

Namespace AkiraNetwork.VirtualStorage Library

Classes

[VirtualCycleDetector](#)

[VirtualDirectory](#)

[VirtualDirectoryAdapter<T>](#)

[VirtualException](#)

[VirtualGroupCondition<T, TKey>](#)

[VirtualItem](#)

[VirtualItemAdapter<T>](#)

[VirtualItem<T>](#)

[VirtualNode](#)

Represents an abstract class for nodes.

[VirtualNodeContext](#)

[VirtualNodeExtensions](#)

[VirtualNodeName](#)

[VirtualNodeNotFoundException](#)

[VirtualPath](#)

[VirtualSortCondition<T>](#)

[VirtualStorageExtensions](#)

[VirtualStorageSettings](#)

[VirtualStorageState](#)

[VirtualStorage<T>](#)

[VirtualSymbolicLink](#)

[VirtualSymbolicLinkAdapter<T>](#)

Structs

[VirtualID](#)

[VirtualNodeListConditions](#)

Interfaces

[IVirtualDeepCloneable<T>](#)

[IVirtualWildcardMatcher](#)

Enums

[VirtualNodeType](#)

[VirtualNodeTypeFilter](#)

Delegates

[ActionNodeDelegate](#)

[NotifyNodeDelegate](#)

[PatternMatch](#)

Delegate ActionNodeDelegate

Namespace: [AkiraNetwork.VirtualStorageLibrary](#).

Assembly: VirtualStorageLibrary.dll

```
public delegate bool ActionNodeDelegate(VirtualDirectory parentDirectory, VirtualNodeName  
nodeName, VirtualPath nodePath)
```

Parameters

parentDirectory [VirtualDirectory](#)

nodeName [VirtualNodeName](#)

nodePath [VirtualPath](#)

Returns

[bool](#) 

Extension Methods

[VirtualTextFormatter.GenerateSingleTableDebugText<T>\(T\)](#).

Interface IVirtualDeepCloneable<T>

Namespace: [AkiraNetwork.VirtualStorageLibrary](#)

Assembly: VirtualStorageLibrary.dll

```
public interface IVirtualDeepCloneable<T>
```

Type Parameters

T

Extension Methods

[VirtualTextFormatter.GenerateSingleTableDebugText<T>\(T\)](#)

Methods

DeepClone(bool)

```
T DeepClone(bool recursive = false)
```

Parameters

recursive [bool](#)

Returns

T

Interface IVirtualWildcardMatcher

Namespace: [AkiraNetwork.VirtualStorageLibrary](#).

Assembly: VirtualStorageLibrary.dll

```
public interface IVirtualWildcardMatcher
```

Extension Methods

[VirtualTextFormatter.GenerateSingleTableDebugText<T>\(T\)](#).

Properties

Count

```
int Count { get; }
```

Property Value

[int](#)

Patterns

```
IEnumerable<string> Patterns { get; }
```

Property Value

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

WildcardDictionary

```
ReadOnlyDictionary<string, string> WildcardDictionary { get; }
```

Property Value

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

Wildcards

```
IEnumerable<string> Wildcards { get; }
```

Property Value

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

Methods

PatternMatcher(string, string)

```
bool PatternMatcher(string nodeName, string pattern)
```

Parameters

[nodeName](#) [string](#)

[pattern](#) [string](#)

Returns

[bool](#)

Delegate NotifyNodeDelegate

Namespace: [AkiraNetwork.VirtualStorageLibrary](#).

Assembly: VirtualStorageLibrary.dll

```
public delegate void NotifyNodeDelegate(VirtualPath path, VirtualNode? node)
```

Parameters

path [VirtualPath](#)

node [VirtualNode](#)

Extension Methods

[VirtualTextFormatter.GenerateSingleTableDebugText<T>\(T\)](#).

Delegate PatternMatch

Namespace: [AkiraNetwork.VirtualStorageLibrary](#).

Assembly: VirtualStorageLibrary.dll

```
public delegate bool PatternMatch(string nodeName, string pattern)
```

Parameters

nodeName [string](#)

pattern [string](#)

Returns

[bool](#)

Extension Methods

[VirtualTextFormatter.GenerateSingleTableDebugText<T>\(T\)](#).


Class VirtualCycleDetector

Namespace: [AkiraNetwork.VirtualStorageLibrary](#).








Assembly: VirtualStorageLibrary.dll

```
public class VirtualCycleDetector
```

Inheritance

[object](#)  ← VirtualCycleDetector

Inherited Members

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

Extension Methods

[VirtualTextFormatter.GenerateSingleTableDebugText<T>\(T\)](#)

Constructors

VirtualCycleDetector()

```
public VirtualCycleDetector()
```

Properties

Count

```
public int Count { get; }
```

Property Value

[int](#) 

CycleDictionary

```
public Dictionary<VirtualID, VirtualSymbolicLink> CycleDictionary { get; }
```

Property Value

[Dictionary](#) [<VirtualID, VirtualSymbolicLink>](#)

Methods

Clear()

```
public void Clear()
```

IsNodeInCycle(VirtualSymbolicLink)

```
public bool IsNodeInCycle(VirtualSymbolicLink link)
```

Parameters

link [VirtualSymbolicLink](#)

Returns

[bool](#)

Class VirtualDirectory

Namespace: [AkiraNetwork.VirtualStorageLibrary](#)

Assembly: VirtualStorageLibrary.dll

```
public class VirtualDirectory : VirtualNode, IVirtualDeepCloneable<VirtualNode>,
    IEnumerable<VirtualNode>, IEnumerable
```







Inheritance

[object](#)  ← [VirtualNode](#)  ← VirtualDirectory

Implements

[IVirtualDeepCloneable](#)  <[VirtualNode](#)>, [IEnumerable](#)  <[VirtualNode](#)>, [IEnumerable](#) 

Inherited Members

[VirtualNode.Name](#) , [VirtualNode.CreatedDate](#) , [VirtualNode.UpdatedDate](#) , [VirtualNode.VID](#) ,
[VirtualNode.IsReferencedInStorage](#) , [object.Equals\(object\)](#)  , [object.Equals\(object, object\)](#)  ,
[object.GetHashCode\(\)](#)  , [object.GetType\(\)](#)  , [object.MemberwiseClone\(\)](#)  ,
[object.ReferenceEquals\(object, object\)](#) 

Extension Methods

[VirtualTextFormatter.GenerateSingleTableDebugText<T>\(T\)](#) ,
[VirtualNodeExtensions.ResolveNodeType\(VirtualNode\)](#) ,
[VirtualTextFormatter.GenerateTableDebugText<T>\(IEnumerable<T>\)](#) ,
[VirtualStorageExtensions.ApplySortConditions<T>\(IEnumerable<T>, List<VirtualSortCondition<T>>?\)](#) ,
[VirtualStorageExtensions.GroupAndSort<T>\(IEnumerable<T>, VirtualGroupCondition<T, object>?, List<VirtualSortCondition<T>>?\)](#).

Constructors

VirtualDirectory()

```
public VirtualDirectory()
```

VirtualDirectory(VirtualNodeName)

```
public VirtualDirectory(VirtualNodeName name)
```

Parameters

name [VirtualNodeName](#)

VirtualDirectory(VirtualNodeName, DateTime, DateTime)

```
public VirtualDirectory(VirtualNodeName name, DateTime createdDate, DateTime updatedDate)
```

Parameters

name [VirtualNodeName](#)

createdDate [DateTime](#)↗

updatedDate [DateTime](#)↗

Properties

Count

```
public int Count { get; }
```

Property Value

[int](#)↗

DirectoryCount

```
public int DirectoryCount { get; }
```

Property Value

[int](#)↗

DirectoryViewCount

```
public int DirectoryViewCount { get; }
```

Property Value

[int](#)

this[VirtualNodeName]

```
public VirtualNode this[VirtualNodeName name] { get; set; }
```

Parameters

name [VirtualNodeName](#)

Property Value

[VirtualNode](#)

ItemCount

```
public int ItemCount { get; }
```

Property Value

[int](#)

ItemViewCount

```
public int ItemViewCount { get; }
```

Property Value

[int](#)

NodeNames

```
public IEnumerable<VirtualNodeName> NodeNames { get; }
```

Property Value

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

NodeType

Gets the node type of node.

```
public override VirtualNodeType NodeType { get; }
```

Property Value

[VirtualNodeType](#)

Nodes

```
public IEnumerable<VirtualNode> Nodes { get; }
```

Property Value

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

NodesView

```
public IEnumerable<VirtualNode> NodesView { get; }
```

Property Value

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

NodesViewCount

```
public int NodesViewCount { get; }
```

Property Value

[int](#)

SymbolicLinkCount

```
public int SymbolicLinkCount { get; }
```

Property Value

[int](#)

SymbolicLinkViewCount

```
public int SymbolicLinkViewCount { get; }
```

Property Value

[int](#)

Methods

Add(VirtualNode, bool)

```
public VirtualNode Add(VirtualNode node, bool allowOverwrite = false)
```

Parameters

node [VirtualNode](#)

allowOverwrite [bool](#) 

Returns

[VirtualNode](#)

AddDirectory(VirtualNodeName, bool)

```
public VirtualDirectory AddDirectory(VirtualNodeName name, bool allowOverwrite = false)
```

Parameters

name [VirtualNodeName](#)

allowOverwrite [bool](#) 

Returns

[VirtualDirectory](#)

AddItem<T>(VirtualNodeName, T?, bool)

```
public VirtualItem<T> AddItem<T>(VirtualNodeName name, T? itemData = default, bool allowOverwrite = false)
```

Parameters

name [VirtualNodeName](#)

itemData T

allowOverwrite [bool](#) 

Returns

[VirtualItem](#)<T>

Type Parameters

T

AddSymbolicLink(VirtualNodeName, VirtualPath, bool)

```
public VirtualSymbolicLink AddSymbolicLink(VirtualNodeName name, VirtualPath targetPath,
bool allowOverwrite = false)
```

Parameters

name [VirtualNodeName](#)

targetPath [VirtualPath](#)

allowOverwrite [bool](#)

Returns

[VirtualSymbolicLink](#)

DeepClone(bool)

Creates a deep clone of the entity. However, the CreatedDate and UpdatedDate should not be cloned as they are set to the current date and time at the time of cloning.

```
public override VirtualNode DeepClone(bool recursive = false)
```

Parameters

recursive [bool](#)

When true, all child nodes are also cloned, creating a deep copy of the entire tree. The default is false. The CreatedDate and UpdatedDate properties are not preserved. They are set to the current date and time at the moment of instantiation or cloning.

