmount = null, bool validateOnly = false, IDictionary<string, string>? driver = null, IDictionary<string, string>? solver = null, bool? reusePuzhash = null, ulong fee = 0, CancellationToken cancellationToken = default)
```

Parameters

walletIdsAndMojoAmounts [IDictionary](#)<uint, long>

The set of wallet ids and amounts (in mojo) representing the offer

minCoinAmount [ulong](#)?

maxCoinAmount [ulong](#)?

validateOnly [bool](#)

Only validate the offer contents. Do not create.

driver [IDictionary](#)<string, string>

Additional data about the puzzle

solver [IDictionary](#)<string, string>

reusePuzhash [bool](#)?

fee [ulong](#)

Transaction fee for offer creation

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<OfferRecord>

An awaitable [Task](#)

CreateOffer(OfferSummary, ulong?, ulong?, bool, IDictionary<string, string>?, IDictionary<string, string>?, bool?, ulong, CancellationToken)

Create an offer file from a set of id's in the form of wallet_id:amount

```
public Task<OfferRecord> CreateOffer(OfferSummary offer, ulong? minCoinAmount = null, ulong?  
maxCoinAmount = null, bool validateOnly = false, IDictionary<string, string>? driver = null,  
IDictionary<string, string>? solver = null, bool? reusePuzhash = null, ulong fee = 0,  
CancellationToken cancellationToken = default)
```

Parameters

offer [OfferSummary](#)

Summary of the offer to create

minCoinAmount [ulong](#)?

maxCoinAmount [ulong](#)?

validateOnly [bool](#)

Only validate the offer contents. Do not create.

driver [IDictionary](#)<[string](#), [string](#)>

Additional data about the puzzle

solver [IDictionary](#)<[string](#), [string](#)>

reusePuzhash [bool](#)?

fee [ulong](#)

Transaction fee for offer creation

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[OfferRecord](#)>

An awaitable [Task](#)

GetCATList(CancellationToken)

Get the default list of CATs

```
public Task<IEnumerable<CATInfo>> GetCATList(CancellationToken cancellationToken = default)
```

Parameters

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <IEnumerable <[CATInfo](#)>>

A list of CATs

GetOffer(string, bool, CancellationToken)

Retrieves an offer

```
public Task<OfferRecord> GetOffer(string tradeId, bool fileContents = false,  
CancellationToken cancellationToken = default)
```

Parameters

`tradeId` [string](#)

The trade id of the offer

`fileContents` [bool](#)

Indicator as to whether to return the offer contents. [Offer](#) will be empty if this is false.

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[OfferRecord](#)>

The OfferRecord

GetOfferSummary(string, bool, CancellationToken)

Retrieves the summary of an offer

```
public Task<OfferSummary> GetOfferSummary(string offer, bool advanced = false,  
CancellationToken cancellationToken = default)
```

Parameters

offer [string](#)

The bech32 encoded offer hex

advanced [bool](#)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[OfferSummary](#)>

The summary of the offer

GetOffers(bool, bool, bool, string?, bool, bool, CancellationToken)

Get the list of offers

```
public Task<IEnumerable<OfferRecord>> GetOffers(bool excludeMyOffers = false, bool  
excludeTakenOffers = false, bool includeCompleted = false, string? sortKey = null, bool  
reverse = false, bool fileContents = false, CancellationToken cancellationToken = default)
```

Parameters

excludeMyOffers [bool](#)

Do not include my offers in the result set

excludeTakenOffers [bool](#)

Do not include taken offers in the result set

includeCompleted [bool](#)

Do not include completed offers in the result set

sortKey [string](#)

Field to sort results by

reverse [bool](#)

Reverse the sort order of the results

fileContents [bool](#)

Include the offer value in the result

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[IEnumerable](#)<[OfferRecord](#)>>

A list of [OfferRecord](#)s

GetOffers(int, int, bool, bool, bool, string?, bool, bool, CancellationToken)

Get the list of offers

```
public Task<IEnumerable<OfferRecord>> GetOffers(int start, int end, bool excludeMyOffers =  
false, bool excludeTakenOffers = false, bool includeCompleted = false, string? sortKey =
```

```
null, bool reverse = false, bool fileContents = false, CancellationToken cancellationToken  
= default)
```

Parameters

start [int](#)

the start index of offers (zero based)

end [int](#)

The end index of offers

excludeMyOffers [bool](#)

Do not include my offers in the result set

excludeTakenOffers [bool](#)

Do not include taken offers in the result set

includeCompleted [bool](#)

Do not include completed offers in the result set

sortKey [string](#)

Field to sort results by

reverse [bool](#)

Reverse the sort order of the results

fileContents [bool](#)

Include the offer value in the result

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <IEnumerable <[OfferRecord](#)>>

A list of [OfferRecords](#)

GetOffersCount(CancellationToken)

Retrieves the number of offers.

```
public Task<(int Total, int MyOffersCount, int TakenOffersCount)>
GetOffersCount(CancellationToken cancellationToken = default)
```

Parameters

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<(int [Total](#), int [MyOffersCount](#), int [TakenOffersCount](#))>

The number of offers

GetStrayCats(CancellationToken)

Get a list of all unacknowledged CATs.

```
public Task<IEnumerable<Token>> GetStrayCats(CancellationToken cancellationToken = default)
```

Parameters

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[IEnumerable](#)<[Token](#)>>

The list of [Tokens](#)

TakeOffer(string, IDictionary<string, object>?, ulong?, ulong?,

bool?, ulong, CancellationToken)

Takes an offer

```
public Task<TradeRecord> TakeOffer(string offer, IDictionary<string, object>? solver = null,  
ulong? minCoinAmount = null, ulong? maxCoinAmount = null, bool? reusePuzhash = null, ulong  
fee = 0, CancellationToken cancellationToken = default)
```

Parameters

offer [string](#)?

solver [IDictionary](#)<[string](#), [object](#)>

minCoinAmount [ulong](#)?

maxCoinAmount [ulong](#)?

reusePuzhash [bool](#)?

fee [ulong](#)

cancellationToken [CancellationToken](#)

Returns

[Task](#)<[TradeRecord](#)>

[TradeRecord](#)

Class TradeRecord

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

Used for storing transaction data and status in wallets.

```
public record TradeRecord : IEquatable<TradeRecord>
```

Inheritance

[object](#) ← TradeRecord

Implements

[IEquatable](#) <[TradeRecord](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

AcceptedAtDateTime

```
[JsonIgnore]  
public DateTime? AcceptedAtDateTime { get; }
```

Property Value

[DateTime](#)?

Used for storing transaction data and status in wallets.

AcceptedAtTime

```
public ulong? AcceptedAtTime { get; init; }
```

Property Value

[ulong](#)?

Used for storing transaction data and status in wallets.

CoinsOfInterest

```
public IEnumerable<Coin> CoinsOfInterest { get; init; }
```

Property Value

[IEnumerable](#)<[Coin](#)>

Used for storing transaction data and status in wallets.

ConfirmedAtIndex

```
public uint ConfirmedAtIndex { get; init; }
```

Property Value

[uint](#)

Used for storing transaction data and status in wallets.

CreatedDateTime

```
[JsonIgnore]  
public DateTime CreatedDateTime { get; }
```

Property Value

[DateTime](#)

Used for storing transaction data and status in wallets.

CreatedAtTime

```
public ulong CreatedAtTime { get; init; }
```

Property Value

[ulong](#)

Used for storing transaction data and status in wallets.

IsMyOffer

```
public bool IsMyOffer { get; init; }
```

Property Value

[bool](#)

Used for storing transaction data and status in wallets.

Offer

Bech32 encoded value of the offer

```
public string Offer { get; init; }
```

Property Value

[string](#)

Used for storing transaction data and status in wallets.

Sent

```
public uint Sent { get; init; }
```

Property Value

[uint](#)

Used for storing transaction data and status in wallets.

SentTo

```
public IEnumerable<SendPeer> SentTo { get; init; }
```

Property Value

[IEnumerable](#)<[SendPeer](#)>

Used for storing transaction data and status in wallets.

Status

```
public TradeStatus Status { get; init; }
```

Property Value

[TradeStatus](#)

Used for storing transaction data and status in wallets.

TakenOffer

```
public string? TakenOffer { get; init; }
```

Property Value

[string](#)

Used for storing transaction data and status in wallets.

TradeId

```
public string TradeId { get; init; }
```

Property Value

[string](#)

Used for storing transaction data and status in wallets.

Enum TradeStatus

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public enum TradeStatus : uint
```

Fields

CANCELLED = 3

CONFIRMED = 4

FAILED = 5

PENDING_ACCEPT = 0

PENDING_CANCEL = 2

PENDING_CONFIRM = 1

Class TransactionRecord

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

Used for storing transaction data and status in wallets.

```
public record TransactionRecord : IEquatable<TransactionRecord>
```

Inheritance

[object](#) ← TransactionRecord

Implements

[IEquatable](#)<[TransactionRecord](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Additions

```
public IEnumerable<Coin> Additions { get; init; }
```

Property Value

[IEnumerable](#)<[Coin](#)>

Used for storing transaction data and status in wallets.

Amount

```
public ulong Amount { get; init; }
```

Property Value

[ulong](#)

Used for storing transaction data and status in wallets.

Confirmed

```
public bool Confirmed { get; init; }
```

Property Value

[bool](#)

Used for storing transaction data and status in wallets.

ConfirmedAtHeight

```
public uint ConfirmedAtHeight { get; init; }
```

Property Value

[uint](#)

Used for storing transaction data and status in wallets.

CreatedAtDateTime

```
[JsonIgnore]  
public DateTime CreatedAtDateTime { get; }
```

Property Value

[DateTime](#)

Used for storing transaction data and status in wallets.

CreatedAtTime

```
public double CreatedAtTime { get; init; }
```

Property Value

[double](#)

Used for storing transaction data and status in wallets.

FeeAmount

```
public ulong FeeAmount { get; init; }
```

Property Value

[ulong](#)

Used for storing transaction data and status in wallets.

