Tungsten

$W$
Welcome to Tungsten.Suite

Tungsten.Suite is a refactoring of the original Tungsten.X libraries. Tungsten is now a suite of NuGet packages which can be referenced individually or as a whole by referencing Tungsten.Suite.

These are the NuGet packages referenced by Tungsten.Suite

- Tungsten.ArrayMethods
- Tungsten.As
- Tungsten.CallResult
- Tungsten.Console
- Tungsten.EventTemplate
- Tungsten.Encryption
- Tungsten.From
- Tungsten.IO.Pipes
- Tungsten.Lockable
- Tungsten.Logging
- Tungsten.Net
- Tungsten.Property
- Tungsten.Threading
- Tungsten.Threading.Lockers

These projects have NuGet packages, but are not included in Tungsten.Suite because they only support the .Net Framework

- Tungsten.Domains
- Tungsten.Firewall
• Tungsten.InterProcess

See Also

Other Resources
Version History

Copyright © 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>$W$</td>
</tr>
</tbody>
</table>
Version History

The topics in this section describe the various changes made to Tungsten over the life of the project.

Version History

Select a version below to see a description of its changes.

- Version 2.0.0

See Also

Other Resources
Welcome to Tungsten.Suite

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
Version 2.0.0

Version 2.0.0 was released on 2/28/2018

Changes in This Release

- Initial release of the refactored Tungsten libraries

See Also

Other Resources
Version History

Copyright @ 2018 Jordan Duerksen
Tungsten

$^n W$
# W Namespace

[Missing <summary> documentation for "N:W"]

## Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>❄️ ArrayMethods</td>
<td>Methods to peek and modify arrays</td>
</tr>
<tr>
<td>❄️ AsExtensions</td>
<td>Extensions which convert objects of one type to another</td>
</tr>
<tr>
<td>❄️ CallResult</td>
<td>A non-generic return value for a function. CallResult encapsulates a success/failure and an exception.</td>
</tr>
<tr>
<td>❄️ CallResultTResult</td>
<td>Generic class to be used as a return value. CallResult encapsulates a success/failure, an exception and a return value.</td>
</tr>
<tr>
<td>❄️ ConsoleStringExtensions</td>
<td>Extension methods related to the Console</td>
</tr>
<tr>
<td>❄️ EventTemplateTSender</td>
<td>Wraps the functionality of delegate, event and RaiseXXX into a single class</td>
</tr>
<tr>
<td>Class/Extension</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>EventTemplateTSendert</td>
<td>Wraps the functionality of delegate, event and RaiseXXX into a single class</td>
</tr>
<tr>
<td>EventTemplateTSender, Arg1</td>
<td>Wraps the functionality of delegate, event and RaiseXXX into a single class</td>
</tr>
<tr>
<td>EventTemplateTSender, Arg1, Arg2</td>
<td>Wraps the functionality of delegate, event and RaiseXXX into a single class</td>
</tr>
<tr>
<td>EventTemplateTSender, Arg1, Arg2, Arg3</td>
<td>Wraps the functionality of delegate, event and RaiseXXX into a single class</td>
</tr>
<tr>
<td>EventTemplateTSender, Arg1, Arg2, Arg3, Arg4</td>
<td>Wraps the functionality of delegate, event and RaiseXXX into a single class</td>
</tr>
<tr>
<td>EventTemplateTSender, Arg1, Arg2, Arg3, Arg4, Arg5</td>
<td>Wraps the functionality of delegate, event and RaiseXXX into a single class</td>
</tr>
<tr>
<td>FromExtensions</td>
<td>Extensions which convert objects of one type to another</td>
</tr>
<tr>
<td>LockableTValue</td>
<td>Extends LockableSlim with ValueChangedDelegate notification</td>
</tr>
<tr>
<td>LockableSlimTValue</td>
<td>Uses ReaderWriterLock to provide thread-safe access to an underlying value</td>
</tr>
<tr>
<td>PropertyTValue</td>
<td>A Property with no owner (self-owned)</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------------------------------</td>
</tr>
<tr>
<td>PropertyTOwner, TValue</td>
<td>A generic Property with an owner</td>
</tr>
<tr>
<td>PropertyBaseTOwner, TValue</td>
<td></td>
</tr>
<tr>
<td>PropertyChangedNotifier</td>
<td>This is a base class for supporting INotifyPropertyChanged</td>
</tr>
<tr>
<td>PropertyHost</td>
<td>Provides a base class to automate the IsDirty, MarkAsClean and InitializeProperties functionality Note that this class does not support INotifyPropertyChanged and is not intened to host owned properties (though nothing prevents you from doing so)</td>
</tr>
<tr>
<td>PropertyHostExtensions</td>
<td>Extension methods related to W.PropertyHost or any class which has multiple W.Property members</td>
</tr>
<tr>
<td>PropertyHostNotifier</td>
<td>Provides a base class to automate the IsDirty, MarkAsClean and InitializeProperties functionality Note that this class inherits PropertyChangedNotifier for INotifyPropertyChanged support</td>
</tr>
</tbody>
</table>
PropertySlim extends 
W.Lockable by adding 
support for 
INotifyPropertyChanged

## Interfaces

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOwnedProperty</td>
<td>Used by PropertyHostMethods.InitializeProperties to find properties on which to set the owner. This interface is not used by self-owned properties.</td>
</tr>
<tr>
<td>IProperty</td>
<td>The base interface which Property must support</td>
</tr>
<tr>
<td>IPropertyTVValue</td>
<td>The base interface which Property must support</td>
</tr>
</tbody>
</table>

## Delegates

<table>
<thead>
<tr>
<th>Delegate</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ValueChangedDelegateTVale</td>
<td>Raised when the value has changed</td>
</tr>
</tbody>
</table>
Tungsten

W
ArrayMethods Class

Methods to peek and modify arrays

Inheritance Hierarchy

- System
- Object
- W
- ArrayMethods

Namespace: W
Version: 2.0.0

Syntax

```
public static class ArrayMethods
```

The ArrayMethods type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>☑ S</td>
<td>AppendT</td>
</tr>
<tr>
<td>☑ S</td>
<td>InsertT</td>
</tr>
<tr>
<td>☑ S</td>
<td>PeekT</td>
</tr>
<tr>
<td>☑ S</td>
<td>PeekEndT</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PeekStartT</td>
<td>Retrieves the specified number of elements from the start of the array without changing the source array</td>
</tr>
<tr>
<td>TakeT</td>
<td>Retrieves and removes the specified range of elements from the array</td>
</tr>
<tr>
<td>TakeFromEndT</td>
<td>Retrieves and removes the specified number of elements from the end of the array</td>
</tr>
<tr>
<td>TakeFromStartT</td>
<td>Retrieves and removes the specified number of elements from the start of the array</td>
</tr>
<tr>
<td>TrimT</td>
<td>Removes the specified range of elements from the array, resizing the array as necessary</td>
</tr>
<tr>
<td>TrimEndT</td>
<td>Removes the specified number of elements from the end of the array, resizing the array as necessary</td>
</tr>
<tr>
<td>TrimStartT</td>
<td>Removes the specified number of elements from the start of the array, resizing the array as necessary</td>
</tr>
</tbody>
</table>

See Also

Reference

W Namespace
Tungsten

$W$
# ArrayMethods Methods

The `ArrayMethods` type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AppendT</code></td>
<td>Appends the items to an array, resizing the array as necessary</td>
</tr>
<tr>
<td><code>InsertT</code></td>
<td>Appends the items to an array, resizing the array as necessary</td>
</tr>
<tr>
<td><code>PeekT</code></td>
<td>Retrieves the specified range of elements from the array</td>
</tr>
<tr>
<td><code>PeekEndT</code></td>
<td>Retrieves the specified number of elements from the end of the array</td>
</tr>
<tr>
<td><code>PeekStartT</code></td>
<td>Retrieves the specified number of elements from the start of the array without changing the source array</td>
</tr>
<tr>
<td><code>TakeT</code></td>
<td>Retrieves and removes the specified range of elements from the array</td>
</tr>
<tr>
<td><code>TakeFromEndT</code></td>
<td>Retrieves and removes the specified number of elements from the end of the array</td>
</tr>
<tr>
<td><code>TakeFromStartT</code></td>
<td>Retrieves and removes the specified number of elements from the start of the array</td>
</tr>
</tbody>
</table>
specified number of elements from the start of the array

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>🔄</td>
<td><strong>TrimT</strong></td>
</tr>
<tr>
<td>🔄</td>
<td>Removes the specified range of elements from the array, resizing the array as necessary</td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>🔄</td>
<td><strong>TrimEndT</strong></td>
</tr>
<tr>
<td>🔄</td>
<td>Removes the specified number of elements from the end of the array, resizing the array as necessary</td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>🔄</td>
<td><strong>TrimStartT</strong></td>
</tr>
<tr>
<td>🔄</td>
<td>Removes the specified number of elements from the start of the array, resizing the array as necessary</td>
</tr>
</tbody>
</table>

**Top**

⚠️ See Also

Reference
ArrayMethods Class
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
ArrayMethodsAppend<T> Method

Appends the items to an array, resizing the array as necessary

**Namespace:**  W  
**Assembly:**  Tungsten.ArrayMethods (in Tungsten.ArrayMethods.dll)  
**Version:**  2.0.0

**Syntax**

```csharp
public static T[] Append<T>(
    ref T[] source,
    T[] itemsToAdd
)
```

**Parameters**

- **source**
  
  Type:  \( T \)
  
  The source array

- **itemsToAdd**
  
  Type:  \( T \)
  
  The array of items to append to the source

---

**Type Parameters**

- **\( T \)**
  
  The data type

**Return Value**

- **Type:**  \( T \)
  
  The modified source array
See Also

Reference

ArrayMethods Class
W Namespace

Copyright © 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
ArrayMethodsInsert\(T\) Method

Appends the items to an array, resizing the array as necessary

Namespace:  \(W\)
Version:  2.0.0

Syntax

C#  

```csharp
public static T[] Insert<T>(
    ref T[] source,
    T[] itemsToInsert,
    int index
)
```

Parameters

- **source**
  - Type:  \(T\)
  - The source array

- **itemsToInsert**
  - Type:  \(T\)
  - The array of items to append to the source

- **index**
  - Type:  System\(\)Int32
  - The index where the items should be inserted

Type Parameters

- \(T\)
  - The data type

Return Value
Type: \textit{T}
The modified source array

\textbf{See Also}

\textbf{Reference}
ArrayMethods Class
W Namespace

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
<th>W</th>
</tr>
</thead>
</table>

ArrayMethodsPeekT Method

Retrieves the specified range of elements from the array

**Namespace:**  W  
**Assembly:**  Tungsten.ArrayMethods (in Tungsten.ArrayMethods.dll)  
**Version:** 2.0.0

## Syntax

```csharp
public static T[] Peek<T>(
    T[] source,
    int startIndex,
    int length
)
```

### Parameters

**source**  
Type:  \( T \)  
The source array

**startIndex**  
Type:  `SystemInt32`  
The index from which to start retrieving elements

**length**  
Type:  `SystemInt32`  
The number of elements to retrieve

### Type Parameters

**\( T \)**  
The data type

### Return Value
Type: $T$
A new array containing only the specified subset of elements

See Also

Reference

ArrayMethods Class
W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

W
ArrayMethods.PeekEnd<T> Method

Retrieves the specified number of elements from the end of the array

**Namespace:** W  
**Assembly:** Tungsten.ArrayMethods (in Tungsten.ArrayMethods.dll)  
**Version:** 2.0.0

**Syntax**

```csharp
public static T[] PeekEnd<T>(
    T[] source,
    int length
)
```

**Parameters**

- **source**
  - Type: T
  - The source array
- **length**
  - Type: System.Int32
  - The number of elements to retrieve

**Type Parameters**

- **T**
  - The data type

**Return Value**

- Type: T
  - A new array containing only the specified subset of elements
See Also

Reference
ArrayMethods Class
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
ArrayMethodsPeekStart<T> Method

Retrieves the specified number of elements from the start of the array without changing the source array.

**Namespace**: W
**Assembly**: Tungsten.ArrayMethods (in Tungsten.ArrayMethods.dll)
**Version**: 2.0.0

**Syntax**

```csharp
public static T[] PeekStart<T>(
    T[] source,
    int length
)
```

**Parameters**

- **source**
  - Type: `T`
  - The source array

- **length**
  - Type: `SystemInt32`
  - The number of elements to retrieve

**Type Parameters**

- **T**
  - The data type

**Return Value**

- Type: `T`
A new array containing only the specified subset of elements

See Also

Reference
ArrayMethods Class
W Namespace
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>( W )</td>
</tr>
</tbody>
</table>
ArrayMethodsTake\( T \) Method

Retrieves and removes the specified range of elements from the array

**Namespace:** W
**Assembly:** Tungsten.ArrayMethods (in Tungsten.ArrayMethods.dll)
**Version:** 2.0.0

### Syntax

```csharp
public static T[] Take<T>(
    ref T[] source,
    int startIndex,
    int length
)
```

### Parameters

**source**
- Type: \( T \)
- The source array

**startIndex**
- Type: System.Int32
- The index from which to start retrieving elements

**length**
- Type: System.Int32
- The number of elements to retrieve

### Type Parameters

\( T \)
- The data type

### Return Value
Type: $T$
A new array containing only the specified subset of elements

See Also

Reference
ArrayMethods Class
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$\text{W}$
ArrayMethods

TakeFromEnd<T> Method

Retrieves and removes the specified number of elements from the end of the array

**Namespace:** W  
**Assembly:** Tungsten.ArrayMethods (in Tungsten.ArrayMethods.dll)  
**Version:** 2.0.0

### Syntax

```csharp
public static T[] TakeFromEnd<T>(
    ref T[] source,
    int length
)
```

### Parameters

- **source**
  - Type: `T`
  - The source array

- **length**
  - Type: `System.Int32`
  - The number of elements to retrieve

### Type Parameters

- **T**
  - The data type

### Return Value

- Type: `T`
A new array containing only the specified subset of elements

See Also

Reference

ArrayMethods Class
W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
ArrayMethods

TakeFromStart<T> Method

Retrieves and removes the specified number of elements from the start of the array

Namespace:  W
Version:  2.0.0

Syntax

```csharp
public static T[] TakeFromStart<T>(
    ref T[] source,
    int length
)
```

Parameters

- source
  - Type:  T
  - The source array
- length
  - Type:  System.Int32
  - The number of elements to retrieve

Type Parameters

- T
  - The data type

Return Value

- Type:  T
A new array containing only the specified subset of elements

See Also

Reference

ArrayMethods Class
W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
ArrayMethodsTrim\(T\) Method

Removes the specified range of elements from the array, resizing the array as necessary

**Namespace:** W
**Assembly:** Tungsten.ArrayMethods (in Tungsten.ArrayMethods.dll)
**Version:** 2.0.0

## Syntax

```csharp
public static T[] Trim<T>(
    ref T[] source,
    int startIndex,
    int length
)
```

### Parameters

**source**
- Type: \(T\)
- The source array

**startIndex**
- Type: System.Int32
- The index from which to start removing elements

**length**
- Type: System.Int32
- The number of elements to remove

### Type Parameters

**T**
- The data type
Return Value
Type: $T$
The modified source array

See Also

Reference
ArrayMethods Class
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$^\text{W}$
ArrayMethodsTrimEnd<T> Method

Removes the specified number of elements from the end of the array, resizing the array as necessary

Namespace: W
Version: 2.0.0

Syntax

C#

```csharp
public static T[] TrimEnd<T>(
    ref T[] source,
    int length
)
```

Parameters

- **source**
  - Type: **T**
  - The source array

- **length**
  - Type: **System.Int32**
  - The number of elements to remove

Type Parameters

- **T**
  - The data type

Return Value

- Type: **T**
  - The modified source array
See Also

Reference
ArrayMethods Class
W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$^W$
ArrayMethodsTrimStart<T> Method

Removes the specified number of elements from the start of the array, resizing the array as necessary

**Namespace:** W
**Assembly:** Tungsten.ArrayMethods (in Tungsten.ArrayMethods.dll)
**Version:** 2.0.0

### Syntax

```csharp
public static T[] TrimStart<T>(
    ref T[] source,
    int length
)
```

### Parameters

- **source**
  - Type: `T`
  - The source array

- **length**
  - Type: `System.Int32`
  - The number of elements to remove

### Type Parameters

- **T**
  - The data type

### Return Value

- **Type:** `T`
  - The modified source array
See Also

Reference
ArrayMethods Class
W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
AsExtensions Class

Extensions which convert objects of one type to another

Inheritance Hierarchy

- System
  - Object
  - WAsExtensions

Namespace: W
Assembly: Tungsten.As (in Tungsten.As.dll) Version: 2.0.1

Syntax

```c#
public static class AsExtensions
```

The `AsExtensions` type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AsTType</code></td>
<td>Use Generic syntax for the <code>as</code> operator.</td>
</tr>
<tr>
<td><code>AsBase64(Byte)</code></td>
<td>Converts a byte array to a Base64 encoded string</td>
</tr>
<tr>
<td><code>AsBase64(String)</code></td>
<td>Converts a string to Base64 encoding</td>
</tr>
<tr>
<td><code>AsBase64(String, Encoding)</code></td>
<td>Converts a string to Base64 encoding</td>
</tr>
<tr>
<td><code>AsBytes(String)</code></td>
<td>Converts a string to an encoded byte array</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>AsBytes(String, Encoding)</code></td>
<td>Converts a string to an encoded byte array</td>
</tr>
<tr>
<td><code>AsCompressed</code></td>
<td>Compresses the byte array using System.IO.Compression.DeflateStream</td>
</tr>
<tr>
<td><code>AsStream(Byte)</code></td>
<td>Creates a MemoryStream object and initializes it with the specified byte array</td>
</tr>
<tr>
<td><code>AsStream(String)</code></td>
<td>Creates a MemoryStream object and initializes it with the specified string</td>
</tr>
<tr>
<td><code>AsString(Byte)</code></td>
<td>Converts an encoded byte array to a string</td>
</tr>
<tr>
<td><code>AsString(Byte, Encoding)</code></td>
<td>Converts an encoded byte array to a string</td>
</tr>
<tr>
<td><code>AsString(Byte, Int32, Int32)</code></td>
<td>Converts an encoded byte array to a string</td>
</tr>
<tr>
<td><code>AsString(Byte, Int32, Int32, Encoding)</code></td>
<td>Converts an encoded byte array to a string</td>
</tr>
</tbody>
</table>

See Also

Reference

W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

W
## AsExtensions Methods

The `AsExtensions` type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AsTType</code></td>
<td>Use Generic syntax for the as operator.</td>
</tr>
<tr>
<td><code>AsBase64(Byte)</code></td>
<td>Converts a byte array to a Base64 encoded string</td>
</tr>
<tr>
<td><code>AsBase64(String)</code></td>
<td>Converts a string to Base64 encoding</td>
</tr>
<tr>
<td><code>AsBase64(String, Encoding)</code></td>
<td>Converts a string to Base64 encoding</td>
</tr>
<tr>
<td><code>AsBytes(String)</code></td>
<td>Converts a string to an encoded byte array</td>
</tr>
<tr>
<td><code>AsBytes(String, Encoding)</code></td>
<td>Converts a string to an encoded byte array</td>
</tr>
<tr>
<td><code>AsCompressed</code></td>
<td>Compresses the byte array using <code>System.IO.Compression.DeflateStream</code></td>
</tr>
<tr>
<td><code>AsStream(Byte)</code></td>
<td>Creates a MemoryStream object and initializes it with the specified byte array</td>
</tr>
<tr>
<td><code>AsStream(String)</code></td>
<td>Creates a MemoryStream object and initializes it with the specified string</td>
</tr>
<tr>
<td><code>AsString(Byte)</code></td>
<td>Converts an encoded byte array to a string</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>AsString(Byte, Encoding)</code></td>
<td>Converts an encoded byte array to a string</td>
</tr>
<tr>
<td><code>AsString(Byte, Int32, Int32)</code></td>
<td>Converts an encoded byte array to a string</td>
</tr>
<tr>
<td><code>AsString(Byte, Int32, Int32, Encoding)</code></td>
<td>Converts an encoded byte array to a string</td>
</tr>
</tbody>
</table>

See Also

Reference
AsExtensions Class
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
AsExtensionsAs<TType> Method

Use Generic syntax for the as operator.

Namespace: W
Assembly: Tungsten.As (in Tungsten.As.dll) Version: 2.0.1

Syntax

```
public static TType As<TType>(
    this Object this
)
where TType : class
```

Parameters

`this`
Type: SystemObject
The item to convert to type TType

Type Parameters

`TType`
The type to convert the item reference to.

Return Value
Type: TType
Null if @this cannot be referenced as TType. Otherwise, the item as TType

Usage Note
In Visual Basic and C#, you can call this method as an instance method on any object of type Object. When you use instance method syntax to call this method, omit the first parameter. For more
information, see Extension Methods (Visual Basic) or Extension Methods (C# Programming Guide).

**Examples**

```csharp
expression as type
```

becomes

```csharp
expression.As<type>()
```

**See Also**

Reference
- AsExtensions Class
- W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
## AsExtensions AsBase64 Method

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AsBase64(Byte)</code></td>
<td>Converts a byte array to a Base64 encoded string</td>
</tr>
<tr>
<td><code>AsBase64(String)</code></td>
<td>Converts a string to Base64 encoding</td>
</tr>
<tr>
<td><code>AsBase64(String, Encoding)</code></td>
<td>Converts a string to Base64 encoding</td>
</tr>
</tbody>
</table>

### See Also

Reference
- AsExtensions Class
- W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
AsExtensionsAsBase64 Method (Byte)

Converts a byte array to a Base64 encoded string

Namespace:  W
Assembly:  Tungsten.As (in Tungsten.As.dll) Version: 2.0.1

Syntax

```csharp
public static string AsBase64(
    this byte[] this
)
```

Parameters

- **this**
  - Type: SystemByte
  - The string to convert to Base64 encoding

Return Value

- Type: String
- The Base64 encoded string

Usage Note

In Visual Basic and C#, you can call this method as an instance method on any object of type . When you use instance method syntax to call this method, omit the first parameter. For more information, see Extension Methods (Visual Basic) or Extension Methods (C# Programming Guide).

See Also
Reference

AsExtensions Class
AsBase64 Overload
W Namespace

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
</tr>
</tbody>
</table>
AsExtensionsAsBase64 Method (String)

Converts a string to Base64 encoding

Namespace:  W
Assembly:  Tungsten.As (in Tungsten.As.dll) Version: 2.0.1

Syntax

```csharp
public static string AsBase64(
    this string this
)
```

Parameters

`this`
Type: `System.String`
The string to convert to Base64 encoding

Return Value
Type: `String`
The Base64 encoded string

Usage Note
In Visual Basic and C#, you can call this method as an instance method on any object of type `String`. When you use instance method syntax to call this method, omit the first parameter. For more information, see Extension Methods (Visual Basic) or Extension Methods (C# Programming Guide).

See Also
Reference
AsExtensions Class
AsBase64 Overload
W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

W
AsExtensionsAsString Base64 Method (String, Encoding)

Converts a string to Base64 encoding

**Namespace:** W
**Assembly:** Tungsten.As (in Tungsten.As.dll) Version: 2.0.1

**Syntax**

```csharp
public static string AsBase64(
    string this,
    Encoding encoding
)
```

**Parameters**

- **this**
  - Type: System.String
  - The string to convert to Base64 encoding

- **encoding**
  - Type: System.Text.Encoding
  - The encoding to use

**Return Value**

- Type: String
  - The Base64 encoded string

**Usage Note**

In Visual Basic and C#, you can call this method as an instance method on any object of type String. When you use instance method syntax to call this method, omit the first parameter. For more information, see Extension Methods (Visual Basic) or Extension
Methods (C# Programming Guide).

See Also

Reference
AsExtensions Class
AsBase64 Overload
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
# AsExtensions\(\text{AsBytes}\) Method

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\text{AsBytes(String)})</td>
<td>Converts a string to an encoded byte array</td>
</tr>
<tr>
<td>(\text{AsBytes(String, Encoding)})</td>
<td>Converts a string to an encoded byte array</td>
</tr>
</tbody>
</table>

## See Also

Reference

- AsExtensions Class
- W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

W
**AsExtensionsAsBytes Method (String)**

Converts a string to an encoded byte array

**Namespace:** W  
**Assembly:** Tungsten.As (in Tungsten.As.dll) Version: 2.0.1

**Syntax**

```csharp
public static byte[] AsBytes(
    this string this
)
```

**Parameters**

*this*

Type: `System.String`  
The string to convert to an encoded byte array

**Return Value**

Type: `Byte`  
A byte array encoding of the specified string

**Usage Note**

In Visual Basic and C#, you can call this method as an instance method on any object of type `String`. When you use instance method syntax to call this method, omit the first parameter. For more information, see [Extension Methods (Visual Basic)](Visual Basic) or [Extension Methods (C# Programming Guide)](C# Programming Guide).

**See Also**
Reference
AsExtensions Class
AsBytes Overload
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$\text{W}$
AsExtensionsAsBytes Method (String, Encoding)

Converts a string to an encoded byte array

**Namespace:**  W  
**Assembly:**  Tungsten.As (in Tungsten.As.dll) Version: 2.0.1

### Syntax

```
public static byte[] AsBytes(
    this string this,
    Encoding encoding
)
```

### Parameters

- **this**
  
  Type:  System.String
  
  The string to convert to an encoded byte array

- **encoding**
  
  Type:  System.Text.Encoding
  
  The encoding to use

### Return Value

Type:  Byte

A byte array encoding of the specified string

### Usage Note

In Visual Basic and C#, you can call this method as an instance method on any object of type String. When you use instance method syntax to call this method, omit the first parameter. For more information, see **Extension Methods (Visual Basic)** or **Extension**
See Also

Reference
AsExtensions Class
AsBytes Overload
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
AsExtensionsAsCompressed Method

Compresses the byte array using System.IO.Compression.DeflateStream

Namespace: W
Assembly: Tungsten.As (in Tungsten.As.dll) Version: 2.0.1

Syntax

C#

```
public static byte[] AsCompressed(
    this byte[] bytes
)
```

Parameters

`bytes`
Type: SystemByte
The byte array to compress

Return Value
Type: Byte
A byte array of compressed data

Usage Note
In Visual Basic and C#, you can call this method as an instance method on any object of type. When you use instance method syntax to call this method, omit the first parameter. For more information, see Extension Methods (Visual Basic) or Extension Methods (C# Programming Guide).

See Also
Reference
AsExtensions Class
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
AsExtensionsAsStream Method

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AsStream(Byte)</code></td>
<td>Creates a MemoryStream object and initializes it with the specified byte array</td>
</tr>
<tr>
<td><code>AsStream(String)</code></td>
<td>Creates a MemoryStream object and initializes it with the specified string</td>
</tr>
</tbody>
</table>

## See Also

Reference

- AsExtensions Class
- W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
AsExtensionsAsStream Method (Byte)

Creates a MemoryStream object and initializes it with the specified byte array

Namespace:  W
Assembly:  Tungsten.As (in Tungsten.As.dll) Version: 2.0.1

Syntax

```csharp
public static MemoryStream AsStream(
    this byte[] this
)
```

Parameters

this
Type: System.Byte
The byte array used in creating the MemoryStream

Return Value

Type: MemoryStream
A new MemoryStream initialized with the specified byte array

Usage Note

In Visual Basic and C#, you can call this method as an instance method on any object of type . When you use instance method syntax to call this method, omit the first parameter. For more information, see Extension Methods (Visual Basic) or Extension Methods (C# Programming Guide).

See Also
Reference
AsExtensions Class
AsStream Overload
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
AsExtensionsAsStream Method (String)

Creates a MemoryStream object and initializes it with the specified string

**Namespace:** W  
**Assembly:** Tungsten.As (in Tungsten.As.dll) Version: 2.0.1

### Syntax

```csharp
public static MemoryStream AsStream(
    this string this
)
```

### Parameters

* **this**  
  Type: System.String  
  The string used in creating the MemoryStream

### Return Value

Type: MemoryStream  
A new MemoryStream initilized with the specified string

### Usage Note

In Visual Basic and C#, you can call this method as an instance method on any object of type String. When you use instance method syntax to call this method, omit the first parameter. For more information, see Extension Methods (Visual Basic) or Extension Methods (C# Programming Guide).

