

# Namespace chia.dotnet.wallet

## Classes

### [AssetCoin](#)

Represents an asset coin, which is a coin that carries an asset ID and lineage proof.

### [AssetToken<T>](#)

Represents a CAT

### [AssetWallet](#)

Represents an asset wallet that manages asset tokens.

### [KeyPair](#)

Represents a key pair consisting of a public key and an optional private key.

### [KeyStore](#)

Represents a key store that holds private and public keys.

### [SpendableAssetCoin](#)

Represents a spendable asset coin.

### [StandardTransaction](#)

Represents a standard transaction in the Chia.NET wallet.

### [StandardWallet](#)

Represents a standard wallet in the Chia.NET Wallet library.

### [Tails](#)

Represents a collection of static methods for creating puzzle programs.

### [WalletOptions](#)

Represents the options for a wallet.

### [Wallet<T>](#)

Represents an abstract wallet class that provides common functionality for different types of wallets.

## Enums

### [CoinSelection](#)

Represents the different strategies for selecting coins.

# Class AssetCoin


Namespace: [chia.dotnet.wallet](#)

Assembly: chia-dotnet-wallet.dll

Represents an asset coin, which is a coin that carries an asset ID and lineage proof.

```
public class AssetCoin
```








## Inheritance

[object](#)  ← AssetCoin

## Derived

[SpendableAssetCoin](#)

## Inherited Members

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

## Constructors

### AssetCoin(CoinSpend, Coin, byte[]?)

Initializes a new instance of the [AssetCoin](#) class.

```
public AssetCoin(CoinSpend parentCoinSpend, Coin coin, byte[]? assetId = null)
```

## Parameters

**parentCoinSpend** CoinSpend

The parent coin spend.

**coin** Coin

The coin.

**assetId** [byte](#)  []

The asset ID.

# Properties

## AssetId

Gets the asset ID of this asset coin.

```
public byte[] AssetId { get; init; }
```

## Property Value

[byte\[\]](#)

Represents an asset coin, which is a coin that carries an asset ID and lineage proof.

## Coin

Gets the underlying coin of this asset coin.

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

## Property Value

Coin

Represents an asset coin, which is a coin that carries an asset ID and lineage proof.

## LineageProof

Gets the lineage proof of this asset coin.

```
public Program LineageProof { get; init; }
```

## Property Value

Program

Represents an asset coin, which is a coin that carries an asset ID and lineage proof.

# ParentCoinSpend

Gets the parent coin spend associated with this asset coin.

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

## Property Value

### CoinSpend

Represents an asset coin, which is a coin that carries an asset ID and lineage proof.

# Class AssetToken<T>

Namespace: [chia.dotnet.wallet](#)

Assembly: chia-dotnet-wallet.dll

Represents a CAT


```
public class AssetToken<T> : Program where T : Program
```

## Type Parameters



















**T**

A Program

## Inheritance

[object](#)  ← Program ← AssetToken<T>

## Inherited Members

Program.True , Program.False , Program.Nil , Program.FromCons(Program, Program) ,  
[Program.FromBytes\(byte\[\]\)](#)  , Program.FromJacobianPoint(JacobianPoint) ,  
Program.FromPrivateKey(PrivateKey) , [Program.FromHex\(string\)](#)  , [Program.FromBool\(bool\)](#)  ,  
[Program.FromInt\(long\)](#)  , [Program.FromBigInt\(BigInteger\)](#)  , [Program.FromText\(string\)](#)  ,  
[Program.FromSource\(string\)](#)  , [Program.FromList\(IEnumerable<Program>\)](#)  ,  
[Program.Deserialize\(byte\[\]\)](#)  , [Program.DeserializeHex\(string\)](#)  , Program.At(Position) ,  
[Program.Curry\(IEnumerable<Program>\)](#)  , Program.Uncurry() , Program.Hash() , Program.HashHex() ,  
Program.Define(Program) , [Program.DefineAll\(IEnumerable<Program>\)](#)  ,  
Program.Compile(CompileOptions) , Program.Run(Program, RunOptions) , Program.ToBytes() ,  
Program.ToJacobianPoint() , Program.ToPrivateKey() , Program.ToHex() , Program.ToBool() ,  
Program.ToInt() , Program.ToBigInt() , Program.ToText() , [Program.ToSource\(bool\)](#)  ,  
[Program.ToList\(bool\)](#)  , Program.Serialize() , Program.SerializeHex() , Program.Equals(Program) ,  
Program.ToString() , Program.Value , Program.IsAtom , Program.IsCons , Program.IsNull , Program.Atom ,  
Program.Cons , Program.First , Program.Rest , Program.PositionSuffix , Program.Position ,  
[object.Equals\(object\)](#)  , [object.Equals\(object, object\)](#)  , [object.GetHashCode\(\)](#)  , [object.GetType\(\)](#)  ,  
[object.MemberwiseClone\(\)](#)  , [object.ReferenceEquals\(object, object\)](#) 

