

# OpcNamespace Members

**Namespace:** Opc.UaFx

**Assemblies:** Opc.UaFx.Advanced.dll, Opc.UaFx.Advanced.dll

The [OpcNamespace](#) type exposes the following members.

## Fields

### UriHost

Specifies that an [Uri](#) is a pointer to a OPC UA namespace without a specific namespace [Host](#).

**C#**

```
public static readonly string UriHost
```

**Field Value**

[String](#)

### UriScheme

Specifies that an [Uri](#) is a pointer to a OPC UA namespace without a specific namespace [Scheme](#).

**C#**

```
public static readonly string UriScheme
```

**Field Value**

[String](#)

## Properties

### Default

Gets the default namespace used in cases there the concrete does not matter or is just unknown.

**C#**

```
public static OpcNamespace Default { get; }
```

**Property Value**

[OpcNamespace](#)

An instance of the [OpcNamespace](#) class its [IsDefault](#) property always returns the value true. While its

[Index](#) property is equals to zero, its [Uri](#) and [Value](#) properties always return a null reference (Nothing in Visual Basic).

## Index

Gets the index of the namespace within all namespaces of a server.

### C#

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

### Property Value

[Int32](#)

The index value which refers to the namespace in the namespace array of the server this [OpcNamespace](#) represents.

### Remarks

The available namespaces of a server can be retrieved through reading the value of the namespaces-node using the node identifier 'OpcObjectTypes.Server.Namespaces'.

## IsAbsolute

Gets a value indicating whether the namespace is absolute.

### C#

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

### Property Value

[Boolean](#)

The value true if an [Uri](#) or [Value](#) is given; otherwise the value false.

## IsDefault

Gets a value indicating whether the namespace defines the default namespace without to refer to a specific address space.

### C#

```
public bool IsDefault { get; }
```

### Property Value

[Boolean](#)

The value true if the namespace points to the generic address space; otherwise the value false if the namespace points to a foundation or user defined address space.

## IsResolved

Gets a value indicating whether the namespace is resolved regarding its [Index](#), [Uri](#) and [Value](#).

### C#

```
public bool IsResolved { get; }
```

### Property Value

#### Boolean

The value true if [Index](#) is not equals zero and [Uri](#) and [Value](#) are not a null reference (Nothing in Visual Basic).

### Remarks

In case there this property provides the value true any call to [Resolve\(IOpcNamespaceResolver\)](#) will have no effect to the [Index](#) nor to the [Uri](#) and [Value](#) properties.

## Scope

Gets scope within the current [OpcNamespace](#) has been declared and its metadata applies to.

### C#

```
public OpcNamespaceScope Scope { get; }
```

### Property Value

#### OpcNamespaceScope

One of the members defined by the [OpcNamespaceScope](#) enumeration.

## Uri

Gets the uniform resource identifier (URI) of the namespace represented and referred to by the [Index](#).

### C#

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

### Property Value

#### Uri

The [Uri](#) to that the [OpcNamespace](#) refers using the [Index](#).

# Value

Gets the intrinsic value of the namespace represented and referred to by the [Index](#).

**C#**

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

## Property Value

[String](#)

The [String](#) to that the [OpcNamespace](#) refers using the [Index](#).

# Methods

## Clear()

Empties the known namespaces.

**C#**

```
public static void Clear()
```

## Remarks

All namespaces retrieved until [Clear](#) is called are no longer referenced by the global / static namespace cache. Any so far retrieved namespaces are kept resolved / existing / valid and kept unchanged by subsequent global namespace operations, because they are not longer tracked nor updated by further global namespace retrieval nor resolve operations.

## CompareTo(Object)

Compares the current [OpcNamespace](#) with the [other](#).

**C#**

```
public int CompareTo(object other)
```

## Parameters

[other](#) [Object](#)

The [OpcNamespace](#) to compare with this [OpcNamespace](#).

## Returns

[Int32](#)

A 32-bit signed integer that indicates the relative order of the objects being compared

(CompareTo(Object)).

## CompareTo(OpcNamespace)

Compares the current [OpcNamespace](#) with another [OpcNamespace](#).