### See Also
Reference

AsExtensions Class
AsStream Overload
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
# AsExtensionsAsString Method

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AsString(Byte)</code></td>
<td>Converts an encoded byte array to a string</td>
</tr>
<tr>
<td><code>AsString(Byte, Encoding)</code></td>
<td>Converts an encoded byte array to a string</td>
</tr>
<tr>
<td><code>AsString(Byte, Int32, Int32)</code></td>
<td>Converts an encoded byte array to a string</td>
</tr>
<tr>
<td><code>AsString(Byte, Int32, Int32, Encoding)</code></td>
<td>Converts an encoded byte array to a string</td>
</tr>
</tbody>
</table>

See Also

Reference
- [AsExtensions Class](#)
- [W Namespace](#)

Copyright © 2018 Jordan Duerksen
Tungsten

\[ W \]
AsExtensionsAsString Method (Byte)

Converts an encoded byte array to a string

**Namespace:** W  
**Assembly:** Tungsten.As (in Tungsten.As.dll) Version: 2.0.1

### Syntax

```
public static string AsString(
    this byte[] this
)
```

### Parameters

- `this`  
  - Type: System.Byte  
  - The encoded byte array to convert to a string

### Return Value

- Type: String  
  - The string representation of the encoded byte array

### Usage Note

In Visual Basic and C#, you can call this method as an instance method on any object of type `byte[]`. When you use instance method syntax to call this method, omit the first parameter. For more information, see [Extension Methods (Visual Basic)] or [Extension Methods (C# Programming Guide)].

### See Also
Reference
AsExtensions Class
AsString Overload
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$\text{W}$
AsExtensionsAsString Method (Byte, Encoding)

Converts an encoded byte array to a string

**Namespace:**  W  
**Assembly:**  Tungsten.As (in Tungsten.As.dll) Version: 2.0.1

### Syntax

```csharp
public static stringAsString(
    this byte[] this,
    Encoding encoding)
```

### Parameters

- **this**
  
  Type:  System.Byte  
  The encoded byte array to convert to a string

- **encoding**
  
  Type:  System.Text.Encoding  
  The encoding to use

### Return Value

Type:  String  
The string representation of the encoded byte array

### Usage Note

In Visual Basic and C#, you can call this method as an instance method on any object of type . When you use instance method syntax to call this method, omit the first parameter. For more information, see  Extension Methods (Visual Basic)  or  Extension
See Also

Reference
AsExtensions Class
AsString Overload
W Namespace
Tungsten

W
AsExtensionsAsString Method (Byte, Int32, Int32)

Converts an encoded byte array to a string

Namespace: W
Assembly: Tungsten.As (in Tungsten.As.dll) Version: 2.0.1

Syntax

```csharp
public static stringAsString(
    this byte[] this,
    int index,
    int count
)
```

Parameters

- **this**
  Type: SystemByte
  The encoded byte array to convert to a string
- **index**
  Type: SystemInt32
  The starting index
- **count**
  Type: SystemInt32
  The number of bytes to convert

Return Value

Type: String
The string representation of the encoded byte array

Usage Note
In Visual Basic and C#, you can call this method as an instance method on any object of type. When you use instance method syntax to call this method, omit the first parameter. For more information, see Extension Methods (Visual Basic) or Extension Methods (C# Programming Guide).

See Also

Reference
AsExtensions Class
AsString Overload
W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
AsExtensionsAsString Method
(Byte, Int32, Int32, Encoding)

Converts an encoded byte array to a string

Namespace: W
Assembly: Tungsten.As (in Tungsten.As.dll) Version: 2.0.1

Syntax

C#

```csharp
public static string AsString(
    this byte[] this,
    int index,
    int count,
    Encoding encoding
)
```

Parameters

- **this**
  - Type: System.Byte
  - The encoded byte array to convert to a string
- **index**
  - Type: System.Int32
  - The starting index
- **count**
  - Type: System.Int32
  - The number of bytes to convert
- **encoding**
  - Type: System.Text.Encoding
  - The encoding to use
Return Value
Type: String
The string representation of the encoded byte array

Usage Note
In Visual Basic and C#, you can call this method as an instance method on any object of type . When you use instance method syntax to call this method, omit the first parameter. For more information, see Extension Methods (Visual Basic) or Extension Methods (C# Programming Guide).

See Also

Reference
AsExtensions Class
AsString Overload
W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

\( W \)
CallResult Class

A non-generic return value for a function. CallResult encapsulates a success/failure and an exception.

Inheritance Hierarchy

- System
- Object
- W
- CallResult
- W
- CallResult
- TResult

Namespace:  W
Assembly:  Tungsten.CallResult (in Tungsten.CallResult.dll) Version: 2.0.0

Syntax

```csharp
public class CallResult
```

The CallResult type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CallResult</td>
<td>Default constructor, initializes Success to false</td>
</tr>
<tr>
<td>CallResult(Boolean)</td>
<td>Constructor which accepts an initial value for Success</td>
</tr>
<tr>
<td>CallResult(Boolean, Exception)</td>
<td>Constructor which accepts an initial value for Success and an initial value for</td>
</tr>
</tbody>
</table>
## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Empty</td>
<td>Provides a new instance of an uninitialized CallResult</td>
</tr>
<tr>
<td>Exception</td>
<td>Provide exception data to the caller if desired</td>
</tr>
<tr>
<td>Success</td>
<td>Set to True if the function succeeds, otherwise False</td>
</tr>
</tbody>
</table>

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTuple</td>
<td>Creates a Tuple from the properties</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object. (Inherited from Object.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object. (Inherited from Object.)</td>
</tr>
</tbody>
</table>

## Top

### Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot; This method should be called in the constructor of any class which has IOwnedProperty members (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTType(FuncTType)</td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTType(FuncTType)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>--------</td>
<td>----------------------------------------------------------</td>
</tr>
</tbody>
</table>

See Also

Reference

W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

W
## CallResult Constructor

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CallResult</td>
<td>Default constructor, initializes Success to false</td>
</tr>
<tr>
<td>CallResult(Boolean)</td>
<td>Constructor which accepts an initial value for Success</td>
</tr>
<tr>
<td>CallResult(Boolean, Exception)</td>
<td>Constructor which accepts an initial value for Success and an initial value for Exception</td>
</tr>
</tbody>
</table>

### See Also

Reference

- CallResult Class
- W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\[ \text{W} \]
CallResult Constructor

Default constructor, initializes Success to false

**Namespace:** W  
**Assembly:** Tungsten.CallResult (in Tungsten.CallResult.dll) Version: 2.0.0

### Syntax

```csharp
public CallResult()
```

### See Also

Reference  
CallResult Class  
CallResult Overload  
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
CallResult Constructor (Boolean)

Constructor which accepts an initial value for Success

Namespace: W
Assembly: Tungsten.CallResult (in Tungsten.CallResult.dll) Version: 2.0.0

Syntax

```csharp
public CallResult(
    bool success
)
```

Parameters

success
  Type: System.Boolean
  The initial value for Success

See Also

Reference
  CallResult Class
  CallResult Overload
  W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
CallResult Constructor (Boolean, Exception)

Constructor which accepts an initial value for Success and an initial value for Exception

**Namespace:**  W  
**Assembly:**  Tungsten.CallResult (in Tungsten.CallResult.dll)  
**Version:**  2.0.0

### Syntax

```csharp
public CallResult(
    bool success,
    Exception e
)
```

### Parameters

**success**
- Type: `SystemBoolean`
- The initial value for Success

**e**
- Type: `SystemException`
- The initial value for Exception

### See Also

**Reference**
- CallResult Class
- CallResult Overload
- W Namespace
Tungsten

\( W \)
# CallResult Properties

The **CallResult** type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Empty</td>
<td>Provides a new instance of an uninitialized CallResult</td>
</tr>
<tr>
<td>Exception</td>
<td>Provide exception data to the caller if desired</td>
</tr>
<tr>
<td>Success</td>
<td>Set to True if the function succeeds, otherwise False</td>
</tr>
</tbody>
</table>

## See Also

**Reference**

- CallResult Class
- W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
CallResultEmpty Property

Provides a new instance of an uninitialized CallResult

Namespace: W
Assembly: Tungsten.CallResult (in Tungsten.CallResult.dll) Version: 2.0.0

Syntax

```csharp
public static CallResult Empty {
  get; }
```

Property Value
Type: CallResult

See Also

Reference
CallResult Class
W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$^{	ext{W}}$
CallResultException Property

Provide exception data to the caller if desired

Namespace:  W
Assembly:  Tungsten.CallResult (in Tungsten.CallResult.dll) Version: 2.0.0

Syntax

```csharp
public Exception Exception { get; set; }
```

Property Value
Type: Exception

See Also

Reference
CallResult Class
W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
CallResultSuccess Property

Set to True if the function succeeds, otherwise False

Namespace: W
Assembly: Tungsten.CallResult (in Tungsten.CallResult.dll) Version: 2.0.0

**Syntax**

```c#
public bool Success { get; set; }
```

Property Value
Type: Boolean

**See Also**

Reference
CallResult Class
W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
# CallResult Methods

The `CallResult` type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTuple</td>
<td>Creates a Tuple from the properties</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations</td>
</tr>
<tr>
<td></td>
<td>before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <code>Type</code> of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current <code>Object</code>.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
</tbody>
</table>
### Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ToString</strong></td>
<td>Returns a string that represents the current object. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>AsTType</strong></td>
<td>Use Generic syntax for the as operator. (Defined by <strong>AsExtensions</strong>.)</td>
</tr>
<tr>
<td><strong>InitializeProperties</strong></td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by <strong>PropertyHostExtensions</strong>.)</td>
</tr>
<tr>
<td><strong>InLock(Action)</strong></td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by <strong>MonitorExtensions</strong>.)</td>
</tr>
<tr>
<td><strong>InLockTType(FuncTType)</strong></td>
<td>Overloaded. Performs the function in a Monitor lock. (Defined by <strong>MonitorExtensions</strong>.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync&lt;TType&gt;(Func&lt;TType&gt;)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>
See Also

Reference
CallResult Class
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
CallResultAsTuple Method

Creates a Tuple from the properties

Namespace: W
Assembly: Tungsten.CallResult (in Tungsten.CallResult.dll) Version: 2.0.0

Syntax

```csharp
public Tuple<bool, Exception> AsTuple()
```

Return Value
Type: TupleBoolean, Exception
A Tuple containing the property values

Remarks
This is essentially a hard-coded Tuple; reflection is not used to obtain the properties. As a result, properties of child classes will not be in the returned Tuple.

See Also

Reference
CallResult Class
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
CallResult<TResult> Class

Generic class to be used as a return value. CallResult encapsulates a success/failure, an exception and a return value.

Inheritance Hierarchy

SystemObject  WCallResult  WCallResult<TResult>

Namespace:  W
Assembly:  Tungsten.CallResult (in Tungsten.CallResult.dll) Version: 2.0.0

Syntax

```csharp
public class CallResult<TResult> : CallResult
```

Type Parameters

<TResult>
The type to be used for the Result member (the return value of the function)

The CallResult<TResult> type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CallResult&lt;TResult&gt;</td>
<td>Default constructor</td>
</tr>
<tr>
<td>CallResult&lt;TResult&gt;(Boolean)</td>
<td>Constructor</td>
</tr>
</tbody>
</table>
accepting an initial Success value

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CallResultTResult(Boolean, TResult)</td>
<td>Constructor accepting an initial Success value and an initial Result value</td>
</tr>
<tr>
<td>CallResultTResult(Boolean, TResult, Exception)</td>
<td>Constructor accepting an initial Success value, an initial Result value and an initial Exception value</td>
</tr>
</tbody>
</table>

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Empty</td>
<td>Provides a new instance of an uninitialized CallResult&lt;TResult&gt;</td>
</tr>
<tr>
<td>Exception</td>
<td>Provide exception data to the caller if desired (Inherited from CallResult.)</td>
</tr>
<tr>
<td>Result</td>
<td>The return value</td>
</tr>
<tr>
<td>Success</td>
<td>Set to True if the function succeeds, otherwise False (Inherited from CallResult.)</td>
</tr>
</tbody>
</table>
## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTuple</td>
<td>Creates a Tuple from the properties (Inherited from CallResult.)</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
</tbody>
</table>
## Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTType(FuncTType)</td>
<td>Overloaded. Performs the function in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock. (Defined by</td>
</tr>
</tbody>
</table>
MonitorExtensions.)

InLockAsyncTTType(FuncTTType) Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)

IsDirty Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)

Lock Performs a Monitor lock (Defined by MonitorExtensions.)

MarkAsClean Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)

Unlock Performs a Monitor unlock (Defined by MonitorExtensions.)

See Also

Reference

W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
CallResultTResult Constructor

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CallResultTResult</td>
<td>Default constructor</td>
</tr>
<tr>
<td>CallResultTResult(Boolean)</td>
<td>Constructor accepting an initial Success value</td>
</tr>
<tr>
<td>CallResultTResult(Boolean, TResult)</td>
<td>Constructor accepting an initial Success value and an initial Result value</td>
</tr>
<tr>
<td>CallResultTResult(Boolean, TResult, Exception)</td>
<td>Constructor accepting an initial Success value, an initial Result value and an initial Exception value</td>
</tr>
</tbody>
</table>

See Also

Reference
- CallResultTResult Class
- W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
CallResultTResult Constructor

Default constructor

Namespace: `W`
Assembly: `Tungsten.CallResult` (in `Tungsten.CallResult.dll`) Version: 2.0.0

### Syntax

```csharp
public CallResult()
```

### See Also

Reference
- `CallResultTResult Class`
- `CallResultTResult Overload`
- `W Namespace`

Copyright @ 2018 Jordan Duerksen
Tungsten

W
CallResult<TResult> Constructor (Boolean)

Constructor accepting an initial Success value

Namespace: W
Assembly: Tungsten.CallResult (in Tungsten.CallResult.dll) Version: 2.0.0

## Syntax

```csharp
public CallResult(
    bool success
)
```

## Parameters

success
Type: SystemBoolean
The initial Success value

## See Also

Reference
CallResultTResult Class
CallResultTResult Overload
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
CallResult<TResult> Constructor (Boolean, TResult)

Constructor accepting an initial Success value and an initial Result value

Namespace: W
Assembly: Tungsten.CallResult (in Tungsten.CallResult.dll) Version: 2.0.0

Syntax

```csharp
public CallResult(
    bool success,
    TResult result
)
```

Parameters

**success**
- Type: System.Boolean
- The initial Success value

**result**
- Type: TResult
- The initial Result value

See Also

Reference
- CallResult<TResult> Class
- CallResult<TResult> Overload
- W Namespace
Tungsten

$^\text{W}$
CallResult{TResult} Constructor (Boolean, TResult, Exception)

Constructor accepting an initial Success value, an initial Result value and an initial Exception value

**Namespace:** W  
**Assembly:** Tungsten.CallResult (in Tungsten.CallResult.dll)  
**Version:** 2.0.0

### Syntax

```csharp
public CallResult(
    bool success,
    TResult result,
    Exception e
)
```

### Parameters

- **success**
  - Type: `System.Boolean`
  - The initial value for Success

- **result**
  - Type: `TResult`
  - The initial Result value

- **e**
  - Type: `System.Exception`
  - An exception object, if an exception occurred

### See Also
Reference

CallResultTResult Class
CallResultTResult Overload
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
The `CallResult<TResult>` generic type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Empty</td>
<td>Provides a new instance of an uninitialized <code>callResult&lt;TResult&gt;</code></td>
</tr>
<tr>
<td>Exception</td>
<td>Provide exception data to the caller if desired (Inherited from <code>callResult</code>.)</td>
</tr>
<tr>
<td>Result</td>
<td>The return value</td>
</tr>
<tr>
<td>Success</td>
<td>Set to True if the function succeeds, otherwise False (Inherited from <code>callResult</code>.)</td>
</tr>
</tbody>
</table>

See Also

Reference
- `CallResult<TResult` Class
- `W` Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
CallResult<TResult> Empty Property

Provides a new instance of an uninitialized CallResult<TResult>

**Namespace:**  W  
**Assembly:**  Tungsten.CallResult (in Tungsten.CallResult.dll)  
**Version:**  2.0.0

### Syntax

```csharp
public static CallResult<TResult> Empty { get; }
```

### Property Value

Type:  CallResult<TResult>

### See Also

Reference

- CallResult<TResult> Class
- W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
CallResult

**Property**

The return value

**Namespace:**  W  
**Assembly:**  Tungsten.CallResult (in Tungsten.CallResult.dll)  
**Version:**  2.0.0

## Syntax

```csharp
public TResult Result { get; set; }
```

**Property Value**

**Type:**  TResult

## See Also

Reference

CallResult

Copyright © 2018 Jordan Duerksen
CallResultTResult Methods

The CallResultTResult generic type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTuple</td>
<td>Creates a Tuple from the properties (Inherited from CallResult.)</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object. (Inherited from Object.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
</tbody>
</table>
### Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ToString</strong></td>
<td>Returns a string that represents the current object. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>AsTType</strong></td>
<td>Use Generic syntax for the as operator. (Defined by <strong>AsExtensions</strong>.)</td>
</tr>
<tr>
<td><strong>InitializeProperties</strong></td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot; This method should be called in the constructor of any class which has IOwnedProperty members (Defined by <strong>PropertyHostExtensions</strong>.)</td>
</tr>
<tr>
<td><strong>InLock(Action)</strong></td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by <strong>MonitorExtensions</strong>.)</td>
</tr>
<tr>
<td><strong>InLockTType(FuncTType)</strong></td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by <strong>MonitorExtensions</strong>.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>InLockAsync(Action)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>InLockAsyncTTType(Func&lt;TType&gt;)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>IsDirty</code></td>
<td>Scans the IsDirty value of each field and property of type <code>IProperty</code> (Defined by <code>PropertyHostExtensions</code>.)</td>
</tr>
<tr>
<td><code>Lock</code></td>
<td>Performs a Monitor lock (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>MarkAsClean</code></td>
<td>Scans each field and property of type <code>IProperty</code> and sets its IsDirty flag to false (Defined by <code>PropertyHostExtensions</code>.)</td>
</tr>
<tr>
<td><code>Unlock</code></td>
<td>Performs a Monitor unlock (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
</tbody>
</table>
See Also

Reference
CallResultT
Result Class
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$^W$
ConsoleStringExtensions Class

Extension methods related to the Console

Inheritance Hierarchy

- System
  - Object
  - ConsoleStringExtensions

Namespace: W

Syntax

C#

```csharp
public static class ConsoleStringExtensions
```

The ConsoleStringExtensions type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>WriteFullConsoleLine</strong></td>
<td>Writes text to the console. Columns which the text doesn't overwrite are filled with the specified padding character.</td>
</tr>
<tr>
<td><strong>WriteToConsole</strong></td>
<td>Writes text to the console at the specified location (x,y)</td>
</tr>
</tbody>
</table>
See Also

Reference

W Namespace

Copyright © 2018 Jordan Duerksen
ConsoleStringExtensions

Methods

The `ConsoleStringExtensions` type exposes the following members.

⚠ Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WriteFullConsoleLine</td>
<td>Writes text to the console. Columns which the text doesn't overwrite are filled with the specified padding character.</td>
</tr>
<tr>
<td>WriteToConsole</td>
<td>Writes text to the console at the specified location (x,y)</td>
</tr>
</tbody>
</table>

See Also

Reference

`ConsoleStringExtensions Class`

W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
ConsoleStringExtensionsWriteFullConsoleLine Method

Writes text to the console. Columns which the text doesn’t overwrite are filled with the specified padding character.

**Namespace:** W

**Assembly:** Tungsten.Console (in Tungsten.Console.dll) Version: 2.0.0

### Syntax

```csharp
public static void WriteFullConsoleLine(
    this string message,
    int verticalOffset = -1,
    char paddingChar = ''
)
```

### Parameters

- **message**
  - Type: SystemString
  - The text to write to the console

- **verticalOffset (Optional)**
  - Type: SystemInt32
  - The line (or row) on which to write the text. 0 is the top of the Console.

- **paddingChar (Optional)**
  - Type: SystemChar
  - The padding character used to fill the unused portion of the line

### Usage Note

In Visual Basic and C#, you can call this method as an instance
method on any object of type `String`. When you use instance method syntax to call this method, omit the first parameter. For more information, see `Extension Methods (Visual Basic)` or `Extension Methods (C# Programming Guide)`.  

See Also

Reference
`ConsoleStringExtensions Class`
`W Namespace`

Copyright © 2018 Jordan Duerksen
Tungsten

W
ConsoleStringExtensions

Method

Writes text to the console at the specified location (x, y)

Namespace: W

Syntax

```csharp
public static void WriteToConsole(
    this string message,
    int x,
    int y
)
```

Parameters

- **message**
  - Type: SystemString
  - The text to write to the console
- **x**
  - Type: SystemInt32
  - The column on which to start writing text
- **y**
  - Type: SystemInt32
  - The row on which to start writing text

Usage Note

In Visual Basic and C#, you can call this method as an instance method on any object of type String. When you use instance method syntax to call this method, omit the first parameter. For more
information, see Extension Methods (Visual Basic) or Extension Methods (C# Programming Guide).

Remarks

The Console origin (0,0) is top-left

See Also

Reference

ConsoleStringExtensions Class
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
EventTemplate TSender Class

Wraps the functionality of delegate, event and RaiseXXX into a single class

Inheritance Hierarchy

```
SystemObject  WEVENTTEMPLATETSENDER
```

Namespace:  W
Assembly:  Tungsten.EventTemplate (in Tungsten.EventTemplate.dll)
Version:  2.0.0

Syntax

```
public class EventTemplate<TSender>
```

Type Parameters

`TSender`
The object raising the event

The EventTemplate TSender type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EventTemplateTSender</td>
<td>Initializes a new instance of the EventTemplateTSender class</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object. (Inherited from Object.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>Raise</td>
<td>Raises the template event</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object. (Inherited from Object.)</td>
</tr>
</tbody>
</table>
### Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>⚡ OnRaised</td>
<td>The template event</td>
</tr>
</tbody>
</table>

### Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>⚡ AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot; This method should be called in the constructor of any class which has IOwnedProperty members (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Overloaded. Performs the function in a Monitor lock</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock</td>
</tr>
<tr>
<td>InLockAsyncTType(FuncTType)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock</td>
</tr>
</tbody>
</table>
Tungsten

$W$
EventTemplate TSender Constructor

Initializes a new instance of the EventTemplate TSender class.

**Namespace:** W
**Assembly:** Tungsten.EventTemplate (in Tungsten.EventTemplate.dll)
**Version:** 2.0.0

### Syntax

```csharp
public EventTemplate()
```

### See Also

Reference
- EventTemplate TSender Class
- W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
# EventTemplate<code>TSender</code> Methods

The `EventTemplate<code>TSender</code>` generic type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from &lt;code&gt;Object&lt;/code&gt;.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from &lt;code&gt;Object&lt;/code&gt;.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from &lt;code&gt;Object&lt;/code&gt;.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the &lt;code&gt;Type&lt;/code&gt; of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from &lt;code&gt;Object&lt;/code&gt;.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current &lt;code&gt;Object&lt;/code&gt;.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from &lt;code&gt;Object&lt;/code&gt;.)</td>
</tr>
<tr>
<td>Raise</td>
<td>Raises the template event</td>
</tr>
</tbody>
</table>
### Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ToString</code></td>
<td>Returns a string that represents the current object. (Inherited from Object.)</td>
</tr>
<tr>
<td><code>AsTType</code></td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td><code>InitializeProperties</code></td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td><code>InLock(Action)</code></td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><code>InLockTType(FuncTType)</code></td>
<td>Overloaded. Performs the function in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>Method Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTTType(FuncTTType)</td>
<td>Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets it's IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>
See Also

Reference
EventTemplateTSender Class
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
EventTemplate \textit{TSender} \textbf{Raise} Method

Raises the template event

\textbf{Namespace: } \textit{W}  \\
\textbf{Assembly: } Tungsten.EventTemplate (in Tungsten.EventTemplate.dll)  \\
Version: 2.0.0

\section*{Syntax}

\begin{verbatim}
public void Raise(
    TSender sender
)
\end{verbatim}

\textbf{Parameters}

\textit{sender}

Type: \textit{TSender}  \\
The object which raised this event

\section*{See Also}

\textbf{Reference}

EventTemplate\textit{TSender Class}  \\
\textit{W Namespace}

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
</tr>
</tbody>
</table>
EventTemplate TSender Events

The EventTemplate TSender generic type exposes the following members.

### Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>🌡️ OnRaised</td>
<td>The template event</td>
</tr>
</tbody>
</table>

See Also

Reference
- EventTemplate TSender Class
- W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

W
EventTemplate<TSender> OnRaised Event

The template event

**Namespace:**  W  
**Assembly:**  Tungsten.EventTemplate (in Tungsten.EventTemplate.dll)  
**Version:**  2.0.0

**Syntax**

```csharp
public event Action<TSender> OnRaised
```

**Value**

Type:  **SystemAction<TSender>**

**See Also**

Reference

EventTemplate<TSender> Class  
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
EventTemplate \textit{TSender, Arg1} Class

Wraps the functionality of delegate, event and RaiseXXX into a single class

\section*{Inheritance Hierarchy}

\begin{itemize}
  \item \texttt{SystemObject}
  \item \texttt{WEventTemplateTSender, Arg1}
\end{itemize}

\textbf{Namespace:} \texttt{W}  
\textbf{Assembly:} Tungsten.EventTemplate (in Tungsten.EventTemplate.dll)  
\textbf{Version:} 2.0.0

\section*{Syntax}

\begin{verbatim}
public class EventTemplate<TSender, Arg1>
\end{verbatim}

\textbf{Type Parameters}

- \texttt{TSender}  
  The object raising the event  
- \texttt{Arg1}  
  The first argument

The \texttt{EventTemplateTSender, Arg1} type exposes the following members.

\section*{Constructors}

\begin{table}[h]
\begin{tabular}{|l|l|}
\hline
\textbf{Name} & \textbf{Description} \\
\hline
\texttt{EventTemplateTSender} & Initializes a new \\
\hline
\end{tabular}
\end{table}
instance of the EventTemplateTSender, Arg1 class

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current Object.</td>
</tr>
<tr>
<td><strong>Raise</strong></td>
<td>Raises the template event</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Returns a string that represents the current object.</td>
</tr>
</tbody>
</table>
Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnRaised</td>
<td>The template event</td>
</tr>
</tbody>
</table>

Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>InLockTType(Func&lt;TType&gt;)</code></td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>InLockAsync(Action)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>InLockAsyncTType(Func&lt;TType&gt;)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>IsDirty</code></td>
<td>Scans the <code>IsDirty</code> value of each field and property of type <code>IProperty</code> (Defined by <code>PropertyHostExtensions</code>.)</td>
</tr>
<tr>
<td><code>Lock</code></td>
<td>Performs a Monitor lock (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>MarkAsClean</code></td>
<td>Scans each field and property of type <code>IProperty</code> and sets its <code>IsDirty</code> flag to false (Defined by <code>PropertyHostExtensions</code>.)</td>
</tr>
<tr>
<td><code>Unlock</code></td>
<td>Performs a Monitor unlock</td>
</tr>
<tr>
<td>Tungsten</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td></td>
</tr>
<tr>
<td>W</td>
<td></td>
</tr>
</tbody>
</table>
EventTemplate<TSender, Arg1> Constructor

Initializes a new instance of the EventTemplate<TSender, Arg1> class

Namespace:  W
Assembly:  Tungsten.EventTemplate (in Tungsten.EventTemplate.dll)
Version:  2.0.0

Syntax

```c#
public EventTemplate()
```

See Also

Reference

EventTemplate<TSender, Arg1> Class
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W^7$
EventTemplate $TSender, Arg1$

Methods

The EventTemplate $TSender, Arg1$ generic type exposes the following members.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified object is equal to the current object. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as the default hash function. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
</tbody>
</table>
Raise

Raises the template event

ToString

Returns a string that represents the current object. (Inherited from Object.)

---

## Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTType(FuncTType)</td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>InLockAsync(Action)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><code>InLockAsyncTTType(Func&lt;TType&gt;)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><code>IsDirty</code></td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td><code>Lock</code></td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><code>MarkAsClean</code></td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td><code>Unlock</code></td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>
See Also

Reference
EventTemplateTSender, Arg1 Class
W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
EventTemplate TSender, Arg1 Raise Method

 Raises the template event

 **Namespace:** W  
 **Assembly:** Tungsten.EventTemplate (in Tungsten.EventTemplate.dll)  
 **Version:** 2.0.0

## Syntax

```csharp
public void Raise(
    TSender sender,
    Arg1 arg1
)
```

### Parameters

**sender**

Type: TSender  
The object which raised this event

**arg1**

Type: Arg1  
The first argument

## See Also

Reference

EventTemplate TSender, Arg1 Class  
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
EventTemplate\textit{T}Sender, \textit{Arg}1

Events

The \texttt{EventTemplate\textit{T}Sender, \textit{Arg}1} generic type exposes the following members.

### Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{OnRaised}</td>
<td>The template event</td>
</tr>
</tbody>
</table>

See Also

Reference

\texttt{EventTemplate\textit{T}Sender, \textit{Arg}1 Class}

\texttt{W Namespace}

Copyright @ 2018 Jordan Duerksen
Tungsten

W
EventTemplate TSender, Arg1 On Raised Event

The template event

**Namespace:**  
W  

**Assembly:**  
Tungsten.EventTemplate (in Tungsten.EventTemplate.dll)  
Version: 2.0.0  

### Syntax

```csharp
public event Action<TSender, Arg1> OnRaised
```

**Value**

Type: SystemAction<TSender, Arg1>

### See Also

Reference

EventTemplate TSender, Arg1 Class  
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$^W$
EventTemplate TSender, Arg1, Arg2 Class

Wraps the functionality of delegate, event and RaiseXXX into a single class

Inheritance Hierarchy

**SystemObject**  **WEventTemplateTSender, Arg1, Arg2**

**Namespace:**  **W**
**Assembly:**  **Tungsten.EventTemplate (in Tungsten.EventTemplate.dll)**
**Version:**  **2.0.0**

Syntax

```c#
public class EventTemplate<TSender, Arg1, Arg2>
```

Type Parameters

- **TSender**
  The object raising the event
- **Arg1**
  The first argument
- **Arg2**
  The second argument

The EventTemplateTSender, Arg1, Arg2 type exposes the following members.

Constructors
### EventTemplate\(T\)\(\text{Sender}, \text{Arg1}, \text{Arg2}\)

Initializes a new instance of the `EventTemplate\(T\)\(\text{Sender}, \text{Arg1}, \text{Arg2}\)` class.

#### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified object is equal to the current object. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as the default hash function. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <code>Type</code> of the current instance. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current <code>Object</code>. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>Raise</strong></td>
<td>Raises the template event</td>
</tr>
</tbody>
</table>
**ToString**

Returns a string that represents the current object. (Inherited from **Object**.)

---

**Top**

**Events**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>⚒ OnRaised</td>
<td>The template event</td>
</tr>
</tbody>
</table>

---

**Top**

**Extension Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>⚒ ⚒ AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by <strong>AsExtensions</strong>.)</td>
</tr>
<tr>
<td>⚒  InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by <strong>PropertyHostExtensions</strong>.)</td>
</tr>
<tr>
<td>⚒  InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>InLockTType(Func&lt;TType&gt;)</td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync&lt;TType&gt;(Func&lt;TType&gt;)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
</tbody>
</table>
Unlock

Performs a Monitor unlock (Defined by MonitorExtensions.)

See Also

Reference
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
EventTemplate TSender, Arg1, Arg2 Constructor

Initializes a new instance of the EventTemplate TSender, Arg1, Arg2 class

Namespace: W
Assembly: Tungsten.EventTemplate (in Tungsten.EventTemplate.dll)
Version: 2.0.0

Syntax

```csharp
public EventTemplate()
```

See Also

Reference
EventTemplate TSender, Arg1, Arg2 Class
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$^7W$
EventTemplate TSender, Arg1, Arg2 Methods

The EventTemplate TSender, Arg1, Arg2 generic type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup</td>
</tr>
<tr>
<td></td>
<td>operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
</tbody>
</table>
### Raise

Raises the template event

### ToString

Returns a string that represents the current object. (Inherited from `Object`.)

---

## Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="https://github.com/andyjorm/media/blob/06549451ef7e61dc12806da1a2f7b8f2915a1613/1.png" alt="" /> AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by <code>AsExtensions</code>.)</td>
</tr>
<tr>
<td><img src="https://github.com/andyjorm/media/blob/06549451ef7e61dc12806da1a2f7b8f2915a1613/1.png" alt="" /> InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by <code>PropertyHostExtensions</code>.)</td>
</tr>
<tr>
<td><img src="https://github.com/andyjorm/media/blob/06549451ef7e61dc12806da1a2f7b8f2915a1613/1.png" alt="" /> InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><img src="https://github.com/andyjorm/media/blob/06549451ef7e61dc12806da1a2f7b8f2915a1613/1.png" alt="" /> InLockTType(FuncTType)</td>
<td>Overloaded. Performs the function in a Monitor lock. (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTTType(FuncTTType)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>
See Also

Reference
EventTemplateTSender, Arg1, Arg2 Class
W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
EventTemplate

EventTemplate\(TSender, Arg1, Arg2\)

Raise Method

Raises the template event

**Namespace:**  \(W\)

**Assembly:**  Tungsten.EventTemplate (in Tungsten.EventTemplate.dll)

**Version:**  2.0.0

**Syntax**

```csharp
public void Raise(
    TSender sender,
    Arg1 arg1,
    Arg2 arg2
)
```

**Parameters**

- **sender**
  - **Type:**  \(TSender\)
  - The object which raised this event

- **arg1**
  - **Type:**  \(Arg1\)
  - The first argument

- **arg2**
  - **Type:**  \(Arg2\)
  - The second argument

**See Also**

Reference
EventTemplate TSender, Arg1, Arg2 Class
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
EventTemplate TSender, Arg1, Arg2 Events

The EventTemplate TSender, Arg1, Arg2 generic type exposes the following members.

Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>✨ OnRaised</td>
<td>The template event</td>
</tr>
</tbody>
</table>

Top

See Also

Reference
EventTemplate TSender, Arg1, Arg2 Class
W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$\text{W}$
EventTemplate TSender, Arg1, Arg2 OnRaised Event

The template event

**Namespace:** W  
**Assembly:** Tungsten.EventTemplate (in Tungsten.EventTemplate.dll)  
**Version:** 2.0.0

### Syntax

```csharp
public event Action<TSender, Arg1, Arg2> OnRaised;
```

- **Value**
  - **Type:** SystemAction<TSender, Arg1, Arg2>

### See Also

**Reference**  
EventTemplate TSender, Arg1, Arg2 Class  
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
EventTemplate\(T\)\(Sender, \ Arg1, \ Arg2, \ Arg3\) Class

Wraps the functionality of delegate, event and RaiseXXX into a single class

**Inheritance Hierarchy**

- `SystemObject`
- `WEventTemplateT\(Sender, \ Arg1, \ Arg2, \ Arg3\)`

**Namespace:** `W`

**Assembly:** `Tungsten.EventTemplate` (in `Tungsten.EventTemplate.dll`)

**Version:** 2.0.0

**Syntax**

```csharp
public class EventTemplate<TSender, Arg1, Arg2, Arg3>
```

Type Parameters

- `T\(Sender\)`
  - The object raising the event
- `Arg1`
  - The first argument
- `Arg2`
  - The second argument
- `Arg3`
  - The third argument

The `EventTemplateT\(Sender, Arg1, Arg2, Arg3\)` type exposes the following members.
### Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EventTemplateTSender, Arg1, Arg2, Arg3</td>
<td>Initializes a new instance of the <code>EventTemplateTSender, Arg1, Arg2, Arg3</code> class</td>
</tr>
</tbody>
</table>

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <code>Type</code> of the current instance. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current <code>Object</code>.</td>
</tr>
</tbody>
</table>
(Inherited from Object.)

- **Raise**
  
  Raises the template event

- **ToString**
  
  Returns a string that represents the current object.
  (Inherited from Object.)

### Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="onraised" /></td>
<td><strong>OnRaised</strong> The template event</td>
</tr>
</tbody>
</table>

### Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="asttype" /></td>
<td><strong>AsTType</strong> Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td><img src="image" alt="initializeproperties" /></td>
<td><strong>InitializeProperties</strong> Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot; This method should be called in the constructor of any class which has IOwnedProperty members (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Method Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTType(FuncTType)</td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTType(FuncTType)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to</td>
</tr>
</tbody>
</table>
false
(Defined by PropertyHostExtensions.)

Unlock
Performs a Monitor unlock
(Defined by MonitorExtensions.)

See Also

Reference
W Namespace
Tungsten

\( W \)
EventTemplate TSender, Arg1, Arg2, Arg3 Constructor

Initializes a new instance of the EventTemplate TSender, Arg1, Arg2, Arg3 class

Namespace: W
Assembly: Tungsten.EventTemplate (in Tungsten.EventTemplate.dll)
Version: 2.0.0

Syntax

```c#
public EventTemplate()
```

See Also

Reference
EventTemplate TSender, Arg1, Arg2, Arg3 Class
W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
### EventTemplate TSender, Arg1, Arg2, Arg3 Methods

The **EventTemplate TSender, Arg1, Arg2, Arg3** generic type exposes the following members.

#### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified object is equal to the current object. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as the default hash function. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <strong>Type</strong> of the current instance. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current <strong>Object</strong>. (Inherited from <strong>Object</strong>.)</td>
</tr>
</tbody>
</table>
Raise

Raises the template event

ToString

Returns a string that represents the current object. (Inherited from Object.)

---

## Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTType(FuncTType)</td>
<td>Overloaded. Performs the function in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(TType FuncTType)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>
See Also

Reference
EventTemplateTSender, Arg1, Arg2, Arg3 Class
W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

W
EventTemplate $TSender$, $Arg1$, $Arg2$, $Arg3$ Raise Method

Raises the template event

**Namespace:** W  
**Assembly:** Tungsten.EventTemplate (in Tungsten.EventTemplate.dll)  
**Version:** 2.0.0

**Syntax**

```csharp
public void Raise(
    TSender sender,
    Arg1 arg1,
    Arg2 arg2,
    Arg3 arg3
)
```

**Parameters**

sender
- **Type:** $TSender$
- The object which raised this event

arg1
- **Type:** $Arg1$
- The first argument

arg2
- **Type:** $Arg2$
- The second argument

arg3
- **Type:** $Arg3$
- The third argument
See Also

Reference
EventTemplateTSender, Arg1, Arg2, Arg3 Class
W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
EventTemplate TSender, Arg1, Arg2, Arg3 Events

The EventTemplate TSender, Arg1, Arg2, Arg3 generic type exposes the following members.

### Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>💥 OnRaised</td>
<td>The template event</td>
</tr>
</tbody>
</table>

See Also

Reference

EventTemplate TSender, Arg1, Arg2, Arg3 Class

W Namespace
EventTemplate TSender, Arg1, Arg2, Arg3

The template event

**Namespace:** W
**Assembly:** Tungsten.EventTemplate (in Tungsten.EventTemplate.dll)
**Version:** 2.0.0

### Syntax

[C#]
```
public event Action<TSender, Arg1, Arg2, Arg3> OnRaised
```

**Value Type:** SystemAction<TSender, Arg1, Arg2, Arg3>

### See Also

**Reference**
- EventTemplate TSender, Arg1, Arg2, Arg3 Class
- W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
EventTemplate\textit{TSender, Arg1, Arg2, Arg3, Arg4} Class

Wraps the functionality of delegate, event and RaiseXXX into a single class

\section*{Inheritance Hierarchy}

\begin{itemize}
  \item System\texttt{Object}
  \item \texttt{WEventTemplate\texttt{TSender, Arg1, Arg2, Arg3, Arg4}}
\end{itemize}

\textbf{Namespace: } \texttt{W}
\textbf{Assembly: } Tungsten.EventTemplate (in Tungsten.EventTemplate.dll)
\textbf{Version: } 2.0.0

\section*{Syntax}

\begin{verbatim}
public class EventTemplate<TSender, Arg1, Arg2, Arg3, Arg4>
\end{verbatim}

Type Parameters

\begin{itemize}
  \item \texttt{TSender}
    \begin{itemize}
      \item The object raising the event
    \end{itemize}
  \item \texttt{Arg1}
    \begin{itemize}
      \item The first argument
    \end{itemize}
  \item \texttt{Arg2}
    \begin{itemize}
      \item The second argument
    \end{itemize}
  \item \texttt{Arg3}
    \begin{itemize}
      \item The third argument
    \end{itemize}
  \item \texttt{Arg4}
    \begin{itemize}
      \item The fourth argument
    \end{itemize}
\end{itemize}

The \texttt{EventTemplate\texttt{TSender, Arg1, Arg2, Arg3, Arg4}} type exposes the
following members.

### Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EventTemplateTSender, Arg1, Arg2, Arg3, Arg4</td>
<td>Initializes a new instance of the <code>EventTemplateTSender, Arg1, Arg2, Arg3, Arg4</code> class</td>
</tr>
</tbody>
</table>

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <code>Type</code> of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
</tbody>
</table>
MemberwiseClone
Creates a shallow copy of the current Object. (Inherited from Object.)

Raise
Raises the template event

ToString
Returns a string that represents the current object. (Inherited from Object.)

Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnRaised</td>
<td>The template event</td>
</tr>
</tbody>
</table>

Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTType(FuncTType)</td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTType(FuncTType)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>

Scans each field and
<table>
<thead>
<tr>
<th>MarkAsClean</th>
<th>property of type IProperty and sets it's IsDirty flag to false (Defined by PropertyHostExtensions.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>

See Also

Reference
W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
EventTemplate<typeparamparam name="TSender">, Arg1, Arg2, Arg3, Arg4 Constructor

Initializes a new instance of the EventTemplate<typeparamparam name="TSender">, Arg1, Arg2, Arg3, Arg4 class

**Namespace:**  W
**Assembly:**  Tungsten.EventTemplate (in Tungsten.EventTemplate.dll)
**Version:**  2.0.0

⚠️ **Syntax**

```c#$
public EventTemplate()
```

⚠️ **See Also**

Reference  
EventTemplate<typeparamparam name="TSender">, Arg1, Arg2, Arg3, Arg4 Class  
W Namespace

Copyright © 2018 Jordan Duerksen
EventTemplate TSender, Arg1, Arg2, Arg3, Arg4 Methods

The EventTemplate TSender, Arg1, Arg2, Arg3, Arg4 generic type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object. (Inherited from Object.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Raise</td>
<td>Raises the template event</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Returns a string that represents the current object. (Inherited from <strong>Object</strong>.)</td>
</tr>
</tbody>
</table>

### Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AsTType</strong></td>
<td>Use Generic syntax for the as operator. (Defined by <strong>AsExtensions</strong>.)</td>
</tr>
<tr>
<td><strong>InitializeProperties</strong></td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by <strong>PropertyHostExtensions</strong>.)</td>
</tr>
<tr>
<td><strong>InLock(Action)</strong></td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by <strong>MonitorExtensions</strong>.)</td>
</tr>
<tr>
<td><strong>InLockTType(FuncTType)</strong></td>
<td>Overloaded. Performs the function in a Monitor lock. (Defined by <strong>MonitorExtensions</strong>.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td>InLockAsyncTTType(Func&lt;TType&gt;)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by <code>PropertyHostExtensions</code>.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by <code>PropertyHostExtensions</code>.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
</tbody>
</table>
See Also

Reference

EventTemplateTSender, Arg1, Arg2, Arg3, Arg4 Class
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
EventTemplate TSender, Arg1, Arg2, Arg3, Arg4

Raise Method

Raises the template event

**Namespace:**  W

**Assembly:**  Tungsten.EventTemplate (in Tungsten.EventTemplate.dll)

**Version:**  2.0.0

### Syntax

```
public void Raise(
    TSender sender,
    Arg1 arg1,
    Arg2 arg2,
    Arg3 arg3,
    Arg4 arg4
)
```

### Parameters

**sender**

Type:  TSender

The object which raised this event

**arg1**

Type:  Arg1

The first argument

**arg2**

Type:  Arg2

The second argument

**arg3**

Type:  Arg3
The third argument

\( \text{arg4} \)

Type: \( \text{Arg4} \)

The fourth argument

\section*{See Also}

Reference

\texttt{EventTemplateTSender, Arg1, Arg2, Arg3, Arg4 Class W Namespace}

Copyright @ 2018 Jordan Duerksen
Tungsten

W
EventTemplate $TSender$, $Arg1$, $Arg2$, $Arg3$, $Arg4$ Events

The EventTemplate $TSender$, $Arg1$, $Arg2$, $Arg3$, $Arg4$ generic type exposes the following members.

#### Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnRaised</td>
<td>The template event</td>
</tr>
</tbody>
</table>

See Also

Reference

EventTemplate $TSender$, $Arg1$, $Arg2$, $Arg3$, $Arg4$ Class

W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
EventTemplate TSender, Arg1, Arg2, Arg3, Arg4

The template event

**Namespace:** W
**Assembly:** Tungsten.EventTemplate (in Tungsten.EventTemplate.dll)
**Version:** 2.0.0

### Syntax

```csharp
public event Action<TSender, Arg1, Arg2, Arg3, Arg4>
```

### Value

**Type:** SystemAction<TSender, Arg1, Arg2, Arg3, Arg4>

### See Also

**Reference**
- EventTemplate TSender, Arg1, Arg2, Arg3, Arg4 Class
- W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$\text{W}$
EventTemplate

Wraps the functionality of delegate, event and RaiseXXX into a single class

Inheritance Hierarchy

System\Object WEventTemplate<TSender, Arg1, Arg2, Arg3, Arg4, Arg5>

Namespace: W
Assembly: Tungsten.EventTemplate (in Tungsten.EventTemplate.dll)
Version: 2.0.0

Syntax

```c#
public class EventTemplate<TSender, Arg1, Arg2, Arg3, Arg4, Arg5>
```

Type Parameters

- **TSender**
  - The object raising the event
- **Arg1**
  - The first argument
- **Arg2**
  - The second argument
- **Arg3**
  - The third argument
- **Arg4**
  - The fourth argument
- **Arg5**
  - The fifth argument
The fifth argument

The EventTemplate TSender, Arg1, Arg2, Arg3, Arg4, Arg5 type exposes the following members.

## Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EventTemplate TSender, Arg1, Arg2, Arg3, Arg4, Arg5</td>
<td>Initializes a new instance of the EventTemplate TSender, Arg1, Arg2, Arg3, Arg4, Arg5 class</td>
</tr>
</tbody>
</table>

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object. (Inherited from Object.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current</td>
</tr>
</tbody>
</table>
instance.  
(Inherited from Object.)

**MemberwiseClone**  
Creates a shallow copy of the current Object.  
(Inherited from Object.)

**Raise**  
Raises the template event

**ToString**  
Returns a string that represents the current object.  
(Inherited from Object.)

---

### Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnRaised</td>
<td>The template event</td>
</tr>
</tbody>
</table>

---

### Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| AsTType               | Use Generic syntax for the as operator.  
(Defined by AsExtensions.) |
| InitializeProperties  | Scans the fields and properties of "owner" and sets the member's Owner property to "owner"  
This method should be called in the constructor of any class which has |
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTType(Func&lt;TType&gt;)</td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTType(Func&lt;TType&gt;)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock. (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>

See Also

- Reference
- W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
EventTemplate TSender, Arg1, Arg2, Arg3, Arg4, Arg5

Constructor

Initializes a new instance of the EventTemplate TSender, Arg1, Arg2, Arg3, Arg4, Arg5 class

**Namespace:** W  
**Assembly:** Tungsten.EventTemplate (in Tungsten.EventTemplate.dll)  
**Version:** 2.0.0

## Syntax

```csharp
public EventTemplate()
```

## See Also

- Reference
  - EventTemplate TSender, Arg1, Arg2, Arg3, Arg4, Arg5 Class
  - W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
EventTemplate\textit{TSender, Arg1, Arg2, Arg3, Arg4, Arg5} Methods

The \textit{EventTemplate\textit{TSender, Arg1, Arg2, Arg3, Arg4, Arg5}} generic type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\textbullet\textbullet Equals</td>
<td>Determines whether the specified object is equal to the current object. (Inherited from Object.)</td>
</tr>
<tr>
<td>\textbullet\textbullet Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td>\textbullet\textbullet GetHashCode</td>
<td>Serves as the default hash function. (Inherited from Object.)</td>
</tr>
<tr>
<td>\textbullet\textbullet GetType</td>
<td>Gets the \textit{Type} of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>\textbullet\textbullet MemberwiseClone</td>
<td>Creates a shallow copy of the current \textit{Object}. (Inherited from Object.)</td>
</tr>
</tbody>
</table>

(Inherited from Object.)
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Raise</code></td>
<td>Raises the template event</td>
</tr>
<tr>
<td><code>ToString</code></td>
<td>Returns a string that represents the current object. (Inherited from <code>Object</code>.)</td>
</tr>
</tbody>
</table>

### Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AsTType</code></td>
<td>Use Generic syntax for the as operator. (Defined by <code>AsExtensions</code>.)</td>
</tr>
<tr>
<td><code>InitializeProperties</code></td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by <code>PropertyHostExtensions</code>.)</td>
</tr>
<tr>
<td><code>InLock(Action)</code></td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>InLockTType(FuncTType)</code></td>
<td>Overloaded. Performs the function in a Monitor lock. (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTTType(FuncTTType)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>
See Also

Reference
EventTemplateTSender, Arg1, Arg2, Arg3, Arg4, Arg5 Class
W Namespace

Copyright © 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
</tr>
</tbody>
</table>
EventTemplate TSender, Arg1, Arg2, Arg3, Arg4, Arg5

Raise Method

Raises the template event

Namespace: W
Assembly: Tungsten.EventTemplate (in Tungsten.EventTemplate.dll)
Version: 2.0.0

Syntax

```csharp
public void Raise(
    TSender sender,
    Arg1 arg1,
    Arg2 arg2,
    Arg3 arg3,
    Arg4 arg4,
    Arg5 arg5
)
```

Parameters

sender
Type: TSender
The object which raised this event

arg1
Type: Arg1
The first argument

arg2
Type: Arg2
The second argument
arg3
Type: Arg3
The third argument
arg4
Type: Arg4
The fourth argument
arg5
Type: Arg5
The fifth argument

See Also

Reference
EventTemplateTSender, Arg1, Arg2, Arg3, Arg4, Arg5 Class
W Namespace

Copyright © 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
</tr>
</tbody>
</table>
EventTemplate $TSender$, $Arg1$, $Arg2$, $Arg3$, $Arg4$, $Arg5$ Events

The EventTemplate $TSender$, $Arg1$, $Arg2$, $Arg3$, $Arg4$, $Arg5$ generic type exposes the following members.

### Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>❡ OnRaised</td>
<td>The template event</td>
</tr>
</tbody>
</table>

Top

### See Also

Reference
EventTemplate $TSender$, $Arg1$, $Arg2$, $Arg3$, $Arg4$, $Arg5$ Class
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
EventTemplate TSender, Arg1, Arg2, Arg3, Arg4, Arg5 On Raised Event

The template event

Namespace:  W
Assembly:  Tungsten.EventTemplate (in Tungsten.EventTemplate.dll)
Version: 2.0.0

Syntax

```c#
public event Action<TSender, Arg1, Arg2, Arg3, Arg4, Arg5>
```

Value
Type: SystemAction<TSender, Arg1, Arg2, Arg3, Arg4, Arg5

See Also

Reference
EventTemplate TSender, Arg1, Arg2, Arg3, Arg4, Arg5 Class
W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

W
FromExtensions Class

Extensions which convert objects of one type to another

▶ Inheritance Hierarchy

    System
      Object
        W
          FromExtensions

Namespace:  W
Assembly:  Tungsten.From (in Tungsten.From.dll) Version: 2.0.1

▶ Syntax

```csharp
public static class FromExtensions
```

The `FromExtensions` type exposes the following members.

▶ Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>➡️ FromBase64(Byte)</td>
<td>Converts a Base64 encoded byte array back to a normal byte array</td>
</tr>
<tr>
<td>➡️ FromBase64(String)</td>
<td>Converts a Base64 encoded string back to a normal string</td>
</tr>
<tr>
<td>➡️ FromBase64(Byte, Encoding)</td>
<td>Converts a Base64 encoded byte array back to a normal byte array</td>
</tr>
<tr>
<td>➡️ FromBase64(String, Encoding)</td>
<td>Converts a Base64 encoded string back to a normal string</td>
</tr>
<tr>
<td>➡️ FromCompressed</td>
<td>Decompresses the byte array using</td>
</tr>
</tbody>
</table>
See Also

Reference

W Namespace
Tungsten

$W$
FromExtensions Methods

The FromExtensions type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FromBase64(Byte)</td>
<td>Converts a Base64 encoded byte array back to a normal byte array</td>
</tr>
<tr>
<td>FromBase64(String)</td>
<td>Converts a Base64 encoded string back to a normal string</td>
</tr>
<tr>
<td>FromBase64(Byte, Encoding)</td>
<td>Converts a Base64 encoded byte array back to a normal byte array</td>
</tr>
<tr>
<td>FromBase64(String, Encoding)</td>
<td>Converts a Base64 encoded string back to a normal string</td>
</tr>
<tr>
<td>FromCompressed</td>
<td>Decompresses the byte array using System.IO.Compression.DeflateStream</td>
</tr>
</tbody>
</table>

See Also

Reference
FromExtensions Class
W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
# FromExtensionsFromBase64 Method

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FromBase64(Byte)</td>
<td>Converts a Base64 encoded byte array back to a normal byte array</td>
</tr>
<tr>
<td>FromBase64(String)</td>
<td>Converts a Base64 encoded string back to a normal string</td>
</tr>
<tr>
<td>FromBase64(Byte, Encoding)</td>
<td>Converts a Base64 encoded byte array back to a normal byte array</td>
</tr>
<tr>
<td>FromBase64(String, Encoding)</td>
<td>Converts a Base64 encoded string back to a normal string</td>
</tr>
</tbody>
</table>

See Also

Reference

FromExtensions Class

W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
FromExtensionsFromBase64 Method (Byte)

Converts a Base64 encoded byte array back to a normal byte array

**Namespace:** W

**Assembly:** Tungsten.From (in Tungsten.From.dll) Version: 2.0.1

**Syntax**

```csharp
public static string FromBase64(
    this byte[] this
)
```

**Parameters**

*this*

Type: `SystemByte`

The Base64 encoded byte array to convert

**Return Value**

Type: `String`

A non-encoded string

**Usage Note**

In Visual Basic and C#, you can call this method as an instance method on any object of type `T`. When you use instance method syntax to call this method, omit the first parameter. For more information, see Extension Methods (Visual Basic) or Extension Methods (C# Programming Guide).

**See Also**
Reference

- FromExtensions Class
- FromBase64 Overload
- W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
FromExtensionsFromBase64 Method (String)

Converts a Base64 encoded string back to a normal string

**Namespace:** W
**Assembly:** Tungsten.From (in Tungsten.From.dll) Version: 2.0.1

### Syntax

```csharp
public static string FromBase64(
    this string this
)
```

### Parameters

**this**
- Type: `System.String`
- The Base64 encoded string to convert

### Return Value

Type: `String`
- A non-encoded string

### Usage Note

In Visual Basic and C#, you can call this method as an instance method on any object of type `String`. When you use instance method syntax to call this method, omit the first parameter. For more information, see [Extension Methods (Visual Basic)](https://docs.microsoft.com/en-us/dotnet/visual-basic/extension-methods) or [Extension Methods (C# Programming Guide)](https://docs.microsoft.com/en-us/dotnet/csharp/extension-methods).

### See Also
Reference

FromExtensions Class
FromBase64 Overload
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
FromExtensionsFromBase64 Method (Byte, Encoding)

Converts a Base64 encoded byte array back to a normal byte array

**Namespace:** W

**Assembly:** Tungsten.From (in Tungsten.From.dll) Version: 2.0.1

### Syntax

```csharp
public static string FromBase64(
    this byte[] this,
    Encoding encoding
)
```

### Parameters

**this**
- **Type:** System.Byte
- The Base64 encoded byte array to convert

**encoding**
- **Type:** System.Text.Encoding
- The encoding to use

### Return Value

**Type:** String

A non-encoded string

### Usage Note

In Visual Basic and C#, you can call this method as an instance method on any object of type . When you use instance method syntax to call this method, omit the first parameter. For more information, see Extension Methods (Visual Basic) or Extension
Methods (C# Programming Guide).