Returns

[VirtualNode](#)

Cloned [VirtualNode](#) instance

DirectoryExists(VirtualNodeName)

```
public bool DirectoryExists(VirtualNodeName name)
```

Parameters

name [VirtualNodeName](#)

Returns

[bool](#) 

Get(VirtualNodeName, bool)

```
public VirtualNode? Get(VirtualNodeName name, bool exceptionEnabled = true)
```

Parameters

name [VirtualNodeName](#)

exceptionEnabled [bool](#) 

Returns

[VirtualNode](#)

GetDirectory(VirtualNodeName)

```
public VirtualDirectory GetDirectory(VirtualNodeName name)
```

Parameters

name [VirtualNodeName](#)

Returns

[VirtualDirectory](#)

GetEnumerator()

Returns an enumerator that iterates through the collection.

```
public IEnumerator<VirtualNode> GetEnumerator()
```

Returns

[IEnumerator](#) [<VirtualNode>](#)

An enumerator that can be used to iterate through the collection.

GetItem<T>(VirtualNodeName)

```
public VirtualItem<T> GetItem<T>(VirtualNodeName name)
```

Parameters

name [VirtualNodeName](#)

Returns

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

Type Parameters

T

GetNodesView()

```
public IEnumerable<VirtualNode> GetNodesView()
```

Returns

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

GetSymbolicLink(VirtualNodeName)

```
public VirtualSymbolicLink GetSymbolicLink(VirtualNodeName name)
```

Parameters

name [VirtualNodeName](#)

Returns

[VirtualSymbolicLink](#)

ItemExists(VirtualNodeName)

```
public bool ItemExists(VirtualNodeName name)
```

Parameters

name [VirtualNodeName](#)

Returns

[bool](#)

NodeExists(VirtualNodeName)

```
public bool NodeExists(VirtualNodeName name)
```

Parameters

name [VirtualNodeName](#)

Returns

[bool](#)

Remove(VirtualNode)

```
public void Remove(VirtualNode node)
```

Parameters

node [VirtualNode](#)

SymbolicLinkExists(VirtualNodeName)

```
public bool SymbolicLinkExists(VirtualNodeName name)
```

Parameters

name [VirtualNodeName](#)

Returns

[bool](#)

ToString()

Returns a string that represents the current object.

```
public override string ToString()
```

Returns

[string](#) 

A string that represents the current object.

Update(VirtualNode)

Updates the [VirtualNode](#).

```
public override void Update(VirtualNode node)
```

Parameters

node [VirtualNode](#)

Value to update

Operators

operator +(VirtualDirectory, VirtualNode)

```
public static VirtualDirectory operator +(VirtualDirectory directory, VirtualNode node)
```

Parameters

directory [VirtualDirectory](#)

node [VirtualNode](#)

Returns

[VirtualDirectory](#)

implicit operator VirtualDirectory(VirtualNodeName)

```
public static implicit operator VirtualDirectory(VirtualNodeName nodeName)
```

Parameters

nodeName [VirtualNodeName](#)

Returns

[VirtualDirectory](#)

operator -(VirtualDirectory, VirtualNode)

```
public static VirtualDirectory operator -(VirtualDirectory directory, VirtualNode node)
```

Parameters

directory [VirtualDirectory](#)

node [VirtualNode](#)

Returns

[VirtualDirectory](#)

Class VirtualDirectoryAdapter<T>

Namespace: [AkiraNetwork.VirtualStorageLibrary](#)

Assembly: VirtualStorageLibrary.dll

```
public class VirtualDirectoryAdapter<T>
```

Type Parameters

T

Inheritance

[object](#) ← VirtualDirectoryAdapter<T>

Inherited Members

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

Extension Methods

[VirtualTextFormatter.GenerateSingleTableDebugText<T>\(T\)](#)

Constructors

VirtualDirectoryAdapter(VirtualStorage<T>)

```
public VirtualDirectoryAdapter(VirtualStorage<T> storage)
```

Parameters

storage [VirtualStorage](#)<T>

Properties

this[VirtualPath, bool]

```
public VirtualDirectory this[VirtualPath path, bool followLinks = true] { get; set; }
```


Parameters

path [VirtualPath](#)

followLinks [bool](#) 

Property Value

[VirtualDirectory](#)

Class VirtualException

Namespace: [AkiraNetwork.VirtualStorageLibrary](#)

Assembly: VirtualStorageLibrary.dll

```
public abstract class VirtualException : Exception, ISerializable
```

Inheritance

[object](#) ← [Exception](#) ← VirtualException

Implements

[ISerializable](#)

Derived

[VirtualNodeNotFoundException](#)

Inherited Members

[Exception.GetBaseException\(\)](#), [Exception.GetType\(\)](#), [Exception.ToString\(\)](#), [Exception.Data](#), [Exception.HelpLink](#), [Exception.HResult](#), [Exception.InnerException](#), [Exception.Message](#), [Exception.Source](#), [Exception.StackTrace](#), [Exception.TargetSite](#), [Exception.SerializeObjectState](#), [object.Equals\(object\)](#), [object.Equals\(object, object\)](#), [object.GetHashCode\(\)](#), [object.MemberwiseClone\(\)](#), [object.ReferenceEquals\(object, object\)](#)

Extension Methods

[VirtualTextFormatter.GenerateSingleTableDebugText<T>\(T\)](#)

Constructors

VirtualException()

```
public VirtualException()
```

VirtualException(string)

```
public VirtualException(string message)
```

Parameters

message [string](#)

VirtualException(string, Exception)

```
public VirtualException(string message, Exception innerException)
```

Parameters

message [string](#)

innerException [Exception](#)

Class VirtualGroupCondition<T, TKey>

Namespace: [AkiraNetwork.VirtualStorageLibrary](#)

Assembly: VirtualStorageLibrary.dll


```
public class VirtualGroupCondition<T, TKey>
```

Type Parameters








T

TKey

Inheritance

[object](#)  ← VirtualGroupCondition<T, TKey>

Inherited Members

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

Extension Methods

[VirtualTextFormatter.GenerateSingleTableDebugText<T>\(T\)](#)

Constructors

VirtualGroupCondition(Expression<Func<T, TKey>>, bool)

```
public VirtualGroupCondition(Expression<Func<T, TKey>> groupBy, bool ascending = true)
```

Parameters

groupBy [Expression](#)  <[Func](#)  <T, TKey>>

ascending [bool](#) 

Properties

Ascending

```
public bool Ascending { get; set; }
```

Property Value

[bool](#)

GroupBy

```
public Expression<Func<T, TKey>> GroupBy { get; set; }
```

Property Value

[Expression](#) <[Func](#) <T, TKey>>

Struct VirtualID

Namespace: [AkiraNetwork.VirtualStorageLibrary](#)






Assembly: VirtualStorageLibrary.dll

```
public readonly record struct VirtualID : IEquatable<VirtualID>
```

Implements

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

Inherited Members

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

Extension Methods

[VirtualTextFormatter.GenerateSingleTableDebugText<T>\(T\)](#)

Constructors

VirtualID()

```
public VirtualID()
```

Properties

ID

```
public Guid ID { get; }
```

Property Value

[Guid](#) 

Methods

ToString()

Returns the fully qualified type name of this instance.

```
public override string ToString()
```

Returns

[string](#) 

The fully qualified type name.

Class VirtualItem

Namespace: [AkiraNetwork.VirtualStorageLibrary](#)

Assembly: VirtualStorageLibrary.dll

```
public abstract class VirtualItem : VirtualNode, IVirtualDeepCloneable<VirtualNode>
```

Inheritance

[object](#)  ← [VirtualNode](#) ← VirtualItem








Implements

[IVirtualDeepCloneable](#)<[VirtualNode](#)>

Derived

[VirtualItem<T>](#)

Inherited Members

[VirtualNode.Name](#) , [VirtualNode.CreatedDate](#) , [VirtualNode.UpdatedDate](#) , [VirtualNode.NodeType](#) , [VirtualNode.VID](#) , [VirtualNode.Update\(VirtualNode\)](#) , [VirtualNode.IsReferencedInStorage](#) , [object.Equals\(object\)](#)  , [object.Equals\(object, object\)](#)  , [object.GetHashCode\(\)](#)  , [object.GetType\(\)](#)  , [object.MemberwiseClone\(\)](#)  , [object.ReferenceEquals\(object, object\)](#)  , [object.ToString\(\)](#) 

Extension Methods

[VirtualTextFormatter.GenerateSingleTableDebugText<T>\(T\)](#) ,
[VirtualNodeExtensions.ResolveNodeType\(VirtualNode\)](#)

Constructors

VirtualItem(VirtualNodeName)

```
protected VirtualItem(VirtualNodeName name)
```

Parameters

name [VirtualNodeName](#)

VirtualItem(VirtualNodeName, DateTime)


```
protected VirtualItem(VirtualNodeName name, DateTime createdDate)
```

Parameters

name [VirtualNodeName](#)

createdDate [DateTime](#)↗

VirtualItem(VirtualNodeName, DateTime, DateTime)

```
protected VirtualItem(VirtualNodeName name, DateTime createdDate, DateTime updatedDate)
```

Parameters

name [VirtualNodeName](#)

createdDate [DateTime](#)↗

updatedDate [DateTime](#)↗

Methods

DeepClone(bool)

Creates a deep clone of the entity. However, the CreatedDate and UpdatedDate should not be cloned as they are set to the current date and time at the time of cloning.

```
public override abstract VirtualNode DeepClone(bool recursive = false)
```

Parameters

recursive [bool](#)↗

When true, all child nodes are also cloned, creating a deep copy of the entire tree. The default is false. The CreatedDate and UpdatedDate properties are not preserved. They are set to the current date and time at the moment of instantiation or cloning.