**C#**

```
public int CompareTo(OpcNamespace other)
```

### Parameters

**other** [OpcNamespace](#)

The [OpcNamespace](#) to compare with this [OpcNamespace](#).

### Returns

[Int32](#)

A 32-bit signed integer that indicates the relative order of the objects being compared ([CompareTo\(\)](#)).

## Create(String)

Creates a [OpcNamespace](#) using the **value** specified.

**C#**

```
public static OpcNamespace Create(string value)
```

### Parameters

**value** [String](#)

The [String](#) value or a null reference (Nothing in Visual Basic) of the namespace to represent.

### Returns

[OpcNamespace](#)

An instance of the [OpcNamespace](#) class.

## Create(String, Int32)

Creates a [OpcNamespace](#) using the **value** and **index** specified.

**C#**

```
public static OpcNamespace Create(string value, int index)
```

### Parameters

value **String**

The **String** value or a null reference (Nothing in Visual Basic) of the namespace to represent and referred to by the **index**.

**index** **Int32**

The positive number which identifies the namespace. A value within the range of zero to **MaxValue**.

## Returns

**OpcNamespace**

An instance of the **OpcNamespace** class.

## Exceptions

**ArgumentOutOfRangeException**

The **index** is less than zero or greater than **MaxValue**.

# Create(Uri)

Creates a **OpcNamespace** from the **Uri** specified. If **uri** provides **Port** information, the **Port** is used for the **Index** of the **OpcNamespace** created.

**C#**

```
public static OpcNamespace Create(Uri uri)
```

## Parameters

**uri** **Uri**

A **Uri** that contains a namespace.

## Returns

**OpcNamespace**

An instance of the **OpcNamespace** class.

## Exceptions

**ArgumentNullException**

The **uri** is a null reference (Nothing in Visual Basic).

**FormatException**

The **uri** is not a valid namespace.

## Create(Uri, Int32)

Creates a [OpcNamespace](#) from the [Uri](#) specified. If [uri](#) provides [Port](#) information, the [Port](#) is used for the [Index](#) of the [OpcNamespace](#) created, if [index](#) not greater than zero.

### C#

```
public static OpcNamespace Create(Uri uri, int index)
```

### Parameters

[uri](#) [Uri](#)

A [Uri](#) that contains a namespace.

[index](#) [Int32](#)

The positive number which identifies the namespace. A value within the range of zero to [MaxValue](#).

### Returns

[OpcNamespace](#)

An instance of the [OpcNamespace](#) class.

### Exceptions

[ArgumentNullException](#)

The [uri](#) is a null reference (Nothing in Visual Basic).

[ArgumentOutOfRangeException](#)

The [index](#) is less than zero or greater than [MaxValue](#).

[FormatException](#)

The [uri](#) is not a valid namespace.

## Equals(Object)

Determines whether the specified [other](#) is equal to this [OpcNamespace](#).

### C#

```
public override bool Equals(object other)
```

### Parameters

[other](#) [Object](#)

The [OpcNamespace](#) to compare to the current [OpcNamespace](#).

## Returns

### Boolean

The value true if the specified [OpcNamespace](#) is equal to the current [OpcNamespace](#); otherwise the value false.

# Equals(OpcNamespace)

Determines whether the specified `other` is equal to this [OpcNamespace](#).

## C#

```
public bool Equals(OpcNamespace other)
```

## Parameters

`other` [OpcNamespace](#)

The [OpcNamespace](#) to compare to the current [OpcNamespace](#).

## Returns

### Boolean

The value true if the specified [OpcNamespace](#) is equal to the current [OpcNamespace](#); otherwise the value false.

# Get(Int32)

Retrieves the [OpcNamespace](#) known under the `namespaceIndex` specified.

## C#

```
public static OpcNamespace Get(int namespaceIndex)
```

## Parameters

`namespaceIndex` [Int32](#)

The index of the [OpcNamespace](#) to retrieve.