See Also

Reference
FromExtensions Class
FromBase64 Overload
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
FromExtensionsFromBase64 Method (String, Encoding)

Converts a Base64 encoded string back to a normal string

**Namespace:** W  
**Assembly:** Tungsten.From (in Tungsten.From.dll) Version: 2.0.1

### Syntax

```csharp
public static string FromBase64(
    this string this,
    Encoding encoding
)
```

### Parameters

**this**
- Type: System.String
  - The Base64 encoded string to convert

**encoding**
- Type: System.Text.Encoding
  - The encoding to use

### Return Value

Type: String  
A non-encoded string

### Usage Note

In Visual Basic and C#, you can call this method as an instance method on any object of type String. When you use instance method syntax to call this method, omit the first parameter. For more information, see Extension Methods (Visual Basic) or Extension
Methods (C# Programming Guide).

See Also

Reference
FromExtensions Class
FromBase64 Overload
W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
FromExtensionsFromCompressed Method

Decompresses the byte array using System.IO.Compression.DeflateStream

**Namespace:**  W  
**Assembly:**  Tungsten.From (in Tungsten.From.dll) Version: 2.0.1

Syntax

```csharp
public static byte[] FromCompressed(
    this byte[] bytes
)
```

Parameters

- `bytes`
  - Type: `SystemByte`
  - The byte array containing compressed data

Return Value

- Type: `Byte`
- A byte array of the decompressed data

Usage Note

In Visual Basic and C#, you can call this method as an instance method on any object of type . When you use instance method syntax to call this method, omit the first parameter. For more information, see Extension Methods (Visual Basic) or Extension Methods (C# Programming Guide).

See Also
Reference

FromExtensions Class
W Namespace

Copyright © 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
</tr>
</tbody>
</table>
IOwnedProperty Interface

Used by PropertyHostMethods.InitializeProperties to find properties on which to set the owner. This interface is not used by self-owned properties.

Namespace:  W
Assembly:  Tungsten.Property (in Tungsten.Property.dll) Version: 2.0.2

Syntax

```csharp
public interface IOwnedProperty
```

The IOwnedProperty type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="SetOwner" /></td>
<td>Sets the property owner to the specified value</td>
</tr>
</tbody>
</table>

See Also

Reference

W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

W
The `IOwnedProperty` type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetOwner</td>
<td>Sets the property owner to the specified value</td>
</tr>
</tbody>
</table>

### See Also

Reference
- `IOwnedProperty Interface`
- `W Namespace`

Copyright © 2018 Jordan Duerksen
Tungsten

W
IOwnedProperty

SetOwner Method

Sets the property owner to the specified value

**Namespace:** W

**Assembly:** Tungsten.Property (in Tungsten.Property.dll) **Version:** 2.0.2

### Syntax

```csharp
void SetOwner(
    Object owner
)
```

### Parameters

*owner*

- Type: **SystemObject**
- The new property owner

### See Also

- Reference
  - IOwnedProperty Interface
  - W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

W
IProperty Interface

The base interface which Property must support

**Namespace:** W
**Assembly:** Tungsten.Property (in Tungsten.Property.dll)  Version: 2.0.2

## Syntax

```c#
public interface IProperty
```

The `IProperty` type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![ ]</td>
<td>IsDirty True if the property's value has changed since initialization or since the last call to MarkAsClean</td>
</tr>
</tbody>
</table>

## See Also

Reference
W Namespace

Copyright © 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
</tr>
</tbody>
</table>
IProperty Properties

The IProperty type exposes the following members.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsDirty</td>
<td>True if the property's value has changed since initialization or since the last call to MarkAsClean</td>
</tr>
</tbody>
</table>

See Also

Reference

IProperty Interface
W Namespace

Copyright © 2018 Jordan Duerksen
IProperty.IsDirty Property

True if the property's value has changed since initialization or since the last call to MarkAsClean

**Namespace:** W

**Assembly:** Tungsten.Property (in Tungsten.Property.dll) Version: 2.0.2

### Syntax

```csharp
bool IsDirty { get; set; }
```

### Property Value

Type: Boolean

### See Also

Reference

IProperty Interface

W Namespace

---

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
IProperty<TValue> Interface

The base interface which Property must support

Namespace:  W
Assembly:  Tungsten.Property (in Tungsten.Property.dll) Version: 2.0.2

Syntax

```csharp
public interface IProperty<TValue> : IProperty
```

Type Parameters

TValue
The type of value for the property

The IProperty<TValue> type exposes the following members.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsDirty</td>
<td>True if the property's value has changed since initialization or since the last call to MarkAsClean (Inherited from IProperty.)</td>
</tr>
<tr>
<td>Value</td>
<td>The value of the property</td>
</tr>
</tbody>
</table>

See Also
Tungsten

$W$
The `IProperty` type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsDirty</td>
<td>True if the property's value has changed since initialization or since the last call to MarkAsClean (Inherited from <code>IProperty</code>.)</td>
</tr>
<tr>
<td>Value</td>
<td>The value of the property</td>
</tr>
</tbody>
</table>

## See Also

Reference
- `IPropertyTValue` Interface
- `W` Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
IProperty<TValue> Value Property

The value of the property

**Namespace:** W  
**Assembly:** Tungsten.Property (in Tungsten.Property.dll)  
**Version:** 2.0.2

**Syntax**

```csharp
TValue Value { get; set; }
```

**Property Value**  
**Type:** TValue

**See Also**

Reference  
IProperty<TValue> Interface  
W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
LockableTValue Class

Extends LockableSlim with ValueChangedDelegate notification

Inheritance Hierarchy

```
System.Object  W.Threading.LockersReaderWriterLockerTValue
   WLockableSlimTValue
   WLockableTValue
   WPropertySlimTValue
```

Namespace: W
Assembly: Tungsten.Lockable (in Tungsten.Lockable.dll) Version: 2.0.1

Syntax

```
public class Lockable<TValue> : LockableSlim<TValue>, IDisposable
```

Type Parameters

```
TValue
   The data type to be used
```

The LockableTValue type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LockableTValue</td>
<td>Constructs a new</td>
</tr>
</tbody>
</table>
Lockable<TValue>

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LockableTValue(ActionObject, TValue, TValue)</td>
<td>Constructs a new Lockable&lt;TValue&gt;</td>
</tr>
<tr>
<td>LockableTValue(TValue)</td>
<td>Constructs a new Lockable&lt;TValue&gt;</td>
</tr>
<tr>
<td>LockableTValue(TValue, ActionObject, TValue, TValue)</td>
<td>Constructs a new Lockable&lt;TValue&gt;</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>Get or Set the value (Inherited from LockableSlimTValue.)</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispose</td>
<td>Disposes the Lockable resources</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether object is equal to the (Inherited from Object)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from <code>Object</code>)</td>
</tr>
<tr>
<td>GetState</td>
<td>Retrieves the internal state of a <code>ReaderWriterLockSlim</code>. (Inherited from <code>ReaderWriterLocker</code>)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <code>Type</code> of the current instance. (Inherited from <code>Object</code>)</td>
</tr>
<tr>
<td>GetValue</td>
<td>Gets the underlying value. (Inherited from <code>LockableSlim</code>)</td>
</tr>
<tr>
<td>InformWaiters</td>
<td>Informs those who are waiting on <code>WaitForChanged</code> that the value has changed</td>
</tr>
<tr>
<td>InLock(LockTypeEnum, ActionTState)</td>
<td>Executes an action from within a <code>ReaderWriterLockSlim</code>. (Inherited from <code>ReaderWriterLocker</code>)</td>
</tr>
<tr>
<td>InLockTValue(LockTypeEnum, FuncTState, TValue)</td>
<td>Executes a function from within a <code>ReaderWriterLockSlim</code>. (Inherited from <code>ReaderWriterLocker</code>)</td>
</tr>
<tr>
<td>InLockAsync(LockTypeEnum, ActionTState)</td>
<td>Executes an action from within a <code>ReaderWriterLockSlim</code>. (Inherited from <code>ReaderWriterLocker</code>)</td>
</tr>
<tr>
<td>InLockAsyncTValue(LockTypeEnum, FuncTState, TValue)</td>
<td>Executes a function from within a <code>ReaderWriterLockSlim</code>. (Inherited from <code>ReaderWriterLocker</code>)</td>
</tr>
</tbody>
</table>
### MemberwiseClone

Creates a shallow copy of the current `Object`. 
(Inherited from `Object`)

### OnValueChanged

Calls `RaiseValueChanged` to raise the `ValueChanged` event

### RaiseValueChanged

Raises the `ValueChanged` event

### setState

Sets the internal state from within a `ReaderWriterLockSlim`.  
(Inherited from `ReaderWriterLockSlim`)

### SetValue

Sets the value and raises the `ValueChanged` event. 
(Overrides `LockableSlim TValue SetValue(TValue)`)

### ToString

Returns a string that represents the current object.  
(Inherited from `Object`)

### WaitForValueChanged

Allows the caller to block until the value changes

---

#### Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ValueChanged</strong></td>
<td>Raised when the value has changed</td>
</tr>
</tbody>
</table>

---

Top
### Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locker</td>
<td>The ReaderWriterLocker used to access the State (Inherited from ReaderWriterLockerTState.)</td>
</tr>
<tr>
<td>State</td>
<td>The internal state (Inherited from ReaderWriterLockerTState.)</td>
</tr>
</tbody>
</table>

### Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>Monitor lock (Defined by MonitorExtensions.)</td>
<td></td>
</tr>
<tr>
<td><strong>InLockTTType(FuncTTType)</strong></td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>InLockAsync(Action)</strong></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>InLockAsyncTTType(FuncTTType)</strong></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>IsDirty</strong></td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td><strong>Lock</strong></td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>MarkAsClean</strong></td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>
Unlock

Performs a Monitor unlock
(Defined by MonitorExtensions.)

See Also

Reference

W Namespace

Copyright @ 2018 Jordan Duerksen
| Tungsten | $W$ |
## Lockable*TValue* Constructor

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lockable<em>TValue</em></td>
<td>Constructs a new Lockable&lt;TValue&gt;</td>
</tr>
<tr>
<td>Lockable<em>TValue</em>(ActionObject, TValue, TValue)</td>
<td>Constructs a new Lockable&lt;TValue&gt;</td>
</tr>
<tr>
<td>Lockable<em>TValue</em>(TValue)</td>
<td>Constructs a new Lockable&lt;TValue&gt;</td>
</tr>
<tr>
<td>Lockable<em>TValue</em>(TValue, ActionObject, TValue, TValue)</td>
<td>Constructs a new Lockable&lt;TValue&gt;</td>
</tr>
</tbody>
</table>

### See Also

- Reference
  - Lockable*TValue* Class
  - W Namespace
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
</tr>
</tbody>
</table>
Lockable\textit{TValue} Constructor

Constructs a new Lockable\textless TValue\textgreater

\textbf{Namespace:} W \\
\textbf{Assembly:} Tungsten.Lockable (in Tungsten.Lockable.dll) Version: 2.0.1

\section*{Syntax}

\begin{verbatim}
public Lockable()
\end{verbatim}

\section*{See Also}

Reference
- Lockable\textit{TValue Class}
- Lockable\textit{TValue Overload}
- W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
Lockable<TValue> Constructor
(ActionObject, TValue, TValue)

Constructs a new Lockable<TValue>

Namespace: W
Assembly: Tungsten.Lockable (in Tungsten.Lockable.dll) Version: 2.0.1

Syntax

```c#
public Lockable(
    Action<Object, TValue, TValue> onValueChanged
)
```

Parameters

onValueChanged
Type: System.Action<Object, TValue, TValue>
The Action to call when the value has changed

See Also

Reference
LockableTValue Class
LockableTValue Overload
W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
Lockable<TValue> Constructor (TValue)

Constructs a new Lockable<TValue>

**Namespace:** W  
**Assembly:** Tungsten.Lockable (in Tungsten.Lockable.dll)  
**Version:** 2.0.1

### Syntax

```csharp
public Lockable(
    TValue initialValue
)
```

### Parameters

- **initialValue**
  - **Type:** TValue  
  - The initial value

### See Also

- **Reference**
  - LockableTValue Class  
  - LockableTValue Overload  
  - W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
Lockable<TValue> Constructor
(TValue, Action<Object, TValue, TValue>)

Constructs a new Lockable<TValue>

Namespace: W
Assembly: Tungsten.Lockable (in Tungsten.Lockable.dll) Version: 2.0.1

Syntax

```c#
public Lockable(
    TValue initialValue,
    Action<Object, TValue, TValue> onValueChanged
)
```

Parameters

initialValue
Type: TValue

[Missing <param name="initialValue"/> documentation for "M:W.Lockable`1.#ctor(`0,System.Action{System.Object,`0,`0})"]

onValueChanged
Type: System.Action<Object, TValue, TValue>

[Missing <param name="onValueChanged"/> documentation for "M:W.Lockable`1.#ctor(`0,System.Action{System.Object,`0,`0})"]

See Also
Reference
LockableTValue Class
LockableTValue Overload
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
Lockable\text{TValue} Properties

The Lockable\text{TValue} generic type exposes the following members.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| Value | Get or Set the value  
(Inherited from LockableSlim\text{TValue}.) |

See Also

Reference
Lockable\text{TValue} Class
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
## LockableTValue Methods

The `LockableTValue` generic type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispose</td>
<td>Disposes the Lockable and releases resources</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether object is equal to the current object.</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td>GetState</td>
<td>Retrieves the internal state from within a <code>ReaderWriterLockSlim</code>.</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <code>Type</code> of the current instance.</td>
</tr>
<tr>
<td>GetValue</td>
<td>Gets the underlying value.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>InformWaiters</code></td>
<td>Informs those who are waiting on <code>WaitForChanged</code> that the value has changed</td>
</tr>
<tr>
<td><code>InLock(LockTypeEnum, Action&lt;TState&gt;)</code></td>
<td>Executes an action from within a <code>ReaderWriterLockSlim</code> (Inherited from <code>ReaderWriterLocker</code>)</td>
</tr>
<tr>
<td><code>InLockTValue(LockTypeEnum, Func&lt;TState, TValue&gt;)</code></td>
<td>Executes a function from within a <code>ReaderWriterLockSlim</code> (Inherited from <code>ReaderWriterLocker</code>)</td>
</tr>
<tr>
<td><code>InLockAsync(LockTypeEnum, Action&lt;TState&gt;)</code></td>
<td>Executes an action from within a <code>ReaderWriterLockSlim</code> (Inherited from <code>ReaderWriterLocker</code>)</td>
</tr>
<tr>
<td><code>InLockAsyncTValue(LockTypeEnum, Func&lt;TState, TValue&gt;)</code></td>
<td>Executes a function from within a <code>ReaderWriterLockSlim</code> (Inherited from <code>ReaderWriterLocker</code>)</td>
</tr>
<tr>
<td><code>MemberwiseClone</code></td>
<td>Creates a shallow copy of the current <code>Object</code> (Inherited from <code>Object</code>)</td>
</tr>
<tr>
<td><code>OnValueChanged</code></td>
<td>Calls <code>RaiseValueChanged</code> to raise the <code>ValueChanged</code> event</td>
</tr>
<tr>
<td><code>RaiseValueChanged</code></td>
<td>Raises the <code>ValueChanged</code> event</td>
</tr>
<tr>
<td><code>SetState</code></td>
<td>Sets the internal state of a <code>ReaderWriterLockSlim</code> (Inherited from <code>ReaderWriterLocker</code>)</td>
</tr>
</tbody>
</table>
SetValue
Sets the value and raises the ValueChanged event.
(Overrides LockableSlimTValue

ToString
Returns a string that represents the current object.
(Inherited from Object

WaitForValueChanged
Allows the caller to block until Value changes

Top

Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>InLockTType(Func&lt;TType&gt;)</td>
<td>Overloaded. Performs the function in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTType(Func&lt;TType&gt;)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false. (Defined by PropertyHostExtensions.)</td>
</tr>
</tbody>
</table>
Unlock

Performs a Monitor unlock
(Defined by MonitorExtensions.)

Top

See Also

Reference
LockableTValue Class
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
Lockable\textit{ TValue} Dispose Method

Disposes the Lockable and releases resources

\textbf{Namespace: } \textit{W}
\textbf{Assembly: } Tungsten.Lockable (in Tungsten.Lockable.dll) Version: 2.0.1

\textbf{Syntax}

\begin{verbatim}
public void Dispose()
\end{verbatim}

Implements
\texttt{IDisposable.Dispose}

\textbf{See Also}

Reference
\texttt{Lockable\textit{ TValue} Class}
\texttt{W Namespace}

Copyright @ 2018 Jordan Duerksen
Tungsten

$\text{W}$
Lockable<TValue> InformWaiters Method

Informs those who are waiting on WaitForChanged that the value has changed

Namespace: W
Assembly: Tungsten.Lockable (in Tungsten.Lockable.dll) Version: 2.0.1

Syntax

C#

```csharp
protected virtual void InformWaiters()
```

See Also

Reference
Lockable<TValue> Class
W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$\text{W}$
Lockable TValue OnValueChanged Method

Calls RaiseValueChanged to raise the ValueChanged event

**Namespace:** W
**Assembly:** Tungsten.Lockable (in Tungsten.Lockable.dll) Version: 2.0.1

### Syntax

```csharp
protected virtual void OnValueChanged(
    Object sender,
    TValue oldValue,
    TValue newValue
)
```

### Parameters

- **sender**
  - Type: System.Object
  - The object initiating the change
- **oldValue**
  - Type: TValue
  - The previous value
- **newValue**
  - Type: TValue
  - The current value

### See Also

Reference
LockableTValue Class
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
## Lockable\(T\)ValueRaiseValueChanged Method

Raises the ValueChanged event

**Namespace:** \(W\)  
**Assembly:** Tungsten.Lockable (in Tungsten.Lockable.dll) Version: 2.0.1

### Syntax

```csharp
protected void RaiseValueChanged(
    Object sender,
    TValue oldValue,
    TValue newValue
)
```

### Parameters

- **sender**  
  Type: `SystemObject`  
  [Missing `<param name="sender"/>` documentation for "M:W.Lockable`1.RaiseValueChanged(System.Object,`0,`0)" ]

- **oldValue**  
  Type: `TValue`  
  The previous value

- **newValue**  
  Type: `TValue`  
  The current value

### See Also
Tungsten

W
LockableSetValue Method

Sets the value and raises the ValueChanged event

Namespace: W
Assembly: Tungsten.Lockable (in Tungsten.Lockable.dll) Version: 2.0.1

Syntax

```csharp
protected override void SetValue(
    TValue newValue
)
```

Parameters

newValue
Type: TValue
The new value

See Also

Reference
LockableTValue Class
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
Lockable\textit{TValue} \textbf{WaitForValueChanged} Method

Allows the caller to block until Value changes

\textbf{Namespace:} \textit{W}  
\textbf{Assembly:} Tungsten.Lockable (in Tungsten.Lockable.dll)  
\textbf{Version:} 2.0.1

\textbf{Syntax}

```
public bool WaitForValueChanged(int msTimeout = -1)
```

\textbf{Parameters}

\textit{msTimeout (Optional)}  
Type: \textbf{SystemInt32}  
The number of milliseconds to wait for the value to change

\textbf{Return Value}

Type: \textbf{Boolean}  
True if the value changed within the specified timeout period, otherwise False

\textbf{See Also}

Reference  
Lockable\textit{TValue} Class  
\textit{W} Namespace
Tungsten

W
The LockableTValue generic type exposes the following members.

### Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ValueChanged</td>
<td>Raised when the value has changed</td>
</tr>
</tbody>
</table>

See Also

Reference
LockableTValue Class
W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

W
Lockable TValueChanged ValueChanged Event

Raised when the value has changed

Namespace: W
Assembly: Tungsten.Lockable (in Tungsten.Lockable.dll) Version: 2.0.1

Syntax

```
public event ValueChangedDelegate<TValue> ValueChanged;
```

Value
Type: WValueChangedDelegate TValue

See Also

Reference
LockableTValue Class
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
LockableTValue Fields

The LockableTValue generic type exposes the following members.

Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locker</td>
<td>The ReaderWriterLocker used to access the State (Inherited from ReaderWriterLockerTState.)</td>
</tr>
<tr>
<td>State</td>
<td>The internal state (Inherited from ReaderWriterLockerTState.)</td>
</tr>
</tbody>
</table>

See Also

Reference
LockableTValue Class
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$^7$W
LockableSlim<TValue> Class

Uses ReaderWriterLock to provide thread-safe access to an underlying value

Inheritance Hierarchy

SystemObject W.Threading.LockersReaderWriterLock<TValue> WLockableSlimTValue WLockableTValue

Namespace: W
Assembly: Tungsten.Lockable (in Tungsten.Lockable.dll) Version: 2.0.1

Syntax

```csharp
public class LockableSlim<TValue> : ReaderWriterLock
```

Type Parameters

*TValue*
The Type of value

The LockableSlim<TValue> type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LockableSlimTValue</td>
<td>Constructs a new LockableSlim with a default initial</td>
</tr>
</tbody>
</table>
### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>Get or Set the value</td>
</tr>
</tbody>
</table>

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispose</td>
<td>Disposes the ReaderWriterLocker and releases resources (Inherited from ReaderWriterLocker)</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether specified object is equal to the current object. (Inherited from Object)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from <code>Object</code>)</td>
</tr>
<tr>
<td>GetState</td>
<td>Retrieves the internal state from within a <code>ReaderWriterLockSlim</code>. (Inherited from <code>ReaderWriterLocker</code>)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <code>Type</code> of the instance. (Inherited from <code>Object</code>)</td>
</tr>
<tr>
<td>GetValue</td>
<td>Gets the underlying value</td>
</tr>
<tr>
<td>InLock(<code>LockType(Enum, Action&lt;TState&gt;)</code>)</td>
<td>Executes an action from within a <code>ReaderWriterLockSlim</code>. (Inherited from <code>ReaderWriterLocker</code>)</td>
</tr>
<tr>
<td>InLockTValue(<code>LockType(Enum, Func&lt;TState, TValue&gt;)</code>)</td>
<td>Executes a function from within a <code>ReaderWriterLockSlim</code>. (Inherited from <code>ReaderWriterLocker</code>)</td>
</tr>
<tr>
<td>InLockAsync(<code>LockType(Enum, Action&lt;TState&gt;)</code>)</td>
<td>Executes an action from within a <code>ReaderWriterLockSlim</code>. (Inherited from <code>ReaderWriterLockSlim</code>)</td>
</tr>
<tr>
<td>InLockAsyncTValue(<code>LockType(Enum, Func&lt;TState, TValue&gt;)</code>)</td>
<td>Executes a function from within a <code>ReaderWriterLockSlim</code>. (Inherited from <code>ReaderWriterLockSlim</code>)</td>
</tr>
</tbody>
</table>
MemberwiseClone

Creates a shallow copy of the current Object.

(Inherited from ReaderWriterLocker)

SetState

Sets the internal state within a ReaderWriterLockSlim instance.

(Inherited from ReaderWriterLocker)

SetValue

Sets the underlying value.

ToString

Returns a string that represents the current object.

(Inherited from Object)

### Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locker</td>
<td>The ReaderWriterLocker object used to access the State.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from ReaderWriterLockerTState.)</td>
</tr>
<tr>
<td>State</td>
<td>The internal state.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from ReaderWriterLockerTState.)</td>
</tr>
</tbody>
</table>
## Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;! This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTType(FuncTType)</td>
<td>Overloaded. Performs the function in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>
InLockAsync(Func<TType>) Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)

IsDirty Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)

Lock Performs a Monitor lock (Defined by MonitorExtensions.)

MarkAsClean Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)

Unlock Performs a Monitor unlock (Defined by MonitorExtensions.)

Remarks

Can be overridden to provide additional functionality

See Also
# LockableSlim<TValue> Constructor

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LockableSlimTValue</td>
<td>Constructs a new LockableSlim with a default initial value</td>
</tr>
<tr>
<td>LockableSlimTValue(TValue)</td>
<td>Constructs a new LockableSlim assigning an initial value</td>
</tr>
</tbody>
</table>

## See Also

Reference

LockableSlimTValue Class

W Namespace

Copyright © 2018 Jordan Duerksen
LockableSlim<TValue> Constructor

Constructs a new LockableSlim with a default initial value

**Namespace:** W

**Assembly:** Tungsten.Lockable (in Tungsten.Lockable.dll) Version: 2.0.1

**Syntax**

```csharp
public LockableSlim()
```

**See Also**

Reference

- LockableSlimTValue Class
- LockableSlimTValue Overload
- W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
LockableSlim\textit{TValue} Constructor (\textit{TValue})

Constructs a new LockableSlim assigning an initial value

\textbf{Namespace: } W  
\textbf{Assembly: } Tungsten.Lockable (in Tungsten.Lockable.dll) Version: 2.0.1

\section*{Syntax}

\begin{verbatim}
public LockableSlim(
    TValue initialValue
)
\end{verbatim}

\section*{Parameters}

\textit{initialValue}

Type: \textit{TValue}

The initial value to assign

\section*{See Also}

Reference

LockableSlim\textit{TValue} Class  
LockableSlim\textit{TValue} Overload  
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\(W\)
LockableSlim\textit{TValue} Properties

The \textit{LockableSlimTValue} generic type exposes the following members.

\section*{Properties}

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>Get or Set the value</td>
</tr>
</tbody>
</table>

\section*{See Also}

Reference

LockableSlimTValue Class
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
LockableSlim TValue Value Property

Get or Set the value

**Namespace:** W
**Assembly:** Tungsten.Lockable (in Tungsten.Lockable.dll) **Version:** 2.0.1

Syntax

```csharp
public TValue Value { get; set; }
```

Return Value
Type: `TValue`
The current value

See Also

Reference
LockableSlim TValue Class
W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
# LockableSlim\textit{TValue} Methods

The \textit{LockableSlim\textit{TValue}} generic type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispose</td>
<td>Disposes the ReaderWriterLocker and releases resources</td>
</tr>
<tr>
<td></td>
<td>(Inherited from ReaderWriterLockSlim)</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from \textit{Object})</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try free resources and perform other cleanup operations</td>
</tr>
<tr>
<td></td>
<td>before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from \textit{Object})</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from \textit{Object})</td>
</tr>
<tr>
<td>GetState</td>
<td>Retrieves the internal state from within a ReaderWriterLockSlim.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from ReaderWriterLockSlim)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Get Type</strong></td>
<td>Gets the <code>Type</code> of the current instance. (Inherited from <code>Object</code>)</td>
</tr>
<tr>
<td><strong>GetValue</strong></td>
<td>Gets the underlying value.</td>
</tr>
</tbody>
</table>
| **InLock(LockTypeEnum, Action<TState>)**    | Executes an action from within a `ReaderWriterLockSlim`. (Inherited from `ReaderWriterLocker`)
| **InLockTValue(LockTypeEnum, Func<TState, TValue>)** | Executes a function from within a `ReaderWriterLockSlim`. (Inherited from `ReaderWriterLocker`)
| **InLockAsync(LockTypeEnum, Action<TState>)** | Executes an action from within a `ReaderWriterLockSlim`. (Inherited from `ReaderWriterLocker`)
| **InLockAsyncTValue(LockTypeEnum, Func<TState, TValue>)** | Executes a function from within a `ReaderWriterLockSlim`. (Inherited from `ReaderWriterLocker`)
| **MemberwiseClone**                         | Creates a shallow copy of the current `Object`. (Inherited from `Object`)   |
| **Set State**                               | Sets the internal state from within a `ReaderWriterLockSlim`.               |
### Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![asoperator] AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>![properties] InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>![lock] InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>Method</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td></td>
</tr>
<tr>
<td><strong>InLock&lt;TType&gt;(Func&lt;TType&gt;)</strong></td>
<td></td>
</tr>
<tr>
<td>Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
<td></td>
</tr>
<tr>
<td><strong>InLockAsync(Action)</strong></td>
<td></td>
</tr>
<tr>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
<td></td>
</tr>
<tr>
<td><strong>InLockAsync&lt;TType&gt;(Func&lt;TType&gt;)</strong></td>
<td></td>
</tr>
<tr>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
<td></td>
</tr>
<tr>
<td><strong>IsDirty</strong></td>
<td></td>
</tr>
<tr>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
<td></td>
</tr>
<tr>
<td><strong>Lock</strong></td>
<td></td>
</tr>
<tr>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
<td></td>
</tr>
<tr>
<td><strong>MarkAsClean</strong></td>
<td></td>
</tr>
<tr>
<td>Scans each field and property of type IProperty and sets it's IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
<td></td>
</tr>
<tr>
<td><strong>Unlock</strong></td>
<td></td>
</tr>
<tr>
<td>Performs a Monitor unlock</td>
<td></td>
</tr>
</tbody>
</table>
See Also

Reference

LockableSlimTValue Class
W Namespace
Tungsten

$W$
LockableSlim<TValue> GetValue Method

Gets the underlying value

**Namespace:** W
**Assembly:** Tungsten.Lockable (in Tungsten.Lockable.dll) Version: 2.0.1

### Syntax

```csharp
protected virtual TValue GetValue()
```

**Return Value**

Type: `TValue`


### See Also

**Reference**

LockableSlim<TValue> Class
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
LockableSlim<TValue> SetValue Method

Sets the underlying value

**Namespace:** W

**Assembly:** Tungsten.Lockable (in Tungsten.Lockable.dll)  Version: 2.0.1

### Syntax

```c#
protected virtual void SetValue(
    TValue value
)
```

### Parameters

**value**
- Type: `TValue`
- The new value

### See Also

Reference
- LockableSlimTValue Class
- W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\[ W \]
LockableSlimTValue Fields

The LockableSlimTValue generic type exposes the following members.

Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locker</td>
<td>The ReaderWriterLocker used to access the State</td>
</tr>
<tr>
<td></td>
<td>(Inherited from ReaderWriterLockerTState.)</td>
</tr>
<tr>
<td>State</td>
<td>The internal state</td>
</tr>
<tr>
<td></td>
<td>(Inherited from ReaderWriterLockerTState.)</td>
</tr>
</tbody>
</table>

See Also

Reference
LockableSlimTValue Class
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
# Property\(TValue\) Class

A Property with no owner (self-owned)

## Inheritance Hierarchy

- **SystemObject**
- **W.Threading.LockersReaderWriterLocker\(TValue\)**
- **WLockableSlim\(TValue\)**
- **WLockable\(TValue\)**
- **WPropertySlim\(TValue\)**
- **WPropertyBase(Property\(TValue\), \(TValue\)**
- **WProperty\(TValue\)**

**Namespace:** W

**Assembly:** Tungsten.Property (in Tungsten.Property.dll) **Version:** 2.0.2

## Syntax

```csharp
public class Property<
\(TValue\)> : PropertyBase<
\(Property\(TValue\)>, 
\(TValue\)>;
```

**Type Parameters**

- **\(TValue\)**
  
  The type of the property value

The Property\(TValue\) type exposes the following members.

## Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property(TValue)</td>
<td>Initializes a new</td>
</tr>
</tbody>
</table>
instance of the `PropertyTValue` class

PropertyTValue( TObject, TValue, TValue) Initializes a new instance of the `PropertyTValue` class

PropertyTValue(TValue) Initializes a new instance of the `PropertyTValue` class

PropertyTValue(TValue, TObject, TValue, TValue) Initializes a new instance of the `PropertyTValue` class

---

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DefaultValue</td>
<td>Allows the programmer to assign a default value which can be reset via the ResetToDefaultValue method. This value does not have to be the initial value. (Inherited from <code>PropertyBaseTOwner, TValue</code>.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>True if Value has changed since initialization or since the last call to MarkAsClean (Inherited from <code>PropertyBaseTOwner, TValue</code>.)</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>Owner</td>
<td>The property owner (Inherited from PropertyBaseTOwner, TValue.)</td>
</tr>
<tr>
<td>Value</td>
<td>Get or Set the value (Inherited from LockableSlimTValue.)</td>
</tr>
</tbody>
</table>

## Top

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispose</td>
<td>Disposes the Lockable and releases resources (Inherited from LockableTValue.)</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether specified object is equal to the current object. (Inherited from Object)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default function. (Inherited from Object)</td>
</tr>
<tr>
<td>GetState</td>
<td>Retrieves the internal state from within a ReaderWriterLockSlim</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>Get Type</strong></td>
<td>Gets the Type of the instance. (Inherited from ReaderWriterLocker)</td>
</tr>
<tr>
<td><strong>Get Value</strong></td>
<td>Gets the underlying value. (Inherited from LockableSlimTValue)</td>
</tr>
<tr>
<td><strong>Inform Waiters</strong></td>
<td>Informs those who are waiting on WaitForChange that the value has changed. (Inherited from LockableTValue)</td>
</tr>
<tr>
<td><strong>InLock</strong> (LockTypeEnum, ActionTState)</td>
<td>Executes an action from within a ReaderWriterLockSlim. (Inherited from ReaderWriterLocker)</td>
</tr>
<tr>
<td><strong>InLockTValue</strong> (LockTypeEnum, FuncTState, TValue)</td>
<td>Executes a function from within a ReaderWriterLockSlim. (Inherited from ReaderWriterLocker)</td>
</tr>
<tr>
<td><strong>InLockAsync</strong> (LockTypeEnum, ActionTState)</td>
<td>Executes an action from within a ReaderWriterLockSlim. (Inherited from ReaderWriterLocker)</td>
</tr>
<tr>
<td><strong>InLockAsyncTValue</strong> (LockTypeEnum, FuncTState, TValue)</td>
<td>Executes a function from within a ReaderWriterLockSlim. (Inherited from ReaderWriterLocker)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LoadValue</td>
<td>Sets Value without raising notification events.</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object.</td>
</tr>
<tr>
<td>OnPropertyChanged</td>
<td>Calls RaisePropertyChanged to raise the PropertyChanged event.</td>
</tr>
<tr>
<td>OnPropertyChanging</td>
<td>Calls RaisePropertyChanged to raise the PropertyChanging event.</td>
</tr>
<tr>
<td>OnValueChanged</td>
<td>Calls RaisePropertyChanged to raise the ValueChanging event.</td>
</tr>
<tr>
<td>RaiseOnPropertyChanged</td>
<td>Raises the PropertyChanged event.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>RaiseOnPropertyChanging</td>
<td>Raises the PropertyChanging event (Inherited from PropertySlimTValue.)</td>
</tr>
<tr>
<td>RaiseValueChanged</td>
<td>Raises the ValueChanged event (Inherited from LockableTValue.)</td>
</tr>
<tr>
<td>ResetToDefaultValue</td>
<td>Resets the Value to the default value provided by DefaultValue (Inherited from PropertyBaseTOwner.TValue.)</td>
</tr>
<tr>
<td>SetState</td>
<td>Sets the internal state within a ReaderWriterLockSlim (Inherited from ReaderWriterLocker.)</td>
</tr>
<tr>
<td>SetValue</td>
<td>Calls OnPropertyChanged on assignment (Inherited from PropertySlimTValue.)</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object. (Inherited from Object.)</td>
</tr>
<tr>
<td>WaitForValueChanged</td>
<td>Allows the caller to block until Value changes (Inherited from LockableTValue.)</td>
</tr>
</tbody>
</table>
### Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PropertyChanged</td>
<td>(Inherited from PropertySlimTValue.)</td>
</tr>
<tr>
<td>PropertyChanging</td>
<td>(Inherited from PropertySlimTValue.)</td>
</tr>
<tr>
<td>ValueChanged</td>
<td>Raised when the value has changed (Inherited from LockableTValue.)</td>
</tr>
</tbody>
</table>

### Operators

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(TValue to PropertyTValue)</td>
<td>Implicit conversion from TValue to Property&lt;TValue&gt;</td>
</tr>
<tr>
<td>(PropertyTValue to TValue)</td>
<td>Implicit conversion from Property&lt;TValue&gt; to TValue</td>
</tr>
</tbody>
</table>

### Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locker</td>
<td>The ReaderWriterLocker used to access the State (Inherited from LockableTValue.)</td>
</tr>
</tbody>
</table>
State The internal state
(Inherited from ReaderWriterLockerTState.)

Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>As&lt;TType&gt;</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>InLock&lt;Action&gt;()</td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLock&lt;TType(Func&lt;TType&gt;)</td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by</td>
</tr>
<tr>
<td>Method Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTType(FuncTType)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>
See Also

Reference
  W Namespace
Tungsten

$W$
## PropertyTValue Constructor

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PropertyTValue</td>
<td>Initializes a new instance of the PropertyTValue class</td>
</tr>
<tr>
<td>PropertyTValue(ActObj, TVal, TVal)</td>
<td>Initializes a new instance of the PropertyTValue class</td>
</tr>
<tr>
<td>PropertyTValue(TVal)</td>
<td>Initializes a new instance of the PropertyTValue class</td>
</tr>
<tr>
<td>PropertyTValue(TVal, ActObj, TVal, TVal)</td>
<td>Initializes a new instance of the PropertyTValue class</td>
</tr>
</tbody>
</table>

### See Also

- Reference
  PropertyTValue Class
  W Namespace

---

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
Property TValue Constructor

Initializes a new instance of the PropertyTValue class

**Namespace:** W  
**Assembly:** Tungsten.Property (in Tungsten.Property.dll)  
**Version:** 2.0.2

### Syntax

```
public Property()
```

### See Also

Reference
- PropertyTValue Class
- PropertyTValue Overload
- W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

\( W \)
**Property**`TValue` Constructor

(_ACTIONOBJECT,`TValue`,`TValue`)  

Initializes a new instance of the `PropertyTValue` class

**Namespace:**  W  
**Assembly:**  Tungsten.Property (in Tungsten.Property.dll)  
**Version:**  2.0.2

### Syntax

```csharp
public Property(
    Action<OBJECT, `TValue`, `TValue>` onValueChanged
)
```

### Parameters

* **onValueChanged**  
  
  Type:  `SystemActionObject, TValue, TValue`  

  [Missing <param name="onValueChanged"/> documentation for  
  "M:W.Property`1.#ctor(System.Action(System.Object,`0,`0))"]

### See Also

**Reference**  

PropertyTValue Class  
PropertyTValue Overload  
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
Property\(T_{Value}\) Constructor
(\(T_{Value}\))

Initializes a new instance of the \(PropertyT_{Value}\) class

**Namespace:** \(W\)
**Assembly:** Tungsten.Property (in Tungsten.Property.dll) **Version:** 2.0.2

### Syntax

```
public Property(
    TValue defaultValue
)
```

**Parameters**

**defaultValue**

Type: \(T_{Value}\)

[Missing <param name="defaultValue"/> documentation for "M:W.Property`1.#ctor(0)""]

### See Also

**Reference**

- Property\(T_{Value}\) Class
- Property\(T_{Value}\) Overload
- \(W\) Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
Property `TValue` Constructor

(*TValue*, *Action* `Object`, *TValue*, *TValue*)

Initializes a new instance of the `Property` class.

**Namespace:** W

**Assembly:** Tungsten.Property (in Tungsten.Property.dll) Version: 2.0.2

### Syntax

```csharp
public Property(
    TValue defaultValue,
    Action<Object, TValue, TValue> onValueChanged
)
```

### Parameters

- `defaultValue`
  
  Type: `TValue`
  
  [Missing <param name="defaultValue"/> documentation for "M:W.Property`1.#ctor(`0,System.Action{System.Object,`0,`0})"]

- `onValueChanged`
  
  Type: `SystemActionObject, TValue, TValue`
  
  [Missing <param name="onValueChanged"/> documentation for "M:W.Property`1.#ctor(`0,System.Action{System.Object,`0,`0})"]

### See Also
Reference
PropertyTValue Class
PropertyTValue Overload
W Namespace

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
<th>W</th>
</tr>
</thead>
</table>

The `PropertyTValue` generic type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DefaultValue</strong></td>
<td>Allows the programmer to assign a default value which can be reset via the <code>ResetToDefaultValue</code> method. This value does not have to be the initial value.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>PropertyBaseTOwner, TValue</code>.)</td>
</tr>
<tr>
<td><strong>IsDirty</strong></td>
<td>True if <code>Value</code> has changed since initialization or since the last call to <code>MarkAsClean</code></td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>PropertyBaseTOwner, TValue</code>.)</td>
</tr>
<tr>
<td><strong>Owner</strong></td>
<td>The property owner</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>PropertyBaseTOwner, TValue</code>.)</td>
</tr>
<tr>
<td><strong>Value</strong></td>
<td>Get or Set the value</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>LockableSlimTValue</code>.)</td>
</tr>
</tbody>
</table>
Tungsten

$W$
The `PropertyTValue` generic type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dispose</strong></td>
<td>Disposes the Lockable and releases resources (Inherited from <code>LockableTValue</code>.)</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether specified object is equal to the current object. (Inherited from <code>Object</code>)</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <code>Object</code>)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as the default hash function. (Inherited from <code>Object</code>)</td>
</tr>
<tr>
<td><strong>GetState</strong></td>
<td>Retrieves the internal state from within a <code>ReaderWriterLockSlim</code>. (Inherited from <code>ReaderWriterLockState</code>)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <code>Type</code> of the instance. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GetValue</td>
<td>Gets the underlying value. (Inherited from <code>LockableSlimTValue</code>.)</td>
</tr>
<tr>
<td>InformWaiters</td>
<td>Informs those who are waiting on <code>WaitForChange</code> that the value has changed. (Inherited from <code>LockableTValue</code>.)</td>
</tr>
<tr>
<td><code>InLock</code>(<code>LockTypeEnum</code>, <code>Action&lt;TState&gt;</code>)</td>
<td>Executes an action from within a <code>ReaderWriterLockSlim</code>. (Inherited from <code>ReaderWriterLocker</code>.)</td>
</tr>
<tr>
<td><code>InLockTValue</code>(<code>LockTypeEnum</code>, <code>Func&lt;TState, TValue&gt;</code>)</td>
<td>Executes a function from within a <code>ReaderWriterLockSlim</code>. (Inherited from <code>ReaderWriterLocker</code>.)</td>
</tr>
<tr>
<td><code>InLockAsync</code>(<code>LockTypeEnum</code>, <code>Action&lt;TState&gt;</code>)</td>
<td>Executes an action from within a <code>ReaderWriterLockSlim</code>. (Inherited from <code>ReaderWriterLocker</code>.)</td>
</tr>
<tr>
<td><code>InLockAsyncTValue</code>(<code>LockTypeEnum</code>, <code>Func&lt;TState, TValue&gt;</code>)</td>
<td>Executes a function from within a <code>ReaderWriterLockSlim</code>. (Inherited from <code>ReaderWriterLocker</code>.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>LoadValue</code></td>
<td>Sets Value without raising notification events (Inherited from <code>PropertyBase&lt;OWNER, TValue&gt;</code>)</td>
</tr>
<tr>
<td><code>MemberwiseClone</code></td>
<td>Creates a shallow copy of the current <code>Object</code>. (Inherited from <code>Object</code>)</td>
</tr>
<tr>
<td><code>OnPropertyChanged</code></td>
<td>Calls <code>RaisePropertyChanged</code> to raise the PropertyChanged event</td>
</tr>
<tr>
<td><code>OnPropertyChanging</code></td>
<td>Calls <code>RaisePropertyChanging</code> to raise the PropertyChanging event</td>
</tr>
<tr>
<td><code>OnValueChanged</code></td>
<td>Calls <code>RaiseValueChanged</code> to raise the ValueChanged event</td>
</tr>
<tr>
<td><code>RaiseOnPropertyChanged</code></td>
<td>Raises the PropertyChanged event</td>
</tr>
<tr>
<td><code>RaiseOnPropertyChanging</code></td>
<td>Raises the PropertyChanging event</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>RaiseValueChanged</td>
<td>Raises the ValueChanged event (Inherited from LockableTValue.)</td>
</tr>
<tr>
<td>ResetToDefaultValue</td>
<td>Resets the Value to the value provided by DefaultValue (Inherited from PropertyBaseToOwner, TValue.)</td>
</tr>
<tr>
<td>SetState</td>
<td>Sets the internal state within a ReaderWriterLockSlim (Inherited from ReaderWriterLocker.)</td>
</tr>
<tr>
<td>SetValue</td>
<td>Calls OnPropertyChanged on assignment (Inherited from PropertySlimTValue.)</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object. (Inherited from Object.)</td>
</tr>
<tr>
<td>WaitForValueChanged</td>
<td>Allows the caller to block until Value changes (Inherited from LockableTValue.)</td>
</tr>
</tbody>
</table>

Top
## Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTType(FuncTType)</td>
<td>Overloaded. Performs the function in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>InLockAsync&lt;TType&gt;(Func&lt;TType&gt;)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>

See Also

Reference
PropertyTValue Class
W Namespace
Tungsten

W
The **PropertyTValue** generic type exposes the following members.

## Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PropertyChanged</td>
<td>(Inherited from PropertySlimTValue.)</td>
</tr>
<tr>
<td>PropertyChanging</td>
<td>(Inherited from PropertySlimTValue.)</td>
</tr>
<tr>
<td>ValueChanged</td>
<td>Raised when the value has changed (Inherited from LockableTValue.)</td>
</tr>
</tbody>
</table>

See Also

Reference

PropertyTValue Class

W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

\( W \)
Property\textit{TValue} Type Conversions

### Operators

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>((\text{TValue to \text{PropertyTValue}}))</td>
<td>Implicit conversion from TValue to Property&lt;TValue&gt;</td>
</tr>
<tr>
<td>((\text{PropertyTValue to TValue}))</td>
<td>Implicit conversion from Property&lt;TValue&gt; to TValue</td>
</tr>
</tbody>
</table>

### See Also

Reference
- PropertyTValue Class
- W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
## PropertyTValue Conversion Operators

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(TValue to PropertyTValue)</td>
<td>Implicit conversion from TValue to Property&lt;TValue&gt;</td>
</tr>
<tr>
<td>(PropertyTValue to TValue)</td>
<td>Implicit conversion from Property&lt;TValue&gt; to TValue</td>
</tr>
</tbody>
</table>

### See Also

- Reference
  - PropertyTValue Class
  - W Namespace

---

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
Property $TValue$ Conversion ($TValue$ to Property$TValue$)

Implicit conversion from $TValue$ to Property$<TValue>$

**Namespace:** W
**Assembly:** Tungsten.Property (in Tungsten.Property.dll) Version: 2.0.2

## Syntax

```csharp
public static implicit operator Property<TValue> (TValue value)
```

### Parameters

$value$
- **Type:** $TValue$
- The value from which to create a new Property$<TValue>$

### Return Value

- **Type:** Property$TValue$

[Missing <returns> documentation for “M:W.Property`1.op_Implicit(`0)~W.Property{`0}”]

## See Also

- **Reference**
  - Property$TValue$ Class
  - Overload
  - W Namespace
Tungsten

$W$
Property<TValue> Conversion (Property<TValue> to TValue)

Implicit conversion from Property<TValue> to TValue

Namespace: W
Assembly: Tungsten.Property (in Tungsten.Property.dll) Version: 2.0.2

Syntax

```csharp
public static implicit operator TValue (Property<TValue> property)
```

Parameters

- `property`
  Type: WProperty<TValue>
  The Property<TValue> from which to obtain the value

Return Value

Type: TValue

[Missing <returns> documentation for "M:W.Property`1.op_Implicit(W.Property{`0})~`0"]

See Also

Reference
- Property<TValue> Class
- Overload
- W Namespace
Tungsten

W
Property TValue Fields

The PropertyTValue generic type exposes the following members.

Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locker</td>
<td>The ReaderWriterLocker used to access the State (Inherited from ReaderWriterLockerTState.)</td>
</tr>
<tr>
<td>State</td>
<td>The internal state (Inherited from ReaderWriterLockerTState.)</td>
</tr>
</tbody>
</table>

See Also

Reference
PropertyTValue Class
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
Property `TOwner, TValue` Class

A generic Property with an owner

### Inheritance Hierarchy

```
SystemObject   W.Threading.LockersReaderWriterLocker `TValue
W.LockableSlim `TValue
W.Lockable `TValue
W.PropertySlim `TValue
W.PropertyBase `TOwner, `TValue
W.Property `TOwner, `TValue
```

**Namespace:** W

**Assembly:** Tungsten.Property (in Tungsten.Property.dll) **Version:** 2.0.2

### Syntax

```csharp
public class Property<`TOwner, `TValue> : PropertyBase
    IOwnedProperty
where `TOwner : class
```

**Type Parameters**

- **`TOwner`**
  - The type of owner

- **`TValue`**
  - The type of the property value

The `Property<`TOwner, `TValue>` type exposes the following members.
## Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PropertyTOwner, TValue</td>
<td>Constructs a new Property</td>
</tr>
<tr>
<td>PropertyTOwner, TValue(ToObject, TValue, TValue)</td>
<td>Initializes a new instance of the PropertyTOwner, TValue class</td>
</tr>
<tr>
<td>PropertyTOwner, TValue(TOwner, TValue)</td>
<td>Constructs a new Property</td>
</tr>
<tr>
<td>PropertyTOwner, TValue(TValue)</td>
<td>Constructs a new Property</td>
</tr>
<tr>
<td>PropertyTOwner, TValue(Towner, TValue, ActionObject, TValue, TValue)</td>
<td>Initializes a new instance of the PropertyTOwner, TValue class</td>
</tr>
<tr>
<td>PropertyTOwner, TValue(TOwner, TValue)</td>
<td>Constructs a new Property</td>
</tr>
<tr>
<td>PropertyTOwner, TValue(TValue, ActionObject, TValue, TValue)</td>
<td>Initializes a new instance of the PropertyTOwner, TValue class</td>
</tr>
</tbody>
</table>

Top
## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DefaultValue</td>
<td>Allows the programmer to assign a default value which can be reset via the ResetToDefault method. This value does not have to be the initial value. (Inherited from PropertyBaseTOwner, TValue.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>True if Value has changed since initialization or since the last call to MarkAsClean (Inherited from PropertyBaseTOwner, TValue.)</td>
</tr>
<tr>
<td>Owner</td>
<td>The property owner (Inherited from PropertyBaseTOwner, TValue.)</td>
</tr>
<tr>
<td>Value</td>
<td>Get or Set the value (Inherited from LockableSlimTValue.)</td>
</tr>
</tbody>
</table>

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispose</td>
<td>Disposes the LockableSlimTValue, releases resources (Inherited from LockableSlimTValue.)</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Specify Object</strong></td>
<td>Returns a specified object if it is equal to the current object. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as the default hash function. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>GetState</strong></td>
<td>Retrieves the internal state from within a <code>ReaderWriterLockSlim</code>. (Inherited from <code>ReaderWriterLocker</code>.)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <code>Type</code> of the current instance. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>GetValue</strong></td>
<td>Gets the underlying value. (Inherited from <code>LockableSlim&lt;TValue&gt;</code>.)</td>
</tr>
<tr>
<td><strong>InformWaiters</strong></td>
<td>Informs those who are waiting on <code>WaitForChanged</code> that the value has changed. (Inherited from <code>LockableTValue</code>.)</td>
</tr>
<tr>
<td><strong>InLock</strong></td>
<td>Executes an action from within a <code>ReaderWriterLockSlim</code>. (Inherited from <code>ReaderWriterLockSlim</code>.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>InLockTValue(LockTypeEnum, FuncTState, TValue)</td>
<td>Executes a function from within a ReaderWriterLockSlim. (Inherited from ReaderWriterLocker&lt;&gt;)</td>
</tr>
<tr>
<td>InLockAsync(LockTypeEnum, ActionTState)</td>
<td>Executes an action from within a ReaderWriterLockSlim. (Inherited from ReaderWriterLocker&lt;&gt;)</td>
</tr>
<tr>
<td>InLockAsyncTValue(LockTypeEnum, FuncTState, TValue)</td>
<td>Executes a function from within a ReaderWriterLockSlim. (Inherited from ReaderWriterLocker&lt;&gt;)</td>
</tr>
<tr>
<td>LoadValue</td>
<td>Sets Value without raising notification events. (Inherited from PropertyBase&lt;TOwner, TValue&gt;.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>OnPropertyChanged</td>
<td>Calls RaisePropertyChanged to raise the PropertyChanged event. (Inherited from PropertyBase&lt;TOwner, TValue&gt;.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>OnPropertyChanging</td>
<td>Calls RaisePropertyChanging to raise the PropertyChanging event</td>
</tr>
<tr>
<td></td>
<td>(Inherited from PropertyBaseTOwner, TValue.)</td>
</tr>
<tr>
<td>OnValueChanged</td>
<td>Calls RaiseValueChanged to raise the ValueChanging event</td>
</tr>
<tr>
<td></td>
<td>(Inherited from PropertyBaseTOwner, TValue.)</td>
</tr>
<tr>
<td>RaiseOnPropertyChanged</td>
<td>Raises the PropertyChanged event</td>
</tr>
<tr>
<td></td>
<td>(Inherited from PropertySlimTValue.)</td>
</tr>
<tr>
<td>RaiseOnPropertyChanging</td>
<td>Raises the PropertyChanging event</td>
</tr>
<tr>
<td></td>
<td>(Inherited from PropertySlimTValue.)</td>
</tr>
<tr>
<td>RaiseValueChanged</td>
<td>Raises the ValueChanging event</td>
</tr>
<tr>
<td></td>
<td>(Inherited from LockableTValue.)</td>
</tr>
<tr>
<td>ResetToDefaultValue</td>
<td>Resets the Value to the value provided by DefaultValue.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from PropertyBaseTOwner, TValue.)</td>
</tr>
<tr>
<td>SetState</td>
<td>Sets the internal state within a</td>
</tr>
</tbody>
</table>
**Events**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PropertyChanged</td>
<td>(Inherited from PropertySlimTValue.)</td>
</tr>
<tr>
<td>PropertyChanging</td>
<td>(Inherited from PropertySlimTValue.)</td>
</tr>
<tr>
<td>ValueChanged</td>
<td>Raised when the value has changed (Inherited from LockableTValue.)</td>
</tr>
</tbody>
</table>
### Operators

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(TValue to PropertyTOwner, TValue)</td>
<td>Implicit conversion from TValue to Property&lt;TOwner, TValue&gt;</td>
</tr>
<tr>
<td>(PropertyTOwner, TValue to TValue)</td>
<td>Implicit conversion from Property&lt;TOwner, TValue&gt; to TValue</td>
</tr>
</tbody>
</table>

### Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locker</td>
<td>The ReaderWriterLocker used to access the State (Inherited from ReaderWriterLockerTState.)</td>
</tr>
<tr>
<td>State</td>
<td>The internal state (Inherited from ReaderWriterLockerTState.)</td>
</tr>
</tbody>
</table>

### Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>Method Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTTType(FuncTTType)</td>
<td>Overloaded. Performs the function in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTTType(FuncTTType)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets it's IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>

See Also

Reference

W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
**Property** $TOwner, TValue

**Constructor**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Property</strong>$TOwner, TValue</td>
<td>Constructs a new Property</td>
</tr>
<tr>
<td><strong>Property</strong>$TOwner, TValue($ActionObject, TValue, TValue)</td>
<td>Initializes a new instance of the <strong>Property</strong>$TOwner, TValue class</td>
</tr>
<tr>
<td><strong>Property</strong>$TOwner, TValue($TOwner)</td>
<td>Constructs a new Property</td>
</tr>
<tr>
<td><strong>Property</strong>$TOwner, TValue(TValue)</td>
<td>Constructs a new Property</td>
</tr>
<tr>
<td><strong>Property</strong>$TOwner, TValue($TOwner, $ActionObject, TValue, TValue)</td>
<td>Initializes a new instance of the <strong>Property</strong>$TOwner, TValue class</td>
</tr>
<tr>
<td><strong>Property</strong>$TOwner, TValue($TOwner, TValue)</td>
<td>Constructs a new Property</td>
</tr>
<tr>
<td><strong>Property</strong>$TOwner, TValue($TValue, $ActionObject, TValue, TValue)</td>
<td>Initializes a new instance of the <strong>Property</strong>$TOwner, TValue class</td>
</tr>
<tr>
<td>PropertyTOwner, TValue( TObject, TValue, ActionObject, TValue)</td>
<td>Initializes a new instance of the PropertyTOwner, TValue class</td>
</tr>
</tbody>
</table>

See Also

Reference

PropertyTOwner, TValue Class
W Namespace

Copyright © 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
</tr>
</tbody>
</table>
Property $T_{Owner}$, $T_{Value}$ Constructor

Constructs a new Property

**Namespace:**  W  
**Assembly:**  Tungsten.Property (in Tungsten.Property.dll)  
**Version:**  2.0.2

### Syntax

```csharp
public Property()
```

### See Also

Reference  
Property $T_{Owner}$, $T_{Value}$ Class  
Property $T_{Owner}$, $T_{Value}$ Overload  
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
Property\textit{TOwner}, \textit{TValue} Constructor (\textit{ActionObject}, \textit{TValue}, \textit{TValue})

Initializes a new instance of the \textit{Property\textit{TOwner}, \textit{TValue}} class

\textbf{Namespace: } \textit{W}
\textbf{Assembly: } Tungsten.Property (in Tungsten.Property.dll) Version: 2.0.2

\section*{Syntax}

\begin{verbatim}
c#

public Property(
    Action<Object, TValue, TValue> onValueChanged)

\end{verbatim}

Parameters

\textit{onValueChanged}

\textbf{Type: } \textit{SystemActionObject, TValue, TValue}

[Missing \texttt\angl{	exttt{<param\ name="onValueChanged"/>\ documentation\ for\ \"M:\textit{W.Property\textit{\`2}.\#ctor(System.Action(System.Object,`1,`1)}\"}}]

\section*{See Also}

\textbf{Reference}

Property\textit{TOwner}, \textit{TValue} Class
Property\textit{TOwner}, \textit{TValue} Overload
\textit{W} Namespace
Tungsten

W
Property `TOwner, TValue` Constructor (**TOwner**)

Constructs a new Property

**Namespace:**  `W`

**Assembly:**  `Tungsten.Property` (in `Tungsten.Property.dll`)  **Version:**  2.0.2

### Syntax

```
public Property(
    TOwner owner
)
```

**Parameters**

*owner*

- **Type:**  `TOwner`
- The owner of the property

### See Also

**Reference**

- Property `TOwner, TValue` Class
- Property `TOwner, TValue` Overload
- `W` Namespace

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
</tr>
</tbody>
</table>
Property \texttt{TOwner, TValue}

Constructor (\texttt{TValue})

Constructs a new Property

\textbf{Namespace:} W
\textbf{Assembly:} Tungsten.Property (in Tungsten.Property.dll) Version: 2.0.2

\section*{Syntax}

```
public Property(
    TValue defaultValue
)
```

\section*{Parameters}

\textit{defaultValue}

Type: \textit{TValue}

The default and initial value of the property

\section*{See Also}

Reference

Property\texttt{TOwner, TValue} Class

Property\texttt{TOwner, TValue} Overload

W Namespace
Tungsten

W
Property \( T_{Owner}, T_{Value} \)

Constructor \( (T_{Owner}, \text{Action} Object, T_{Value}, T_{Value}) \)

Initializes a new instance of the \( PropertyT_{Owner}, T_{Value} \) class

**Namespace:** \( W \)

**Assembly:** Tungsten.Property (in Tungsten.Property.dll) Version: 2.0.2

### Syntax

```csharp
public Property(
    TOwner owner,
    Action<Object, TValue, TValue> onValueChanged
)
```

### Parameters

- **owner**
  - Type: \( T_{Owner} \)
    - [Missing <param name="owner"/> documentation for "M:W.Property`2.#ctor(`0,System.Action{System.Object,`1,`1})"]

- **onValueChanged**
  - Type: \( \text{SystemActionObject, TValue, TValue} \)
    - [Missing <param name="onValueChanged"/> documentation for "M:W.Property`2.#ctor(`0,System.Action{System.Object,`1,`1})"]

### See Also
Tungsten

$W$
Property TOwner, TValue Constructor (TOwner, TValue)

Constructs a new Property

Namespace: W
Assembly: Tungsten.Property (in Tungsten.Property.dll) Version: 2.0.2

Syntax

```csharp
public Property(
    TOwner owner,
    TValue defaultValue
)
```

Parameters

owner
Type: TOwner
The owner of the property
defaultValue
Type: TValue
The default and initial value of the property

See Also

Reference
Property TOwner, TValue Class
Property TOwner, TValue Overload
W Namespace
Tungsten

W
Property `T.Owner, TValue` Constructor (`T.Value, ActionObject, TValue, TValue`)

Initializes a new instance of the `Property<T.Owner, TValue>` class.