Returns

[VirtualNode](#)

Cloned [VirtualNode](#) instance

Class VirtualItemAdapter<T>

Namespace: [AkiraNetwork.VirtualStorageLibrary](#)


Assembly: VirtualStorageLibrary.dll

```
public class VirtualItemAdapter<T>
```








Type Parameters

T

Inheritance

[object](#)  ← VirtualItemAdapter<T>

Inherited Members

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

Extension Methods

[VirtualTextFormatter.GenerateSingleTableDebugText<T>\(T\)](#)

Constructors

VirtualItemAdapter(VirtualStorage<T>)

```
public VirtualItemAdapter(VirtualStorage<T> storage)
```

Parameters

storage [VirtualStorage](#)<T>

Properties

this[VirtualPath, bool]

```
public VirtualItem<T> this[VirtualPath path, bool followLinks = true] { get; set; }
```

Parameters

path [VirtualPath](#)

followLinks [bool](#) 

Property Value

[VirtualItem](#)<T>

Class VirtualItem<T>

Namespace: [AkiraNetwork.VirtualStorageLibrary](#)

Assembly: VirtualStorageLibrary.dll

```
public class VirtualItem<T> : VirtualItem, IVirtualDeepCloneable<VirtualNode>, IDisposable
```

Type Parameters

T







Inheritance

[object](#)  ← [VirtualNode](#) ← [VirtualItem](#) ← VirtualItem<T>

Implements

[IVirtualDeepCloneable<VirtualNode>](#), [IDisposable](#) 

Inherited Members

[VirtualNode.Name](#), [VirtualNode.CreatedDate](#), [VirtualNode.UpdatedDate](#), [VirtualNode.VID](#), [VirtualNode.IsReferencedInStorage](#), [object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) 

Extension Methods

[VirtualTextFormatter.GenerateSingleTableDebugText<T>\(T\)](#),
[VirtualNodeExtensions.ResolveNodeType\(VirtualNode\)](#)

Constructors

VirtualItem()

```
public VirtualItem()
```

VirtualItem(VirtualNodeName)

```
public VirtualItem(VirtualNodeName name)
```

Parameters

name [VirtualNodeName](#)

VirtualItem(VirtualNodeName, T?)

```
public VirtualItem(VirtualNodeName name, T? item)
```

Parameters

name [VirtualNodeName](#)

item T

VirtualItem(VirtualNodeName, T?, DateTime)

```
public VirtualItem(VirtualNodeName name, T? item, DateTime createdDate)
```

Parameters

name [VirtualNodeName](#)

item T

createdDate [DateTime](#)[↗](#)

VirtualItem(VirtualNodeName, T?, DateTime, DateTime)

```
public VirtualItem(VirtualNodeName name, T? item, DateTime createdDate,  
DateTime updatedDate)
```

Parameters

name [VirtualNodeName](#)

item T

createdDate [DateTime](#)

updatedAt [DateTime](#)

Properties

ItemData

```
public T? ItemData { get; set; }
```

Property Value

T

NodeType

Gets the node type of node.

```
public override VirtualNodeType NodeType { get; }
```

Property Value

[VirtualNodeType](#)

Methods

DeepClone(bool)

Creates a deep clone of the entity. However, the CreatedDate and UpdatedDate should not be cloned as they are set to the current date and time at the time of cloning.

```
public override VirtualNode DeepClone(bool recursive = false)
```

Parameters

recursive [bool](#)

When true, all child nodes are also cloned, creating a deep copy of the entire tree. The default is false. The CreatedDate and UpdatedDate properties are not preserved. They are set to the current date and time at the moment of instantiation or cloning.

Returns

[VirtualNode](#)

Cloned [VirtualNode](#) instance

Dispose()

Performs application-defined tasks associated with freeing, releasing, or resetting unmanaged resources.

```
public void Dispose()
```

Dispose(bool)

```
protected virtual void Dispose(bool disposing)
```

Parameters

disposing [bool](#)

~VirtualItem()

```
protected ~VirtualItem()
```

ToString()

Returns a string that represents the current object.

```
public override string ToString()
```


Returns

[string](#) 

A string that represents the current object.

Update(VirtualNode)

Updates the [VirtualNode](#).

```
public override void Update(VirtualNode node)
```

Parameters

node [VirtualNode](#)

Value to update

Operators

implicit operator VirtualItem<T>(VirtualNodeName)

```
public static implicit operator VirtualItem<T>(VirtualNodeName name)
```

Parameters

name [VirtualNodeName](#)

Returns

[VirtualItem](#)<T>

implicit operator VirtualItem<T>((VirtualNodeName nodeName, T? itemData))

```
public static implicit operator VirtualItem<T>((VirtualNodeName nodeName, T?
```

```
itemData) tuple)
```

Parameters

tuple ([VirtualNodeName nodeName](#)[↗], T [itemData](#)[↗])

Returns

[VirtualItem](#)<T>

implicit operator VirtualItem<T>(T?)

```
public static implicit operator VirtualItem<T>(T? itemData)
```

Parameters

itemData T

Returns

[VirtualItem](#)<T>

Class VirtualNode

Namespace: [AkiraNetwork.VirtualStorageLibrary](#)

Assembly: VirtualStorageLibrary.dll

Represents an abstract class for nodes.

```
public abstract class VirtualNode : IVirtualDeepCloneable<VirtualNode>
```

Inheritance

[object](#)  ← VirtualNode








Implements

[IVirtualDeepCloneable](#) <[VirtualNode](#)>

Derived

[VirtualDirectory](#), [VirtualItem](#), [VirtualSymbolicLink](#)

Inherited Members

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

Extension Methods

[VirtualTextFormatter.GenerateSingleTableDebugText<T>\(T\)](#) ,
[VirtualNodeExtensions.ResolveNodeType\(VirtualNode\)](#)

Constructors

VirtualNode(VirtualNodeName)

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

```
protected VirtualNode(VirtualNodeName name)
```

Parameters

name [VirtualNodeName](#)

The name of node.

VirtualNode(VirtualNodeName, DateTime)

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

```
protected VirtualNode(VirtualNodeName name, DateTime createdDate)
```

Parameters

name [VirtualNodeName](#)

The name of node.

createdDate [DateTime](#) 

The created date of node.

VirtualNode(VirtualNodeName, DateTime, DateTime)

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

```
protected VirtualNode(VirtualNodeName name, DateTime createdDate, DateTime updatedDate)
```

Parameters

name [VirtualNodeName](#)

The name of node.

createdDate [DateTime](#) 

The created date of node.

updatedAt [DateTime](#) 

The updated date of node.

Properties

CreatedDate

Gets the created date of the node. This date represents when the node was first created.

```
public DateTime CreatedDate { get; }
```

Property Value

[DateTime](#)

IsReferencedInStorage

Gets a value indicating whether referenced in storage. If this property is true, the node is referenced from storage. Otherwise, it is not.

```
public bool IsReferencedInStorage { get; }
```

Property Value

[bool](#)

Name

Gets the name of node.

```
public VirtualNodeName Name { get; }
```

Property Value

[VirtualNodeName](#)

NodeType

Gets the node type of node.

```
public abstract VirtualNodeType NodeType { get; }
```

Property Value

[VirtualNodeType](#)

UpdatedDate

Gets the updated date of the node. This date represents the last time the node was modified. It is set to the current date and time at the moment of instantiation or cloning.

```
public DateTime UpdatedDate { get; }
```

Property Value

[DateTime](#)

VID

Gets the VID of node.

```
public VirtualID VID { get; }
```

Property Value

[VirtualID](#)

Methods

DeepClone(bool)

Creates a deep clone of the entity. However, the CreatedDate and UpdatedDate should not be cloned as they are set to the current date and time at the time of cloning.

```
public abstract VirtualNode DeepClone(bool recursive = false)
```

Parameters

`recursive` [bool](#)

When true, all child nodes are also cloned, creating a deep copy of the entire tree. The default is false. The `CreatedDate` and `UpdatedDate` properties are not preserved. They are set to the current date and time at the moment of instantiation or cloning.

Returns

[VirtualNode](#)

Cloned [VirtualNode](#) instance

Update(VirtualNode)

Updates the [VirtualNode](#).

```
public abstract void Update(VirtualNode node)
```

Parameters

`node` [VirtualNode](#)

Value to update

Class VirtualNodeContext

Namespace: [AkiraNetwork.VirtualStorageLibrary](#)

Assembly: VirtualStorageLibrary.dll

```
public class VirtualNodeContext
```

Inheritance

[object](#) ← VirtualNodeContext

Inherited Members

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

Extension Methods

[VirtualTextFormatter.GenerateSingleTableDebugText<T>\(T\)](#)

Constructors

VirtualNodeContext(VirtualNode?, VirtualPath, VirtualDirectory?, int, int, VirtualPath?, bool, VirtualSymbolicLink?)

```
public VirtualNodeContext(VirtualNode? node, VirtualPath traversalPath, VirtualDirectory?
parentNode = null, int depth = 0, int index = 0, VirtualPath? resolvedPath = null, bool
resolved = false, VirtualSymbolicLink? resolvedLink = null)
```

Parameters

node [VirtualNode](#)

traversalPath [VirtualPath](#)

parentNode [VirtualDirectory](#)

depth [int](#)

index [int](#)

resolvedPath [VirtualPath](#)

resolved [bool](#)

resolvedLink [VirtualSymbolicLink](#)

Properties

Depth

```
public int Depth { get; set; }
```

Property Value

[int](#)

Index

```
public int Index { get; set; }
```

Property Value

[int](#)

Node

```
public VirtualNode? Node { get; set; }
```

Property Value

[VirtualNode](#)

ParentDirectory

```
public VirtualDirectory? ParentDirectory { get; set; }
```

Property Value

[VirtualDirectory](#)

Resolved

```
public bool Resolved { get; set; }
```

Property Value

[bool](#)

ResolvedLink

```
public VirtualSymbolicLink? ResolvedLink { get; set; }
```

Property Value

[VirtualSymbolicLink](#)

ResolvedPath

```
public VirtualPath? ResolvedPath { get; set; }
```

Property Value

[VirtualPath](#)

TraversalPath

```
public VirtualPath TraversalPath { get; set; }
```

Property Value

Methods

ToString()

Returns a string that represents the current object.

```
public override string ToString()
```

Returns

[string](#) 

A string that represents the current object.