IsInMempool

If one of the nodes we sent it to responded with success, we set it to success

```
[JsonIgnore]  
public bool IsInMempool { get; }
```

Property Value

[bool](#)

Used for storing transaction data and status in wallets.

Remarks

Note, transactions pending inclusion (pending) return false

Name

```
public string Name { get; init; }
```

Property Value

[string](#)

Used for storing transaction data and status in wallets.

Removals

```
public IEnumerable<Coin> Removals { get; init; }
```

Property Value

[IEnumerable](#)<[Coin](#)>

Used for storing transaction data and status in wallets.

Sent

```
public uint Sent { get; init; }
```

Property Value

[uint](#)

Used for storing transaction data and status in wallets.

SentTo

Represents the list of peers that we sent the transaction to, whether each one included it in the mempool, and what the error message (if any) was

```
public IEnumerable<SendPeer> SentTo { get; init; }
```

PropertyValue

[IEnumerable](#) <SendPeer>

Used for storing transaction data and status in wallets.

SpendBundle

```
public SpendBundle? SpendBundle { get; init; }
```

PropertyValue

[SpendBundle](#)

Used for storing transaction data and status in wallets.

ToAddress

```
public string ToAddress { get; init; }
```

PropertyValue

[string](#)

Used for storing transaction data and status in wallets.

ToPuzzleHash

```
public string ToPuzzleHash { get; init; }
```

PropertyValue

[string](#)

Used for storing transaction data and status in wallets.

TradeId

```
public string? TradeId { get; init; }
```

Property Value

[string](#) ↗

Used for storing transaction data and status in wallets.

TransactionId

chia python aliases the [Name](#) property to return this along with the record

```
[JsonIgnore]  
public string TransactionId { get; }
```

Property Value

[string](#) ↗

Used for storing transaction data and status in wallets.

Type

TransactionType

```
public TransactionType Type { get; init; }
```

Property Value

[TransactionType](#)

Used for storing transaction data and status in wallets.

ValidTimes

```
public ConditionValidTimes ValidTimes { get; init; }
```

Property Value

[ConditionValidTimes](#)

Used for storing transaction data and status in wallets.

WalletId

```
public uint WalletId { get; init; }
```

Property Value

[uint](#)

Used for storing transaction data and status in wallets.

Enum TransactionType

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public enum TransactionType
```

Fields

COINBASE_REWARD = 2

FEE_REWARD = 3

INCOMING_TRADE = 4

INCOMING_TX = 0

OUTGOING_TRADE = 5

OUTGOING_TX = 1

Class TransactionTypeFilter

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record TransactionTypeFilter : IEquatable<TransactionTypeFilter>
```

Inheritance

[object](#) ← TransactionTypeFilter

Implements

[IEquatable](#)<[TransactionTypeFilter](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Mode

```
public FilterMode Mode { get; init; }
```

Property Value

[FilterMode](#)

Values

```
public IEnumerable<byte> Values { get; init; }
```

Property Value

[IEnumerable](#)<[byte](#)>

Class TransactionsInfo

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

Information that goes along with each transaction block

```
public record TransactionsInfo : IEquatable<TransactionsInfo>
```

Inheritance

[object](#) ← TransactionsInfo

Implements

[IEquatable](#)<[TransactionsInfo](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

AggregatedSignature

```
public string AggregatedSignature { get; init; }
```

Property Value

[string](#)

Information that goes along with each transaction block

Cost

This is the total cost of this block, including CLVM cost, cost of program size and conditions

```
public ulong Cost { get; init; }
```

Property Value

[ulong](#)

Information that goes along with each transaction block

Fees

This only includes user fees, not block rewards

```
public ulong Fees { get; init; }
```

Property Value

[ulong](#)

Information that goes along with each transaction block

GeneratorRefsRoot

sha256 of the concatenation of the generator ref list entries

```
public string GeneratorRefsRoot { get; init; }
```

Property Value

[string](#)

Information that goes along with each transaction block

GeneratorRoot

sha256 of the block generator in this block

```
public string GeneratorRoot { get; init; }
```

Property Value

[string](#)

Information that goes along with each transaction block

RewardClaimsIncorporated

These can be in any order

```
public IEnumerable<Coin> RewardClaimsIncorporated { get; init; }
```

Property Value

[IEnumerable](#) <[Coin](#)>

Information that goes along with each transaction block

Class UInt32Range

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record UInt32Range : IEquatable<UInt32Range>
```

Inheritance

[object](#) ← UInt32Range

Implements

[IEquatable](#)<[UInt32Range](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Start

```
public uint Start { get; init; }
```

Property Value

[uint](#)

Stop

```
public uint Stop { get; init; }
```

Property Value

[uint](#)

Class UInt64Range

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record UInt64Range : IEquatable<UInt64Range>
```

Inheritance

[object](#) ← UInt64Range

Implements

[IEquatable](#)<[UInt64Range](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Start

```
public ulong Start { get; init; }
```

Property Value

[ulong](#)

Stop

```
public ulong Stop { get; init; }
```

Property Value

[ulong](#)

Class UnfinishedHeaderBlock

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

Same as a FullBlock but without TransactionInfo and Generator, used by light clients

```
public record UnfinishedHeaderBlock : IEquatable<UnfinishedHeaderBlock>
```

Inheritance

[object](#) ← UnfinishedHeaderBlock

Implements

[IEquatable](#) <[UnfinishedHeaderBlock](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

ChallengeChainSpProof

If not first sp in sub-slot

```
public VDFProof? ChallengeChainSpProof { get; init; }
```

Property Value

[VDFProof](#)

Same as a FullBlock but without TransactionInfo and Generator, used by light clients

FinishedSubSlots

If first sb

```
public IEnumerable<EndOfSubSlotBundle> FinishedSubSlots { get; init; }
```

Property Value

[IEnumerable](#)<[EndOfSubSlotBundle](#)>

Same as a FullBlock but without TransactionInfo and Generator, used by light clients

Foliage

Reward chain foliage data

```
public Foliage Foliage { get; init; }
```

Property Value

[Foliage](#)

Same as a FullBlock but without TransactionInfo and Generator, used by light clients

FoliageTransactionBlock

Reward chain foliage data (tx block)

```
public FoliageTransactionBlock? FoliageTransactionBlock { get; init; }
```

Property Value

[FoliageTransactionBlock](#)

Same as a FullBlock but without TransactionInfo and Generator, used by light clients

RewardChainBlock

Reward chain trunk data

```
public RewardChainBlockUnfinished RewardChainBlock { get; init; }
```

Property Value

[RewardChainBlockUnfinished](#)

Same as a FullBlock but without TransactionInfo and Generator, used by light clients

RewardChainSpProof

If not first sp in sub-slot

```
public VDFProof? RewardChainSpProof { get; init; }
```

Property Value

[VDFProof](#)

Same as a FullBlock but without TransactionInfo and Generator, used by light clients

TransactionsFilter

Filter for block transactions

```
public string TransactionsFilter { get; init; }
```

Property Value

[string](#)

Same as a FullBlock but without TransactionInfo and Generator, used by light clients

Class VCLineageProof

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

The covenant layer for exigent metadata layers requires to be passed the previous parent's metadata too

```
public record VCLineageProof : LineageProof, IEquatable<LineageProof>,
IEquatable<VCLineageProof>
```

Inheritance

[object](#) ← [LineageProof](#) ← VCLineageProof

Implements

[IEquatable](#)<[LineageProof](#)>, [IEquatable](#)<[VCLineageProof](#)>

Inherited Members

[LineageProof.ParentName](#), [LineageProof.InnerPuzzleHash](#), [LineageProof.Amount](#),
[object.Equals\(object\)](#), [object.Equals\(object, object\)](#), [object.GetHashCode\(\)](#), [object.GetType\(\)](#),
[object.MemberwiseClone\(\)](#), [object.ReferenceEquals\(object, object\)](#), [object.ToString\(\)](#)

Properties

ParentProofHash

```
public string? ParentProofHash { get; init; }
```

Property Value

[string](#)

The covenant layer for exigent metadata layers requires to be passed the previous parent's metadata too

Class VCProofs

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record VCProofs : IEquatable<VCProofs>
```

Inheritance

[object](#) ← VCProofs

Implements

[IEquatable](#)<[VCProofs](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

KeyValuePair

```
public IDictionary<string, string> KeyValuePair { get; init; }
```

Property Value

[IDictionary](#)<[string](#), [string](#)>

Class VCRecord

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record VCRecord : IEquatable<VCRecord>
```

Inheritance

[object](#) ← VCRecord

Implements

[IEquatable](#)<[VCRecord](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

CoinId

```
public string? CoinId { get; init; }
```

Property Value

[string](#)

ConfirmedAtHeight

```
public uint ConfirmedAtHeight { get; init; }
```

Property Value

[uint](#)

VC

```
public VerifiedCredential VC { get; init; }
```

Property Value

[VerifiedCredential](#)

Class VDFInfo

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record VDFInfo : IEquatable<VDFInfo>
```

Inheritance

[object](#) ↗ ← VDFInfo

Implements

[IEquatable](#) ↗ <[VDFInfo](#)>

Inherited Members

[object.Equals\(object\)](#) ↗ , [object.Equals\(object, object\)](#) ↗ , [object.GetHashCode\(\)](#) ↗ , [object.GetType\(\)](#) ↗ ,
[object.MemberwiseClone\(\)](#) ↗ , [object.ReferenceEquals\(object, object\)](#) ↗ , [object.ToString\(\)](#) ↗

Properties

Challenge

Used to generate the discriminant (VDF group)

```
public string Challenge { get; init; }
```

Property Value

[string](#) ↗

NumberOfIterations

```
public ulong NumberOfIterations { get; init; }
```

Property Value

[ulong](#) ↗

Output

```
public ClassgroupElement Output { get; init; }
```

Property Value

[ClassgroupElement](#)

Class VDFProof

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record VDFProof : IEquatable<VDFProof>
```

Inheritance

[object](#) ← VDFProof

Implements

[IEquatable](#)<[VDFProof](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

NormalizedToIdentity

```
public bool NormalizedToIdentity { get; init; }
```

Property Value

[bool](#)

Witness

```
public string Witness { get; init; }
```

Property Value

[string](#)

WitnessType

```
public byte WitnessType { get; init; }
```

Property Value

[byte](#) ↗

Class VerifiedCredential

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

This class serves as the main driver for the entire VC puzzle stack. Given the information below, it can sync and spend VerifiedCredentials in any specified manner. Trying to sync from a spend that this class did not create will likely result in an error.

```
public record VerifiedCredential : IEquatable<VerifiedCredential>
```

Inheritance

[object](#) ← VerifiedCredential

Implements

[IEquatable](#) <[VerifiedCredential](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Coin

```
public Coin Coin { get; init; }
```

Property Value

[Coin](#)

This class serves as the main driver for the entire VC puzzle stack. Given the information below, it can sync and spend VerifiedCredentials in any specified manner. Trying to sync from a spend that this class did not create will likely result in an error.