## Returns

### [OpcNamespace](#)

If `namespaceIndex` is equals zero the [Default](#) namespace; otherwise if there is a known namespace with the index specified the reference to the existing [OpcNamespace](#). If there does no namespace exist under the index specified by `namespaceIndex` a new namespace with that index is created, added to the indexed namespaces and the reference to the new [OpcNamespace](#) is returned.



# Get(Int32, String)

Retrieves the [OpcNamespace](#) known under the [namespaceIndex](#) and [namespaceValue](#) specified.

C#

```
public static OpcNamespace Get(int namespaceIndex, string namespaceValue)
```

## Parameters

[namespaceIndex](#) [Int32](#)

The index of the [OpcNamespace](#) to retrieve.

[namespaceValue](#) [String](#)

The [Uri](#)-formatted or [String](#)-based representation of the [OpcNamespace](#) to retrieve.

## Returns

[OpcNamespace](#)

If [namespaceIndex](#) is equals zero and the [Port](#) portion of the [namespaceValue](#) is zero too the [Default](#) namespace; otherwise if there is a known namespace with the index specified by [namespaceIndex](#) or (if this parameter is equals zero) a known namespace with the index defined by the [Port](#) portion of the [namespaceValue](#) the reference to the existing [OpcNamespace](#). If there does no namespace exist under the index and value specified by [namespaceIndex](#) and [namespaceValue](#) a new namespace with that index and value is created, added to the indexed / unindexed namespaces and the reference to the new [OpcNamespace](#) is returned.

## Exceptions

[ArgumentException](#)

The [namespaceValue](#) is an empty string.

[ArgumentNullException](#)

The [namespaceValue](#) is a null reference (Nothing in Visual Basic).

[FormatException](#)

The [namespaceValue](#) is not a valid namespace.

# Get(Int32, Uri)

Retrieves the [OpcNamespace](#) known under the [namespaceIndex](#) and [namespaceUri](#) specified.

C#

```
public static OpcNamespace Get(int namespaceIndex, Uri namespaceUri)
```

## Parameters

## namespaceIndex Int32

The index of the [OpcNamespace](#) to retrieve.

## namespaceUri Uri

The [Uri](#) of the [OpcNamespace](#) to retrieve.

## Returns

### [OpcNamespace](#)

If [namespaceIndex](#) is equals zero and the [Port](#) portion of the [namespaceUri](#) is zero too the [Default](#) namespace; otherwise if there is a known namespace with the index specified by [namespaceIndex](#) or (if this parameter is equals zero) a known namespace with the index defined by the [Port](#) portion of the [namespaceUri](#) the reference to the existing [OpcNamespace](#). If there does no namespace exist under the index and uri specified by [namespaceIndex](#) and [namespaceUri](#) a new namespace with that index and uri is created, added to the indexed / unindexed namespaces and the reference to the new [OpcNamespace](#) is returned.

## Exceptions

### [ArgumentNullException](#)

The [namespaceUri](#) is a null reference (Nothing in Visual Basic).

### [FormatException](#)

The [namespaceUri](#) is not a valid namespace.

# Get(String)

Retrieves the [OpcNamespace](#) using the information provided by the [namespaceUriOrValue](#) specified.

## C#

```
public static OpcNamespace Get(string namespaceUriOrValue)
```

## Parameters

### [namespaceUriOrValue](#) [String](#)

The [String](#)-based representation or [Uri](#)-formatted string of the [OpcNamespace](#) to retrieve.

## Returns

### [OpcNamespace](#)

If [Host](#) portion of the [namespaceUriOrValue](#) is equals to [UriHost](#) and the [Port](#) does not refer to an indexed namespace the [Default](#) namespace; otherwise if there is a known namespace with the index specified by the [Port](#) portion the reference to the existing [OpcNamespace](#). If there does no namespace exist under the index or [namespaceUriOrValue](#) specified a new namespace with that [Uri](#) is created, added to the indexed /

unindexed namespaces and the reference to the new [OpcNamespace](#) is returned.

## Exceptions

### [ArgumentException](#)

The [namespaceUriOrValue](#) is an empty string.

### [ArgumentNullException](#)

The [namespaceUriOrValue](#) is a null reference (Nothing in Visual Basic).