Class VirtualNodeExtensions

Namespace: [AkiraNetwork.VirtualStorageLibrary](#)








Assembly: VirtualStorageLibrary.dll

```
public static class VirtualNodeExtensions
```

Inheritance

[object](#)  ← VirtualNodeExtensions

Inherited Members

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

Methods

ResolveNodeType(VirtualNode)

```
public static VirtualNodeType ResolveNodeType(this VirtualNode node)
```

Parameters

node [VirtualNode](#)

Returns

[VirtualNodeType](#)







Struct VirtualNodeListConditions

Namespace: [AkiraNetwork.VirtualStorageLibrary](#)

Assembly: VirtualStorageLibrary.dll

```
public struct VirtualNodeListConditions
```

Inherited Members

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

Extension Methods

[VirtualTextFormatter.GenerateSingleTableDebugText<T>\(T\)](#)

Constructors

VirtualNodeListConditions()

```
public VirtualNodeListConditions()
```

VirtualNodeListConditions(VirtualNodeTypeFilter, VirtualGroupCondition<VirtualNode, object>?, List<VirtualSortCondition<VirtualNode>>?)

```
public VirtualNodeListConditions(VirtualNodeTypeFilter filter,  
VirtualGroupCondition<VirtualNode, object>? groupCondition,  
List<VirtualSortCondition<VirtualNode>>? sortConditions)
```

Parameters

filter [VirtualNodeTypeFilter](#)

groupCondition [VirtualGroupCondition](#)<[VirtualNode](#), [object](#)>

sortConditions [List](#)<[VirtualSortCondition](#)<[VirtualNode](#)>>

Properties

Filter

```
public VirtualNodeTypeFilter Filter { readonly get; set; }
```

Property Value

[VirtualNodeTypeFilter](#)

GroupCondition

```
public VirtualGroupCondition<VirtualNode, object>? GroupCondition { readonly get; set; }
```

Property Value

[VirtualGroupCondition](#) <[VirtualNode](#), [object](#)>

SortConditions

```
public List<VirtualSortCondition<VirtualNode>>? SortConditions { readonly get; set; }
```

Property Value

[List](#) <[VirtualSortCondition](#) <[VirtualNode](#)>>

Class VirtualNodeName

Namespace: [AkiraNetwork.VirtualStorageLibrary](#)

Assembly: VirtualStorageLibrary.dll

```
public class VirtualNodeName : IEquatable<VirtualNodeName>, IComparable<VirtualNodeName>,
    IComparable
```

Inheritance

[object](#) ← VirtualNodeName

Implements

[IEquatable](#) <[VirtualNodeName](#)>, [IComparable](#) <[VirtualNodeName](#)>, [IComparable](#)

Inherited Members

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

Extension Methods

[VirtualTextFormatter.GenerateSingleTableDebugText<T>\(T\)](#)

Constructors

VirtualNodeName(string)

```
public VirtualNodeName(string name)
```

Parameters

name [string](#)

Properties

IsRoot

```
public bool IsRoot { get; }
```

Property Value

[bool](#)

Name

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

Property Value

[string](#)

Methods

CompareTo(VirtualNodeName?)

Compares the current instance with another object of the same type and returns an integer that indicates whether the current instance precedes, follows, or occurs in the same position in the sort order as the other object.

```
public int CompareTo(VirtualNodeName? other)
```

Parameters

other [VirtualNodeName](#)

An object to compare with this instance.

Returns

[int](#)

A value that indicates the relative order of the objects being compared. The return value has these meanings:

Value	Meaning
Less than zero	This instance precedes other in the sort order.

Value	Meaning
Zero	This instance occurs in the same position in the sort order as other .
Greater than zero	This instance follows other in the sort order.

CompareTo(object?)

Compares the current instance with another object of the same type and returns an integer that indicates whether the current instance precedes, follows, or occurs in the same position in the sort order as the other object.

```
public int CompareTo(object? obj)
```

Parameters

obj [object](#)

An object to compare with this instance.

Returns

[int](#)

A value that indicates the relative order of the objects being compared. The return value has these meanings:

Value	Meaning
Less than zero	This instance precedes obj in the sort order.
Zero	This instance occurs in the same position in the sort order as obj .
Greater than zero	This instance follows obj in the sort order.

Exceptions

[ArgumentException](#)

`obj` is not the same type as this instance.

Equals(VirtualNodeName?)

Indicates whether the current object is equal to another object of the same type.

```
public bool Equals(VirtualNodeName? other)
```

Parameters

`other` [VirtualNodeName](#)

An object to compare with this object.

Returns

[bool](#)

[true](#) if the current object is equal to the `other` parameter; otherwise, [false](#).

Equals(object?)

Determines whether the specified object is equal to the current object.

```
public override bool Equals(object? obj)
```

Parameters

`obj` [object](#)

The object to compare with the current object.

Returns

[bool](#)

[true](#) if the specified object is equal to the current object; otherwise, [false](#).

GenerateNodeName(string)

```
public static VirtualNodeName GenerateNodeName(string prefix)
```

Parameters

prefix [string](#)

Returns

[VirtualNodeName](#)

GetHashCode()

Serves as the default hash function.

```
public override int GetHashCode()
```

Returns

[int](#)

A hash code for the current object.

IsValidNodeName(VirtualNodeName)

```
public static bool IsValidNodeName(VirtualNodeName nodeName)
```

Parameters

nodeName [VirtualNodeName](#)

Returns

[bool](#)

ResetCounter()

```
public static void ResetCounter()
```

ToString()

Returns a string that represents the current object.

```
public override string ToString()
```

Returns

[string](#)

A string that represents the current object.

Operators

operator ==(VirtualNodeName?, VirtualNodeName?)

```
public static bool operator ==(VirtualNodeName? left, VirtualNodeName? right)
```

Parameters

left [VirtualNodeName](#)

right [VirtualNodeName](#)

Returns

[bool](#)

implicit operator string(VirtualNodeName)

```
public static implicit operator string(VirtualNodeName nodeName)
```

Parameters

nodeName [VirtualNodeName](#)

Returns

[string](#)

implicit operator VirtualNodeName(string)

```
public static implicit operator VirtualNodeName(string name)
```

Parameters

name [string](#)

Returns

[VirtualNodeName](#)

operator !=(VirtualNodeName?, VirtualNodeName?)

```
public static bool operator !=(VirtualNodeName? left, VirtualNodeName? right)
```

Parameters

left [VirtualNodeName](#)

right [VirtualNodeName](#)

Returns

[bool](#)

Class VirtualNodeNotFoundException

Namespace: [AkiraNetwork.VirtualStorageLibrary](#)

Assembly: VirtualStorageLibrary.dll

```
public class VirtualNodeNotFoundException : VirtualException, ISerializable
```

Inheritance

[object](#) < [Exception](#) < [VirtualException](#) < VirtualNodeNotFoundException

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](#), [Exception.SerializeObjectState](#), [object.Equals\(object\)](#), [object.Equals\(object, object\)](#), [object.GetHashCode\(\)](#), [object.MemberwiseClone\(\)](#), [object.ReferenceEquals\(object, object\)](#)

Extension Methods

[VirtualTextFormatter.GenerateSingleTableDebugText<T>\(T\)](#)

Constructors

VirtualNodeNotFoundException()

```
public VirtualNodeNotFoundException()
```

VirtualNodeNotFoundException(string)

```
public VirtualNodeNotFoundException(string message)
```

Parameters


message [string](#)

VirtualNodeNotFoundException(string, Exception)

```
public VirtualNodeNotFoundException(string message, Exception innerException)
```

Parameters

message [string](#) 

innerException [Exception](#) 

Enum VirtualNodeType

Namespace: [AkiraNetwork.VirtualStorageLibrary](#).

Assembly: VirtualStorageLibrary.dll

```
public enum VirtualNodeType
```

Extension Methods

[VirtualTextFormatter.GenerateSingleTableDebugText<T>\(T\)](#).

Fields

Directory = 0

Item = 1

None = 3

SymbolicLink = 2

Enum VirtualNodeTypeFilter

Namespace: [AkiraNetwork.VirtualStorageLibrary](#)

Assembly: VirtualStorageLibrary.dll

```
public enum VirtualNodeTypeFilter
```

Extension Methods

[VirtualTextFormatter.GenerateSingleTableDebugText<T>\(T\)](#)

Fields

All = 7

Directory = 2

Item = 1

None = 0

SymbolicLink = 4

Class VirtualPath

Namespace: [AkiraNetwork.VirtualStorageLibrary](#)

Assembly: VirtualStorageLibrary.dll

```
public class VirtualPath : IEquatable<VirtualPath>, IComparable<VirtualPath>, IComparable
```




Inheritance

[object](#)  ← VirtualPath

Implements

[IEquatable](#)  <[VirtualPath](#)>, [IComparable](#)  <[VirtualPath](#)>, [IComparable](#) 

Inherited Members

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

Extension Methods

[VirtualTextFormatter.GenerateSingleTableDebugText<T>\(T\)](#)

Constructors

VirtualPath(IEnumerable<VirtualNodeName>)

```
public VirtualPath(IEnumerable<VirtualNodeName> parts)
```

Parameters

parts [IEnumerable](#)  <[VirtualNodeName](#)>

VirtualPath(string)

```
public VirtualPath(string path)
```

Parameters

path [string](#)

Properties

BaseDepth

```
public int BaseDepth { get; }
```

Property Value

[int](#)

Depth

```
public int Depth { get; }
```

Property Value

[int](#)

DirectoryPath

```
public VirtualPath DirectoryPath { get; }
```

Property Value

[VirtualPath](#)

Dot

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

Property Value

[string](#)