EMLLineageProof

```
public VCLineageProof EMLLineageProof { get; init; }
```

Property Value

[VCLineageProof](#)

This class serves as the main driver for the entire VC puzzle stack. Given the information below, it can sync and spend VerifiedCredentials in any specified manner.Trying to sync from a spend that this class did not create will likely result in an error.

InnerPuzzleHash

```
public string InnerPuzzleHash { get; init; }
```

Property Value

[string](#)

This class serves as the main driver for the entire VC puzzle stack. Given the information below, it can sync and spend VerifiedCredentials in any specified manner.Trying to sync from a spend that this class did not create will likely result in an error.

LauncherId

```
public string LauncherId { get; init; }
```

Property Value

[string](#)

This class serves as the main driver for the entire VC puzzle stack. Given the information below, it can sync and spend VerifiedCredentials in any specified manner.Trying to sync from a spend that this class did not create will likely result in an error.

ProofHash

```
public string? ProofHash { get; init; }
```

Property Value

[string](#) ↗

This class serves as the main driver for the entire VC puzzle stack. Given the information below, it can sync and spend VerifiedCredentials in any specified manner.Trying to sync from a spend that this class did not create will likely result in an error.

ProofProvider

```
public string ProofProvider { get; init; }
```

Property Value

[string](#) ↗

This class serves as the main driver for the entire VC puzzle stack. Given the information below, it can sync and spend VerifiedCredentials in any specified manner.Trying to sync from a spend that this class did not create will likely result in an error.

SingletonLineageProof

```
public LineageProof SingletonLineageProof { get; init; }
```

Property Value

[LineageProof](#)

This class serves as the main driver for the entire VC puzzle stack. Given the information below, it can sync and spend VerifiedCredentials in any specified manner.Trying to sync from a spend that this class did not create will likely result in an error.

Class VerifiedCredentialManager

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

API wrapper for those wallet RPC methods dealing with verified credentials

```
public sealed class VerifiedCredentialManager
```

Inheritance

[object](#) ← VerifiedCredentialManager

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Constructors

VerifiedCredentialManager(WalletProxy)

API wrapper for those wallet RPC methods dealing with verified credentials

```
public VerifiedCredentialManager(WalletProxy walletProxy)
```

Parameters

walletProxy [WalletProxy](#)

API wrapper for those wallet RPC methods dealing with verified credentials

Properties

WalletProxy

```
public WalletProxy WalletProxy { get; init; }
```

Property Value

[WalletProxy](#)

API wrapper for those wallet RPC methods dealing with verified credentials

Methods

AddProofs(VCProofs, CancellationToken)

Add a set of proofs to the DB that can be used when spending a VC. VCs are near useless until their proofs have been added.

```
public Task AddProofs(VCProofs proofs, CancellationToken cancellationToken = default)
```

Parameters

`proofs` [VCProofs](#)

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

An awaitable Task

Get(string, CancellationToken)

Given a launcher ID get the verified credential.

```
public Task<VCRecord> Get(string vcId, CancellationToken cancellationToken = default)
```

Parameters

`vcId` [string](#)

launcher ID

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[VCRecord](#)>

[VCRecord](#)

GetList(uint, uint, CancellationToken)

Get a list of verified credentials in the specified range and any 'proofs' associated with the roots contained within.

```
public Task<IEnumerable<VCRecord>> GetList(uint start = 0, uint end = 50, CancellationToken  
cancellationToken = default)
```

Parameters

start [uint](#)

end [uint](#)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[IEnumerable](#) <[VCRecord](#)>>

A list of [VCRecord](#)

GetProofsForRoot(string, CancellationToken)

Given a specified vc root, get any proofs associated with that root.

```
public Task<IDictionary<string, string>> GetProofsForRoot(string root, CancellationToken cancellationToken = default)
```

Parameters

root [string](#)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[IDictionary](#)<string, string>>

A Dictionary of strings

Mint(string, string, ulong, CancellationToken)

Mint a verified credential using the assigned DID.

```
public Task<(VCRecord VCRecord, IEnumerable<TradeRecord> Transactions)> Mint(string targetAddress, string didId, ulong fee = 0, CancellationToken cancellationToken = default)
```

Parameters

targetAddress [string](#)

didId [string](#)

fee [ulong](#)

Fee (in units of mojos)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<([VCRecord](#) [VCRecord](#), [IEnumerable](#)<[TradeRecord](#)> [Transactions](#))>

a VCRecord and list of TradeRecord

Revoke(string, bool?, ulong, CancellationToken)

Revoke an on chain VC provided the correct DID is available.

```
public Task<IEnumerable<TransactionRecord>> Revoke(string vcParentId, bool? reusePuzhash = null, ulong fee = 0, CancellationToken cancellationToken = default)
```

Parameters

vcParentId [string](#)

reusePuzhash [bool](#)?

fee [ulong](#)

Fee (in units of mojos)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[IEnumerable](#)<[TransactionRecord](#)>>

A list of [TransactionRecord](#)

Spend(string, string?, string?, string?, bool?, ulong, CancellationToken)

Spend a verified credential.

```
public Task<IEnumerable<TransactionRecord>> Spend(string vcId, string? newPuzhash = null, string? providerInnerPuzhash = null, string? newProofHash = null, bool? reusePuzhash = null, ulong fee = 0, CancellationToken cancellationToken = default)
```

Parameters

`vcId` [string](#)

`newPuzhash` [string](#)

`providerInnerPuzhash` [string](#)

`newProofHash` [string](#)

`reusePuzhash` [bool](#)?

`fee` [ulong](#)

Fee (in units of mojos)

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[IEnumerable](#) <[TransactionRecord](#)>>

A list of [TransactionRecord](#)

Class Wallet

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

Base class representing a specific wallet (i.e. anything with a WalletID)

```
public class Wallet
```

Inheritance

[object](#) ← Wallet

Derived

[CATWallet](#), [CRCATWallet](#), [DAOWallet](#), [DIDWallet](#), [DataLayerWallet](#), [NFTWallet](#), [PoolWallet](#)

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Remarks

When not derived from this represents a [STANDARD WALLET](#)

Constructors

Wallet(uint, WalletProxy)

Base class representing a specific wallet (i.e. anything with a WalletID)

```
public Wallet(uint walletId, WalletProxy walletProxy)
```

Parameters

walletId [uint](#)

The wallet_id to wrap

walletProxy [WalletProxy](#)

Wallet RPC proxy to use for communication

Remarks

When not derived from this represents a [STANDARD_WALLET](#)

Properties

WalletId

The id of the wallet

```
public uint WalletId { get; init; }
```

Property Value

[uint](#)

Base class representing a specific wallet (i.e. anything with a WalletID)

WalletProxy

Wallet RPC proxy for communication

```
public WalletProxy WalletProxy { get; init; }
```

Property Value

[WalletProxy](#)

Base class representing a specific wallet (i.e. anything with a WalletID)

Methods

CreateWalletDataObject()

Creates a dynamic object and sets its wallet_id property to [WalletId](#)

```
protected dynamic CreateWalletDataObject()
```

Returns

dynamic

A dynamic object

DeleteUnconfirmedTransactions(CancellationToken)

Delete unconfirmed transactions from the wallet

```
public Task DeleteUnconfirmedTransactions(CancellationToken cancellationToken = default)
```

Parameters

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

An awaitable [Task](#)

GetBalance(CancellationToken)

Get the balance of a specific wallet

```
public Task<WalletBalance> GetBalance(CancellationToken cancellationToken = default)
```

Parameters

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[WalletBalance](#)>

The wallet balance (in units of mojos)

GetNextAddress(bool, CancellationToken)

Get the last address or create a new one

```
public Task<string> GetNextAddress(bool newAddress, CancellationToken cancellationToken  
= default)
```

Parameters

newAddress [bool](#)

Whether to generate a new address

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[string](#)>

An address

GetSpendableCoins(ulong?, ulong?, IEnumerable<ulong>?, IEnumerable<Coin>?, IEnumerable<string>?, CancellationToken)

Get the list of spendable coins

```
public Task<(IEnumerable<CoinRecord> ConfirmedRecords, IEnumerable<CoinRecord>  
UnconfirmedRecords, IEnumerable<Coin> UnconfirmedAdditions)> GetSpendableCoins(ulong?  
minCoinAmount, ulong? maxCoinAmount, IEnumerable<ulong>? excludedCoinAmounts = null,  
IEnumerable<Coin>? excludedCoins = null, IEnumerable<string>? excludedCoinIds = null,  
CancellationToken cancellationToken = default)
```

Parameters

`minCoinAmount` [ulong](#)?

The minimum coin amount

`maxCoinAmount` [ulong](#)?