### [FormatException](#)

The [namespaceUriOrValue](#) is not a valid namespace.

## Get(Uri)

Retrieves the [OpcNamespace](#) with the information provided by the [namespaceUri](#) specified.

### C#

```
public static OpcNamespace Get(Uri namespaceUri)
```

## Parameters

[namespaceUri](#) Uri

The [Uri](#) of the [OpcNamespace](#) to retrieve.

## Returns

[OpcNamespace](#)

If [Host](#) portion of the [namespaceUri](#) is equals to [UriHost](#) and the [Port](#) does not refer to an indexed namespace the [Default](#) namespace; otherwise if there is a known namespace with the index specified by the [Port](#) portion the reference to the existing [OpcNamespace](#). If there does no namespace exist under the index or [namespaceUri](#) specified a new namespace with that [Uri](#) is created, added to the indexed / unindexed namespaces and the reference to the new [OpcNamespace](#) is returned.

## Exceptions

### [ArgumentNullException](#)

The [namespaceUri](#) is a null reference (Nothing in Visual Basic).

## GetHashCode()

Retrieves a hash code for this [OpcNamespace](#).

### C#

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

## Returns

[Int32](#)

An [Int32](#) that contains the hash code for the [OpcNamespace](#).

## GetId(Byte[])

Retrieves a new [OpcNodeId](#) using the opaque [value](#) and this [OpcNamespace](#).

### C#

```
public OpcNodeId GetId(byte[] value)
```

## Parameters

[value](#) [Byte\[\]](#)

The opaque value of the identifier.

## Returns

[OpcNodeId](#)

A new instance of the [OpcNodeId](#) using the [value](#) as the identifier value and this [OpcNamespace](#) as the [Namespace](#).

## GetId(Guid)

Retrieves a new [OpcNodeId](#) using the general unique identifier (= GUID) [value](#) and this [OpcNamespace](#).

### C#

```
public OpcNodeId GetId(Guid value)
```

## Parameters

[value](#) [Guid](#)

The general unique identifier (= GUID) value of the identifier.

## Returns

[OpcNodeId](#)

A new instance of the [OpcNodeId](#) using the [value](#) as the identifier value and this [OpcNamespace](#) as the [Namespace](#).

# GetId(Int32)

Retrieves a new [OpcNodeId](#) using the numeric [value](#) and this [OpcNamespace](#).

## C#

```
public OpcNodeId GetId(int value)
```

### Parameters

[value](#) [Int32](#)

The numeric value of the identifier.

### Returns

[OpcNodeId](#)

A new instance of the [OpcNodeId](#) using the [value](#) as the identifier value and this [OpcNamespace](#) as the [Namespace](#).

# GetId(Object)

Retrieves a new [OpcNodeId](#) using the [value](#) and this [OpcNamespace](#).

## C#

```
public OpcNodeId GetId(object value)
```

### Parameters

[value](#) [Object](#)

The value of the identifier.

### Returns

[OpcNodeId](#)

A new instance of the [OpcNodeId](#) using the [value](#) as the identifier value and this [OpcNamespace](#) as the [Namespace](#).

### Exceptions

[ArgumentException](#)

The type of [value](#) is not supported.

## GetId(String)

Retrieves a new [OpcNodeId](#) using the [String](#)[value](#) and this [OpcNamespace](#).

**C#**

```
public OpcNodeId GetId(string value)
```

### Parameters

[value](#) [String](#)

The [String](#) value of the identifier.

### Returns

[OpcNodeId](#)

A new instance of the [OpcNodeId](#) using the [value](#) as the identifier value and this [OpcNamespace](#) as the [Namespace](#).

## GetId(String, OpcName[])

Retrieves a new [OpcNodeId](#) using the [String](#)[value](#), this [OpcNamespace](#) and the [pathElements](#).

**C#**

```
public OpcNodeId GetId(string value, params OpcName[] pathElements)
```

### Parameters

[value](#) [String](#)

The [String](#) value of the identifier.

[pathElements](#) [OpcName\[\]](#)

The elements of the [OpcNamePath](#) to use as the logical (= physically not available in the address space) portion of the identifier.