## Constructors

# AssetToken(byte[], T)

Represents a CAT

```
public AssetToken(byte[] assetId, T innerPuzzle)
```

## Parameters

**assetId** [byte\[\]](#)

The asset id

**innerPuzzle** T

The inner puzzle program

## Properties

### AssetId

The Token's asset id

```
public byte[] AssetId { get; init; }
```

### Property Value

[byte\[\]](#)

Represents a CAT

### InnerPuzzle

The tokens inner puzzle Program

```
public T InnerPuzzle { get; init; }
```

### Property Value

T

Represents a CAT

# Class AssetWallet

Namespace: [chia.dotnet.wallet](#)

Assembly: chia-dotnet-wallet.dll








Represents an asset wallet that manages asset tokens.

```
public class AssetWallet : Wallet<AssetToken<StandardTransaction>>
```

## Inheritance

[object](#)  ← [Wallet](#) [<AssetToken<StandardTransaction>>](#) > ← AssetWallet

## Inherited Members

[Wallet<AssetToken<StandardTransaction>>.Node](#) ,  
[Wallet<AssetToken<StandardTransaction>>.KeyStore](#) ,  
[Wallet<AssetToken<StandardTransaction>>.Options](#) ,  
[Wallet<AssetToken<StandardTransaction>>.CoinRecords](#) ,  
[Wallet<AssetToken<StandardTransaction>>.ArtificialCoinRecords](#) ,  
[Wallet<AssetToken<StandardTransaction>>.PuzzleCache](#) ,  
[Wallet<AssetToken<StandardTransaction>>.CoinRecordIndex\(CoinRecord\)](#) ,  
[Wallet<AssetToken<StandardTransaction>>.Sync\(WalletOptions, CancellationToken\)](#) ,  
[Wallet<AssetToken<StandardTransaction>>.FetchCoinRecords\(CancellationToken\)](#) ,  
[Wallet<AssetToken<StandardTransaction>>.ClearUnconfirmedTransactions\(CancellationToken\)](#) ,  
[Wallet<AssetToken<StandardTransaction>>.CreateSpend\(\)](#) ,  
[Wallet<AssetToken<StandardTransaction>>.FindUnusedIndices\(int, List<int>, bool, CancellationToken\)](#) ,  
[Wallet<AssetToken<StandardTransaction>>.GetBalance\(\)](#) ,  
[Wallet<AssetToken<StandardTransaction>>.SelectCoinRecords\(BigInteger, CoinSelection, int, bool\)](#) ,  
[Wallet<AssetToken<StandardTransaction>>.CompleteSpend\(SpendBundle, CancellationToken\)](#) ,  
[object.Equals\(object\)](#)  , [object.Equals\(object, object\)](#)  , [object.GetHashCode\(\)](#)  , [object.GetType\(\)](#)  ,  
[object.MemberwiseClone\(\)](#)  , [object.ReferenceEquals\(object, object\)](#)  , [object.ToString\(\)](#) 

## Constructors

AssetWallet(FullNodeProxy, KeyStore, byte[], byte[]?,  
WalletOptions?)

Initializes a new instance of the [AssetWallet](#) class.



```
public AssetWallet(FullNodeProxy node, KeyStore keyStore, byte[] assetId, byte[]?
hiddenPuzzleHash = null, WalletOptions? walletOptions = null)
```

## Parameters

**node** FullNodeProxy

The full node proxy.

**keyStore** [KeyStore](#)

The key store.

**assetId** [byte](#)[]

The asset ID.

**hiddenPuzzleHash** [byte](#)[]

The hidden puzzle hash.