The maximum coin amount>

`excludedCoinAmounts` [IEnumerable](#)<[ulong](#)>

Amounts to exlcude

`excludedCoins` [IEnumerable](#)<[Coin](#)>

Coins to exclude

`excludedCoinIds` [IEnumerable](#)<[string](#)>

Coin ids to exclude

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<([IEnumerable](#)<[CoinRecord](#)> [ConfirmedRecords](#), [IEnumerable](#)<[CoinRecord](#)> [UnconfirmedRecords](#), [IEnumerable](#)<[Coin](#)> [UnconfirmedAdditions](#))>

Information about spendable coins

GetTransactionCount(TransactionTypeFilter?, CancellationToken)

Get the number of transactions

```
public Task<uint> GetTransactionCount(TransactionTypeFilter? typeFilter = null,  
CancellationToken cancellationToken = default)
```

Parameters

`typeFilter` [TransactionTypeFilter](#)

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[uint](#)>

The number of transactions

GetTransactions(string?, TransactionTypeFilter?, bool, string?, uint, uint, bool?, CancellationToken)

Retrieves a list of transactions from a wallet.

```
public Task<IEnumerable<TransactionRecord>> GetTransactions(string? toAddress = null,
    TransactionTypeFilter? typeFilter = null, bool reverse = false, string? sortKey = null,
    uint start = 0, uint end = 50, bool? confirmed = null, CancellationToken cancellationToken
    = default)
```

Parameters

toAddress [string](#)

Restrict results only to this address

typeFilter [TransactionTypeFilter](#)

reverse [bool](#)

Reverse the sort order of the results

sortKey [string](#)

Field to sort results by

start [uint](#)

the start index of transactions (zero based)

end [uint](#)

The end index of transactions

confirmed [bool](#)?

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[IEnumerable](#) <[TransactionRecord](#)>>

A list of [TransactionRecord](#)

GetWalletInfo(CancellationToken)

Retrieves information about this wallet

```
public Task<WalletInfo> GetWalletInfo(CancellationToken cancellationToken = default)
```

Parameters

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[WalletInfo](#)>

[WalletInfo](#) and the wallet pubkey fingerprint

Remarks

Throws an exception if the wallet does not exist

SelectCoins(ulong, IEnumerable<Coin>?, IEnumerable<ulong>?, ulong?, ulong?, CancellationToken)

Returns a set of coins that can be used for generating a new transaction.

```
public Task<IEnumerable<Coin>> SelectCoins(ulong amount, IEnumerable<Coin>? excludedCoins = null, IEnumerable<ulong>? excludedCoinAmounts = null, ulong? minCoinAmount = null, ulong?
```

```
maxCoinAmount = null, CancellationToken cancellationToken = default)
```

Parameters

amount [ulong](#)

An amount

excludedCoins [IEnumerable](#)<Coin>

excludedCoinAmounts [IEnumerable](#)<[ulong](#)>

minCoinAmount [ulong](#)?

maxCoinAmount [ulong](#)?

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[IEnumerable](#)<Coin>>

The list of [IEnumerable](#)<T>s

SendTransaction(string, ulong, IEnumerable<string>?,
IEnumerable<ulong>?, IEnumerable<string>?, ulong?, ulong?,
bool, ulong, CancellationToken)

Sends a transaction

```
public Task<TransactionRecord> SendTransaction(string address, ulong amount,  
IEnumerable<string>? memos = null, IEnumerable<ulong>? excludeCoinAmounts = null,  
IEnumerable<string>? excludeCoinsIds = null, ulong? minCoinAmount = null, ulong?  
maxCoinAmount = null, bool resusePuzHash = false, ulong fee = 0, CancellationToken  
cancellationToken = default)
```

Parameters

address [string](#)

The receiving address

amount [ulong](#)

The amount to send (in units of mojos)

memos [IEnumerable](#)<[string](#)>

Memos to go along with the transaction

excludeCoinAmounts [IEnumerable](#)<[ulong](#)>

excludeCoinsIds [IEnumerable](#)<[string](#)>

minCoinAmount [ulong](#)?

maxCoinAmount [ulong](#)?

reusePuzHash [bool](#)

fee [ulong](#)

Fee amount (in units of mojos)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[TransactionRecord](#)>

The [TransactionRecord](#)

**SendTransactionMulti(IEnumerable<Coin>,
IEnumerable<Coin>?, ulong, CancellationToken)**

Sends a transaction

```
public Task<TransactionRecord> SendTransactionMulti(IEnumerable<Coin> additions,  
IEnumerable<Coin>? coins = null, ulong fee = 0, CancellationToken cancellationToken  
= default)
```

Parameters

additions [IEnumerable](#) <[Coin](#)>

Additions to the block chain

coins [IEnumerable](#) <[Coin](#)>

Coin to include

fee [ulong](#)

Fee amount (in units of mojos)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[TransactionRecord](#)>

The [TransactionRecord](#)

Validate(CancellationToken)

Validates that [WalletId](#) is a [STANDARD WALLET](#)

```
public virtual Task Validate(CancellationToken cancellationToken = default)
```

Parameters

cancellationToken [CancellationToken](#)

Base class representing a specific wallet (i.e. anything with a WalletID)

Returns

[Task](#)

True if the wallet is of the expected type

Remarks

Intended to be overriden by derived classes of specific [WalletType](#)

Validate(WalletType, CancellationToken)

Validates that [WalletId](#) exists and is of the correct [WalletType](#)

```
protected Task Validate(WalletType walletType, CancellationToken cancellationToken)
```

Parameters

walletType [WalletType](#)

The expected type of wallet

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

true if the wallet is of the expected type

Remarks

Throws an exception if the wallet does not exist

Class WalletAddress

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record WalletAddress : IEquatable<WalletAddress>
```

Inheritance

[object](#) ← WalletAddress

Implements

[IEquatable](#) <[WalletAddress](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Address

```
public string Address { get; init; }
```

Property Value

[string](#)

HdPath

```
public string HdPath { get; init; }
```

Property Value

[string](#)

Class WalletBalance

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

```
public record WalletBalance : IEquatable<WalletBalance>
```

Inheritance

[object](#) ← WalletBalance

Implements

[IEquatable](#) <[WalletBalance](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

AssetId

```
public string? AssetId { get; init; }
```

Property Value

[string](#)

ConfirmedWalletBalance

```
public BigInteger ConfirmedWalletBalance { get; init; }
```

Property Value

[BigInteger](#)

Fingerprint

```
public uint? Fingerprint { get; init; }
```

Property Value

[uint](#)?

MaxSendAmount

```
public BigInteger MaxSendAmount { get; init; }
```

Property Value

[BigInteger](#)

PendingChange

```
public BigInteger PendingChange { get; init; }
```

Property Value

[BigInteger](#)

PendingCoinRemovalCount

```
public int PendingCoinRemovalCount { get; init; }
```

Property Value

[int](#)

SpendableBalance

```
public BigInteger SpendableBalance { get; init; }
```

Property Value

[BigInteger](#) ↗

UnconfirmedWalletBalance

```
public BigInteger UnconfirmedWalletBalance { get; init; }
```

Property Value

[BigInteger](#) ↗

UnspentCoinCount

```
public int UnspentCoinCount { get; init; }
```

Property Value

[int](#) ↗

WalletId

```
public uint WalletId { get; init; }
```

Property Value

[uint](#) ↗

WalletType

```
public WalletType WalletType { get; init; }
```

Property Value

[WalletType](#)

Class WalletInfo

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

This object represents the wallet data as it is stored in DB. ID: Main wallet (Standard) is stored at index 1, every wallet created after done has auto incremented id. Name: can be a user provided or default generated name. (can be modified) Type: is specified during wallet creation and should never be changed. Data: this filed is intended to be used for storing any wallet specific information required for it. This data should be json encoded string.

```
public record WalletInfo : IEquatable<WalletInfo>
```

Inheritance

[object](#) ← WalletInfo

Implements

[IEquatable](#)<[WalletInfo](#)>

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Properties

Data

```
public string Data { get; init; }
```

Property Value

[string](#)

This object represents the wallet data as it is stored in DB. ID: Main wallet (Standard) is stored at index 1, every wallet created after done has auto incremented id. Name: can be a user provided or default generated name. (can be modified) Type: is specified during wallet creation and should never be changed. Data: this filed is intended to be used for storing any wallet specific information required for it. This data should be json encoded string.

Id

```
public uint Id { get; init; }
```

Property Value

[uint](#)

This object represents the wallet data as it is stored in DB. ID: Main wallet (Standard) is stored at index 1, every wallet created after done has auto incremented id. Name: can be a user provided or default generated name. (can be modified) Type: is specified during wallet creation and should never be changed. Data: this filed is intended to be used for storing any wallet specific information required for it. This data should be json encoded string.

Name

```
public string Name { get; init; }
```

Property Value

[string](#)

This object represents the wallet data as it is stored in DB. ID: Main wallet (Standard) is stored at index 1, every wallet created after done has auto incremented id. Name: can be a user provided or default generated name. (can be modified) Type: is specified during wallet creation and should never be changed. Data: this filed is intended to be used for storing any wallet specific information required for it. This data should be json encoded string.

Type

```
public WalletType Type { get; init; }
```

Property Value

[WalletType](#)

This object represents the wallet data as it is stored in DB. ID: Main wallet (Standard) is stored at index 1, every wallet created after done has auto incremented id. Name: can be a user provided or default generated name. (can be modified) Type: is specified during wallet creation and should never be changed. Data: this file is intended to be used for storing any wallet specific information required for it. This data should be json encoded string.

Class WalletProxy

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

Proxy that communicates with the wallet endpoint

```
public sealed class WalletProxy : ServiceProxy
```

Inheritance

[object](#) ← [ServiceProxy](#) ← WalletProxy

Inherited Members

[ServiceProxy.OriginService](#) , [ServiceProxy.DestinationService](#) , [ServiceProxy.RpcClient](#) ,
[ServiceProxy.HealthZ\(CancellationToken\)](#) , [ServiceProxy.StopNode\(CancellationToken\)](#) ,
[ServiceProxy.GetConnections\(CancellationToken\)](#) , [ServiceProxy.GetRoutes\(CancellationToken\)](#) ,
[ServiceProxy.OpenConnection\(string, int, CancellationToken\)](#) ,
[ServiceProxy.CloseConnection\(string, CancellationToken\)](#) , [object.Equals\(object\)](#) ,
[object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Remarks

ctor

Constructors

WalletProxy(IRpcClient, string)

Proxy that communicates with the wallet endpoint

```
public WalletProxy(IRpcClient rpcClient, string originService)
```

Parameters

rpcClient [IRpcClient](#)

[IRpcClient](#) instance to use for rpc communication

`originService` [string](#)

[Origin](#)

Remarks

ctor

Methods

AddKey(IEnumerable<string>, CancellationToken)

Adds a new key to the wallet

```
public Task<uint> AddKey(IEnumerable<string> mnemonic, CancellationToken cancellationToken = default)
```

Parameters

`mnemonic` [IEnumerable](#)<[string](#)>

The key mnemonic

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[uint](#)>

The new key's fingerprint

CalculateRoyalties(IEnumerable<FungibleAsset>, IEnumerable<RoyaltyAsset>, CancellationToken)

Transfers an NFT to another address.