**Namespace:** `W`  
**Assembly:** `Tungsten.Property` (in `Tungsten.Property.dll`)  
**Version:** 2.0.2

### Syntax

```csharp
public Property(
    TValue defaultValue,
    Action<Object, TValue, TValue> onValueChanged
)
```

### Parameters

- **defaultValue**
  - Type: `T.Value`
  - [Missing <param name="defaultValue"/> documentation for "M:W.Property`2.#ctor(1,0bject,1 Value,1 Value)""]

- **onValueChanged**
  - Type: `System.Action<Object, TValue, TValue>`
  - [Missing <param name="onValueChanged"/> documentation for "M:W.Property`2.#ctor(1,0bject,1 Value,1 Value)""]

### See Also
Reference
PropertyTOwner, TValue Class
PropertyTOwner, TValue Overload
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
Property `TOwner, TValue` Constructor (`TOwner, TValue, ActionObject, TValue, TValue`)

Initializes a new instance of the `Property<TOwner, TValue>` class

**Namespace:** W

**Assembly:** Tungsten.Property (in Tungsten.Property.dll) Version: 2.0.2

**Syntax**

```csharp
public Property(
    TOwner owner,
    TValue defaultValue,
    Action<Object, TValue, TValue> onValueChanged
)
```

**Parameters**

- **owner**
  - Type: `TOwner`
  - [Missing <param name="owner"/> documentation for "M:W.Property`2.#ctor(`0,`1,System.Action{System.Object,`1,`1})"]

- **defaultValue**
  - Type: `TValue`
  - [Missing <param name="defaultValue"/> documentation for "M:W.Property`2.#ctor(`0,`1,System.Action{System.Object,`1,`1})"]

- **onValueChanged**
  - Type: `System.ActionObject, TValue, TValue`
  - [Missing <param name="onValueChanged"/> documentation for]
See Also

Reference
PropertyTOwner, TValue Class
PropertyTOwner, TValue Overload
W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
**Property** $T$, $T$ **Properties**

The Property $T$, $T$ generic type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="DefaultValue" /></td>
<td>Allows the programmer to assign a default value which can be reset via the ResetToDefaultValue method. This value does not have to be the initial value. (Inherited from PropertyBase $T$, $T$.)</td>
</tr>
<tr>
<td><img src="image" alt="IsDirty" /></td>
<td>True if Value has changed since initialization or since the last call to MarkAsClean. (Inherited from PropertyBase $T$, $T$.)</td>
</tr>
<tr>
<td><img src="image" alt="Owner" /></td>
<td>The property owner (Inherited from PropertyBase $T$, $T$.)</td>
</tr>
<tr>
<td><img src="image" alt="Value" /></td>
<td>Get or Set the value (Inherited from LockableSlim $T$)</td>
</tr>
</tbody>
</table>
See Also

Reference

PropertyTOwner, TValue Class
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
Property `TOwner, TValue` Methods

The `PropertyTOwner, TValue` generic type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dispose</strong></td>
<td>Disposes the Lockable and releases resources (Inherited from <code>LockableTValue</code>.)</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether specified object is equal to the current object. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as the default hash function. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>GetState</strong></td>
<td>Retrieves the internal state from within a <code>ReaderWriterLockSlim</code> object.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <code>Type</code> of the instance.</td>
</tr>
<tr>
<td><strong>GetValue</strong></td>
<td>Gets the underlying value.</td>
</tr>
<tr>
<td><strong>InformWaiters</strong></td>
<td>Informs those who are waiting on <code>WaitForChanged</code> that the value has changed.</td>
</tr>
<tr>
<td><strong>InLock(LockTypeEnum, Action&lt;TState&gt;)</strong></td>
<td>Executes an action from within a ReaderWriterLockSlim.</td>
</tr>
<tr>
<td><strong>InLockTValue(LockTypeEnum, Func&lt;TState, TValue&gt;)</strong></td>
<td>Executes a function from within a ReaderWriterLockSlim.</td>
</tr>
<tr>
<td><strong>InLockAsync(LockTypeEnum, Action&lt;TState&gt;)</strong></td>
<td>Executes an action from within a ReaderWriterLockSlim.</td>
</tr>
<tr>
<td><strong>InLockAsyncTValue(LockTypeEnum, Func&lt;TState, TValue&gt;)</strong></td>
<td>Executes a function from within a ReaderWriterLockSlim.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LoadValue</td>
<td>Sets Value without raising notification events (Inherited from PropertyBase&lt;TOwner, TValue&gt;.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>OnPropertyChanged</td>
<td>Calls RaisePropertyChanged to raise the PropertyChanged event</td>
</tr>
<tr>
<td>OnPropertyChanging</td>
<td>Calls RaisePropertyChanged to raise the PropertyChanging event</td>
</tr>
<tr>
<td>OnValueChanged</td>
<td>Calls RaisePropertyChanged to raise the Value Changed event</td>
</tr>
<tr>
<td>RaiseOnPropertyChanged</td>
<td>Raises the PropertyChanged event</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>RaiseOnPropertyChanging</td>
<td>Raises the PropertyChanging event (Inherited from PropertySlimTValue.)</td>
</tr>
<tr>
<td>RaiseValueChanged</td>
<td>Raises the ValueChanged event (Inherited from LockableTValue.)</td>
</tr>
<tr>
<td>ResetToDefaultValue</td>
<td>Resets the Value to the value provided by DefaultValue (Inherited from PropertyBaseTOwner, TValue.)</td>
</tr>
<tr>
<td>SetState</td>
<td>Sets the internal state within a ReaderWriterLockSlim (Inherited from ReaderWriterLocker)</td>
</tr>
<tr>
<td>SetValue</td>
<td>Calls OnPropertyChanged on assignment (Inherited from PropertySlimTValue.)</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object. (Inherited from Object)</td>
</tr>
<tr>
<td>WaitForValueChanged</td>
<td>Allows the caller to block until Value changes (Inherited from LockableTValue.)</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>AsTType</code></td>
<td>Use Generic syntax for the as operator. (Defined by <code>AsExtensions</code>.)</td>
</tr>
<tr>
<td><code>InitializeProperties</code></td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot; This method should be called in the constructor of any class which has <code>IOwnedProperty</code> members. (Defined by <code>PropertyHostExtensions</code>.)</td>
</tr>
<tr>
<td><code>InLock(Action)</code></td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>InLockTType(FuncTType)</code></td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>InLockAsync(Action)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>InLockAsync&lt;TType&gt;(Func&lt;TType&gt;)</strong></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>IsDirty</strong></td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td><strong>Lock</strong></td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>MarkAsClean</strong></td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td><strong>Unlock</strong></td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>

Top

See Also

Reference
PropertyTOwner, TValue Class
W Namespace
Tungsten

$W$
Property TOwner, TValue Events

The Property TOwner, TValue generic type exposes the following members.

### Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PropertyChanged</td>
<td>(Inherited from PropertySlimTValue.)</td>
</tr>
<tr>
<td>PropertyChanging</td>
<td>(Inherited from PropertySlimTValue.)</td>
</tr>
<tr>
<td>ValueChanged</td>
<td>Raised when the value has changed (Inherited from LockableTValue.)</td>
</tr>
</tbody>
</table>

See Also

Reference
Property TOwner, TValue Class
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
Property $TOwner, TValue$ Type Conversions

### Operators

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(TValue to Property$TOwner$, TValue)</td>
<td>Implicit conversion from TValue to Property$TOwner$, TValue</td>
</tr>
</tbody>
</table>

See Also

Reference

Property$TOwner$, TValue Class

W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
## Property\( TOwner, TValue \)

### Conversion Operators

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implicit conversion from ( TValue ) to ( \text{Property&lt;}TOwner, TValue)</td>
<td>Implicit conversion from ( TValue ) to ( \text{Property&lt;}TOwner, TValue)</td>
</tr>
<tr>
<td>Implicit conversion from ( \text{Property&lt;}TOwner, TValue) to ( TValue )</td>
<td>Implicit conversion from ( \text{Property&lt;}TOwner, TValue) to ( TValue )</td>
</tr>
</tbody>
</table>

### See Also

See Also:

Reference

Property\( TOwner, TValue \) Class

\( W \) Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
Property `TOwner, TValue` Conversion (`TValue` to Property `TOwner, TValue`)

Implicit conversion from `TValue` to Property `TOwner, TValue`

**Namespace:** W  
**Assembly:** Tungsten.Property (in Tungsten.Property.dll)  
**Version:** 2.0.2

### Syntax

```csharp
public static implicit operator Property<TOwner, TValue> (TValue value)
```

#### Parameters

**value**  
Type: `TValue`  
The value from which to create a new Property `TOwner, TValue`

#### Return Value

Type: `Property<TOwner, TValue>`  
[Missing <returns> documentation for `M:W.Property`2.op_Implicit(`1)~W.Property{`0,`1}`]

### See Also

Reference
Tungsten

W
Property \texttt{TOwner, TValue} Conversion (Property\texttt{TOwner, TValue} to \texttt{TValue})

Implicit conversion from Property\texttt{<TOwner, TValue>} to \texttt{TValue}

\textbf{Namespace:} \texttt{W}  
\textbf{Assembly:} Tungsten.Property (in Tungsten.Property.dll) Version: 2.0.2

\section*{Syntax}

\begin{verbatim}
public static implicit operator TValue (Property<TOwner, TValue> property)
)
\end{verbatim}

\section*{Parameters}

\textit{property}  
Type: \texttt{WProperty< TOwner, TValue>}  
The Property\texttt{<TOwner, TValue>} from which to obtain the value

\section*{Return Value}

Type: \texttt{TValue}  
[Missing <returns> documentation for “M:W.Property`2.op_Implicit(W.Property{`0,`1})~`1”]

\section*{See Also}

Reference
Property\texttt{<TOwner, TValue} Class
Copyright © 2018 Jordan Duerksen
Tungsten

$W$
Property\textit{TOwner}, \textit{TValue} Fields

The \textit{Property\textit{TOwner}, \textit{TValue}} generic type exposes the following members.

### Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locker</td>
<td>The ReaderWriterLocker used to access the State (Inherited from ReaderWriterLocker\textit{TState}.)</td>
</tr>
<tr>
<td>State</td>
<td>The internal state (Inherited from ReaderWriter Locker\textit{TState}.)</td>
</tr>
</tbody>
</table>

### See Also

Reference

\textit{Property\textit{TOwner}, \textit{TValue} Class}

\textit{W Namespace}

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
PropertyBase TOwner, TValue  
Class

[Missing <summary> documentation for "T:W.PropertyBase`2"]

## Inheritance Hierarchy

```
System Object  W.Threading.Lockers.ReaderWriterLocker TValue
WLockableSlim TValue
WLockable TValue
WPropertySlim TValue
WPropertyBase TOwner, TValue
WProperty TValue
WProperty TOwner, TValue
```

**Namespace:** W  
**Assembly:** Tungsten.Property (in Tungsten.Property.dll)  
**Version:** 2.0.2

## Syntax

```c#
public abstract class PropertyBase<TOwner, TValue>
    : IProperty<TValue>, IProperty, IDisposable
```

### Type Parameters

**TOwner**

[Missing <typeparam name="TOwner"/> documentation for "T:W.PropertyBase`2"]

**TValue**

[Missing <typeparam name="TValue"/> documentation for "T:W.PropertyBase`2"]
The `PropertyBase` type exposes the following members.

## Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>PropertyBase(TOwner, TValue)</code></td>
<td>Initializes a new instance of the <code>PropertyBase</code> class</td>
</tr>
<tr>
<td><code>PropertyBase(TOwner, TValue(ActionObject, TValue, TValue))</code></td>
<td>Initializes a new instance of the <code>PropertyBase</code> class</td>
</tr>
<tr>
<td><code>PropertyBase(TOwner)</code></td>
<td>Initializes a new instance of the <code>PropertyBase</code> class</td>
</tr>
<tr>
<td><code>PropertyBase(TValue)</code></td>
<td>Initializes a new instance of the <code>PropertyBase</code> class</td>
</tr>
<tr>
<td><code>PropertyBase(TOwner, TValue(TOwner, TValue, TValue))</code></td>
<td>Initializes a new instance of the <code>PropertyBase</code> class</td>
</tr>
<tr>
<td><code>PropertyBase(TOwner, TValue(TOwner, ActionObject, TValue, TValue))</code></td>
<td>Initializes a new instance of the <code>PropertyBase</code> class</td>
</tr>
<tr>
<td><code>PropertyBase(TOwner, TValue(TOwner, TValue))</code></td>
<td>Initializes a new instance of the <code>PropertyBase</code> class</td>
</tr>
</tbody>
</table>
**Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DefaultValue</td>
<td>Allows the programmer to assign a default value which can be reset via the ResetToDefaultValue method. This value does not have to be the initial value.</td>
</tr>
<tr>
<td>IsDirty</td>
<td>True if Value has changed since initialization or since the last call to MarkAsClean</td>
</tr>
<tr>
<td>Owner</td>
<td>The property owner</td>
</tr>
<tr>
<td>Value</td>
<td>Get or Set the value (Inherited from LockableSlimTValue.)</td>
</tr>
</tbody>
</table>

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispose</td>
<td>Disposes the LockableSlimTValue.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup</td>
</tr>
<tr>
<td></td>
<td>operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td><strong>GetState</strong></td>
<td>Retrieves the internal state from within a ReaderWriterLockSlim.</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <code>Type</code> of the current instance.</td>
</tr>
<tr>
<td><strong>GetValue</strong></td>
<td>Gets the underlying value.</td>
</tr>
<tr>
<td><strong>InformWaiters</strong></td>
<td>Informs those who are waiting on <code>WaitForChanged</code> that the value has</td>
</tr>
<tr>
<td></td>
<td>changed.</td>
</tr>
<tr>
<td><strong>InLock</strong> <code>LockType</code> <code>Action</code> <code>TState</code></td>
<td>Executes an action from within a ReaderWriterLockSlim.</td>
</tr>
<tr>
<td>**InLockTValue<code>LockType</code> <code>Func</code> <code>TState</code> <code>TValue</code></td>
<td>Executes a function from within a ReaderWriterLockSlim.</td>
</tr>
<tr>
<td>**InLockAsync<code>LockType</code> <code>Action</code> <code>TState</code></td>
<td>Executes an action from within a ReaderWriterLockSlim.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>InLockAsyncTValue(LockTypeEnum, FuncTState, TValue)</code></td>
<td>Executes a function from within a ReaderWriterLockSlim (Inherited from ReaderWriterLocker)</td>
</tr>
<tr>
<td><code>LoadValue</code></td>
<td>Sets Value without raising notification events</td>
</tr>
<tr>
<td><code>MemberwiseClone</code></td>
<td>Creates a shallow copy of the current object (Inherited from Object)</td>
</tr>
<tr>
<td><code>OnPropertyChanged</code></td>
<td>Calls RaisePropertyChanged to raise the PropertyChanged event (Overrides PropertySlimTValueOnPropertyChanged)</td>
</tr>
<tr>
<td><code>OnPropertyChanging</code></td>
<td>Calls RaisePropertyChanged to raise the PropertyChanging event (Overrides PropertySlimTValueOnPropertyChanging)</td>
</tr>
<tr>
<td><code>OnValueChanged</code></td>
<td>Calls RaisePropertyChanged to raise the ValueChanged event (Overrides LockableTValueOnValueChanged, TValue, TValue).</td>
</tr>
<tr>
<td><code>RaiseOnPropertyChanged</code></td>
<td>Raises the PropertyChanged event (Inherited from PropertySlim)</td>
</tr>
<tr>
<td><code>RaiseOnPropertyChanging</code></td>
<td>Raises the PropertyChanging event (Inherited from PropertySlim)</td>
</tr>
<tr>
<td><code>RaiseValueChanged</code></td>
<td>Raises the ValueChanged event (Inherited from Lockable)</td>
</tr>
</tbody>
</table>
| `ResetToDefaultValue` | Resets the Value to the DefaultValue
### Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetState</td>
<td>Sets the internal state from within a ReaderWriterLockSlim. (Inherited from ReaderWriterLocker)</td>
</tr>
<tr>
<td>SetValue</td>
<td>Calls OnPropertyChanged on assignment. (Inherited from PropertySlim)</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object. (Inherited from Object)</td>
</tr>
<tr>
<td>WaitForValueChanged</td>
<td>Allows the caller to block until Value changes. (Inherited from Lockable)</td>
</tr>
</tbody>
</table>

### Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PropertyChanged</td>
<td>(Inherited from PropertySlimTValue.)</td>
</tr>
<tr>
<td>PropertyChanging</td>
<td>(Inherited from PropertySlimTValue.)</td>
</tr>
<tr>
<td>ValueChanged</td>
<td>Raised when the value has changed. (Inherited from LockableTValue.)</td>
</tr>
</tbody>
</table>

### Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locker</td>
<td>The ReaderWriterLocker used to access</td>
</tr>
</tbody>
</table>
State
The internal state
(Inherited from ReaderWriterLockerTState.)

Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTTType(FuncTTType)</td>
<td>Overloaded. Performs the function in</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>Monitor lock</strong>&lt;br&gt;(Defined by MonitorExtensions.)&lt;br&gt;&lt;br&gt;[\text{InLockAsync}(\text{Action})]</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock&lt;br&gt;(Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>[\text{InLockAsync}^{\text{TType}}(\text{Func}^{\text{TType}})]</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock&lt;br&gt;(Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>IsDirty</strong></td>
<td>Scans the IsDirty value of each field and property of type IProperty&lt;br&gt;(Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td><strong>Lock</strong></td>
<td>Performs a Monitor lock&lt;br&gt;(Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>MarkAsClean</strong></td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false&lt;br&gt;(Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td><strong>Unlock</strong></td>
<td>Performs a Monitor unlock&lt;br&gt;(Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>
See Also

Reference
W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

W
## PropertyBase

### Constructor

#### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PropertyBaseTOwner, TValue</td>
<td>Initializes a new instance of the PropertyBaseTOwner, TValue class</td>
</tr>
<tr>
<td>PropertyBaseTOwner, TValue(addAction, TValue, TValue)</td>
<td>Initializes a new instance of the PropertyBaseTOwner, TValue class</td>
</tr>
<tr>
<td>PropertyBaseTOwner, TValue(TOwner)</td>
<td>Initializes a new instance of the PropertyBaseTOwner, TValue class</td>
</tr>
<tr>
<td>PropertyBaseTOwner, TValue(TValue)</td>
<td>Initializes a new instance of the PropertyBaseTOwner, TValue class</td>
</tr>
<tr>
<td>PropertyBaseTOwner, TValue(TOwner, addAction, TValue, TValue)</td>
<td>Initializes a new instance of the PropertyBaseTOwner, TValue class</td>
</tr>
<tr>
<td>PropertyBaseTOwner, TValue(TOwner, TValue)</td>
<td>Initializes a new instance of the</td>
</tr>
<tr>
<td>PropertyBase&lt;OWNER, TValue&gt; constructor</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>PropertyBase&lt;OWNER, TValue&gt;(TValue, ActionObject, TValue, TValue)</td>
<td>Initializes a new instance of the PropertyBase&lt;OWNER, TValue&gt; class</td>
</tr>
<tr>
<td>PropertyBase&lt;OWNER, TValue&gt;(OWNER, TValue, ActionObject, TValue, TValue)</td>
<td>Initializes a new instance of the PropertyBase&lt;OWNER, TValue&gt; class</td>
</tr>
</tbody>
</table>

**See Also**

Reference

PropertyBase<OWNER, TValue> Class

W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
PropertyBase\texttt{TOwner, TValue} Constructor

Initializes a new instance of the \texttt{PropertyBaseTOwner, TValue} class

\textbf{Namespace:} \texttt{W}
\textbf{Assembly:} Tungsten.Property (in Tungsten.Property.dll) Version: 2.0.2

\section*{Syntax}

\begin{verbatim}
public PropertyBase()
\end{verbatim}

\section*{See Also}

Reference
\texttt{PropertyBaseTOwner, TValue Class}
\texttt{PropertyBaseTOwner, TValue Overload}
\texttt{W Namespace}

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>$W$</td>
</tr>
</tbody>
</table>
PropertyBase<TOwner, TValue> Constructor (Action<Object, TValue, TValue>)

Initializes a new instance of the PropertyBase<TOwner, TValue> class

**Namespace:** W  
**Assembly:** Tungsten.Property (in Tungsten.Property.dll)  
**Version:** 2.0.2

### Syntax

```csharp
public PropertyBase(
    Action<Object, TValue, TValue> onValueChanged
)
```

### Parameters

*onValueChanged*

**Type:** System.Action<Object, TValue, TValue>  
[Missing <param name="onValueChanged"/> documentation for "M:W.PropertyBase`2.#ctor(System.Action{System.Object,`1,`1})"]

### See Also

**Reference**

PropertyBase<TOwner, TValue> Class  
PropertyBase<TOwner, TValue> Overload  
W Namespace
Tungsten

$W$
PropertyBase\texttt{TOwner}, \texttt{TValue}

Constructor (\texttt{TOwner})

Initializes a new instance of the \texttt{PropertyBase\texttt{TOwner}, \texttt{TValue}} class

**Namespace:** \texttt{W}

**Assembly:** Tungsten.Property (in Tungsten.Property.dll) Version: 2.0.2

### Syntax

```csharp
public PropertyBase(
    TOwner owner
)
```

### Parameters

**owner**

Type: \texttt{TOwner}

[Missing <param name="owner"/> documentation for "M:W.PropertyBase`2.#ctor`0"]

### See Also

**Reference**

- PropertyBase\texttt{TOwner}, \texttt{TValue} Class
- PropertyBase\texttt{TOwner}, \texttt{TValue} Overload
- \texttt{W} Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
PropertyBase\texttt{TOwner, TValue} Constructor (\texttt{TValue})

Initializes a new instance of the PropertyBase\texttt{TOwner, TValue} class

**Namespace: W**  
**Assembly:** Tungsten.Property (in Tungsten.Property.dll)  
**Version:** 2.0.2

### Syntax

[C#]
```csharp
public PropertyBase(  
    TValue defaultValue
)
```

### Parameters

**defaultValue**

Type: \texttt{TValue}  

[Missing <param name="defaultValue"/> documentation for "M:W.PropertyBase`2.#ctor(1)"]

### See Also

- Reference
- PropertyBase\texttt{TOwner, TValue} Class
- PropertyBase\texttt{TOwner, TValue Overload}
- W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\[ W \]
PropertyBase\( T_{\text{Owner}}, T_{\text{Value}} \)
Constructor (\( T_{\text{Owner}}, \) 
\( \text{ActionObject}, T_{\text{Value}}, T_{\text{Value}} \))

Initializes a new instance of the PropertyBase\( T_{\text{Owner}}, T_{\text{Value}} \) class

Namespace: \( \text{W} \)
Assembly: Tungsten.Property (in Tungsten.Property.dll) Version: 2.0.2

### Syntax

```c#
public PropertyBase(
    TOwner owner,
    Action<Object, TValue, TValue> onValueChanged)
```

### Parameters

**owner**

Type: \( T_{\text{Owner}} \)

[Missing <param name="owner"/> documentation for "M:W.PropertyBase\(2\).ctor(0,System.Action{System.Object,1,1})"]

**onValueChanged**

Type: \( \text{SystemActionObject, TValue, TValue} \)

[Missing <param name="onValueChanged"/> documentation for "M:W.PropertyBase\(2\).ctor(0,System.Action{System.Object,1,1})"]

### See Also
Tungsten

$^nW$
PropertyBase\(T_{Owner},\ T_{Value}\) Constructor (\(T_{Owner},\ T_{Value}\))

Initializes a new instance of the `PropertyBase\(T_{Owner},\ T_{Value}\)` class

**Namespace:** W  
**Assembly:** Tungsten.Property (in Tungsten.Property.dll)  
**Version:** 2.0.2

### Syntax

```csharp
public PropertyBase(
    TOwner owner,
    TValue defaultValue
)
```

### Parameters

**owner**  
Type: `TOwner`  
[Missing <param name="owner"/> documentation for "M:W.PropertyBase`2.#ctor(`0,`1)"]

**defaultValue**  
Type: `TValue`  
[Missing <param name="defaultValue"/> documentation for "M:W.PropertyBase`2.#ctor(`0,`1)"]

### See Also

Reference  
- PropertyBase\(T_{Owner},\ T_{Value}\) Class  
- PropertyBase\(T_{Owner},\ T_{Value}\) Overload
Tungsten

W
PropertyBase $T_{Owner}$, $T_{Value}$ Constructor ($T_{Value}$, $ActionObject$, $T_{Value}$, $T_{Value}$)

Initializes a new instance of the PropertyBase $T_{Owner}$, $T_{Value}$ class

Namespace: W
Assembly: Tungsten.Property (in Tungsten.Property.dll) Version: 2.0.2

Syntax

```csharp
public PropertyBase(
    $T_{Value}$ defaultValue,
    $ActionObject$, $T_{Value}$, $T_{Value}$> onValueChanged)
```

Parameters

defaultValue

Type: $T_{Value}$

[Missing <param name="defaultValue"/> documentation for "M:W.PropertyBase`2.#ctor`1,System.Action`1,`1,`1)"

onValueChanged

Type: System $ActionObject$, $T_{Value}$, $T_{Value}$

[Missing <param name="onValueChanged"/> documentation for "M:W.PropertyBase`2.#ctor`1,System.Action`1,`1,`1)"

See Also
Reference
PropertyBaseTOwner, TValue Class
PropertyBaseTOwner, TValue Overload
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
PropertyBase

Constructor (TOwner, TValue, Action<Object, TValue, TValue>)

Initializes a new instance of the PropertyBase class

Namespace: W
Assembly: Tungsten.Property (in Tungsten.Property.dll) Version: 2.0.2

Syntax

```csharp
public PropertyBase(
    TOwner owner,
    TValue defaultValue,
    Action<Object, TValue, TValue> onValueChanged)
```

Parameters

owner
Type: TOwner
[Missing <param name="owner"/> documentation for "M:W.PropertyBase`2.#ctor(0,1,System.Action{System.Object,1,1})"]

defaultValue
Type: TValue
[Missing <param name="defaultValue"/> documentation for "M:W.PropertyBase`2.#ctor(0,1,System.Action{System.Object,1,1})"]

onValueChanged
Type: System.Object, TValue, TValue
[Missing <param name="onValueChanged"/> documentation for "M:W.PropertyBase`2.#ctor(0,1,System.Action{System.Object,1,1})"]
See Also

Reference

PropertyBase<TOwner, TValue> Class
PropertyBase<TOwner, TValue> Overload
W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
# PropertyBase\textit{TOwner, TValue} Properties

The \textit{PropertyBase\textit{TOwner, TValue}} generic type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DefaultValue</td>
<td>Allows the programmer to assign a default value which can be reset via the \textit{ResetToDefaultValue} method. This value does not have to be the initial value.</td>
</tr>
<tr>
<td>IsDirty</td>
<td>True if Value has changed since initialization or since the last call to \textit{MarkAsClean}</td>
</tr>
<tr>
<td>Owner</td>
<td>The property owner</td>
</tr>
<tr>
<td>Value</td>
<td>Get or Set the value (Inherited from \textit{LockableSlim\textit{TValue}}.)</td>
</tr>
</tbody>
</table>

## See Also

Reference

\textit{PropertyBase\textit{TOwner, TValue Class}}

\textit{W Namespace}

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
PropertyBase<TOwner, TValue> DefaultValue Property

Allows the programmer to assign a default value which can be reset via the ResetToDefaultValue method. This value does not have to be the initial value.