**walletOptions** [WalletOptions](#)

The wallet options.

## Properties

### AssetId

Gets the asset ID.

```
public byte[] AssetId { get; init; }
```

### Property Value

[byte](#)[]

Represents an asset wallet that manages asset tokens.

### HiddenPuzzleHash

Gets the hidden puzzle hash.

```
public byte[] HiddenPuzzleHash { get; init; }
```

Property Value

[byte](#)[]

Represents an asset wallet that manages asset tokens.

## Methods

### CreatePuzzle(KeyPair)

Creates a puzzle for the specified key pair.

```
public override AssetToken<StandardTransaction> CreatePuzzle(KeyPair keyPair)
```

Parameters

**keyPair** [KeyPair](#)

The key pair.

Returns

[AssetToken](#) <[StandardTransaction](#)>

The created asset token puzzle.

### FindTail(CancellationToken)

Finds the tail of the asset token.

```
public Task<Program?> FindTail(Cancellation token = default)
```

Parameters

`cancellationToken` [CancellationToken](#)

The cancellation token.

Returns

[Task](#) <Program>

The tail of the asset token.

## GetParentCoinSpend(CoinRecord, CancellationToken)

Gets the parent coin spend for the specified coin record.

```
public Task<CoinSpend> GetParentCoinSpend(CoinRecord coinRecord, CancellationTokencancellationToken = default)
```

Parameters

`coinRecord` CoinRecord

The coin record.

`cancellationToken` [CancellationToken](#)

The cancellation token.

Returns

[Task](#) <CoinSpend>

The parent coin spend.

## SignSpend(SpendBundle, byte[])

Signs the spend bundle with the private keys in the key store.

```
public override SpendBundle SignSpend(SpendBundle spendBundle, byte[] aggSigMeExtraData)
```

Parameters

`spendBundle` SpendBundle

The spend bundle to sign.

`aggSigMeExtraData` [byte](#)[]

The aggregated signature me extra data.

Returns

SpendBundle

The signed spend bundle.

# Enum CoinSelection

Namespace: [chia.dotnet.wallet](#)

Assembly: chia-dotnet-wallet.dll

Represents the different strategies for selecting coins.

```
public enum CoinSelection
```

## Fields

**Largest = 1**

Select the largest coins first.

**Newest = 2**

Select the newest coins first.

**Oldest = 3**

Select the oldest coins first.

**Smallest = 0**

Select the smallest coins first.

# Class KeyPair

Namespace: [chia.dotnet.wallet](#)

Assembly: chia-dotnet-wallet.dll

Represents a key pair consisting of a public key and an optional private key.

```
public record KeyPair : IEquatable<KeyPair>
```








## Inheritance

[object](#)  ← KeyPair

## Implements

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

## Inherited Members

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

# Constructors

## KeyPair(JacobianPoint, PrivateKey?)

Initializes a new instance of the [KeyPair](#) class.

```
public KeyPair(JacobianPoint publicKey, PrivateKey? privateKey = null)
```

## Parameters

**publicKey** JacobianPoint

The public key.

**privateKey** PrivateKey

The private key. Can be null.

# Properties

# PrivateKey

Gets or sets the private key. Can be null if the private key is not available.

```
public PrivateKey? PrivateKey { get; init; }
```

## Property Value

PrivateKey

Represents a key pair consisting of a public key and an optional private key.

# PublicKey

Gets or sets the public key.

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

## Property Value

JacobianPoint

Represents a key pair consisting of a public key and an optional private key.

# Class KeyStore

Namespace: [chia.dotnet.wallet](#)

Assembly: chia-dotnet-wallet.dll

Represents a key store that holds private and public keys.

```
public class KeyStore
```

## Inheritance

[object](#)  ← KeyStore

## Inherited Members

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

# Constructors

## KeyStore(object, bool)

Initializes a new instance of the [KeyStore](#) class.

```
public KeyStore(object key, bool hardened = false)
```

## Parameters

key [object](#) 

The key to initialize the key store with.

hardened [bool](#) 

A value indicating whether the key store is hardened.

## Exceptions

[ArgumentException](#) 

Thrown when the key is neither a PrivateKey nor a JacobianPoint.



# Properties

## Hardened

Gets or sets a value indicating whether the key store is hardened.

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

## Property Value

[bool](#)

Represents a key store that holds private and public keys.