```
public Task<IDictionary<string, IEnumerable<AssetInfo>>> CalculateRoyalties(IEnumerable<FungibleAsset> fungibleAssets, IEnumerable<RoyaltyAsset>
```

```
royaltyAssets, CancellationToken cancellationToken = default)
```

Parameters

fungibleAssets [IEnumerable](#)<[FungibleAsset](#)>

royaltyAssets [IEnumerable](#)<[RoyaltyAsset](#)>

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[IDictionary](#)<[string](#), [IEnumerable](#)<[AssetInfo](#)>>>

An awaitable Task

CheckDeleteKey(uint, CancellationToken)

Check the key use prior to possible deletion checks whether key is used for either farm or pool rewards checks if any wallets have a non-zero balance

```
public Task<(uint Fingerprint, bool UsedForFarmerRewards, bool UsedForPoolRewards, bool WalletBalance)> CheckDeleteKey(uint fingerprint, CancellationToken cancellationToken = default)
```

Parameters

fingerprint [uint](#)

The key's fingerprint

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<(uint Fingerprint, bool UsedForFarmerRewards, bool UsedForPoolRewards, bool WalletBalance)>

Indicators of how the wallet is used

CreateCATWallet(string, ulong, ulong, CancellationToken)

Create a new CAT wallet

```
public Task<(WalletType Type, string AssetId, uint WalletId)> CreateCATWallet(string name, ulong amount, ulong fee = 0, CancellationToken cancellationToken = default)
```

Parameters

name [string](#)

The wallet name

amount [ulong](#)

The amount to put in the wallet (in units of mojos)

fee [ulong](#)

Fee (in units of mojos)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<(WalletType Type, string AssetId, uint WalletId)>

Information about the wallet

CreateCATWallet(ulong, ulong, CancellationToken)

Create a new CAT wallet

```
public Task<(WalletType Type, string AssetId, uint WalletId)> CreateCATWallet(ulong amount, ulong fee = 0, CancellationToken cancellationToken = default)
```

Parameters

amount [ulong](#)

The amount to put in the wallet (in units of mojos)

fee [ulong](#)

Fee (in units of mojos)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<(WalletType Type, string AssetId, uint WalletId)>

Information about the wallet

CreateDIDWallet(IEnumerable<string>, ulong, string, IDictionary<string, string>?, ulong, CancellationToken)

Creates a new DID wallet

```
public Task<(WalletType Type, string myDID, uint walletId)>
CreateDIDWallet(IEnumerable<string> backupDIDs, ulong num0fBackupIdsNeeded, string name,
IDictionary<string, string>? metaData = null, ulong fee = 0, CancellationToken
cancellationToken = default)
```

Parameters

backupDIDs [IEnumerable](#)<[string](#)>

Backup DIDs

numOfBackupIdsNeeded [ulong](#)

The number of back ids needed to create the wallet

name [string](#)

metaData [IDictionary](#)<[string](#), [string](#)>

`fee` `ulong`

`cancellationToken` `CancellationToken`

A token to allow the call to be cancelled

Returns

`Task`<(`WalletType` `Type`, `string` `AssetId`, `uint` `WalletId`)>

Information about the wallet

CreateExistingDAOWallet(string, ulong, CancellationToken)

Create a new CAT wallet

```
public Task<(WalletType Type, string TreasuryId, uint WalletId, uint CatWalletId, uint DaoCatWalletId)> CreateExistingDAOWallet(string treasuryId, ulong filterAmount = 1, CancellationToken cancellationToken = default)
```

Parameters

`treasuryId` `string`

`filterAmount` `ulong`

`cancellationToken` `CancellationToken`

A token to allow the call to be cancelled

Returns

`Task`<(`WalletType` `Type`, `string` `TreasuryId`, `uint` `WalletId`, `uint` `CatWalletId`, `uint` `DaoCatWalletId`)>

Information about the wallet

CreateNFTWallet(string?, CancellationToken)

Creates a new NFT wallet

```
public Task<(uint Id, WalletType Type)> CreateNFTWallet(string? didId = null,  
CancellationToken cancellationToken = default)
```

Parameters

didId [string](#)

An optional DID ID

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<(uint Id, WalletType Type)>

Information about the wallet

CreateNewDAOWallet(DAORules?, ulong?, ulong, ulong, ulong, CancellationToken)

Create a new CAT wallet

```
public Task<(WalletType Type, string TreasuryId, uint WalletId, uint CatWalletId, uint  
DaoCatWalletId)> CreateNewDAOWallet(DAORules? daoRules = null, ulong? amountOfCats = null,  
ulong filterAmount = 1, ulong feeForCat = 0, ulong fee = 0, CancellationToken  
cancellationToken = default)
```

Parameters

daoRules [DAORules](#)

amountOfCats [ulong](#)?

filterAmount [ulong](#)

feeForCat [ulong](#)

Fee (in units of mojos)

fee [ulong](#)

Fee (in units of mojos)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<([WalletType](#) [Type](#), [string](#) [TreasuryId](#), [uint](#) [WalletId](#), [uint](#) [CatWalletId](#), [uint](#) [DaoCatWalletId](#))>

Information about the wallet

CreateNewDL(string, ulong, CancellationToken)

Initialize the new data layer wallets.

```
public Task<(IEnumerable<TransactionRecord> Transactions, string LauncherId)>
CreateNewDL(string root, ulong fee = 0, CancellationToken cancellationToken = default)
```

Parameters

root [string](#)

fee [ulong](#)

Fee (in units of mojos)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<([IEnumerable](#)<[TransactionRecord](#)> [Transactions](#), [string](#) [LauncherId](#))>

An awaitable Task

CreatePoolWallet(PoolState, ulong?, string?, CancellationToken)

Creates a new pool wallet

```
public Task<(TransactionRecord transaction, string launcherId, string p2SingletonHash)>
CreatePoolWallet(PoolState initialTargetState, ulong? p2SingletonDelayTime = null, string?
p2SingletonDelayedPH = null, CancellationToken cancellationToken = default)
```

Parameters

initialTargetState [PoolState](#)

The desired intial state of the wallet

p2SingletonDelayTime [ulong](#)?

Delay time to create the wallet

p2SingletonDelayedPH [string](#)

A delayed address (can be null or empty to not use)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<(TransactionRecord [transaction](#), string [launcherId](#), string [p2SingletonHash](#))>

Information about the wallet

CreateSignedTransaction(IEnumerable<AmountWithPuzzlehash>, IEnumerable<ulong>?, IEnumerable<Coin>?,
IEnumerable<PuzzleAnnouncement>?,
IEnumerable<CoinAnnouncement>?, IEnumerable<Coin>?,
ulong?, ulong?, ulong, CancellationToken)

Creates and signs a transaction.

```
public Task<(TransactionRecord SignedTx, IEnumerable<TransactionRecord> SignedTxs)>
CreateSignedTransaction(IEnumerable<AmountWithPuzzlehash> additions, IEnumerable<ulong>?
excludeCoinAmounts = null, IEnumerable<Coin>? excludeCoins = null,
```

```
IEnumerable<PuzzleAnnouncement>? puzzleAnnouncements = null, IEnumerable<CoinAnnouncement>? coinAnnouncements = null, IEnumerable<Coin>? coins = null, ulong? minCoinAmount = null, ulong? maxCoinAmount = null, ulong fee = 0, CancellationToken cancellationToken = default)
```

Parameters

additions [IEnumerable](#)<[AmountWithPuzzlehash](#)>

excludeCoinAmounts [IEnumerable](#)<[ulong](#)>

excludeCoins [IEnumerable](#)<[Coin](#)>

puzzleAnnouncements [IEnumerable](#)<[PuzzleAnnouncement](#)>

coinAnnouncements [IEnumerable](#)<[CoinAnnouncement](#)>

coins [IEnumerable](#)<[Coin](#)>

minCoinAmount [ulong](#)?

maxCoinAmount [ulong](#)?

fee [ulong](#)

Fee (in units of mojos)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<([TransactionRecord](#) [SignedTx](#), [IEnumerable](#)<[TransactionRecord](#)> [SignedTxs](#))>

The signed [TransactionRecord](#)

CreateWalletForCAT(string, CancellationToken)

Create a wallet for an existing CAT

```
public Task<(WalletType Type, string AssetID, uint WalletId)> CreateWalletForCAT(string assetId, CancellationToken cancellationToken = default)
```

Parameters

assetId [string](#)

The id of the CAT

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <([WalletType](#) [Type](#), [string](#) [AssetId](#), [uint](#) [WalletId](#))>

The wallet type

DeleteAllKeys(CancellationToken)

Deletes all keys from the wallet

```
public Task DeleteAllKeys(CancellationToken cancellationToken = default)
```

Parameters

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

An awaitable [Task](#)

DeleteKey(uint, CancellationToken)

Deletes a specific key from the wallet

```
public Task DeleteKey(uint fingerprint, CancellationToken cancellationToken = default)
```

Parameters

fingerprint [uint](#)

The key's fingerprint

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

An awaitable [Task](#)

DeleteNotifications(IEnumerable<string>, CancellationToken)

Deletes notifications.