DotDot

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

Property Value

[string](#)

FixedPath

```
public VirtualPath FixedPath { get; }
```

Property Value

[VirtualPath](#)

IsAbsolute

```
public bool IsAbsolute { get; }
```

Property Value

[bool](#)

IsDot

```
public bool IsDot { get; }
```

Property Value

[bool](#)

IsDotDot

```
public bool IsDotDot { get; }
```

Property Value

[bool](#)

IsEmpty

```
public bool IsEmpty { get; }
```

Property Value

[bool](#)

IsEndsWithSlash

```
public bool IsEndsWithSlash { get; }
```

Property Value

[bool](#)

IsRoot

```
public bool IsRoot { get; }
```

Property Value

[bool](#)

NodeName

```
public VirtualNodeName NodeName { get; }
```

Property Value

[VirtualNodeName](#)

PartsList

```
public List<VirtualNodeName> PartsList { get; }
```

Property Value

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

Path

```
public string Path { get; }
```

Property Value

[string](#)

Root

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

Property Value

[string](#)

Separator

```
public static char Separator { get; }
```

Property Value

[char](#)

Methods

AddEndSlash()

```
public VirtualPath AddEndSlash()
```

Returns

[VirtualPath](#)

AddStartSlash()

```
public VirtualPath AddStartSlash()
```

Returns

[VirtualPath](#)

ArePathsSubdirectories(VirtualPath, VirtualPath)

```
public static bool ArePathsSubdirectories(VirtualPath path1, VirtualPath path2)
```

Parameters

path1 [VirtualPath](#)

path2 [VirtualPath](#)

Returns

[bool](#)

Combine(params VirtualPath[])

```
public VirtualPath Combine(params VirtualPath[] paths)
```

Parameters

paths [VirtualPath\[\]](#)

Returns

[VirtualPath](#)

Combine(params string[])

```
public static string Combine(params string[] paths)
```

Parameters

paths [string\[\]](#)

Returns

[string](#)

CombineFromIndex(VirtualPath, int)

```
public VirtualPath CombineFromIndex(VirtualPath path, int index)
```

Parameters

path [VirtualPath](#)

[index](#) [int](#)

Returns

[VirtualPath](#)

CompareTo(VirtualPath?)

Compares the current instance with another object of the same type and returns an integer that indicates whether the current instance precedes, follows, or occurs in the same position in the sort order as the other object.

```
public int CompareTo(VirtualPath? other)
```

Parameters

other [VirtualPath](#)

An object to compare with this instance.

Returns

[int](#)

A value that indicates the relative order of the objects being compared. The return value has these meanings:

Value	Meaning
Less than zero	This instance precedes other in the sort order.
Zero	This instance occurs in the same position in the sort order as other .
Greater than zero	This instance follows other in the sort order.

CompareTo(object?)

Compares the current instance with another object of the same type and returns an integer that indicates whether the current instance precedes, follows, or occurs in the same position in the sort order

as the other object.

```
public int CompareTo(object? obj)
```

Parameters

obj [object](#)

An object to compare with this instance.

Returns

[int](#)

A value that indicates the relative order of the objects being compared. The return value has these meanings:

Value	Meaning
Less than zero	This instance precedes obj in the sort order.
Zero	This instance occurs in the same position in the sort order as obj .
Greater than zero	This instance follows obj in the sort order.

Exceptions

[ArgumentException](#)

obj is not the same type as this instance.

Equals(VirtualPath?)

Indicates whether the current object is equal to another object of the same type.

```
public bool Equals(VirtualPath? other)
```

Parameters

other [VirtualPath](#)

An object to compare with this object.

Returns

[bool](#)

[true](#) if the current object is equal to the `other` parameter; otherwise, [false](#).

Equals(object?)

Determines whether the specified object is equal to the current object.

```
public override bool Equals(object? obj)
```

Parameters

`obj` [object](#)

The object to compare with the current object.

Returns

[bool](#)

[true](#) if the specified object is equal to the current object; otherwise, [false](#).

GetHashCode()

Serves as the default hash function.

```
public override int GetHashCode()
```

Returns

[int](#)

A hash code for the current object.

GetNodeName()

```
public VirtualNodeName GetNodeName()
```

Returns

[VirtualNodeName](#)

GetParentPath()

```
public VirtualPath GetParentPath()
```

Returns

[VirtualPath](#)

GetPartsLinkedList()

```
public LinkedList<VirtualNodeName> GetPartsLinkedList()
```

Returns

[LinkedList](#) <[VirtualNodeName](#)>

GetPartsList()

```
public List<VirtualNodeName> GetPartsList()
```

Returns

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

GetRelativePath(VirtualPath)

```
public VirtualPath GetRelativePath(VirtualPath basePath)
```

Parameters

basePath [VirtualPath](#)

Returns

[VirtualPath](#)

IsSubdirectory(VirtualPath)

```
public bool IsSubdirectory(VirtualPath parentPath)
```

Parameters

parentPath [VirtualPath](#)

Returns

[bool](#)

NormalizePath()

```
public VirtualPath NormalizePath()
```

Returns

[VirtualPath](#)

NormalizePath(string)

```
public static string NormalizePath(string path)
```

Parameters

path [string](#)

Returns

[string](#)

StartsWith(VirtualPath)

```
public bool StartsWith(VirtualPath path)
```

Parameters

path [VirtualPath](#)

Returns

[bool](#)

ToString()

Returns a string that represents the current object.

```
public override string ToString()
```

Returns

[string](#)

A string that represents the current object.

TrimEndSlash()

```
public VirtualPath TrimEndSlash()
```

Returns

[VirtualPath](#)

Operators

operator +(VirtualPath, VirtualNodeName)

```
public static VirtualPath operator +(VirtualPath path, VirtualNodeName nodeName)
```

Parameters

path [VirtualPath](#)

nodeName [VirtualNodeName](#)

Returns

[VirtualPath](#)

operator +(VirtualPath, VirtualPath)

```
public static VirtualPath operator +(VirtualPath path1, VirtualPath path2)
```

Parameters

path1 [VirtualPath](#)

path2 [VirtualPath](#)

Returns

[VirtualPath](#)

operator +(VirtualPath, char)

```
public static VirtualPath operator +(VirtualPath path, char chr)
```

Parameters

path [VirtualPath](#)

chr [char](#)

Returns

[VirtualPath](#)

operator +(VirtualPath, string)

```
public static VirtualPath operator +(VirtualPath path, string str)
```

Parameters

path [VirtualPath](#)

str [string](#)

Returns

[VirtualPath](#)

operator +(char, VirtualPath)

```
public static VirtualPath operator +(char chr, VirtualPath path)
```

Parameters

chr [char](#)

path [VirtualPath](#)

Returns

[VirtualPath](#)

operator +(string, VirtualPath)

```
public static VirtualPath operator +(string str, VirtualPath path)
```

Parameters

str [string](#)

path [VirtualPath](#)

Returns

[VirtualPath](#)

operator ==(VirtualPath?, VirtualPath?)

```
public static bool operator ==(VirtualPath? left, VirtualPath? right)
```

Parameters

left [VirtualPath](#)

right [VirtualPath](#)

Returns

[bool](#)

implicit operator string(VirtualPath?)

```
public static implicit operator string(VirtualPath? virtualPath)
```

Parameters

`virtualPath` [VirtualPath](#)

Returns

[string](#)

implicit operator VirtualPath(string)

```
public static implicit operator VirtualPath(string path)
```

Parameters

`path` [string](#)

Returns

[VirtualPath](#)

operator !=(VirtualPath?, VirtualPath?)

```
public static bool operator !=(VirtualPath? left, VirtualPath? right)
```

Parameters

`left` [VirtualPath](#)

`right` [VirtualPath](#)

Returns

[bool](#)

Class VirtualSortCondition<T>

Namespace: [AkiraNetwork.VirtualStorageLibrary](#)

Assembly: VirtualStorageLibrary.dll

```
public class VirtualSortCondition<T>
```

Type Parameters

T

Inheritance

[object](#) ← VirtualSortCondition<T>

Inherited Members

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

Extension Methods

[VirtualTextFormatter.GenerateSingleTableDebugText<T>\(T\)](#)

Constructors

VirtualSortCondition(Expression<Func<T, object>>, bool)

```
public VirtualSortCondition(Expression<Func<T, object>> sortBy, bool ascending = true)
```

Parameters

sortBy [Expression](#) <[Func](#) <T, [object](#)>>

ascending [bool](#)

Properties

Ascending

```
public bool Ascending { get; set; }
```

Property Value

[bool](#)

SortBy

```
public Expression<Func<T, object>> SortBy { get; set; }
```

Property Value

[Expression](#) <[Func](#) <T, [object](#)>>

Class VirtualStorageExtensions

Namespace: [AkiraNetwork.VirtualStorageLibrary](#)

Assembly: VirtualStorageLibrary.dll

```
public static class VirtualStorageExtensions
```

Inheritance

[object](#) ← VirtualStorageExtensions

Inherited Members

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

Methods

ApplySortConditions<T>(IEnumerable<T>, List<VirtualSortCondition<T>>?)

```
public static IEnumerable<T> ApplySortConditions<T>(this IEnumerable<T> source, List<VirtualSortCondition<T>>? sortConditions = null)
```

Parameters

source [IEnumerable](#)<T>

sortConditions [List](#)<[VirtualSortCondition](#)<T>>

Returns

[IEnumerable](#)<T>

Type Parameters

T

GroupAndSort<T>(IEnumerable<T>, VirtualGroupCondition<T, object>?, List<VirtualSortCondition<T>>?)

```
public static IEnumerable<T> GroupAndSort<T>(this IEnumerable<T> source,  
VirtualGroupCondition<T, object>? groupCondition = null, List<VirtualSortCondition<T>>?  
sortConditions = null)
```

Parameters

source [IEnumerable](#)<T>

groupCondition [VirtualGroupCondition](#)<T, [object](#)>

sortConditions [List](#)<[VirtualSortCondition](#)<T>>

Returns

[IEnumerable](#)<T>

Type Parameters

T


Class VirtualStorageSettings

Namespace: [AkiraNetwork.VirtualStorageLibrary](#).