## Keys

Gets or sets the list of key pairs.

```
public List<KeyPair> Keys { get; init; }
```

## Property Value

[List](#) [KeyPair](#)

Represents a key store that holds private and public keys.

## PrivateKey

Gets or sets the private key.

```
public PrivateKey? PrivateKey { get; init; }
```

## Property Value

PrivateKey

Represents a key store that holds private and public keys.

# PublicKey

Gets or sets the public key.

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

## Property Value

JacobianPoint

Represents a key store that holds private and public keys.

## Methods

### Generate(int)

Generates the specified number of key pairs and adds them to the key store.

```
public void Generate(int amount)
```

#### Parameters

amount [int](#)

The number of key pairs to generate.

### GenerateUntil(int)

Generates key pairs until the specified number of key pairs is reached.

```
public void GenerateUntil(int amount)
```

#### Parameters

amount [int](#)

The number of key pairs to generate.



# Class SpendableAssetCoin


Namespace: [chia.dotnet.wallet](#)

Assembly: chia-dotnet-wallet.dll








Represents a spendable asset coin.

```
public class SpendableAssetCoin : AssetCoin
```

## Inheritance

[object](#)  ← [AssetCoin](#) ← SpendableAssetCoin

## Inherited Members

[AssetCoin.ParentCoinSpend](#) , [AssetCoin.AssetId](#) , [AssetCoin.LineageProof](#) , [AssetCoin.Coin](#) , [object.Equals\(object\)](#)  , [object.Equals\(object, object\)](#)  , [object.GetHashCode\(\)](#)  , [object.GetType\(\)](#)  , [object.MemberwiseClone\(\)](#)  , [object.ReferenceEquals\(object, object\)](#)  , [object.ToString\(\)](#) 

## Constructors

SpendableAssetCoin(CoinSpend, Coin, Program, Program, int, byte[]?)

Initializes a new instance of the [SpendableAssetCoin](#) class.

```
public SpendableAssetCoin(CoinSpend parentCoinSpend, Coin coin, Program innerPuzzle, Program innerSolution, int extraDelta = 0, byte[]? assetId = null)
```

## Parameters

**parentCoinSpend** CoinSpend

The parent coin spend.

**coin** Coin

The coin.

**innerPuzzle** Program

The inner puzzle program.

`innerSolution` Program

The inner solution program.

`extraDelta` [int](#)

The extra delta value.

`assetId` [byte](#) []

The asset ID.

## Properties

### ExtraDelta

Gets or sets the extra delta value.

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

### Property Value

[int](#)

Represents a spendable asset coin.

### InnerPuzzle

Gets or sets the inner puzzle program.

```
public Program InnerPuzzle { get; init; }
```

### Property Value

Program

Represents a spendable asset coin.

# InnerSolution

Gets or sets the inner solution program.

```
public Program InnerSolution { get; init; }
```

## Property Value

Program

Represents a spendable asset coin.

# Puzzle

Gets or sets the puzzle program.

```
public Program Puzzle { get; init; }
```

## Property Value

Program

Represents a spendable asset coin.

# Methods

## CalculateDeltas(List<SpendableAssetCoin>)

Calculates the deltas based on the given spendable asset coins.

```
public static long[] CalculateDeltas(List<SpendableAssetCoin> spendableAssetCoins)
```

## Parameters

**spendableAssetCoins** [List](#) [<SpendableAssetCoin>](#)

The spendable asset coins.

## Returns

[long](#)[]

An array of deltas.

## CalculateSubtotals(long[])

Calculates the subtotals based on the given deltas.

```
public static long[] CalculateSubtotals(long[] deltas)
```

## Parameters

deltas [long](#)[]

The deltas.

## Returns

[long](#)[]

An array of subtotals.