```
public Task DeleteNotifications(IEnumerable<string> ids, CancellationToken cancellationToken = default)
```

Parameters

ids [IEnumerable](#)<[string](#)>

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

An awaitable Task

DidFindLostDid(string, CancellationToken)

Recover a missing or unspendable DID wallet by a coin id of the DID.

```
public Task<string> DidFindLostDid(string coinId, CancellationToken cancellationToken = default)
```

Parameters

coinId [string](#)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[string](#)>

[string](#)

DidGetInfo(string, bool, CancellationToken)

Retrieves information about a DID.

```
public Task<DidInfo> DidGetInfo(string coinId, bool latest = true, CancellationToken cancellationToken = default)
```

Parameters

coinId [string](#)

latest [bool](#)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[DidInfo](#)>

An awaitable Task

ExtendDerivationIndex(uint, CancellationToken)

Extends the current derivation index.

```
public Task<uint> ExtendDerivationIndex(uint index, CancellationToken cancellationToken  
= default)
```

Parameters

`index` [uint](#)

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[uint](#)>

[uint](#)

GenerateMnemonic(CancellationToken)

Generates a new mnemonic phrase

```
public Task<IEnumerable<string>> GenerateMnemonic(CancellationToken cancellationToken  
= default)
```

Parameters

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[IEnumerable](#)<[string](#)>>

The new mnemonic as an [IEnumerable<T>](#) of 24 words

GetAutoClaim(CancellationToken)

Get auto claim merkle coins config

```
public Task<AutoClaimSettings> GetAutoClaim(CancellationToken cancellationToken = default)
```

Parameters

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <AutoClaimSettings>

[AutoClaimSettings](#)

GetCoinRecords(UInt32Range?, UInt32Range?, UInt64Range?, AmountFilter?, HashFilter?, HashFilter?, HashFilter?, CoinType?, WalletType?, uint?, uint?, CoinRecordOrder, uint, bool, bool, CancellationToken)

```
public Task<(IEnumerable<CoinRecord> CoinRecords, int? TotalCount)>  
GetCoinRecords(UInt32Range? spentRange = null, UInt32Range? confirmedRange = null,  
UInt64Range? amountRange = null, AmountFilter? amountFilter = null, HashFilter?  
parentCoinIdFilter = null, HashFilter? puzzleHashFilter = null, HashFilter? coinIdFilter =  
null, CoinType? coinType = null, WalletType? walletType = null, uint? walletId = null, uint?  
limit = null, CoinRecordOrder order = CoinRecordOrder.ConfirmedHeight, uint offset = 0,  
bool includeTotalCount = false, bool reverse = false, CancellationToken cancellationToken  
= default)
```

Parameters

spentRange [UInt32Range](#)

confirmedRange [UInt32Range](#)

amountRange [UInt64Range](#)

amountFilter [AmountFilter](#)

parentCoinIdFilter [HashFilter](#)

puzzleHashFilter [HashFilter](#)

coinIdFilter [HashFilter](#)

coinType [CoinType](#)?

walletType [WalletType](#)?

walletId [uint](#)?

limit [uint](#)?

order [CoinRecordOrder](#)

offset [uint](#)

includeTotalCount [bool](#)

reverse [bool](#)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<([IEnumerable](#)<[CoinRecord](#)> [CoinRecords](#), [int](#)? [TotalCount](#))>

[IEnumerable](#)<[T](#)>

GetCoinRecordsByNames(IEnumerable<string>, bool, uint?, uint?, CancellationToken)

Retrieves the coins for given coin IDs

```
public Task<IEnumerable<CoinRecord>> GetCoinRecordsByNames(IEnumerable<string> names, bool
includeSpentCoins, uint? startHeight = null, uint? endHeight = null, CancellationToken
cancellationToken = default)
```

Parameters

`names` [IEnumerable](#)<[string](#)>

The coin names

`includeSpentCoins` [bool](#)

Flag indicating whether to include spent coins or not

`startHeight` [uint](#)?

confirmation start height for search

`endHeight` [uint](#)?

confirmation end height for search

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[IEnumerable](#)<[CoinRecord](#)>>

A list of [CoinRecord](#)s

GetCurrentDerivationIndex(CancellationToken)

Gets the current derivation index.

```
public Task<uint> GetCurrentDerivationIndex(CancellationToken cancellationToken = default)
```

Parameters

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[uint](#)>

[uint](#)

GetFarmedAmount(CancellationToken)

Get the amount farmed

```
public Task<(ulong FarmedAmount, ulong FarmerRewardAmount, ulong FeeAmount, uint LastHeightFarmed, ulong PoolRewardAmount)> GetFarmedAmount(CancellationToken cancellationToken = default)
```

Parameters

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<(FarmedAmount, ulong> FarmerRewardAmount, ulong> FeeAmount, uint> LastHeightFarmed, ulong> PoolRewardAmount)>

The amount farmed

GetHeightInfo(CancellationToken)

Get blockchain height info

```
public Task<uint> GetHeightInfo(CancellationToken cancellationToken = default)
```

Parameters

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<uint>

Current block height

GetLoggedInFingerprint(CancellationToken)

Retrieves the logged in fingerprint

```
public Task<uint?> GetLoggedInFingerprint(CancellationToken cancellationToken = default)
```

Parameters

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[uint](#)?>

The logged in fingerprint

GetNFTByDID(string, CancellationToken)

Get an NFT wallet by DID ID

```
public Task<uint> GetNFTByDID(string didId, CancellationToken cancellationToken = default)
```

Parameters

`didId` [string](#)

The DID ID

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[uint](#)>

The wallet id

GetNFTInfo(string, bool, bool, bool?, CancellationToken)

Get info about an NFT

```
public Task<NFTInfo> GetNFTInfo(string coinId, bool latest = true, bool ignoreSizeLimit = false, bool? reusePuzhash = null, CancellationToken cancellationToken = default)
```

Parameters

coinId [string](#)

latest [bool](#)

Get latest NFT

ignoreSizeLimit [bool](#)

reusePuzhash [bool](#)?

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[NFTInfo](#)>

The wallet id

GetNetworkInfo(CancellationToken)

Retrieves information about the current network

```
public Task<(string NetworkName, string NetworkPrefix)> GetNetworkInfo(CancellationToken cancellationToken = default)
```

Parameters

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <(string [MyDid](#), string [CoinID](#))>

The current network name and prefix

GetPoolInfo(Uri, CancellationToken)

Gets basic info about a pool that is used for pool wallet creation

```
public static Task<PoolInfo> GetPoolInfo(Uri poolUri, CancellationToken cancellationToken  
= default)
```

Parameters

`poolUri` [Uri](#)

The uri of the pool (not including 'pool_info')

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[PoolInfo](#)>

[PoolInfo](#) that can be used to create a pool wallet and join this pool

GetPrivateKey(uint, CancellationToken)

Get the private key accessible by the wallet

```
public Task<PrivateKey> GetPrivateKey(uint fingerprint, CancellationToken cancellationToken  
= default)
```

Parameters

fingerprint [uint](#)

The fingerprint

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[PrivateKey](#)>

The private key for the fingerprint

GetPublicKeys(CancellationToken)

Get all root public keys accessible by the wallet

```
public Task<IEnumerable<uint>> GetPublicKeys(CancellationToken cancellationToken = default)
```

Parameters

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[IEnumerable](#) <[uint](#)>>

All root public keys accessible by the wallet

GetSyncStatus(CancellationToken)

Get the wallet's sync status

```
public Task<(bool GenesisInitialized, bool Synced, bool Syncing)>
GetSyncStatus(CancellationToken cancellationToken = default)
```

Parameters

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<(bool [GenesisInitialized](#), bool [Synced](#), bool [Syncing](#))>

The current sync status

GetTimestampForHeight(uint, CancellationToken)

Retrieve the timestamp for a given block height.

```
public Task<(ulong Timestamp, DateTime DateTimestamp)> GetTimestampForHeight(uint height,  
CancellationToken cancellationToken = default)
```

Parameters

height [uint](#)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<(ulong [Timestamp](#), DateTime [DateTimestamp](#))>

A timestamp

GetTransaction(string, CancellationToken)

Get a specific transaction

```
public Task<TransactionRecord> GetTransaction(string transactionId, CancellationToken  
cancellationToken = default)
```

Parameters

`transactionId` [string](#)

The id of the transaction to find

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[TransactionRecord](#)>

The [TransactionRecord](#)

GetTransactionMemo(string, CancellationToken)

Retrieves the memo from a transaction.

```
public Task<IDictionary<string, IDictionary<string, IEnumerable<string>>>>
GetTransactionMemo(string transactionId, CancellationToken cancellationToken = default)
```

Parameters

`transactionId` [string](#)

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[IDictionary](#) <[string](#), [IDictionary](#) <[string](#), [IEnumerable](#) <[string](#)>>>>

An awaitable Task

GetWalletBalances(IEnumerable<uint>, CancellationToken)

Retrieves the balance of a specific list of wallets.

```
public Task<IDictionary<string, WalletBalance>> GetWalletBalances(IEnumerable<uint>  
walletIds, CancellationToken cancellationToken = default)
```

Parameters

walletIds [IEnumerable](#)<uint>

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<IDictionary<string, WalletBalance>>

A list of [WalletBalance](#)

GetWallets(bool, CancellationToken)

Get the list of wallets

```
public Task<(IEnumerable<WalletInfo> Wallets, uint Fingerprint)> GetWallets(bool includeData  
= true, CancellationToken cancellationToken = default)
```

Parameters

includeData [bool](#)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<(IEnumerable<WalletInfo> Wallets, uint Fingerprint)>

The list of wallets

GetWallets(WalletType, bool, CancellationToken)

Get the list of wallets

```
public Task<IEnumerable<WalletInfo>> GetWallets(WalletType type, bool includeData = true,  
CancellationToken cancellationToken = default)
```

Parameters

type [WalletType](#)

Return only wallets of this type

includeData [bool](#)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[IEnumerable](#)<[WalletInfo](#)>>

The list of wallets

GetWalletsWithIDs(CancellationToken)

Gets all the wallets with DIDs

```
public Task<IEnumerable<(uint WalletId, string DIDId, uint DIDWalletID)>>  
GetWalletsWithIDs(CancellationToken cancellationToken = default)
```

Parameters

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[IEnumerable](#)<([uint](#) [WalletId](#), [string](#) [DIDId](#), [uint](#) [DIDWalletID](#))>>

The list of wallets

LogIn(uint, CancellationToken)

Sets a key to active.