Namespace: W
Assembly: Tungsten.Property (in Tungsten.Property.dll) Version: 2.0.2

Syntax

```csharp
public TValue DefaultValue { get; set; }
```

Property Value
Type: TValue

See Also

Reference

PropertyBase<TOwner, TValue> Class
W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
PropertyBase<
owner, TValue>

IsDirty Property

True if Value has changed since initialization or since the last call to MarkAsClean

Namespace: W
Assembly: Tungsten.Property (in Tungsten.Property.dll) Version: 2.0.2

Syntax

```c#
public bool IsDirty { get; set; }
```

Property Value
Type: Boolean
Implements IPropertyIsDirty

See Also

Reference
PropertyBase<TOwner, TValue> Class
W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

W"
PropertyBase<TOwner, TValue> Property

The property owner

**Namespace:** W

**Assembly:** Tungsten.Property (in Tungsten.Property.dll) Version: 2.0.2

### Syntax

```
public TOwner Owner { get; protected set; }
```

**Property Value**

**Type:** TOwner

### See Also

**Reference**

- PropertyBase<TOwner, TValue> Class
- W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
### PropertyBase<TOwner, TValue> Methods

The `PropertyBase<TOwner, TValue>` generic type exposes the following members.

#### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispose</td>
<td>Disposes the Lockable and releases resources</td>
<td>(Inherited from Lockable)</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
<td>(Inherited from Object)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to perform other cleanup operations before being reclaimed by garbage collection.</td>
<td>(Inherited from Object)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function.</td>
<td>(Inherited from Object)</td>
</tr>
<tr>
<td>GetState</td>
<td>Retrieves the internal state from within a <code>ReaderWriterLockSlim</code>.</td>
<td>(Inherited from ReaderWriterLockSlim)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance.</td>
<td>(Inherited from Object)</td>
</tr>
<tr>
<td>GetValue</td>
<td>Gets the underlying value.</td>
<td>(Inherited from Lockable)</td>
</tr>
<tr>
<td>Method Name</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>InformWaiters</td>
<td>Informs those who are waiting for the value to change.</td>
<td></td>
</tr>
<tr>
<td>InLock(LockTypeEnum, ActionTState)</td>
<td>Executes an action from within a ReaderWriterLockSlim.</td>
<td></td>
</tr>
<tr>
<td>InLockTValue(LockTypeEnum, FuncTState, TValue)</td>
<td>Executes a function from within a ReaderWriterLockSlim.</td>
<td></td>
</tr>
<tr>
<td>InLockAsync(LockTypeEnum, ActionTState)</td>
<td>Executes an action from within a ReaderWriterLockSlim.</td>
<td></td>
</tr>
<tr>
<td>InLockAsyncTValue(LockTypeEnum, FuncTState, TValue)</td>
<td>Executes a function from within a ReaderWriterLockSlim.</td>
<td></td>
</tr>
<tr>
<td>LoadValue</td>
<td>Sets Value without raising notification events.</td>
<td></td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current object.</td>
<td></td>
</tr>
<tr>
<td>OnPropertyChanged</td>
<td>Calls RaisePropertyChanged method.</td>
<td></td>
</tr>
<tr>
<td>OnPropertyChanging</td>
<td>Calls RaisePropertyChanged method.</td>
<td></td>
</tr>
<tr>
<td>OnPropertyChanging</td>
<td>Calls RaisePropertyChanged method.</td>
<td></td>
</tr>
<tr>
<td>OnValueChanged</td>
<td>Calls RaisePropertyChanged method.</td>
<td></td>
</tr>
</tbody>
</table>
### Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AsTType</code></td>
<td>Use Generic syntax for the as operator. (Defined by <code>AsExtensions</code>.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>InitializeProperties</strong></td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td><strong>InLock(Action)</strong></td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>InLockTType(Func&lt;TType&gt;)</strong></td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>InLockAsync(Action)</strong></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>InLockAsyncTType(Func&lt;TType&gt;)</strong></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>IsDirty</strong></td>
<td>Scans the IsDirty value of</td>
</tr>
<tr>
<td>Action</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>

### See Also

- **Reference**
  - PropertyBaseTOwner, TValue Class
  - W Namespace

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
</tr>
</tbody>
</table>
PropertyBase\textit{TOwner}, \textit{TValue}LoadValue Method

Sets Value without raising notification events

\textbf{Namespace: } W  \\
\textbf{Assembly: } Tungsten.Property (in Tungsten.Property.dll) Version: 2.0.2

\section*{Syntax}

C#  

\begin{verbatim}
public void LoadValue(
  TValue value
)
\end{verbatim}

\section*{Parameters}

\textit{value}  

Type: \textit{TValue}  

[Missing <param name="value"/> documentation for "M:\textsf{W.PropertyBase\textbar{2}.LoadValue\textbar{1}"\textbar{1}"

\section*{See Also}

\textbf{Reference}  

PropertyBase\textit{TOwner}, \textit{TValue} Class  

W Namespace
Tungsten

$W$
PropertyBase TOwner, TValue OnPropertyChanged Method

Calls RaisePropertyChanged to raise the PropertyChanged event

Namespace: W
Assembly: Tungsten.Property (in Tungsten.Property.dll) Version: 2.0.2

Syntax

```csharp
protected override void OnPropertyChanged(
    string propertyName = null
)
```

Parameters

`propertyName (Optional)`
Type: System.String
The name of the caller (the property which changed)

See Also

Reference
PropertyBase TOwner, TValue Class
W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
PropertyBase<TOwner, TValue> OnPropertyChanging Method

Calls RaisePropertyChanging to raise the PropertyChanging event

**Namespace:** W  
**Assembly:** Tungsten.Property (in Tungsten.Property.dll)  
**Version:** 2.0.2

### Syntax

```
protected override void OnPropertyChanging(  
    string propertyName = null)
```

### Parameters

- **propertyName** *(Optional)*  
  - **Type:** System.String  
  - The name of the caller (the property which changed)

### See Also

- Reference  
  - PropertyBase<TOwner, TValue> Class  
  - W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
PropertyBase\textit{TOwner}, \textit{TValue}\texttt{OnValueChanged} Method

Calls \texttt{RaiseValueChanged} to raise the \texttt{ValueChanged} event

\textbf{Namespace:} \textit{W}  
\textbf{Assembly:} Tungsten.Property (in Tungsten.Property.dll) Version: 2.0.2

\section*{Syntax}

\begin{verbatim}
protected override void OnValueChanged(
    Object sender,
    TValue oldValue,
    TValue newValue
)
\end{verbatim}

\section*{Parameters}

\textit{sender}
\begin{itemize}
  \item Type: \texttt{SystemObject}
  \item The property owner
\end{itemize}

\textit{oldValue}
\begin{itemize}
  \item Type: \texttt{TValue}
  \item The previous value
\end{itemize}

\textit{newValue}
\begin{itemize}
  \item Type: \texttt{TValue}
  \item The current value
\end{itemize}

\section*{See Also}

Reference
Tungsten

$W$
PropertyBase<TOwner, TValue> ResetToDefaultValue Method

Resets the Value to the value provided by DefaultValue

**Namespace:** W  
**Assembly:** Tungsten.Property (in Tungsten.Property.dll)  
**Version:** 2.0.2

### Syntax

```csharp
public void ResetToDefaultValue(
    bool raise
)
```

### Parameters

*raise*

- **Type:** System.Boolean
- If True, raise OnPropertyChanged event and call the OnValueChanged callback

### See Also

**Reference**

PropertyBase<TOwner, TValue> Class  
W Namespace

Copyright © 2018 Jordan Duerksen
| Tungsten | W |
PropertyBase{TOwner, TValue}

Events

The PropertyBase{TOwner, TValue} generic type exposes the following members.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PropertyChanged</td>
<td>(Inherited from PropertySlim{TValue}.)</td>
</tr>
<tr>
<td>PropertyChanging</td>
<td>(Inherited from PropertySlim{TValue}.)</td>
</tr>
<tr>
<td>ValueChanged</td>
<td>Raised when the value has changed (Inherited from Lockable{TValue}.)</td>
</tr>
</tbody>
</table>

See Also

Reference

PropertyBase{TOwner, TValue} Class

W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
**PropertyBase**<sub>TOwner, TValue</sub> Fields

The PropertyBase<sub>TOwner, TValue</sub> generic type exposes the following members.

### Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locker</td>
<td>The ReaderWriterLocker used to access the State (Inherited from ReaderWriterLockerTState.)</td>
</tr>
<tr>
<td>State</td>
<td>The internal state (Inherited from ReaderWriterLockerTState.)</td>
</tr>
</tbody>
</table>

### See Also

Reference

PropertyBase<sub>TOwner, TValue</sub> Class

W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

W
PropertyChangedNotifier Class

This is a base class for supporting INotifyPropertyChanged

Inheritance Hierarchy

SystemObject WPropertyChangedNotifier WPropertyHostNotifier

Namespace: W
Assembly: Tungsten.Property (in Tungsten.Property.dll) Version: 2.0.2

Syntax

```csharp
public abstract class PropertyChangedNotifier : INotifyPropertyChanged
```

The PropertyChangedNotifier type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PropertyChangedNotifier</td>
<td>Initializes a new instance of the PropertyChangedNotifier class</td>
</tr>
</tbody>
</table>

Methods
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>GetValue</strong></td>
<td>Override this method to provide Get functionality</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current <strong>Object</strong>.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>OnPropertyChanged</strong></td>
<td>Calls <strong>RaisePropertyChanged</strong> to raise the <strong>PropertyChanged</strong> event</td>
</tr>
<tr>
<td><strong>RaiseOnPropertyChanged</strong></td>
<td>Raises the <strong>PropertyChanged</strong> event</td>
</tr>
</tbody>
</table>
### Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PropertyChanged</td>
<td>Raised when a property changes</td>
</tr>
</tbody>
</table>

### Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;! This method should be called in the constructor of any class</td>
</tr>
</tbody>
</table>
which has IOwnedProperty members (Defined by PropertyHostExtensions.)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTType(Func&lt;TType&gt;)</td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTType(Func&lt;TType&gt;)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock</td>
</tr>
</tbody>
</table>
MarkAsClean

MarkAsClean scans each field and property of type IProperty and sets its IsDirty flag to false.

(Defined by PropertyHostExtensions.)

Unlock

Unlock performs a Monitor unlock.

(Defined by MonitorExtensions.)

See Also

Reference

W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
PropertyChangedNotifier Constructor

Initializes a new instance of the PropertyChangedNotifier class

**Namespace:**  W  
**Assembly:**  Tungsten.Property (in Tungsten.Property.dll)  **Version:**  2.0.2

### Syntax

```csharp
protected PropertyChangedNotifier()
```

### See Also

- Reference
  - PropertyChangedNotifier Class
  - W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

W
# PropertyChangedNotifier Methods

The `PropertyChangedNotifier` type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <code>Type</code> of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>GetValue</strong></td>
<td>Override this method to provide Get functionality</td>
</tr>
</tbody>
</table>
MemberwiseClone

Creates a shallow copy of the current \texttt{Object}. (Inherited from \texttt{Object}.)

OnPropertyChanged

Calls \texttt{RaisePropertyChanged} to raise the \texttt{PropertyChanged} event.

RaiseOnPropertyChanged

Raises the \texttt{PropertyChanged} event.

SetValue

Calls \texttt{OnPropertyChanged}. This method does not make assignments. Override this method to make assignments.

ToString

Returns a string that represents the current object. (Inherited from \texttt{Object}.)

\textbf{Extension Methods}

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathbb{A}$ $\mathbb{T}$Type</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method</td>
</tr>
</tbody>
</table>
should be called in the constructor of any class which has IOwnedProperty members
(Defined by PropertyHostExtensions.)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>InLock(Action)</strong></td>
<td>Overloaded. Performs the action in a Monitor lock</td>
</tr>
<tr>
<td><strong>InLockTTType(Func&lt;TType&gt;)</strong></td>
<td>Overloaded. Performs the function in a Monitor lock</td>
</tr>
<tr>
<td><strong>InLockAsync(Action)</strong></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock</td>
</tr>
<tr>
<td><strong>InLockAsyncTTType(Func&lt;TType&gt;)</strong></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock</td>
</tr>
<tr>
<td><strong>IsDirty</strong></td>
<td>Scans the IsDirty value of each field and property of type IProperty</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets it's IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>

**See Also**

Reference
- PropertyChangedNotifier Class
- W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
(PropertyChangedNotifierGetValue Method

Override this method to provide Get functionality

Namespace: W
Assembly: Tungsten.Property (in Tungsten.Property.dll) Version: 2.0.2

Syntax

```csharp
protected virtual Object GetValue()
```

Return Value
Type: Object
Unless overridden, this function will always return null

See Also

Reference
PropertyChangedNotifier Class
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
PropertyChangedNotifier.OnPropertyChanged Method

Calls RaisePropertyChanged to raise the PropertyChanged event

**Namespace:** W
**Assembly:** Tungsten.Property (in Tungsten.Property.dll) Version: 2.0.2

### Syntax

```csharp
protected virtual void OnPropertyChanged(
    string propertyName = null
)
```

### Parameters

- **propertyName (Optional)**
  - Type: System.String
  - The name of the caller (the property which changed)

### See Also

Reference
- PropertyChangedNotifier Class
- W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

W
PropertyChangedNotifierRaiseOnPropertyChanged Method

Raises the PropertyChanged event

**Namespace:**  W  
**Assembly:**  Tungsten.Property (in Tungsten.Property.dll)  
**Version:**  2.0.2

### Syntax

```csharp
protected virtual void RaiseOnPropertyChanged(
    Object sender,
    string propertyName
)
```

### Parameters

**sender**
- Type: `SystemObject`
- The sender is the owner of the property

**propertyName**
- Type: `SystemString`
- The name of the caller (the property which changed)

### See Also

**Reference**
- PropertyChangedEventArgsNotifier Class
- W Namespace

---

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
PropertyChangedNotifier.SetValue Method

Calls OnPropertyChanged. This method does not make assignments. Override this method to make assignments.

Namespace: W
Assembly: Tungsten.Property (in Tungsten.Property.dll) Version: 2.0.2

Syntax

```csharp
protected virtual void SetValue(
    Object value,
    string propertyName = ""
)
```

Parameters

`value`
- Type: SystemObject
  - The new value

`(propertyName Optional)`
- Type: SystemString
  - The name of the caller (the property being set)

See Also

Reference
PropertyChangedNotifier Class
W Namespace
Tungsten

$W$
PropertyChangedNotifier Events

The PropertyChangedNotifier type exposes the following members.

## Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PropertyChanged</td>
<td>Raised when a property changes</td>
</tr>
</tbody>
</table>

## See Also

Reference

PropertyChangedNotifier Class

W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
PropertyChangedNotifier

Event

Raised when a property changes

**Namespace:** W

**Assembly:** Tungsten.Property (in Tungsten.Property.dll) **Version:** 2.0.2

### Syntax

```c#
public event PropertyChangedEventHandler PropertyChanged
```

### Value

- **Type:** `System.ComponentModel.PropertyChangedEventHandler`
- **Implements** `INotifyPropertyChanged`;
- `PropertyChangedPropertyChangedChangedPropertyChangedChanged`

### See Also

**Reference**
- PropertyChangedNotifier Class
- W Namespace

Copyright @ 2018 Jordan Duerksen
PropertyHost Class

Provides a base class to automate the IsDirty, MarkAsClean and InitializeProperties functionality. Note that this class does not support INotifyPropertyChanged and is not intended to host owned properties (though nothing prevents you from doing so).

Inheritance Hierarchy

- System
- Object
- WPropertyHost

Namespace: W
Assembly: Tungsten.Property (in Tungsten.Property.dll) Version: 2.0.2

Syntax

```csharp
public class PropertyHost
```

The PropertyHost type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PropertyHost</td>
<td>Calls PropertyHostExtensions.InitializeProperties so you don't have to</td>
</tr>
</tbody>
</table>

Properties
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsDirty</td>
<td>Finds all Properties and checks their IsDirty flag</td>
</tr>
</tbody>
</table>

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup</td>
</tr>
<tr>
<td></td>
<td>operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Uses reflection to find all Properties and mark them as clean</td>
</tr>
<tr>
<td></td>
<td>(call Property.MarkAsClean())</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object.</td>
</tr>
</tbody>
</table>
(Inherited from `Object`.)

**ToString**

Returns a string that represents the current object.  
(Inherited from `Object`.)

---

## Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| **AsTType**                 | Use Generic syntax for the as operator.  
   (Defined by `AsExtensions`.) |
| **InitializeProperties**    | Scans the fields and properties of "owner" and sets the member's Owner property to "owner". This method should be called in the constructor of any class which has IOwnedProperty members.  
   (Defined by `PropertyHostExtensions`.) |
| **InLock(Action)**          | Overloaded. Performs the action in a Monitor lock.  
   (Defined by `MonitorExtensions`.) |
| **InLockTType(FuncTType)**  | Overloaded. Performs the function in a Monitor lock.  
   (Defined by |
<table>
<thead>
<tr>
<th>MdName</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTTType(FuncTTType)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets it's IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>
See Also

Reference

W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
PropertyHost Constructor

Calls PropertyHostExtensions.InitializeProperties so you don't have to

**Namespace:**  W  
**Assembly:**  Tungsten.Property (in Tungsten.Property.dll)  
**Version:**  2.0.2

## Syntax

```csharp
public PropertyHost()
```

## See Also

Reference  
PropertyHost Class
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
PropertyHost Properties

The PropertyHost type exposes the following members.

⚠ Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>📜 IsDirty</td>
<td>Finds all Properties and checks their IsDirty flag</td>
</tr>
</tbody>
</table>

Top

⚠ See Also

Reference
- PropertyHost Class
- W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$^7_{	ext{W}}$
PropertyHost.IsDirty Property

Finds all Properties and checks their IsDirty flag

**Namespace:** W  
**Assembly:** Tungsten.Property (in Tungsten.Property.dll)  
**Version:** 2.0.2

### Syntax

```csharp
public bool IsDirty { get; }
```

**Return Value**

Type: **Boolean**  
True if any Property's IsDirty flag is true. Otherwise false.

### See Also

**Reference**  
PropertyHost Class  
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
**PropertyHost Methods**

The **PropertyHost** type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Equals</code></td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><code>Finalize</code></td>
<td>Allows an object to try to free resources and perform other cleanup</td>
</tr>
<tr>
<td></td>
<td>operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><code>GetHashCode</code></td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><code>GetType</code></td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><code>MarkAsClean</code></td>
<td>Uses reflection to find all Properties and mark them as clean (call</td>
</tr>
<tr>
<td></td>
<td>Property.MarkAsClean()</td>
</tr>
<tr>
<td><code>MemberwiseClone</code></td>
<td>Creates a shallow copy of the current <strong>Object</strong>.</td>
</tr>
</tbody>
</table>
ToString

Returns a string that represents the current object.
(Inherited from Object.)

---

Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTType(FuncTType)</td>
<td>Overloaded. Performs the function in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>InLockAsync(Action)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>InLockAsyncTTType(FuncTTType)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>IsDirty</code></td>
<td>Scans the IsDirty value of each field and property of type <code>IProperty</code> (Defined by <code>PropertyHostExtensions</code>.)</td>
</tr>
<tr>
<td><code>Lock</code></td>
<td>Performs a Monitor lock (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>MarkAsClean</code></td>
<td>Scans each field and property of type <code>IProperty</code> and sets its IsDirty flag to false (Defined by <code>PropertyHostExtensions</code>.)</td>
</tr>
<tr>
<td><code>Unlock</code></td>
<td>Performs a Monitor unlock (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
</tbody>
</table>
See Also

Reference

PropertyHost Class
W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
PropertyHostMarkAsClean Method

Uses reflection to find all Properties and mark them as clean (call Property.MarkAsClean())

**Namespace:** W
**Assembly:** Tungsten.Property (in Tungsten.Property.dll) Version: 2.0.2

⚠️ **Syntax**

```csharp
public void MarkAsClean()
```

⚠️ **See Also**

Reference
PropertyHost Class
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
PropertyHostExtensions Class

Extension methods related to W.PropertyHost or any class which has multiple W.Property members

Inheritance Hierarchy

SystemObject  WPropertyHostExtensions

Namespace:  W
Assembly:  Tungsten.Property (in Tungsten.Property.dll) Version: 2.0.2

Syntax

```c#
public static class PropertyHostExtensions
```

The PropertyHostExtensions type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![S] InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot; This method should be called in the constructor of any class which has IOwnedProperty members</td>
</tr>
<tr>
<td>![S] IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty</td>
</tr>
</tbody>
</table>
MarkAsClean

Scans each field and property of type IProperty and sets its IsDirty flag to false

Top

See Also

Reference

W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
PropertyHostExtensions
Methods

The PropertyHostExtensions type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members.</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty.</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets it's IsDirty flag to false.</td>
</tr>
</tbody>
</table>

See Also

Reference
PropertyHostExtensions Class
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
PropertyHostExtensionsInitializeProperties Method

Scans the fields and properties of "owner" and sets the member's Owner property to "owner" This method should be called in the constructor of any class which has IOwnedProperty members

Namespace: W
Assembly: Tungsten.Property (in Tungsten.Property.dll) Version: 2.0.2

Syntax

```csharp
public static void InitializeProperties(
    this Object this
)
```

Parameters

this

Type: SystemObject
The object on which to find and initialize properties

Usage Note

In Visual Basic and C#, you can call this method as an instance method on any object of type Object. When you use instance method syntax to call this method, omit the first parameter. For more information, see Extension Methods (Visual Basic) or Extension Methods (C# Programming Guide).

See Also
Tungsten

\[ W \]
PropertyHostExtensions.IsDirty Method

Scans the IsDirty value of each field and property of type IProperty

**Namespace:** W  
**Assembly:** Tungsten.Property (in Tungsten.Property.dll)  
**Version:** 2.0.2

### Syntax

```csharp
public static bool IsDirty(
    this Object this
)
```

### Parameters

`this`
- **Type:** System.Object  
  - The object on which to inspect for dirty properties

### Return Value

- **Type:** Boolean  
  - True if any IProperty member's IsDirty value is true, otherwise false

### Usage Note

In Visual Basic and C#, you can call this method as an instance method on any object of type Object. When you use instance method syntax to call this method, omit the first parameter. For more information, see [Extension Methods (Visual Basic)](https://docs.microsoft.com/en-us/dotnet/visual-basic/extension-methods) or [Extension Methods (C# Programming Guide)](https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/extension-methods).
See Also

Reference
PropertyHostExtensions Class
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
PropertyHostExtensionsMarkAsClean Method

Scans each field and property of type IProperty and sets its IsDirty flag to false

Namespace: W
Assembly: Tungsten.Property (in Tungsten.Property.dll) Version: 2.0.2

▲ Syntax

```csharp
public static void MarkAsClean(
    this Object this
)
```

Parameters

this

Type: SystemObject
The object on which to mark all properties as clean

Usage Note
In Visual Basic and C#, you can call this method as an instance method on any object of type Object. When you use instance method syntax to call this method, omit the first parameter. For more information, see Extension Methods (Visual Basic) or Extension Methods (C# Programming Guide).

▲ See Also

Reference
PropertyHostExtensions Class
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
PropertyHostNotifier Class

Provides a base class to automate the IsDirty, MarkAsClean and InitializeProperties functionality. Note that this class inherits PropertyChangedNotifier for INotifyPropertyChanged support.

▲ Inheritance Hierarchy

```
SystemObject  WPropertyChangedNotifier
             WPropertyHostNotifier
```

Namespace:  W
Assembly:  Tungsten.Property (in Tungsten.Property.dll) Version: 2.0.2

▲ Syntax

```
public class PropertyHostNotifier : PropertyChangedNotifier
```

The `PropertyHostNotifier` type exposes the following members.

▲ Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PropertyHostNotifier</td>
<td>Calls PropertyHostMethods.InitializeProperties so you don't have to</td>
</tr>
</tbody>
</table>

▲ Properties
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IsDirty</strong></td>
<td>Finds all Properties and checks their IsDirty flag</td>
</tr>
</tbody>
</table>

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>Object</strong>)</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup</td>
</tr>
<tr>
<td></td>
<td>operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>Object</strong>)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>Object</strong>)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>Object</strong>)</td>
</tr>
<tr>
<td><strong>GetValue</strong></td>
<td>Override this method to provide Get functionality</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>PropertyChangedNotifier</strong>)</td>
</tr>
<tr>
<td><strong>MarkAsClean</strong></td>
<td>Uses reflection to find all</td>
</tr>
</tbody>
</table>
Properties and mark them as clean (call Property.MarkAsClean())

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>OnPropertyChanged</strong></td>
<td>Calls RaisePropertyChanged to raise the PropertyChanged event (Inherited from PropertyChangedNotifier.)</td>
</tr>
<tr>
<td><strong>RaiseOnPropertyChanged</strong></td>
<td>Raises the PropertyChanged event (Inherited from PropertyChangedNotifier.)</td>
</tr>
<tr>
<td><strong>SetValue</strong></td>
<td>Calls OnPropertyChanged. This method does not make assignments. Override this method to make assignments. (Inherited from PropertyChangedNotifier.)</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Returns a string that represents the current object. (Inherited from Object.)</td>
</tr>
</tbody>
</table>

### Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
PropertyChanged Raised when a property changes 
(Inherited from PropertyChangedNotifier.)

Top

Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>❌ AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
</tbody>
</table>
| 📦 InitializeProperties | Scans the fields and properties of "owner" and sets the member's Owner property to "owner" This method should be called in the constructor of any class which has IOwnedProperty members  
(Defined by PropertyHostExtensions.) |
| 📦 InLock(Action) | Overloaded. Performs the action in a Monitor lock  
(Defined by MonitorExtensions.)                                                             |
| 📦 InLockTTType(FuncTTType) | Overloaded. Performs the function in a Monitor lock  
(Defined by MonitorExtensions.)                                                              |
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync&lt;TType&gt;(Func&lt;TType&gt;)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>
See Also

Reference

W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$\text{W}$
PropertyHostNotifier Constructor

Calls PropertyHostMethods.InitializeProperties so you don't have to

**Namespace:**  
W

**Assembly:**  
Tungsten.Property (in Tungsten.Property.dll)  
**Version:**  
2.0.2

### Syntax

```c#
public PropertyHostNotifier()
```

### See Also

**Reference**

PropertyHostNotifier Class

W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

W
The PropertyHostNotifier type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsDirty</td>
<td>Finds all Properties and checks their IsDirty flag</td>
</tr>
</tbody>
</table>

## See Also

Reference

- PropertyHostNotifier Class
- W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
PropertyHostNotifier.IsDirty Property

Finds all Properties and checks their IsDirty flag

**Namespace:**  W  
**Assembly:**  Tungsten.Property (in Tungsten.Property.dll)  
**Version:**  2.0.2

### Syntax

```
public bool IsDirty { get; }
```

**Return Value**

Type:  **Boolean**

True if any Property's IsDirty flag is true. Otherwise false.