# Class StandardTransaction

Namespace: [chia.dotnet.wallet](#)

Assembly: chia-dotnet-wallet.dll

Represents a standard transaction in the Chia.NET wallet.

```
public class StandardTransaction : Program
```

## Inheritance

[object](#) ← Program ← StandardTransaction

## Inherited Members

Program.True , Program.False , Program.Nil , Program.FromCons(Program, Program) ,  
[Program.FromBytes\(byte\[\]\)](#) , Program.FromJacobianPoint(JacobianPoint) ,  
Program.FromPrivateKey(PrivateKey) , [Program.FromHex\(string\)](#) , [Program.FromBool\(bool\)](#) ,  
[Program.FromInt\(long\)](#) , [Program.FromBigInt\(BigInteger\)](#) , [Program.FromText\(string\)](#) ,  
[Program.FromSource\(string\)](#) , [Program.FromList\(IEnumerable<Program>\)](#) ,  
[Program.Deserialize\(byte\[\]\)](#) , [Program.DeserializeHex\(string\)](#) , Program.At(Position) ,  
[Program.Curry\(IEnumerable<Program>\)](#) , Program.Uncurry() , Program.Hash() , Program.HashHex() ,  
Program.Define(Program) , [Program.DefineAll\(IEnumerable<Program>\)](#) ,  
Program.Compile(CompileOptions) , Program.Run(Program, RunOptions) , Program.ToBytes() ,  
Program.ToJacobianPoint() , Program.ToPrivateKey() , Program.ToHex() , Program.ToBool() ,  
Program.ToInt() , Program.ToBigInt() , Program.ToText() , [Program.ToSource\(bool\)](#) ,  
[Program.ToList\(bool\)](#) , Program.Serialize() , Program.SerializeHex() , Program.Equals(Program) ,  
Program.ToString() , Program.Value , Program.IsAtom , Program.IsCons , Program.IsNull , Program.Atom ,  
Program.Cons , Program.First , Program.Rest , Program.PositionSuffix , Program.Position ,  
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

## Constructors

### StandardTransaction(JacobianPoint)

Initializes a new instance of the [StandardTransaction](#) class with the specified synthetic public key.

```
public StandardTransaction(JacobianPoint syntheticPublicKey)
```



## Parameters

**syntheticPublicKey** `JacobianPoint`

The synthetic public key to use.

## Properties

### SyntheticPublicKey

Gets or sets the synthetic public key associated with the transaction.

```
public JacobianPoint SyntheticPublicKey { get; init; }
```

## Property Value

`JacobianPoint`

Represents a standard transaction in the Chia.NET wallet.

## Methods

### GetSolution(List<Program>)

Gets the solution program for the specified conditions.

```
public static Program GetSolution(List<Program> conditions)
```

## Parameters

**conditions** [List](#)<`Program`>

The conditions to use in the solution.

## Returns

`Program`

The solution program.

# Spend(Coin, Program)

Creates a coin spend using the specified coin and solution program.

```
public CoinSpend Spend(Coin coin, Program solution)
```

## Parameters

**coin** Coin

The coin to spend.

**solution** Program

The solution program to use.

## Returns

CoinSpend

The created coin spend.

# Class StandardWallet

Namespace: [chia.dotnet.wallet](#)

Assembly: chia-dotnet-wallet.dll








Represents a standard wallet in the Chia.NET Wallet library.

```
public class StandardWallet : Wallet<StandardTransaction>
```

## Inheritance

[object](#)  ← [Wallet](#)  <[StandardTransaction](#)> ← StandardWallet

## Inherited Members

[Wallet<StandardTransaction>.Node](#) , [Wallet<StandardTransaction>.KeyStore](#) ,  
[Wallet<StandardTransaction>.Options](#) , [Wallet<StandardTransaction>.CoinRecords](#) ,  
[Wallet<StandardTransaction>.ArtificialCoinRecords](#) , [Wallet<StandardTransaction>.PuzzleCache](#) ,  
[Wallet<StandardTransaction>.CoinRecordIndex\(CoinRecord\)](#) ,  
[Wallet<StandardTransaction>.Sync\(WalletOptions, CancellationToken\)](#) ,  
[Wallet<StandardTransaction>.FetchCoinRecords\(CancellationTokentoken\)](#) ,  
[Wallet<StandardTransaction>.ClearUnconfirmedTransactions\(CancellationTokentoken\)](#) ,  
[Wallet<StandardTransaction>.CreateSpend\(\)](#) ,  
[Wallet<StandardTransaction>.FindUnusedIndices\(int, List<int>, bool, CancellationToken\)](#) ,  
[Wallet<StandardTransaction>.GetBalance\(\)](#) ,  
[Wallet<StandardTransaction>.SelectCoinRecords\(BigInteger, CoinSelection, int, bool\)](#) ,  
[Wallet<StandardTransaction>.CompleteSpend\(SpendBundle, CancellationToken\)](#) ,  
[object.Equals\(object\)](#)  , [object.Equals\(object, object\)](#)  , [object.GetHashCode\(\)](#)  , [object.GetType\(\)](#)  ,  
[object.MemberwiseClone\(\)](#)  , [object.ReferenceEquals\(object, object\)](#)  , [object.ToString\(\)](#) 

## Constructors

StandardWallet(FullNodeProxy, KeyStore, byte[]?,  
WalletOptions?)