```
public Task<uint> LogIn(uint fingerprint, CancellationToken cancellationToken = default)
```

Parameters

fingerprint [uint](#)

The fingerprint

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[uint](#)>

The key fingerprint

LogInAndWaitForSync(uint, int, CancellationToken)

Sets a fingerprint to active. Waits for the wallet to sync.

```
public Task<uint> LogInAndWaitForSync(uint fingerprint, int millisecondsDelay = 10000,  
CancellationToken cancellationToken = default)
```

Parameters

fingerprint [uint](#)

The fingerprint

millisecondsDelay [int](#)

The number of milliseconds to wait each time before checking sync status

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[uint](#)>

The key fingerprint

NftSetDidBulk(string, IEnumerable<NftCoinInfo>, bool?, ulong, CancellationToken)

Bulk set DID for NFTs across different wallets.

```
public Task<(int TxNum, SpendBundle SpendBundle)> NftSetDidBulk(string didId,
IEnumerable<NftCoinInfo> nftCoinList, bool? reusePuzhash = null, ulong fee = 0,
CancellationToken cancellationToken = default)
```

Parameters

didId [string](#)

nftCoinList [IEnumerable](#)<[NftCoinInfo](#)>

reusePuzhash [bool](#)?

fee [ulong](#)

Fee (in units of mojos)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <([int](#) [TxNum](#), [SpendBundle](#) [SpendBundle](#))>

Transaction number and [SpendBundle](#)

NftTransferBulk(string, IEnumerable<NftCoinInfo>, bool?, ulong, CancellationToken)

Bulk transfer NFTs to an address.

```
public Task<(int TxNum, SpendBundle SpendBundle)> NftTransferBulk(string targetAddress, IEnumerable<NftCoinInfo> nftCoinList, bool? reusePuzhash = null, ulong fee = 0, CancellationToken cancellationToken = default)
```

Parameters

targetAddress [string](#)

nftCoinList [IEnumerable](#)<[NftCoinInfo](#)>

reusePuzhash [bool](#)?

fee [ulong](#)

Fee (in units of mojos)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<(int [TxNum](#), [SpendBundle](#) [SpendBundle](#))>

Transaction number and a [SpendBundle](#)

PushTransactions(IEnumerable<TransactionRecord>, CancellationToken)

Pushes a list of transactions to the mempool and blockchain.

```
public Task PushTransactions(IEnumerable<TransactionRecord> transactions, CancellationToken cancellationToken = default)
```

Parameters

transactions [IEnumerable](#)<[TransactionRecord](#)>

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

An awaitable task

PushTx(SpendBundle, CancellationToken)

Pushes a transaction / spend bundle to the mempool and blockchain. Returns whether the spend bundle was successfully included into the mempool

```
public Task<bool> PushTx(SpendBundle spendBundle, CancellationToken cancellationToken  
= default)
```

Parameters

`spendBundle` [SpendBundle](#)

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <bool>

Indicator of whether the spend bundle was successfully included in the mempool

RecoverDIDWallet(string, CancellationToken)

```
public Task<(WalletType Type, string myDID, uint walletId, string coinName, Coin coin,  
string newPuzHash, string pubkey, IEnumerable<byte> backupIDs, ulong  
numVerificationsRequired)> RecoverDIDWallet(string backupData, CancellationToken  
cancellationToken = default)
```

Parameters

`backupData` [string](#)

`cancellationToken` [CancellationToken](#)

Returns

[Task](#) <(WalletType Type, string myDID, uint walletId, string coinName, Coin coin, string newPuzHash, string pubkey, IEnumerable<byte> backupDIDs, ulong numVerifications Required)>

Exceptions

[ArgumentNullException](#)

SendNotification(ulong, string, string, ulong, CancellationToken)

Sends a notification.

```
public Task<TransactionRecord> SendNotification(ulong amount, string message, string target, ulong fee = 0, CancellationToken cancellationToken = default)
```

Parameters

amount [ulong](#)

message [string](#)

In hex

target [string](#)

fee [ulong](#)

Fee (in units of mojos)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[TransactionRecord](#)>

[TransactionRecord](#)

SetAutoClaim(bool, CancellationToken)

Set auto claim merkle coins config

```
public Task<AutoClaimSettings> SetAutoClaim(bool enabled, CancellationToken cancellationToken = default)
```

Parameters

enabled [bool](#)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<[AutoClaimSettings](#)>

[AutoClaimSettings](#)

SetAutoClaim(bool, ushort, ulong, ulong, CancellationToken)

Set auto claim merkle coins config

```
public Task<AutoClaimSettings> SetAutoClaim(bool enabled = true, ushort batchSize = 50, ulong minAmount = 0, ulong txFee = 0, CancellationToken cancellationToken = default)
```

Parameters

enabled [bool](#)

batchSize [ushort](#)

minAmount [ulong](#)

txFee [ulong](#)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <AutoClaimSettings>

[AutoClaimSettings](#)

SetWalletResyncOnStartup(bool, CancellationToken)

Resync the current logged in wallet. The transaction and offer records will be kept.

```
public Task SetWalletResyncOnStartup(bool enable = true, CancellationToken cancellationToken = default)
```

Parameters

enable [bool](#)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

An awaitable Task

SignMessageByAddress(string, string, bool, CancellationToken)

Given a derived P2 address, sign the message by its private key.

```
public Task<(string PubKey, string Signature, string SigningMode)>
SignMessageByAddress(string message, string address, bool isHex = false, CancellationToken cancellationToken = default)
```

Parameters

message [string](#)

address [string](#)

isHex [bool](#)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<([string](#) [PubKey](#), [string](#) [Signature](#), [string](#) [SigningMode](#))>

PubKey, Signature, and SigningMode

SignMessageById(string, string, bool, CancellationToken)

Given a NFT/DID ID, sign the message by the P2 private key.

```
public Task<(string PubKey, string Signature, string SigningMode, string LatestCoinId)>
SignMessageById(string message, string id, bool isHex = false, CancellationToken
cancellationToken = default)
```

Parameters

message [string](#)

id [string](#)

isHex [bool](#)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<([string](#) [PubKey](#), [string](#) [Signature](#), [string](#) [SigningMode](#), [string](#) [LatestCoinId](#))>

PubKey, Signature, and SigningMode

SpendClawbackCoins(IEnumerable<string>, ushort, ulong, CancellationToken)

Spend clawback coins that were sent (to claw them back) or received (to claim them).

```
public Task<IEnumerable<string>> SpendClawbackCoins(IEnumerable<string> coinIds, ushort  
batchSize = 50, ulong fee = 0, CancellationToken cancellationToken = default)
```

Parameters

coinIds [IEnumerable<string>](#)

batchSize [ushort](#)

fee [ulong](#)

Fee (in units of mojos)

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task<IEnumerable<string>>](#)

A list of [string](#)

VerifySignature(string, string, string, string?, string?, CancellationToken)

Given a public key, message and signature, verify if it is valid.

```
public Task<bool> VerifySignature(string signature, string message, string pubkey, string?  
address = null, string? signingMode = null, CancellationToken cancellationToken = default)
```

Parameters

signature [string](#)

message [string](#)

pubkey [string](#)

`address` [string](#)

`signingMode` [string](#)

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#) <[bool](#)>

[bool](#)

WaitForSync(int, CancellationToken)

Will wait until [GetSyncStatus\(CancellationToken\)](#) indicates that the wallet has synced or until the cancellation token is canceled

```
public Task WaitForSync(int millisecondsDelay = 10000, CancellationToken cancellationToken  
= default)
```

Parameters

`millisecondsDelay` [int](#)

The number of milliseconds to wait each time before checking sync status

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

An awaitable [Task](#)

Exceptions

[TaskCanceledException](#)

When cancellation token expires or is cancelled

Enum WalletType

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

Wallet Types

```
public enum WalletType : byte
```

Fields

ATOMIC_SWAP = 2

AUTHORIZED_PAYEE = 3

CAT = 6

CRCAT = 57

CUSTODY = 5

DAO = 14

DAO_CAT = 15

DATA_LAYER = 11

DATA_LAYER_OFFER = 12

DISTRIBUTED_ID = 8

MULTI_SIG = 4

NFT = 10

POOLING_WALLET = 9

RECOVERABLE = 7

STANDARD_WALLET = 0

VC = 13

Class WebSocketRpcClient

Namespace: [chia.dotnet](#)

Assembly: chia-dotnet.dll

Class that handles core communication with the rpc endpoint using a websocket (wss). Only the daemon endpoint supports websockets, but it can proxy communication to other services. [Destination](#)

```
public class WebSocketRpcClient : IRpcClient, IDisposable
```

Inheritance

[object](#) ← WebSocketRpcClient

Implements

[IRpcClient](#), [IDisposable](#)

Inherited Members

[object.Equals\(object\)](#), [object.Equals\(object, object\)](#), [object.GetHashCode\(\)](#), [object.GetType\(\)](#),
[object.MemberwiseClone\(\)](#), [object.ReferenceEquals\(object, object\)](#), [object.ToString\(\)](#)

Constructors

WebSocketRpcClient(EndpointInfo)

ctor

```
public WebSocketRpcClient(EndpointInfo endpoint)
```

Parameters

endpoint [EndpointInfo](#)

Details of the websocket endpoint

Properties

Endpoint

Details of the RPC service endpoint

```
public EndpointInfo Endpoint { get; init; }
```

Property Value

[EndpointInfo](#)

Class that handles core communication with the rpc endpoint using a websocket (wss). Only the daemon endpoint supports websockets, but it can proxy communication to other services.

Methods

Close(CancellationToken)

Cancels the receive loop and closes the websocket