Assembly: VirtualStorageLibrary.dll

```
public class VirtualStorageSettings
```

Inheritance

[object](#)  ← VirtualStorageSettings

Inherited Members

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

Extension Methods

[VirtualTextFormatter.GenerateSingleTableDebugText<T>\(T\)](#)

Properties

InvalidFullNodeNames

```
public string[] InvalidFullNodeNames { get; set; }
```

Property Value

[string](#)  []

InvalidNodeNameCharacters

```
public char[] InvalidNodeNameCharacters { get; set; }
```

Property Value

[char](#)  []

NodeListConditions

```
public VirtualNodeListConditions NodeListConditions { get; set; }
```

Property Value

[VirtualNodeListConditions](#)

PathDot

```
public string PathDot { get; set; }
```

Property Value

[string](#) 

PathDotDot

```
public string PathDotDot { get; set; }
```

Property Value

[string](#) 

PathRoot

```
public string PathRoot { get; set; }
```

Property Value

[string](#) 

PathSeparator


```
public char PathSeparator { get; set; }
```

Property Value

[char](#)

PrefixDirectory

```
public string PrefixDirectory { get; set; }
```

Property Value

[string](#)

PrefixItem

```
public string PrefixItem { get; set; }
```

Property Value

[string](#)

PrefixSymbolicLink

```
public string PrefixSymbolicLink { get; set; }
```

Property Value

[string](#)

Settings

```
public static VirtualStorageSettings Settings { get; }
```

Property Value

[VirtualStorageSettings](#)

WildcardMatcher

```
public IVirtualWildcardMatcher? WildcardMatcher { get; set; }
```

Property Value

[IVirtualWildcardMatcher](#)

Methods

Initialize()

```
public static void Initialize()
```


Class VirtualStorageState

Namespace: [AkiraNetwork.VirtualStorageLibrary](#).








Assembly: VirtualStorageLibrary.dll

```
public class VirtualStorageState
```

Inheritance

[object](#)  ← VirtualStorageState

Inherited Members

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

Extension Methods

[VirtualTextFormatter.GenerateSingleTableDebugText<T>\(T\)](#)

Properties

InvalidFullNodeNames

```
public string[] InvalidFullNodeNames { get; set; }
```

Property Value

[string](#) []

InvalidNodeNameCharacters

```
public char[] InvalidNodeNameCharacters { get; set; }
```

Property Value

[char](#) []

NodeListConditions

```
public VirtualNodeListConditions NodeListConditions { get; set; }
```

Property Value

[VirtualNodeListConditions](#)

PathDot

```
public string PathDot { get; set; }
```

Property Value

[string](#) 

PathDotDot

```
public string PathDotDot { get; set; }
```

Property Value

[string](#) 

PathRoot

```
public string PathRoot { get; set; }
```

Property Value

[string](#) 

PathSeparator

```
public char PathSeparator { get; set; }
```

Property Value

[char](#)

PrefixDirectory

```
public string PrefixDirectory { get; set; }
```

Property Value

[string](#)

PrefixItem

```
public string PrefixItem { get; set; }
```

Property Value

[string](#)

PrefixSymbolicLink

```
public string PrefixSymbolicLink { get; set; }
```

Property Value

[string](#)

State

```
public static VirtualStorageState State { get; }
```

Property Value

[VirtualStorageState](#)

WildcardMatcher

```
public IVirtualWildcardMatcher? WildcardMatcher { get; set; }
```

Property Value

[IVirtualWildcardMatcher](#)

Methods

SetNodeListConditions(VirtualNodeListConditions)

```
public static void SetNodeListConditions(VirtualNodeListConditions conditions)
```

Parameters

conditions [VirtualNodeListConditions](#)

SetNodeListConditions(VirtualNodeTypeFilter, VirtualGroupCondition<VirtualNode, object>?, List<VirtualSortCondition<VirtualNode>>?)

```
public static void SetNodeListConditions(VirtualNodeTypeFilter filter,  
VirtualGroupCondition<VirtualNode, object>? groupCondition = null,  
List<VirtualSortCondition<VirtualNode>>? sortConditions = null)
```

Parameters

filter [VirtualNodeTypeFilter](#)

groupCondition [VirtualGroupCondition](#) <[VirtualNode](#), [object](#) >

sortConditions [List](#) <[VirtualSortCondition](#) <[VirtualNode](#)> >

Class VirtualStorage<T>

Namespace: [AkiraNetwork.VirtualStorageLibrary](#).

Assembly: VirtualStorageLibrary.dll

```
public class VirtualStorage<T>
```








Type Parameters

T




Inheritance

[object](#)  ← VirtualStorage<T>

Inherited Members

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

Extension Methods

[VirtualTextFormatter.GenerateLinkTableDebugText<T>\(VirtualStorage<T>\)](#) ,
[VirtualTextFormatter.GenerateSingleTableDebugText<T>\(T\)](#) ,
[VirtualTextFormatter.GenerateTreeDebugText<T>\(VirtualStorage<T>, VirtualPath, bool, bool\)](#) 

Constructors

VirtualStorage()

```
public VirtualStorage()
```

Properties

CurrentPath

```
public VirtualPath CurrentPath { get; }
```


Property Value

[VirtualPath](#)

CycleDetectorForTarget

```
public VirtualCycleDetector CycleDetectorForTarget { get; }
```

Property Value

[VirtualCycleDetector](#)

CycleDetectorForTree

```
public VirtualCycleDetector CycleDetectorForTree { get; }
```

Property Value

[VirtualCycleDetector](#)

Dir

```
public VirtualDirectoryAdapter<T> Dir { get; }
```

Property Value

[VirtualDirectoryAdapter](#)<T>

this[VirtualPath, bool]

```
public VirtualNode this[VirtualPath path, bool followLinks = true] { get; set; }
```

Parameters

path [VirtualPath](#)

followLinks [bool](#)[↗]

Property Value

[VirtualNode](#)

Item

```
public VirtualItemAdapter<T> Item { get; }
```

Property Value

[VirtualItemAdapter](#)<T>

Link

```
public VirtualSymbolicLinkAdapter<T> Link { get; }
```

Property Value

[VirtualSymbolicLinkAdapter](#)<T>

LinkDictionary

```
public Dictionary<VirtualPath, HashSet<VirtualPath>> LinkDictionary { get; }
```

Property Value

[Dictionary](#)[↗] <[VirtualPath](#), [HashSet](#)[↗] <[VirtualPath](#)>>

Root

```
public VirtualDirectory Root { get; }
```

Property Value

[VirtualDirectory](#)

Methods

AddDirectory(VirtualPath, VirtualDirectory, bool)

```
public void AddDirectory(VirtualPath directoryPath, VirtualDirectory directory, bool  
createSubdirectories = false)
```

Parameters

directoryPath [VirtualPath](#)

directory [VirtualDirectory](#)

createSubdirectories [bool](#)

AddDirectory(VirtualPath, bool)

```
public void AddDirectory(VirtualPath path, bool createSubdirectories = false)
```

Parameters

path [VirtualPath](#)

createSubdirectories [bool](#)

AddItem(VirtualPath, VirtualItem<T>, bool)

```
public void AddItem(VirtualPath itemDirectoryPath, VirtualItem<T> item, bool overwrite  
= false)
```

Parameters

itemDirectoryPath [VirtualPath](#)

item [VirtualItem](#)<T>

overwrite [bool](#)[↗](#)

AddItem(VirtualPath, T?, bool)

```
public void AddItem(VirtualPath itemPath, T? data = default, bool overwrite = false)
```

Parameters

itemPath [VirtualPath](#)

data T

overwrite [bool](#)[↗](#)

AddLinkToDictionary(VirtualPath, VirtualPath)

```
public void AddLinkToDictionary(VirtualPath targetPath, VirtualPath linkPath)
```

Parameters

targetPath [VirtualPath](#)

linkPath [VirtualPath](#)

AddNode(VirtualPath, VirtualNode, bool)

```
public void AddNode(VirtualPath nodeDirectoryPath, VirtualNode node, bool overwrite = false)
```

Parameters

nodeDirectoryPath [VirtualPath](#)

node [VirtualNode](#)

overwrite [bool](#)

AddSymbolicLink(VirtualPath, VirtualPath?, bool)

```
public void AddSymbolicLink(VirtualPath linkPath, VirtualPath? targetPath = null, bool  
    overwrite = false)
```

Parameters

linkPath [VirtualPath](#)

targetPath [VirtualPath](#)

overwrite [bool](#)

AddSymbolicLink(VirtualPath, VirtualSymbolicLink, bool)

```
public void AddSymbolicLink(VirtualPath linkDirectoryPath, VirtualSymbolicLink link, bool  
    overwrite = false)
```

Parameters

linkDirectoryPath [VirtualPath](#)

link [VirtualSymbolicLink](#)

overwrite [bool](#)

ChangeDirectory(VirtualPath)

```
public void ChangeDirectory(VirtualPath path)
```

Parameters

path [VirtualPath](#)

ConvertToAbsolutePath(VirtualPath?, VirtualPath?)

```
public VirtualPath ConvertToAbsolutePath(VirtualPath? relativePath, VirtualPath? basePath  
= null)
```

Parameters

relativePath [VirtualPath](#)

basePath [VirtualPath](#)

Returns

[VirtualPath](#)

CopyNode(VirtualPath, VirtualPath, bool, bool, bool, List<VirtualNodeContext>?)

```
public void CopyNode(VirtualPath sourcePath, VirtualPath destinationPath, bool overwrite =  
false, bool recursive = false, bool followLinks = false, List<VirtualNodeContext>?  
destinationContextList = null)
```

Parameters

sourcePath [VirtualPath](#)

destinationPath [VirtualPath](#)

overwrite [bool](#) 

recursive [bool](#) 

followLinks [bool](#) 

destinationContextList [List](#)  <[VirtualNodeContext](#)>

DirectoryExists(VirtualPath, bool)

```
public bool DirectoryExists(VirtualPath path, bool followLinks = false)
```