Represents a standard wallet in the Chia.NET Wallet library.

```
public StandardWallet(FullNodeProxy node, KeyStore keyStore, byte[]? hiddenPuzzleHash =  
null, WalletOptions? walletOptions = null)
```

## Parameters

**node** FullNodeProxy

Represents a standard wallet in the Chia.NET Wallet library.

**keyStore** [KeyStore](#)

Represents a standard wallet in the Chia.NET Wallet library.

**hiddenPuzzleHash** [byte](#)[]

Represents a standard wallet in the Chia.NET Wallet library.

**walletOptions** [WalletOptions](#)

Represents a standard wallet in the Chia.NET Wallet library.

## Properties

### HiddenPuzzleHash

Gets the hidden puzzle hash.

```
public byte[] HiddenPuzzleHash { get; }
```

### Property Value

[byte](#)[]

Represents a standard wallet in the Chia.NET Wallet library.

## Methods

### CreatePuzzle(KeyPair)

Creates a puzzle for the specified key pair.

```
public override StandardTransaction CreatePuzzle(KeyPair keyPair)
```

## Parameters

**keyPair** [KeyPair](#)

The key pair.

## Returns

[StandardTransaction](#)

A new instance of the [StandardTransaction](#) class.

# Send(byte[], long, long, CancellationToken)

Sends a transaction.

```
public Task<List<CoinSpend>> Send(byte[] puzzleHash, long amount, long fee,
CancellationToken cancellationToken = default)
```

## Parameters

**puzzleHash** [byte](#)[]

The puzzle hash.

**amount** [long](#)

The amount to send.

**fee** [long](#)

The fee amount.

**cancellationToken** [CancellationToken](#)

The cancellation token.

## Returns

[Task](#) <[List](#) <CoinSpend> >

A list of coin spends.

## SendFee(long, CancellationToken)

Sends a fee transaction.

```
public Task<List<CoinSpend>> SendFee(long amount, CancellationToken cancellationToken  
= default)
```

### Parameters

**amount** [long](#)

The amount to send as a fee.

**cancellationToken** [CancellationToken](#)

The cancellation token.

### Returns

[Task](#) <[List](#) <CoinSpend> >

A list of coin spends.

## SignSpend(SpendBundle, byte[])

Signs a spend bundle with the specified aggregated signature me extra data.

```
public override SpendBundle SignSpend(SpendBundle spendBundle, byte[] aggSigMeExtraData)
```

### Parameters

**spendBundle** SpendBundle

The spend bundle to sign.

**aggSigMeExtraData** [byte](#)[]

The aggregated signature me extra data.

### Returns

SpendBundle

The signed spend bundle.

# Class Tails

Namespace: [chia.dotnet.wallet](#)

Assembly: chia-dotnet-wallet.dll

Represents a collection of static methods for creating puzzle programs.

```
public static class Tails
```

## Inheritance

[object](#)  ← Tails

## Inherited Members

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

# Methods

## Delegated(JacobianPoint)

Creates a puzzle program using the "delegated" puzzle with the specified public key.

```
public static Program Delegated(JacobianPoint publicKey)
```

## Parameters

**publicKey** JacobianPoint

The public key to be used in the puzzle program.

## Returns

Program

A puzzle program.

## EverythingWithSignature(JacobianPoint)



Creates a puzzle program using the "everythingWithSignature" puzzle with the specified public key.

```
public static Program EverythingWithSignature(JacobianPoint publicKey)
```

## Parameters

**publicKey** JacobianPoint

The public key to be used in the puzzle program.

## Returns

Program

A puzzle program.

## GenesisByCoinId(byte[])

Creates a puzzle program using the "genesisByCoinId" puzzle with the specified coin ID.

```
public static Program GenesisByCoinId(byte[] coinId)
```

## Parameters

**coinId** [byte](#)[]

The coin ID to be used in the puzzle program.