### Returns

[OpcNodeId](#)

A new instance of the [OpcNodeId](#) using the [value](#) as the identifier value, this [OpcNamespace](#) as the [Namespace](#) and the [pathElements](#) for the [Path](#).

## GetId(String, OpcNamePath)

Retrieves a new [OpcNodeId](#) using the [String](#)[value](#), this [OpcNamespace](#) and the [path](#).

**C#**

```
public OpcNodeId GetId(string value, OpcNamePath path)
```

## Parameters

**value** String

The String value of the identifier.

**path** OpcNamePath

The logical (= physically not available in the address space) portion of the identifier.

## Returns

OpcNodeId

A new instance of the OpcNodeId using the **value** as the identifier value, this OpcNamespace as the Namespace and the **path** specified for the Path.

# GetId(UInt32)

Retrieves a new OpcNodeId using the numeric **value** and this OpcNamespace.

## C#

```
public OpcNodeId GetId(uint value)
```

## Parameters

**value** UInt32

The numeric value of the identifier.

## Returns

OpcNodeId

A new instance of the OpcNodeId using the **value** as the identifier value and this OpcNamespace as the Namespace.

# GetName(String)

Retrieves a new OpcName using the **name** and this OpcNamespace.

## C#

```
public OpcName GetName(string name)
```

## Parameters

**name** String

The value of the [OpcName](#) to create.

## Returns

[OpcName](#)

A new instance of the [OpcName](#) using the [name](#) as the [Value](#) and this [OpcNamespace](#) as the [Namespace](#).

## Parse(String)

Converts a namespace string to a [OpcNamespace](#) instance.

### C#

```
public static OpcNamespace Parse(string value)
```

## Parameters

[value](#) [String](#)

A string that contains a namespace.

## Returns

[OpcNamespace](#)

An instance of the [OpcNamespace](#) class.

## Exceptions

[FormatException](#)

The [value](#) is not a valid namespace.

## Resolve(IOpcNamespaceResolver)

Resolves missing OPC UA namespace information using the object specified by [resolver](#).

### C#

```
public void Resolve(IOpcNamespaceResolver resolver)
```

## Parameters

[resolver](#) [IOpcNamespaceResolver](#)

An instance implementing the [IOpcNamespaceResolver](#) interface to use to retrieve additional namespace information so far missed by the [OpcNamespace](#).



## Exceptions

### ArgumentNullException

The `resolver` is a null reference (Nothing in Visual Basic).

## Remarks

Only missing information is resolved using the `resolver` specified. A once resolved `Index` or `Value` is not discarded or re-resolved by subsequent calls of this method (see `IsResolved`).

## ToString()

Returns a string representing the namespace.

### C#

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

## Returns

### String

The `Value` and `Index` (if non-zero) formatted as string.

## TryCreate(Uri, Int32, out OpcNamespace)

Determines whether a `Uri` is a valid namespace.

### C#

```
public static bool TryCreate(Uri uri, int index, out OpcNamespace nodeNamespace)
```

## Parameters

### uri Uri

The `Uri` to validate.

### index Int32

The index which applies to the `uri` of the namespace to validate.

### nodeNamespace OpcNamespace

The `OpcNamespace` version of the `uri` with the `index` specified.

## Returns

### Boolean

The value true; if the `uri` is a valid namespace; otherwise the value false.

## Remarks

The `index` is only used if it is greater than zero; otherwise the `Port` of the `uri` is used instead.

# TryCreate(Uri, out OpcNamespace)

Determines whether a `Uri` is a valid namespace.

## C#

```
public static bool TryCreate(Uri uri, out OpcNamespace nodeNamespace)
```

## Parameters

`uri` `Uri`

The `Uri` to validate.

`nodeNamespace` `OpcNamespace`

The `OpcNamespace` version of the `uri`.

## Returns

`Boolean`

The value true, if the `uri` is a valid namespace; otherwise the value false.

# TryParse(String, out OpcNamespace)

Determines whether a string is a valid namespace.