Parameters

path [VirtualPath](#)

followLinks [bool](#)

Returns

[bool](#)

ExpandPath(VirtualPath, VirtualNodeTypeFilter, bool, bool)


```
public IEnumerable<VirtualPath> ExpandPath(VirtualPath path, VirtualNodeTypeFilter filter =  
VirtualNodeTypeFilter.All, bool followLinks = true, bool resolveLinks = true)
```

Parameters

path [VirtualPath](#)

filter [VirtualNodeTypeFilter](#)

followLinks [bool](#)

resolveLinks [bool](#)

Returns

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

ExpandPathTree(VirtualPath, VirtualNodeTypeFilter, bool, bool)


```
public IEnumerable<VirtualNodeContext> ExpandPathTree(VirtualPath path,  
VirtualNodeTypeFilter filter = VirtualNodeTypeFilter.All, bool followLinks = true, bool  
resolveLinks = true)
```

Parameters

path [VirtualPath](#)

filter [VirtualNodeTypeFilter](#)

followLinks [bool](#)

resolveLinks [bool](#)

Returns

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

GetDirectory(VirtualPath, bool)

```
public VirtualDirectory GetDirectory(VirtualPath path, bool followLinks = false)
```

Parameters

path [VirtualPath](#)

followLinks [bool](#)

Returns

[VirtualDirectory](#)

GetItem(VirtualPath, bool)

```
public VirtualItem<T> GetItem(VirtualPath path, bool followLinks = false)
```

Parameters

path [VirtualPath](#)

followLinks [bool](#)

Returns

[VirtualItem](#)<T>

GetLinksFromDictionary(VirtualPath)

```
public HashSet<VirtualPath> GetLinksFromDictionary(VirtualPath targetPath)
```

Parameters

targetPath [VirtualPath](#)

Returns

[HashSet](#) [<VirtualPath>](#)

GetNode(VirtualPath, bool)

```
public VirtualNode GetNode(VirtualPath path, bool followLinks = false)
```

Parameters

path [VirtualPath](#)

followLinks [bool](#)

Returns

[VirtualNode](#)

GetNodeType(VirtualPath, bool)

```
public VirtualNodeType GetNodeType(VirtualPath path, bool followLinks = false)
```

Parameters

path [VirtualPath](#)

followLinks [bool](#)

Returns

[VirtualNodeType](#)

GetNodes(VirtualNodeTypeFilter, bool, bool)

```
public IEnumerable<VirtualNode> GetNodes(VirtualNodeTypeFilter nodeType =  
VirtualNodeTypeFilter.All, bool recursive = false, bool followLinks = false)
```

Parameters

nodeType [VirtualNodeTypeFilter](#)

recursive [bool](#)

followLinks [bool](#)

Returns

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

GetNodes(VirtualPath, VirtualNodeTypeFilter, bool, bool)

```
public IEnumerable<VirtualNode> GetNodes(VirtualPath basePath, VirtualNodeTypeFilter  
nodeType = VirtualNodeTypeFilter.All, bool recursive = false, bool followLinks = false)
```

Parameters

basePath [VirtualPath](#)

nodeType [VirtualNodeTypeFilter](#)

recursive [bool](#)

followLinks [bool](#)

Returns

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

GetNodesWithPaths(VirtualNodeTypeFilter, bool, bool)

```
public IEnumerable<VirtualPath> GetNodesWithPaths(VirtualNodeTypeFilter nodeType =  
VirtualNodeTypeFilter.All, bool recursive = false, bool followLinks = false)
```

Parameters

nodeType [VirtualNodeTypeFilter](#)

recursive [bool](#)

followLinks [bool](#)

Returns

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

GetNodesWithPaths(VirtualPath, VirtualNodeTypeFilter, bool, bool)

```
public IEnumerable<VirtualPath> GetNodesWithPaths(VirtualPath basePath,  
VirtualNodeTypeFilter nodeType = VirtualNodeTypeFilter.All, bool recursive = false, bool  
followLinks = false)
```

Parameters

basePath [VirtualPath](#)

nodeType [VirtualNodeTypeFilter](#)

recursive [bool](#)

followLinks [bool](#)

Returns

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

GetSymbolicLink(VirtualPath)

```
public VirtualSymbolicLink GetSymbolicLink(VirtualPath path)
```

Parameters

path [VirtualPath](#)

Returns

[VirtualSymbolicLink](#)

ItemExists(VirtualPath, bool)

```
public bool ItemExists(VirtualPath path, bool followLinks = false)
```

Parameters

path [VirtualPath](#)

followLinks [bool](#) 

Returns

[bool](#) 

MoveNode(VirtualPath, VirtualPath, bool, bool)


```
public void MoveNode(VirtualPath sourcePath, VirtualPath destinationPath, bool overwrite = false, bool resolveLinks = true)
```

Parameters

sourcePath [VirtualPath](#)

destinationPath [VirtualPath](#)

overwrite [bool](#)

resolveLinks [bool](#)

NodeExists(VirtualPath, bool)

```
public bool NodeExists(VirtualPath path, bool followLinks = false)
```

Parameters

path [VirtualPath](#)

followLinks [bool](#)

Returns

[bool](#)

RemoveLinkByLinkPath(VirtualPath)

```
public void RemoveLinkByLinkPath(VirtualPath linkPath)
```

Parameters

linkPath [VirtualPath](#)

RemoveLinkFromDictionary(VirtualPath, VirtualPath)

```
public void RemoveLinkFromDictionary(VirtualPath targetPath, VirtualPath linkPath)
```

Parameters

targetPath [VirtualPath](#)

linkPath [VirtualPath](#)

RemoveNode(VirtualPath, bool, bool, bool)


```
public void RemoveNode(VirtualPath nodePath, bool recursive = false, bool followLinks = false, bool resolveLinks = true)
```

Parameters

nodePath [VirtualPath](#)

recursive [bool](#)

followLinks [bool](#)

resolveLinks [bool](#)

ResolveLinkTarget(VirtualPath)

```
public VirtualPath ResolveLinkTarget(VirtualPath path)
```

Parameters

path [VirtualPath](#)

Returns

[VirtualPath](#)

SetLinkTargetNodeType(HashSet<VirtualPath>, VirtualNodeType)

```
public void SetLinkTargetNodeType(HashSet<VirtualPath> linkPathSet, VirtualNodeType nodeType)
```

Parameters

linkPathSet [HashSet](#) <[VirtualPath](#)>

nodeType [VirtualNodeType](#)

SetNode(VirtualPath, VirtualNode)

```
public void SetNode(VirtualPath destinationPath, VirtualNode node)
```

Parameters

destinationPath [VirtualPath](#)

node [VirtualNode](#)

SetNodeName(VirtualPath, VirtualNodeName, bool)

```
public void SetNodeName(VirtualPath nodePath, VirtualNodeName newName, bool resolveLinks  
= true)
```

Parameters

nodePath [VirtualPath](#)

newName [VirtualNodeName](#)

resolveLinks [bool](#)

SymbolicLinkExists(VirtualPath)

```
public bool SymbolicLinkExists(VirtualPath path)
```

Parameters

path [VirtualPath](#)

Returns

[bool](#)[↗]

TryGetDirectory(VirtualPath, bool)

```
public VirtualDirectory? TryGetDirectory(VirtualPath path, bool followLinks = false)
```

Parameters

path [VirtualPath](#)

followLinks [bool](#)[↗]

Returns

[VirtualDirectory](#)

TryGetItem(VirtualPath, bool)

```
public VirtualItem<T>? TryGetItem(VirtualPath path, bool followLinks = false)
```

Parameters

path [VirtualPath](#)

followLinks [bool](#)[↗]

Returns

[VirtualItem](#)<T>

TryGetNode(VirtualPath, bool)

```
public VirtualNode? TryGetNode(VirtualPath path, bool followLinks = false)
```

Parameters

path [VirtualPath](#)

followLinks [bool](#)

Returns

[VirtualNode](#)

TryGetSymbolicLink(VirtualPath)

```
public VirtualSymbolicLink? TryGetSymbolicLink(VirtualPath path)
```

Parameters

path [VirtualPath](#)

Returns

[VirtualSymbolicLink](#)

TryResolveLinkTarget(VirtualPath)

```
public VirtualPath? TryResolveLinkTarget(VirtualPath path)
```

Parameters

path [VirtualPath](#)

Returns

[VirtualPath](#)

UpdateAllTargetNodeTypesInDictionary()

```
public void UpdateAllTargetNodeTypesInDictionary()
```

UpdateDirectory(VirtualPath, VirtualDirectory)

```
public void UpdateDirectory(VirtualPath directoryPath, VirtualDirectory newDirectory)
```

Parameters

directoryPath [VirtualPath](#)

newDirectory [VirtualDirectory](#)

UpdateItem(VirtualPath, VirtualItem<T>)

```
public void UpdateItem(VirtualPath itemPath, VirtualItem<T> newItem)
```

Parameters

itemPath [VirtualPath](#)

newItem [VirtualItem](#)<T>

UpdateLinkInDictionary(VirtualPath, VirtualPath)

```
public void UpdateLinkInDictionary(VirtualPath linkPath, VirtualPath newTargetPath)
```

Parameters

linkPath [VirtualPath](#)

newTargetPath [VirtualPath](#)

UpdateLinksToTarget(VirtualPath, VirtualPath)

```
public void UpdateLinksToTarget(VirtualPath oldTargetPath, VirtualPath newTargetPath)
```

Parameters

oldTargetPath [VirtualPath](#)

newTargetPath [VirtualPath](#)

UpdateSymbolicLink(VirtualPath, VirtualSymbolicLink)

```
public void UpdateSymbolicLink(VirtualPath linkPath, VirtualSymbolicLink newLink)
```

Parameters

linkPath [VirtualPath](#)

newLink [VirtualSymbolicLink](#)

UpdateTargetNodeTypesInDictionary(VirtualPath)

```
public void UpdateTargetNodeTypesInDictionary(VirtualPath targetPath)
```

Parameters

targetPath [VirtualPath](#)