## Returns

Program

A puzzle program.

## IndexedWithSignature(JacobianPoint, int)

Creates a puzzle program using the "indexedWithSignature" puzzle with the specified public key and index.

```
public static Program IndexedWithSignature(JacobianPoint publicKey, int index)
```

## Parameters

**publicKey** JacobianPoint

The public key to be used in the puzzle program.

**index** [int](#)

The index to be used in the puzzle program.

## Returns

Program

A puzzle program.

# MeltableGenesisByCoinId(byte[])

Creates a puzzle program using the "meltableGenesisByCoinId" puzzle with the specified coin ID.

```
public static Program MeltableGenesisByCoinId(byte[] coinId)
```

## Parameters

**coinId** [byte](#)[]

The coin ID to be used in the puzzle program.

## Returns

Program

A puzzle program.

# Class WalletOptions

Namespace: [chia.dotnet.wallet](#)

Assembly: chia-dotnet-wallet.dll

Represents the options for a wallet.

```
public class WalletOptions
```

## Inheritance

[object](#)  ← WalletOptions

## Inherited Members

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

## Fields

### DefaultWalletOptions

Gets the default wallet options.

```
public static readonly WalletOptions DefaultWalletOptions
```

## Field Value

[WalletOptions](#)

Represents the options for a wallet.

## Properties

### InstantCoinRecords

Gets or sets a value indicating whether instant coin records are enabled.

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

## Property Value

[bool](#)

Represents the options for a wallet.

## MaxAddressCount

Gets or sets the maximum address count.

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

## Property Value

[int](#)

Represents the options for a wallet.

## MinAddressCount

Gets or sets the minimum address count.

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

## Property Value

[int](#)

Represents the options for a wallet.

## UnusedAddressCount

Gets or sets the count of unused addresses.

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

## Property Value

[int](#)

Represents the options for a wallet.

# Class Wallet<T>

Namespace: [chia.dotnet.wallet](#)

Assembly: chia-dotnet-wallet.dll

Represents an abstract wallet class that provides common functionality for different types of wallets.

```
public abstract class Wallet<T> where T : Program
```

## Type Parameters

**T**

The type of program associated with the wallet.








## Inheritance

[object](#)  ← Wallet<T>

## Derived

[AssetWallet](#), [StandardWallet](#)

## Inherited Members

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

# Constructors

## Wallet(FullNodeProxy, KeyStore, WalletOptions?)

Represents an abstract wallet class that provides common functionality for different types of wallets.

```
protected Wallet(FullNodeProxy node, KeyStore keyStore, WalletOptions? walletOptions = null)
```

## Parameters

**node** FullNodeProxy

Represents an abstract wallet class that provides common functionality for different types of wallets.

keyStore [KeyStore](#)

Represents an abstract wallet class that provides common functionality for different types of wallets.

walletOptions [WalletOptions](#)

Represents an abstract wallet class that provides common functionality for different types of wallets.

## Properties

### ArtificialCoinRecords

Gets the list of chia.dotnet.CoinRecord instances representing the artificial coin records associated with the wallet.

```
public List<CoinRecord> ArtificialCoinRecords { get; }
```

#### Property Value

[List](#) <CoinRecord>

Represents an abstract wallet class that provides common functionality for different types of wallets.

### CoinRecords

Gets or sets the list of lists of chia.dotnet.CoinRecord instances representing the coin records associated with the wallet.

```
public List<List<CoinRecord>> CoinRecords { get; }
```

#### Property Value

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

Represents an abstract wallet class that provides common functionality for different types of wallets.

## KeyStore

Gets or sets the [KeyStore](#) instance used for managing keys.

```
public KeyStore KeyStore { get; init; }
```

## Property Value

[KeyStore](#)

Represents an abstract wallet class that provides common functionality for different types of wallets.

## Node

Gets or sets the `chia.dotnet.FullNodeProxy` instance used for interacting with the Chia full node.

```
public FullNodeProxy Node { get; init; }
```

## Property Value

`FullNodeProxy`

Represents an abstract wallet class that provides common functionality for different types of wallets.