```
public Task Close(CancellationToken cancellationToken = default)
```

Parameters

[cancellationToken CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

Awaitable [Task](#)

Connect(CancellationToken)

Opens the websocket and starts the receive loop

```
public Task Connect(CancellationToken cancellationToken = default)
```

Parameters

`cancellationToken` [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

An awaitable [Task](#)

Dispose()

[Dispose\(\)](#)

```
public void Dispose()
```

Dispose(bool)

Called when the instance is being disposed or finalized

```
protected virtual void Dispose(bool disposing)
```

Parameters

`disposing` [bool](#)

Invoke from [Dispose\(\)](#)

OnBroadcastMessageReceived(Message)

Raises the [BroadcastMessageReceived](#) event

```
protected virtual void OnBroadcastMessageReceived(Message message)
```

Parameters

`message` [Message](#)

The message to broadcast

OnConnected()

Called after [Connect\(CancellationToken\)](#) completes successfully. Lets derived classes know that they can do post connection initialization

```
protected virtual void OnConnected()
```

PostMessage(Message, CancellationToken)

Posts a [Message](#) to the websocket but does not wait for a response

```
public virtual Task PostMessage(Message message, CancellationToken cancellationToken  
= default)
```

Parameters

message [Message](#)

The message to post

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)

Awaitable [Task](#)

Remarks

Awaiting this method waits for the message to be sent only. It doesn't await a response.

SendMessage(Message, CancellationToken)

Sends a [Message](#) to the endpoint and waits for a response

```
public Task<dynamic> SendMessage(Message message, CancellationToken cancellationToken  
= default)
```

Parameters

message [Message](#)

The message to send

cancellationToken [CancellationToken](#)

A token to allow the call to be cancelled

Returns

[Task](#)<dynamic>

The response message

Remarks

Awaiting this method will block until a response is received from the [WebSocket](#) or the A token to allow the call to be cancelled is cancelled

Exceptions

[ResponseException](#)

Throws when [IsSuccessfulResponse](#) is False

Events

BroadcastMessageReceived

Event raised when a message is received from the endpoint that was either not in response to a request or was a response from a posted message (i.e. we didn't register to receive the response) Pooling state_changed messages come through this event

```
public event EventHandler<Message>? BroadcastMessageReceived
```

Event Type

[EventHandler](#) <Message>

Class that handles core communication with the rpc endpoint using a websocket (wss). Only the daemon endpoint supports websockets, but it can proxy communication to other services.

Namespace chia.dotnet.bech32

Classes

[Bech32M](#)

Bech32M implementation for encoding addresses

Structs

[HexBytes](#)

Utility to perform operations on an array of bytes and represent them as a Hex string

Class Bech32M

Namespace: [chia.dotnet.bech32](#)

Assembly: chia-dotnet.dll

Bech32M implementation for encoding addresses

```
public class Bech32M
```

Inheritance

[object](#) ← Bech32M

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

Remarks

adapted from <https://github.com/Playwo/ChiaRPC.Net/blob/master/ChiaRPC.Net/Utils/Bech32M.cs>

Constructors

Bech32M(string)

Bech32M implementation for encoding addresses

```
public Bech32M(string prefix = "xch")
```

Parameters

prefix [string](#)

Address prefix

Remarks

adapted from <https://github.com/Playwo/ChiaRPC.Net/blob/master/ChiaRPC.Net/Utils/Bech32M.cs>

Properties

AddressPrefix

Address prefix to use

```
public string AddressPrefix { get; init; }
```

Property Value

[string](#)

xch

Methods

AddressToPuzzleHash(string)

Converts an address to a puzzle hash

```
public static HexBytes AddressToPuzzleHash(string address)
```

Parameters

[address](#) [string](#)

The address to convert

Returns

[HexBytes](#)

The puzzle hash

Exceptions

[ArgumentException](#)

Thrown when [address](#) is not valid

AddressToPuzzleHashString(string)

Converts an address to a puzzle hash

```
public static string AddressToPuzzleHashString(string address)
```

Parameters

address [string](#)

The address to convert

Returns

[string](#)

The puzzle hash

Exceptions

[ArgumentException](#)

Thrown when **address** is not valid

PuzzleHashToAddress(string)

Converts a puzzle hash to an address using [AddressPrefix](#)

```
public string PuzzleHashToAddress(string puzzleHash)
```

Parameters

puzzleHash [string](#)

Returns

[string](#)

The puzzle hash

PuzzleHashToAddress(HexBytes)

Converts a puzzle hash to an address using [AddressPrefix](#)

```
public string PuzzleHashToAddress(HexBytes puzzleHash)
```

Parameters

puzzleHash [HexBytes](#)

Returns

[string](#) ↗

The puzzle hash

Struct HexBytes

Namespace: [chia.dotnet.bech32](#)

Assembly: chia-dotnet.dll

Utility to perform operations on an array of bytes and represent them as a Hex string

```
public readonly struct HexBytes
```

Inherited Members

[object.Equals\(object, object\)](#) , [object.GetType\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

Remarks

adapted from <https://github.com/Playwo/ChiaRPC.Net/blob/master/ChiaRPC.Net/Utils/Bech32M.cs>

Constructors

HexBytes(string, byte[])

Utility to perform operations on an array of bytes and represent them as a Hex string

```
public HexBytes(string hex, byte[] bytes)
```

Parameters

hex [string](#)

bytes [byte](#)[]

Remarks

adapted from <https://github.com/Playwo/ChiaRPC.Net/blob/master/ChiaRPC.Net/Utils/Bech32M.cs>

Properties

Bytes

The array of bytes

```
public byte[] Bytes { get; init; }
```

Property Value

[byte](#)[]

Utility to perform operations on an array of bytes and represent them as a Hex string

Empty

```
public static HexBytes Empty { get; }
```

Property Value

[HexBytes](#)

Utility to perform operations on an array of bytes and represent them as a Hex string

Hex

[Bytes](#) hex string representation

```
public string Hex { get; init; }
```

Property Value

[string](#)[]

Utility to perform operations on an array of bytes and represent them as a Hex string

IsEmpty

Flag indicating that the array is empty

```
public bool IsEmpty { get; }
```

Property Value

[bool](#)

Utility to perform operations on an array of bytes and represent them as a Hex string

Methods

Equals(object?)

Indicates whether this instance and a specified object are equal.

```
public override bool Equals(object? obj)
```

Parameters

[obj](#) [object](#)

The object to compare with the current instance.

Returns

[bool](#)

[true](#) if [obj](#) and this instance are the same type and represent the same value; otherwise, [false](#).

FromBytes(byte[])

```
public static HexBytes FromBytes(byte[] bytes)
```

Parameters

[bytes](#) [byte](#)[]

Utility to perform operations on an array of bytes and represent them as a Hex string

Returns

[HexBytes](#)

Utility to perform operations on an array of bytes and represent them as a Hex string

FromHex(string)

```
public static HexBytes FromHex(string hex)
```

Parameters

hex [string](#)

Utility to perform operations on an array of bytes and represent them as a Hex string

Returns

[HexBytes](#)

Utility to perform operations on an array of bytes and represent them as a Hex string

GetHashCode()

Returns the hash code for this instance.

```
public override int GetHashCode()
```

Returns

[int](#)

A 32-bit signed integer that is the hash code for this instance.

Sha256()

SHA256 encoded copy

```
public HexBytes Sha256()
```

Returns

[HexBytes](#)

Encoded copy

ToString()

Returns the fully qualified type name of this instance.

```
public override string ToString()
```

Returns

[string](#)

The fully qualified type name.

TryFromHex(string, out HexBytes)

```
public static bool TryFromHex(string hex, out HexBytes result)
```

Parameters

hex [string](#)

Utility to perform operations on an array of bytes and represent them as a Hex string

result [HexBytes](#)

Utility to perform operations on an array of bytes and represent them as a Hex string

Returns

[bool](#)

Utility to perform operations on an array of bytes and represent them as a Hex string

Operators

operator +(HexBytes, byte[])

```
public static HexBytes operator +(HexBytes a, byte[] b)
```

Parameters

a [HexBytes](#)

Utility to perform operations on an array of bytes and represent them as a Hex string

b [byte\[\]](#)

Utility to perform operations on an array of bytes and represent them as a Hex string

Returns

[HexBytes](#)

Utility to perform operations on an array of bytes and represent them as a Hex string

operator +(HexBytes, string)

```
public static HexBytes operator +(HexBytes a, string b)
```

Parameters

a [HexBytes](#)

Utility to perform operations on an array of bytes and represent them as a Hex string

b [string](#)

Utility to perform operations on an array of bytes and represent them as a Hex string

Returns

[HexBytes](#)

Utility to perform operations on an array of bytes and represent them as a Hex string

operator +(HexBytes, HexBytes)

```
public static HexBytes operator +(HexBytes a, HexBytes b)
```

Parameters

a [HexBytes](#)

Utility to perform operations on an array of bytes and represent them as a Hex string

b [HexBytes](#)

Utility to perform operations on an array of bytes and represent them as a Hex string

Returns

[HexBytes](#)

Utility to perform operations on an array of bytes and represent them as a Hex string

operator ==(HexBytes, HexBytes)

```
public static bool operator ==(HexBytes a, HexBytes b)
```

Parameters

a [HexBytes](#)

Utility to perform operations on an array of bytes and represent them as a Hex string

b [HexBytes](#)

Utility to perform operations on an array of bytes and represent them as a Hex string

Returns

[bool](#)

Utility to perform operations on an array of bytes and represent them as a Hex string

operator !=(HexBytes, HexBytes)

```
public static bool operator !=(HexBytes a, HexBytes b)
```

Parameters

a [HexBytes](#)

Utility to perform operations on an array of bytes and represent them as a Hex string

b [HexBytes](#)

Utility to perform operations on an array of bytes and represent them as a Hex string

Returns

[bool](#)

Utility to perform operations on an array of bytes and represent them as a Hex string