WalkPathToTarget(VirtualPath, NotifyNodeDelegate?, ActionNodeDelegate?, bool, bool)

```
public VirtualNodeContext WalkPathToTarget(VirtualPath targetPath, NotifyNodeDelegate?  
notifyNode = null, ActionNodeDelegate? actionNode = null, bool followLinks = true, bool  
exceptionEnabled = true)
```

Parameters

targetPath [VirtualPath](#)

notifyNode [NotifyNodeDelegate](#)

actionNode [ActionNodeDelegate](#)

followLinks [bool](#)

exceptionEnabled [bool](#)

Returns

[VirtualNodeContext](#)

WalkPathTree(VirtualPath, VirtualNodeTypeFilter, bool, bool, bool)

```
public IEnumerable<VirtualNodeContext> WalkPathTree(VirtualPath basePath,
VirtualNodeTypeFilter filter = VirtualNodeTypeFilter.All, bool recursive = true, bool
followLinks = true, bool resolveLinks = true)
```

Parameters

basePath [VirtualPath](#)

filter [VirtualNodeTypeFilter](#)

recursive [bool](#)

followLinks [bool](#)

resolveLinks [bool](#)

Returns

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

Class VirtualSymbolicLink

Namespace: [AkiraNetwork.VirtualStorageLibrary](#)

Assembly: VirtualStorageLibrary.dll

```
public class VirtualSymbolicLink : VirtualNode, IVirtualDeepCloneable<VirtualNode>
```







Inheritance

[object](#)  ← [VirtualNode](#) ← VirtualSymbolicLink

Implements

[IVirtualDeepCloneable](#) <[VirtualNode](#)>

Inherited Members

[VirtualNode.Name](#) , [VirtualNode.CreatedDate](#) , [VirtualNode.UpdatedDate](#) , [VirtualNode.VID](#) , [VirtualNode.IsReferencedInStorage](#) , [object.Equals\(object\)](#)  , [object.Equals\(object, object\)](#)  , [object.GetHashCode\(\)](#)  , [object.GetType\(\)](#)  , [object.MemberwiseClone\(\)](#)  , [object.ReferenceEquals\(object, object\)](#) 

Extension Methods

[VirtualNodeExtensions.ResolveNodeType\(VirtualNode\)](#) ,
[VirtualTextFormatter.GenerateSingleTableDebugText<T>\(T\)](#)

Constructors

VirtualSymbolicLink()

```
public VirtualSymbolicLink()
```

VirtualSymbolicLink(VirtualNodeName)

```
public VirtualSymbolicLink(VirtualNodeName name)
```

Parameters

name [VirtualNodeName](#)

VirtualSymbolicLink(VirtualNodeName, VirtualPath?)

```
public VirtualSymbolicLink(VirtualNodeName name, VirtualPath? targetPath)
```

Parameters

name [VirtualNodeName](#)

targetPath [VirtualPath](#)

VirtualSymbolicLink(VirtualNodeName, VirtualPath?, DateTime, DateTime)

```
public VirtualSymbolicLink(VirtualNodeName name, VirtualPath? targetPath, DateTime  
    createdAt, DateTime updatedAt)
```

Parameters

name [VirtualNodeName](#)

targetPath [VirtualPath](#)

createdAt [DateTime](#)↗

updatedAt [DateTime](#)↗

Properties

NodeType

Gets the node type of node.

```
public override VirtualNodeType NodeType { get; }
```

Property Value

[VirtualNodeType](#)

TargetNodeType

```
public VirtualNodeType TargetNodeType { get; set; }
```

Property Value

[VirtualNodeType](#)

TargetPath

```
public VirtualPath? TargetPath { get; set; }
```

Property Value

[VirtualPath](#)

Methods

DeepClone(bool)

Creates a deep clone of the entity. However, the CreatedDate and UpdatedDate should not be cloned as they are set to the current date and time at the time of cloning.

```
public override VirtualNode DeepClone(bool recursive = false)
```

Parameters

recursive [bool](#)

When true, all child nodes are also cloned, creating a deep copy of the entire tree. The default is false. The CreatedDate and UpdatedDate properties are not preserved. They are set to the current date and time at the moment of instantiation or cloning.

Returns

[VirtualNode](#)

Cloned [VirtualNode](#) instance

ToString()

Returns a string that represents the current object.

```
public override string ToString()
```

Returns

[string](#) 

A string that represents the current object.

Update(VirtualNode)

Updates the [VirtualNode](#).

```
public override void Update(VirtualNode node)
```

Parameters

node [VirtualNode](#)

Value to update

Operators

implicit operator VirtualSymbolicLink(VirtualPath?)

```
public static implicit operator VirtualSymbolicLink(VirtualPath? targetPath)
```

Parameters

targetPath [VirtualPath](#)

Returns

[VirtualSymbolicLink](#)

implicit operator VirtualSymbolicLink((VirtualNodeName nodeName, VirtualPath? targetPath))

```
public static implicit operator VirtualSymbolicLink((VirtualNodeName nodeName, VirtualPath? targetPath) tuple)
```

Parameters

tuple ([VirtualNodeName nodeName](#)[↗], [VirtualPath targetPath](#)[↗])

Returns

[VirtualSymbolicLink](#)

Class VirtualSymbolicLinkAdapter<T>

Namespace: [AkiraNetwork.VirtualStorageLibrary](#)

Assembly: VirtualStorageLibrary.dll

```
public class VirtualSymbolicLinkAdapter<T>
```








Type Parameters

T

Inheritance

[object](#)  ← VirtualSymbolicLinkAdapter<T>

Inherited Members

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

Extension Methods

[VirtualTextFormatter.GenerateSingleTableDebugText<T>\(T\)](#)

Constructors

VirtualSymbolicLinkAdapter(VirtualStorage<T>)

```
public VirtualSymbolicLinkAdapter(VirtualStorage<T> storage)
```

Parameters

storage [VirtualStorage](#)<T>

Properties

this[VirtualPath, bool]

```
public VirtualSymbolicLink this[VirtualPath path, bool followLinks = true] { get; set; }
```

Parameters

path [VirtualPath](#)

followLinks [bool](#) 

Property Value

[VirtualSymbolicLink](#)

Namespace AkiraNetwork.VirtualStorage Library.Utilities

Classes

[VirtualTextFormatter](#)


Class VirtualTextFormatter

Namespace: [AkiraNetwork.VirtualStorageLibrary.Utilities](#)








Assembly: VirtualStorageLibrary.dll

```
public static class VirtualTextFormatter
```

Inheritance

[object](#)  ← VirtualTextFormatter

Inherited Members

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

Methods

GenerateLinkTableDebugText<T>(VirtualStorage<T>)

```
public static string GenerateLinkTableDebugText<T>(this VirtualStorage<T> vs)
```

Parameters

vs [VirtualStorage](#)<T>

Returns

[string](#) 

Type Parameters

T

GenerateSingleTableDebugText<T>(T)

```
public static string GenerateSingleTableDebugText<T>(this T singleObject)
```

Parameters

`singleObject T`

Returns

`string` [↗](#)

Type Parameters

`T`

GenerateTableDebugText<T>(IEnumerable<T>)

```
public static string GenerateTableDebugText<T>(this IEnumerable<T> enumerableObject)
```

Parameters

`enumerableObject IEnumerable ↗ <T>`

Returns

`string` [↗](#)

Type Parameters

`T`

GenerateTreeDebugText<T>(VirtualStorage<T>, VirtualPath, bool, bool)

```
public static string GenerateTreeDebugText<T>(this VirtualStorage<T> vs, VirtualPath  
basePath, bool recursive = true, bool followLinks = false)
```

Parameters

`vs VirtualStorage ↗ <T>`

basePath [VirtualPath](#)

recursive [bool](#)

followLinks [bool](#)

Returns

[string](#)

Type Parameters

T

Namespace AkiraNetwork.VirtualStorage Library.WildcardMatchers

Classes

[DefaultWildcardMatcher](#)

[PowerShellWildcardMatcher](#)

Class DefaultWildcardMatcher

Namespace: [AkiraNetwork.VirtualStorageLibrary.WildcardMatchers](#)

Assembly: VirtualStorageLibrary.dll

```
public class DefaultWildcardMatcher : IVirtualWildcardMatcher
```

Inheritance

[object](#) ← DefaultWildcardMatcher

Implements

[IVirtualWildcardMatcher](#)

Inherited Members

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

Extension Methods

[VirtualTextFormatter.GenerateSingleTableDebugText<T>\(T\)](#)

Properties

Count

```
public int Count { get; }
```

Property Value

[int](#)

Patterns

```
public IEnumerable<string> Patterns { get; }
```

Property Value

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

WildcardDictionary

```
public ReadOnlyDictionary<string, string> WildcardDictionary { get; }
```

Property Value

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

Wildcards

```
public IEnumerable<string> Wildcards { get; }
```

Property Value

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

Methods

PatternMatcher(string, string)

```
public bool PatternMatcher(string nodeName, string pattern)
```

Parameters

nodeName [string](#)

pattern [string](#)

Returns

[bool](#)

Class PowerShellWildcardMatcher

Namespace: [AkiraNetwork.VirtualStorageLibrary.WildcardMatchers](#)

Assembly: VirtualStorageLibrary.dll

```
public class PowerShellWildcardMatcher : IVirtualWildcardMatcher
```

Inheritance

[object](#) ← PowerShellWildcardMatcher

Implements

[IVirtualWildcardMatcher](#)

Inherited Members

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

Extension Methods

[VirtualTextFormatter.GenerateSingleTableDebugText<T>\(T\)](#)

Properties

Count

```
public int Count { get; }
```

Property Value

[int](#)

Patterns

```
public IEnumerable<string> Patterns { get; }
```

Property Value

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

WildcardDictionary

```
public ReadOnlyDictionary<string, string> WildcardDictionary { get; }
```

Property Value

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

Wildcards

```
public IEnumerable<string> Wildcards { get; }
```

Property Value

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

Methods

PatternMatcher(string, string)

```
public bool PatternMatcher(string nodeName, string pattern)
```

Parameters

nodeName [string](#)

pattern [string](#)

Returns

[bool](#)