## Options

Gets or sets the [WalletOptions](#) instance used for configuring wallet options.

```
public WalletOptions Options { get; init; }
```

## Property Value

[WalletOptions](#)

Represents an abstract wallet class that provides common functionality for different types of wallets.

## PuzzleCache

Gets the list of `T` instances representing the puzzle cache associated with the wallet.



```
public List<T> PuzzleCache { get; }
```

## Property Value

[List](#)<T>

Represents an abstract wallet class that provides common functionality for different types of wallets.

## Methods

### ClearUnconfirmedTransactions(CancellationToken)

Clears the unconfirmed transactions associated with the wallet.

```
public Task ClearUnconfirmedTransactions(Cancellation token = default)
```

## Parameters

**cancellationToken** [CancellationToken](#)

## Returns

[Task](#)

### CoinRecordIndex(CoinRecord)

Gets the index of the given coin record in the puzzle cache.

```
public int CoinRecordIndex(CoinRecord coinRecord)
```

## Parameters

**coinRecord** CoinRecord

The coin record to find the index of.

## Returns

[int](#)

The index of the coin record in the puzzle cache.

## CompleteSpend(SpendBundle, CancellationToken)

Completes the spend of the given spend bundle.

```
public Task CompleteSpend(SpendBundle spendBundle, CancellationTok... cancellationToken = default)
```

### Parameters

**spendBundle** SpendBundle

**cancellationToken** [CancellationToken](#)

### Returns

[Task](#)

### Exceptions

[Exception](#)

## CreatePuzzle(KeyPair)

Creates a new instance of the program associated with the wallet using the given key pair.

```
public abstract T CreatePuzzle(KeyPair keyPair)
```

### Parameters

**keyPair** [KeyPair](#)

The key pair to use for creating the program.

### Returns

T

The created program instance.

## CreateSpend()

Creates a new spend bundle.

```
public SpendBundle CreateSpend()
```

Returns

SpendBundle

SpendBundle

## FetchCoinRecords(CancellationToken)

Fetches the coin records associated with the wallet from the Chia network.

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

Parameters

cancellationToken [CancellationToken](#)

Returns

[Task](#)

An awaitable Task

## FindUnusedIndices(int, List<int>, bool, CancellationToken)

Finds the unused indices for the wallet.

```
public Task<List<int>> FindUnusedIndices(int amount, List<int> used, bool presynced = false,
```

```
CancellationToken cancellationToken = default)
```

## Parameters

**amount** [int](#)

**used** [List](#) <[int](#)>

**presynced** [bool](#)

**cancellationToken** [CancellationToken](#)

## Returns

[Task](#) <[List](#) <[int](#)>>

The list of indices

## Exceptions

[Exception](#)

## GetBalance()

Gets the balance of the wallet.

```
public BigInteger GetBalance()
```

## Returns

[BigInteger](#)

The balance

## SelectCoinRecords(BigInteger, CoinSelection, int, bool)

Selects coin records for spending.

```
public List<CoinRecord> SelectCoinRecords(BigInteger amount, CoinSelection coinSelection,
```

```
int minimumCoinRecords = 0, bool required = true)
```

## Parameters

**amount** [BigInteger](#)

**coinSelection** [CoinSelection](#)

**minimumCoinRecords** [int](#)

**required** [bool](#)

## Returns

[List](#) <CoinRecord>

## Exceptions

[Exception](#)

# SignSpend(SpendBundle, byte[])

Signs the given spend bundle with the wallet's private key and returns the signed spend bundle.

```
public abstract SpendBundle SignSpend(SpendBundle spendBundle, byte[] aggSigMeExtraData)
```

## Parameters

**spendBundle** SpendBundle

The spend bundle to sign.

**aggSigMeExtraData** [byte](#)[]

The extra data to include in the aggregated signature.

## Returns

SpendBundle

The signed spend bundle.

# Sync(WalletOptions?, CancellationToken)

Synchronizes the wallet with the Chia network, fetching new coin records and updating the puzzle cache.

```
public Task Sync(WalletOptions? overrideOptions = null, CancellationToken cancellationToken = default)
```

## Parameters

**overrideOptions** [WalletOptions](#)

The wallet options to use for synchronization. If null, the default wallet options will be used.

**cancellationToken** [CancellationToken](#)

The cancellation token.

## Returns

[Task](#)

A task representing the asynchronous synchronization operation.