## C#

```
public static bool TryParse(string value, out OpcNamespace nodeNamespace)
```

## Parameters

`value` `String`

The string to validate.

`nodeNamespace` `OpcNamespace`

The `OpcNamespace` version of the string.

## Returns

`Boolean`

The value true, if `value` is a valid namespace; otherwise the value false.

# Operators

## Equality(OpcNamespace, OpcNamespace)

Returns a value indicating whether two instance of [OpcNamespace](#) are equal.

**C#**

```
public static bool operator ==(OpcNamespace left, OpcNamespace right)
```

## GreaterThan(OpcNamespace, OpcNamespace)

Determines whether the first specified [OpcNamespace](#) object is greater than the second specified [OpcNamespace](#) object.

**C#**

```
public static bool operator >(OpcNamespace left, OpcNamespace right)
```

## GreaterThanOrEqual(OpcNamespace, OpcNamespace)

Determines whether the first specified [OpcNamespace](#) object is greater than or equal to the second specified [OpcNamespace](#) object.

**C#**

```
public static bool operator >=(OpcNamespace left, OpcNamespace right)
```

## Implicit(Int32 to OpcNamespace)

Converts a [Int32](#) to an [OpcNamespace](#) object.

**C#**

```
public static implicit operator OpcNamespace(int value)
```

## Implicit(String to OpcNamespace)

Converts a [String](#) to an [OpcNamespace](#) object.

**C#**

```
public static implicit operator OpcNamespace(string value)
```

## Inequality(OpcNamespace, OpcNamespace)

Returns a value indicating whether two instances of [OpcNamespace](#) are not equal.

**C#**

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

## LessThan(OpcNamespace, OpcNamespace)

Determines whether the first specified [OpcNamespace](#) object is less than the second specified [OpcNamespace](#) object.

### C#

```
public static bool operator <(OpcNamespace left, OpcNamespace right)
```

### Exceptions

[ArgumentNullException](#)

The [Left](#) is a null reference (Nothing in Visual Basic).

## LessThanOrEqual(OpcNamespace, OpcNamespace)

Determines whether the first specified [OpcNamespace](#) object is less than or equal to the second [OpcNamespace](#) object.

### C#

```
public static bool operator <=(OpcNamespace left, OpcNamespace right)
```

### Exceptions

[ArgumentNullException](#)

The [Left](#) is a null reference (Nothing in Visual Basic).

# Table of Contents

<b>Fields</b>	1
UriHost	1
UriScheme	1
<b>Properties</b>	1
Default	1
Index	2
IsAbsolute	2
IsDefault	2
IsResolved	3
Scope	3
Uri	3
Value	4
<b>Methods</b>	4
Clear()	4
CompareTo(Object)	4
CompareTo(OpcNamespace)	5
Create(String)	5
Create(String, Int32)	5
Create(Uri)	6
Create(Uri, Int32)	7
Equals(Object)	7
Equals(OpcNamespace)	8
Get(Int32)	8
Get(Int32, String)	9
Get(Int32, Uri)	9
Get(String)	10
Get(Uri)	11
GetHashCode()	11
GetId(Byte[])	12
GetId(Guid)	12
GetId(Int32)	13
GetId(Object)	13
GetId(String)	14
GetId(String, OpcName[])	14
GetId(String, OpcNamePath)	14
GetId(UInt32)	15
GetName(String)	15
Parse(String)	16
Resolve(IOpcNamespaceResolver)	16
ToString()	17
TryCreate(Uri, Int32, out OpcNamespace)	17
TryCreate(Uri, out OpcNamespace)	18
TryParse(String, out OpcNamespace)	18
<b>Operators</b>	19
Equality(OpcNamespace, OpcNamespace)	19
GreaterThan(OpcNamespace, OpcNamespace)	19
GreaterThanOrEqual(OpcNamespace, OpcNamespace)	19
Implicit(Int32 to OpcNamespace)	19
Implicit(String to OpcNamespace)	19
Inequality(OpcNamespace, OpcNamespace)	19
LessThan(OpcNamespace, OpcNamespace)	20

LessThanOrEqual(OpcNamespace, OpcNamespace) .....	20
---	----