### See Also

Reference

- PropertyHostNotifier Class
- W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
The **PropertyHostNotifier** type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>➔ <strong>Equals</strong></td>
<td>Determines whether the specified object is equal to the current object. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>➔ <strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>➔ <strong>GetHashCode</strong></td>
<td>Serves as the default hash function. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>➔ <strong>GetType</strong></td>
<td>Gets the <strong>Type</strong> of the current instance. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>➔ <strong>GetValue</strong></td>
<td>Override this method to provide Get functionality (Inherited from <strong>PropertyChangedNotifier</strong>.)</td>
</tr>
<tr>
<td>➔ <strong>MarkAsClean</strong></td>
<td>Uses reflection to find all</td>
</tr>
</tbody>
</table>
Properties and mark them as clean (call Property.MarkAsClean())

**MemberwiseClone**

Creates a shallow copy of the current `Object`. (Inherited from `Object`.)

**OnPropertyChanged**

Calls `RaisePropertyChanged` to raise the PropertyChanged event (Inherited from `PropertyChangedNotifier`).

**RaiseOnPropertyChanged**

Raises the PropertyChanged event (Inherited from `PropertyChangedNotifier`).

**SetValue**

Calls OnPropertyChanged. This method does not make assignments. Override this method to make assignments. (Inherited from `PropertyChangedNotifier`.)

**ToString**

Returns a string that represents the current object. (Inherited from `Object`.)

---

### Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Top
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;! This method should be called in the constructor of any class which has IOwnedProperty members.  (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTTType(FuncTTType)</td>
<td>Overloaded. Performs the function in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTTType(FuncTTType)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock</td>
</tr>
</tbody>
</table>

See Also

Reference

PropertyHostNotifier Class
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
PropertyHostNotifier MarkAsClean Method

Uses reflection to find all Properties and mark them as clean (call Property.MarkAsClean())

**Namespace:** W  
**Assembly:** Tungsten.Property (in Tungsten.Property.dll)  
**Version:** 2.0.2

### Syntax

```csharp
public void MarkAsClean()
```

### See Also

Reference  
PropertyHostNotifier Class  
W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
PropertyHostNotifier Events

The PropertyHostNotifier type exposes the following members.

## Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PropertyChanged</td>
<td>Raised when a property changes (Inherited from PropertyChangedNotifier.)</td>
</tr>
</tbody>
</table>

See Also

- Reference
- PropertyHostNotifier Class
- W Namespace

Copyright @ 2018 Jordan Duerksen
PropertySlim<TValue> Class

PropertySlim extends W.Lockable by adding support for INotifyPropertyChanged

▲ Inheritance Hierarchy

SystemObject  W.Threading.LockersReaderWriterLocker TValue
             WLockableSlim TValue
             WLockable TValue
             WPropertySlim TValue
             WPropertyBase TOwner, TValue

Namespace:  W
Assembly:  Tungsten.Property (in Tungsten.Property.dll) Version: 2.0.2

▲ Syntax

```csharp
public class PropertySlim<TValue> : Lockable<TValue>, INotifyPropertyChanging, INotifyPropertyChanged
```

Type Parameters

TValue

The Type of Value

The PropertySlim<TValue> type exposes the following members.

▲ Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>


**Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>Get or Set the value (Inherited from LockableSlimTValue.)</td>
</tr>
</tbody>
</table>

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispose</td>
<td>Disposes the LockableSlimTValue, releases resources</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>object is equal to the object.</td>
<td>(Inherited from Object)</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as the default hash function. (Inherited from Object)</td>
</tr>
<tr>
<td><strong>GetState</strong></td>
<td>Retrieves the internal state from within a ReaderWriterLockSlim. (Inherited from ReaderWriterLocker)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the Type of the current instance. (Inherited from Object)</td>
</tr>
<tr>
<td><strong>GetValue</strong></td>
<td>Gets the underlying value. (Inherited from LockableSlimTValue)</td>
</tr>
<tr>
<td><strong>InformWaiters</strong></td>
<td>Informs those who are waiting on WaitForChanged that the value has changed. (Inherited from LockableSlimTValue)</td>
</tr>
<tr>
<td><strong>InLock</strong></td>
<td>Executes an action from within a ReaderWriterLockSlim. (Inherited from ReaderWriterLocker)</td>
</tr>
<tr>
<td><strong>InLockTValue</strong></td>
<td>Executes a function from within a ReaderWriterLockSlim. (Inherited from ReaderWriterLocker)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>Func&lt;TState, TValue&gt;</td>
<td>ReaderWriterLockSlim (Inherited from ReaderWriterLocker)</td>
</tr>
<tr>
<td>InLockAsync(LockTypeEnum, Action&lt;TState&gt;)</td>
<td>Executes an action from within a ReaderWriterLockSlim</td>
</tr>
<tr>
<td>InLockAsync&lt;TValue&gt;(LockTypeEnum, Func&lt;TState, TValue&gt;)</td>
<td>Executes a function from within a ReaderWriterLockSlim</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object)</td>
</tr>
<tr>
<td>OnPropertyChanged</td>
<td>Calls RaisePropertyChanged to raise the PropertyChanged event</td>
</tr>
<tr>
<td>OnPropertyChanging</td>
<td>Calls RaisePropertyChanged to raise the PropertyChanging event</td>
</tr>
<tr>
<td>OnValueChanged</td>
<td>Calls RaisePropertyChanged to raise the ValueChanged event</td>
</tr>
<tr>
<td>RaiseOnPropertyChanged</td>
<td>Raises the PropertyChanged event</td>
</tr>
<tr>
<td>RaiseOnPropertyChanging</td>
<td>Raises the PropertyChanging event</td>
</tr>
<tr>
<td>RaiseValueChanged</td>
<td>Raises the ValueChanged event</td>
</tr>
<tr>
<td>SetState</td>
<td>Sets the internal state of a ReaderWriterLockSlim.</td>
</tr>
</tbody>
</table>
Inherited from ReaderWriterLocker

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SetValue</strong></td>
<td>Calls OnPropertyChanged on assignment (Overrides LockableTValueSetValue)</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Returns a string that represents the current object. (Inherited from Object)</td>
</tr>
<tr>
<td><strong>WaitForValueChanged</strong></td>
<td>Allows the caller to block until Value changes (Inherited from Lockable)</td>
</tr>
</tbody>
</table>

## Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PropertyChanged</strong></td>
<td></td>
</tr>
<tr>
<td><strong>PropertyChanging</strong></td>
<td></td>
</tr>
<tr>
<td><strong>ValueChanged</strong></td>
<td>Raised when the value has changed (Inherited from LockableTValue.)</td>
</tr>
</tbody>
</table>

## Operators

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(TValue to <em>S</em>)</td>
<td>Implicit conversion from</td>
</tr>
</tbody>
</table>
### Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AsTType</code></td>
<td>Use Generic syntax for the <code>as</code> operator. (Defined by <code>AsExtensions</code>.)</td>
</tr>
<tr>
<td><code>InitializeProperties</code></td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTTType(Func&lt;TType&gt;)</td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTTType(Func&lt;TType&gt;)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>

See Also

Reference

W Namespace
| Tungsten | W |
PropertySlim\textit{TValue} Constructor

\section*{Overload List}

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PropertySlim\textit{TValue}</td>
<td>Construct a new PropertySlim with a default initial value</td>
</tr>
<tr>
<td>PropertySlim\textit{TValue}(ActionObject, \textit{TValue}, \textit{TValue})</td>
<td>Construct a new PropertySlim</td>
</tr>
<tr>
<td>PropertySlim\textit{TValue}(\textit{TValue})</td>
<td>Construct a new PropertySlim</td>
</tr>
<tr>
<td>PropertySlim\textit{TValue}(\textit{TValue}, ActionObject, \textit{TValue}, \textit{TValue})</td>
<td>Construct a new PropertySlim</td>
</tr>
</tbody>
</table>

\section*{See Also}

Reference
PropertySlim\textit{TValue} Class
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
PropertySlim<TValue> Constructor

Construct a new PropertySlim with a default initial value

Namespace: W
Assembly: Tungsten.Property (in Tungsten.Property.dll) Version: 2.0.2

Syntax

```
public PropertySlim()
```

See Also

Reference
- PropertySlimTValue Class
- PropertySlimTValue Overload
- W Namespace
Tungsten

$^W$
PropertySlim $TValue$ Constructor (Action$Object$, $TValue$, $TValue$)

Construct a new PropertySlim

**Namespace:** W  
**Assembly:** Tungsten.Property (in Tungsten.Property.dll)  
**Version:** 2.0.2

### Syntax

```csharp
public PropertySlim(  
    Action<Object, TValue, TValue> onValueChanged  
)
```

### Parameters

**onValueChanged**
- **Type:** System$Action<Object, TValue, TValue>$
- Called when the value is changed

### See Also

**Reference**
- PropertySlim$TValue$ Class
- PropertySlim$TValue$ Overload
- W Namespace

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>$W$</td>
</tr>
</tbody>
</table>
PropertySlim<TValue> Constructor
(<TValue>)

Construct a new PropertySlim

Namespace:  W
Assembly:  Tungsten.Property (in Tungsten.Property.dll) Version: 2.0.2

Syntax

C#

```csharp
public PropertySlim(
    TValue initialValue
)
```

Parameters

initialValue
  Type: TValue
  The initial value

See Also

Reference
  PropertySlim<TValue> Class
  PropertySlim<TValue> Overload
  W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
PropertySlim\(TValue\) Constructor
\((TValue, Action\ Object, TValue, TValue)\)

Construct a new PropertySlim.

**Namespace:** W

**Assembly:** Tungsten.Property (in Tungsten.Property.dll) Version: 2.0.2

**Syntax**

```c#
public PropertySlim(
    TValue initialValue,
    Action<Object, TValue, TValue> onValueChanged
)
```

**Parameters**

- **initialValue**
  - Type: \(TValue\)
  - The initial value

- **onValueChanged**
  - Type: System\Action\Object, TValue, TValue\)
  - Called when the value is changed

**See Also**

Reference
PropertySlim\(TValue\) Class
Tungsten

\( W \)
PropertySlim\textit{TValue} Properties

The \textit{PropertySlimTValue} generic type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>Get or Set the value (Inherited from \textit{LockableSlimTValue}).</td>
</tr>
</tbody>
</table>

See Also

Reference

\textit{PropertySlimTValue Class}

\textit{W Namespace}

Copyright @ 2018 Jordan Duerksen
Tungsten

$W^n$
## PropertySlim<TValue> Methods

The `PropertySlim<TValue>` generic type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dispose</strong></td>
<td>Disposes the Lockable and releases resources (Inherited from <code>Lockable</code>*)</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether object is equal to the object. (Inherited from <code>Object</code>*)</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to free resources and perform cleanup operations before it is reclaimed by garbage collection. (Inherited from <code>Object</code>*)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as the default hash function. (Inherited from <code>Object</code>*)</td>
</tr>
<tr>
<td><strong>GetState</strong></td>
<td>Retrieves the internal state from within a <code>ReaderWriterLockSlim</code> (Inherited from <code>ReaderWriterLocker</code>*)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <code>Type</code> of the instance. (Inherited from <code>Object</code>*)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GetValue</td>
<td>Gets the underlying value (Inherited from LockableSlimTvValue)</td>
</tr>
<tr>
<td>InformWaiters</td>
<td>Informs those who are waiting on WaitForChanged that the value has changed (Inherited from LockableSlimTvValue)</td>
</tr>
<tr>
<td>InLock(LockTypeEnum, ActionTState)</td>
<td>Executes an action from within a ReaderWriterLockSlim (Inherited from ReaderWriterLocker)</td>
</tr>
<tr>
<td>InLockTvValue(LockTypeEnum, FuncTState, TValue)</td>
<td>Executes a function from within a ReaderWriterLockSlim (Inherited from ReaderWriterLocker)</td>
</tr>
<tr>
<td>InLockAsync(LockTypeEnum, ActionTState)</td>
<td>Executes an action from within a ReaderWriterLockSlim (Inherited from ReaderWriterLocker)</td>
</tr>
<tr>
<td>InLockAsyncTvValue(LockTypeEnum, FuncTState, TValue)</td>
<td>Executes a function from within a ReaderWriterLockSlim (Inherited from ReaderWriterLocker)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object (Inherited from Object)</td>
</tr>
<tr>
<td>OnPropertyChanged</td>
<td>Calls RaisePropertyChanged to raise the PropertyChanged event</td>
</tr>
<tr>
<td>OnPropertyChanging</td>
<td>Calls RaisePropertyChanged to raise the PropertyChanging event</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>OnValueChanged</td>
<td>Calls RaiseValueChanged to raise the ValueChanged event. (Inherited from Lockable)</td>
</tr>
<tr>
<td>RaiseOnPropertyChanged</td>
<td>Raises the PropertyChanged event.</td>
</tr>
<tr>
<td>RaiseOnPropertyChanging</td>
<td>Raises the PropertyChanging event.</td>
</tr>
<tr>
<td>RaiseValueChanged</td>
<td>Raises the ValueChanged event. (Inherited from Lockable)</td>
</tr>
<tr>
<td>SetState</td>
<td>Sets the internal state to a ReaderWriterLockSlim. (Inherited from ReaderWriterLocker)</td>
</tr>
<tr>
<td>SetValue</td>
<td>Calls OnPropertyChanged on assignment. (Overrides Lockable TValueSetValue)</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object. (Inherited from Object)</td>
</tr>
<tr>
<td>WaitForValueChanged</td>
<td>Allows the caller to block until Value changes. (Inherited from Lockable)</td>
</tr>
</tbody>
</table>

**Top**

### Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>InitializeProperties</strong></td>
<td><strong>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;</strong> This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td><strong>InLock(Action)</strong></td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>InLockTTType(FuncTTType)</strong></td>
<td>Overloaded. Performs the function in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>InLockAsync(Action)</strong></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>InLockAsyncTTType(FuncTTType)</strong></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets it's IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>

See Also

Reference
PropertySlimTValue Class
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
PropertySlim TValue OnPropertyChanged Method

Calls RaisePropertyChanged to raise the PropertyChanged event

Namespace: W
Assembly: Tungsten.Property (in Tungsten.Property.dll) Version: 2.0.2

Syntax

```csharp
protected virtual void OnPropertyChanged(
    string propertyName = null
)
```

Parameters

`propertyName (Optional)`
Type: System.String
The name of the caller (the property which changed)

See Also

Reference
PropertySlim TValue Class
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( \text{Tungsten} \)

\( \text{W} \)
**PropertySlim TValue OnPropertyChanging Method**

Calls `RaisePropertyChanging` to raise the `PropertyChanging` event

**Namespace:** W  
**Assembly:** Tungsten.Property (in Tungsten.Property.dll)  
**Version:** 2.0.2

### Syntax

```csharp
protected virtual void OnPropertyChanging(
    string propertyName = null
)
```

### Parameters

- `propertyName` (**Optional**)
  - Type: `System.String`
  - The name of the caller (the property which changed)

### See Also

- Reference  
  - PropertySlim TValue Class  
  - W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
PropertySlim TValue RaiseOnPropertyChanged Method

Raises the PropertyChanged event

Namespace:  W
Assembly:  Tungsten.Property (in Tungsten.Property.dll) Version: 2.0.2

Syntax

```csharp
protected void RaiseOnPropertyChanged(
    Object sender,
    string propertyName
)
```

Parameters

sender
Type: SystemObject
The sender is the owner of the property

propertyName
Type: SystemString
The name of the caller (the property which changed)

See Also

Reference
PropertySlim TValue Class
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
PropertySlim $TValue$ RaiseOnPropertyChanging Method

Raises the PropertyChanging event

**Namespace:** W  
**Assembly:** Tungsten.Property (in Tungsten.Property.dll)  
**Version:** 2.0.2

### Syntax

```csharp
protected void RaiseOnPropertyChanging(  
    Object sender,  
    string propertyName  
)
```

### Parameters

**sender**
- Type: `SystemObject`  
The sender is the owner of the property

**propertyName**
- Type: `SystemString`  
The name of the caller (the property which changing)

### See Also

Reference  
PropertySlim $TValue$ Class  
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
PropertySlim TValue SetValue Method

Calls OnPropertyChanged on assignment

**Namespace:** `W`  
**Assembly:** Tungsten.Property (in Tungsten.Property.dll)  
**Version:** 2.0.2

### Syntax

```csharp
protected override void SetValue(TValue value)
```

#### Parameters

- `value`  
  Type: `TValue`  
  The new value

### See Also

Reference
- PropertySlimTValue Class
- W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W^+$
PropertySlim<TValue> Events

The PropertySlim<TValue> generic type exposes the following members.

## Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PropertyChanged</td>
<td>Raised when the value has changed (Inherited from LockableTValue.)</td>
</tr>
<tr>
<td>PropertyChanging</td>
<td></td>
</tr>
<tr>
<td>ValueChanged</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Reference

PropertySlim<TValue> Class

W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

W
PropertySlim<TValue>PropertyChanged Event

[Missing <summary> documentation for "E:W.PropertySlim`1.PropertyChanged"]

Namespace:  W
Assembly:  Tungsten.Property (in Tungsten.Property.dll) Version: 2.0.2

Syntax

```csharp
public event PropertyChangedEventHandler PropertyChanged
```

Value
Type: System.ComponentModel.PropertyChangedEventHandler
Implements INotifyPropertyChanged

See Also

Reference
PropertySlim<TValue> Class
W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
PropertySlim TValue PropertyChanging Event

[Missing <summary> documentation for "E:W.PropertySlim`1.PropertyChanging"]

Namespace:  W  
Assembly:  Tungsten.Property (in Tungsten.Property.dll)  Version:  2.0.2

.markdown

## Syntax

C#  

```csharp
public event PropertyChangingEventHandler PropertyChanging
```

Value  
Type:  System.ComponentModel.PropertyChangingEventHandler  
Implements  
INotifyPropertyChanged  
PropertyChanging

## See Also

Reference  
PropertySlim TValue Class  
W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

W
## PropertySlim<TValue> Type Conversions

### Operators

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>(TValue to PropertySlim&lt;TValue&gt;)</code></td>
<td>Implicit conversion from TValue to PropertySlim&lt;TValue&gt;</td>
</tr>
<tr>
<td><code>(PropertySlim&lt;TValue&gt; to TValue)</code></td>
<td>Implicit conversion from PropertySlim&lt;TValue&gt; to TValue</td>
</tr>
</tbody>
</table>

## See Also

Reference
- PropertySlim<TValue> Class
- W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

W
# PropertySlim<sup>TValue</sup> Conversion Operators

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(TValue to PropertySlim&lt;TValue&gt;)</td>
<td>Implicit conversion from TValue to PropertySlim&lt;TValue&gt;</td>
</tr>
<tr>
<td>(PropertySlim&lt;TValue&gt; to TValue)</td>
<td>Implicit conversion from PropertySlim&lt;TValue&gt; to TValue</td>
</tr>
</tbody>
</table>

## See Also

- Reference
  - PropertySlim<TValue> Class
  - W Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

W
PropertySlim<TValue> Conversion

(TValue to PropertySlim<TValue>)

Implicit conversion from TValue to PropertySlim<TValue>

**Namespace:** W
**Assembly:** Tungsten.Property (in Tungsten.Property.dll) Version: 2.0.2

### Syntax

```csharp
public static implicit operator PropertySlim<TValue>(TValue value)
```

### Parameters

- **value**
  - Type: TValue
  - The value from which to create a new PropertySlim<TValue>

### Return Value

- Type: PropertySlim<TValue>

[Missing <returns> documentation for "M:W.PropertySlim`1.op_Implicit`0~W.PropertySlim`0"]

### See Also

- **Reference**
  - PropertySlim<TValue> Class
  - Overload
  - W Namespace
Tungsten

W
PropertySlim TValue Conversion (PropertySlim TValue to TValue)

Implicit conversion from PropertySlim<TValue> to TValue

Namespace: W
Assembly: Tungsten.Property (in Tungsten.Property.dll) Version: 2.0.2

Syntax

```csharp
public static implicit operator TValue (PropertySlim<TValue> property)
```

Parameters

property
  Type: WPropertySlim<TValue>
The PropertySlim<TValue> from which to obtain the value

Return Value
  Type: TValue

[Missing <returns> documentation for "M:W.PropertySlim`1.op_Implicit(W.PropertySlim{`0})~`0"]

See Also

Reference
PropertySlimTValue Class
Overload
W Namespace
Tungsten

$W$
PropertySlim\textit{TValue} Fields

The \textit{PropertySlimTValue} generic type exposes the following members.

### Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locker</td>
<td>The ReaderWriterLocker used to access the State (Inherited from ReaderWriter Locker TState.)</td>
</tr>
<tr>
<td>State</td>
<td>The internal state (Inherited from ReaderWriter Locker TState.)</td>
</tr>
</tbody>
</table>

### See Also

Reference
- PropertySlimTValue Class
- W Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
ValueChangedDelegate<TValue> Delegate

Raised when the value has changed

**Namespace:** W
**Assembly:** Tungsten.Lockable (in Tungsten.Lockable.dll) Version: 2.0.1

**Syntax**

```csharp
public delegate void ValueChangedDelegate<TValue>
    (Object sender,
     TValue oldValue,
     TValue newValue)
```

**Parameters**

- **sender**
  - Type: `System.Object`
  - The object which raised the event

- **oldValue**
  - Type: `TValue`
  - The old value

- **newValue**
  - Type: `TValue`
  - The new value

**Type Parameters**

- `TValue`
See Also

Reference

W Namespace
Tungsten

W
W.Domains Namespace

[Missing <summary> documentation for "N:W.Domains"]

## Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DomainLoader</td>
<td>An AppDomain helper class which makes it easy to host reloadable AppDomains. Supports ShadowCopy.</td>
</tr>
</tbody>
</table>

## Interfaces

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDomainLoader</td>
<td>Defines the interface for a DomainLoader</td>
</tr>
</tbody>
</table>
Tungsten

$W$
DomainLoader Class

An AppDomain helper class which makes it easy to host relodable AppDomains. Supports ShadowCopy.

Inheritance Hierarchy

```
SystemObject  W.Domains.DomainLoader
```

Namespace:  W.Domains
Assembly:  Tungsten.Domains (in Tungsten.Domains.dll) Version: 2.0.0

Syntax

```c#
public class DomainLoader : IDomainLoader, IDisposable
```

The DomainLoader type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DomainLoader(String, Boolean)</td>
<td>Creates an AppDomain under the current AppDomain</td>
</tr>
<tr>
<td>DomainLoader(String, String, Boolean)</td>
<td>Creates an AppDomain under the current AppDomain</td>
</tr>
</tbody>
</table>
### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DomainName</td>
<td>The name of the new AppDomain</td>
</tr>
</tbody>
</table>

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create(String)</td>
<td>Instantiates a class and returns a handle to it. This handle must be cast to an interface in order to work across AppDomains.</td>
</tr>
<tr>
<td>CreateTInterfaceType(String)</td>
<td>Instantiates a class and returns a handle to it. This handle must be cast to an interface in order to work across AppDomains.</td>
</tr>
<tr>
<td>Dispose</td>
<td>Performs application-defined tasks</td>
</tr>
</tbody>
</table>
associated with freeing, releasing, or resetting unmanaged resources.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DoCallback</td>
<td>Executes an action in the context of the hosted AppDomain</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td>Execute(String, String, Object)</td>
<td>Instantiates a class and calls a method exposed by it.</td>
</tr>
<tr>
<td>ExecuteTResult(String, String, Object)</td>
<td>Instantiates a class and calls a method exposed by it.</td>
</tr>
<tr>
<td>ExecuteStaticMethod(String, String, Object)</td>
<td>Executes a static method on the specified type across the AppDomain</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>ExecuteStaticMethodTResult(String, String, Object)</td>
<td>Executes a static method on the specified type across the AppDomain</td>
</tr>
<tr>
<td>Finalize</td>
<td>Destructs the DomainLoader instance. Calls Dispose. (Overrides ObjectFinalize.)</td>
</tr>
<tr>
<td>GetDataTData</td>
<td>Gets the value stored in the current application domain for the specified name</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>Load</td>
<td>Loads the dlls into the new AppDomain</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a</td>
</tr>
</tbody>
</table>
shallow copy of the current Object. (Inherited from Object.)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetData</td>
<td>Sets the value of the specified application domain property</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object. (Inherited from Object.)</td>
</tr>
<tr>
<td>Unload</td>
<td>Unloads the AppDomain and deletes files in the cache folder. The cache folder is where dlls are copied, and run, when using shadow copying.</td>
</tr>
</tbody>
</table>

Top

## Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for</td>
</tr>
</tbody>
</table>
the as operator.  
(Defined by 
AsExtensions.)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
</table>
| InitializeProperties | Scans the fields and properties of "owner" and sets the member's Owner property to "owner". This method should be called in the constructor of any class which has IOwnedProperty members.  
(Defined by PropertyHostExtensions.) |

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
</table>
| InLock(Action) | Overloaded. Performs the action in a Monitor lock.  
(Defined by MonitorExtensions.) |

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
</table>
| InLockTTType(FuncTTType) | Overloaded. Performs the function in a Monitor lock.  
(Defined by MonitorExtensions.) |

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
</table>
| InLockAsync(Action) | Overloaded. Asynchronously performs the action in a Monitor lock.  
(Defined by MonitorExtensions.) |

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLockAsyncTTType(FuncTTType)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty</td>
</tr>
<tr>
<td></td>
<td>(Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock</td>
</tr>
<tr>
<td></td>
<td>(Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to</td>
</tr>
<tr>
<td></td>
<td>false</td>
</tr>
<tr>
<td></td>
<td>(Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock</td>
</tr>
<tr>
<td></td>
<td>(Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>

**Top**

**See Also**

**Reference**

*W.Domains Namespace*

Copyright © 2018 Jordan Duerksen
Tungsten

W
DomainLoader Constructor

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DomainLoader(String, Boolean)</td>
<td>Creates an AppDomain under the current AppDomain</td>
</tr>
<tr>
<td>DomainLoader(String, String, Boolean)</td>
<td>Creates an AppDomain under the current AppDomain</td>
</tr>
</tbody>
</table>

See Also

Reference
- DomainLoader Class
- W.Domains Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
DomainLoader Constructor (String, Boolean)

Creates an AppDomain under the current AppDomain

**Namespace:** W.Domains  
**Assembly:** Tungsten.Domains (in Tungsten.Domains.dll)  
**Version:** 2.0.0

### Syntax

```csharp
public DomainLoader(
    string relativeSubFolderForDomain,
    bool useShadowCopy = false
)
```

### Parameters

- **relativeSubFolderForDomain**  
  Type: System.String  
  The relative path to the subfolder which will be the root folder for the new AppDomain

- **useShadowCopy (Optional)**  
  Type: System.Boolean  
  True to shadow copy files. This allows dlls to be added, removed or modified while the AppDomain is still loaded.

### See Also

- Reference  
  DomainLoader Class  
  DomainLoader Overload
W.Domains Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
DomainLoader Constructor
(String, String, Boolean)

Creates an AppDomain under the current AppDomain

**Namespace:**   W.Domains  
**Assembly:**  Tungsten.Domains (in Tungsten.Domains.dll) Version: 2.0.0

### Syntax

```csharp
public DomainLoader(
    string domainName,
    string relativeSubFolderForDomain,
    bool useShadowCopy = false
)
```

### Parameters

**domainName**

Type: **System.String**

The name for the domain. If not assigned, or null or empty, a Guid is assigned.

**relativeSubFolderForDomain**

Type: **System.String**

The relative path to the subfolder which will be the root folder for the new AppDomain

**useShadowCopy (Optional)**

Type: **System.Boolean**

True to shadow copy files. This allows dlls to be added, removed or modified while the AppDomain is still loaded.
See Also

Reference
DomainLoader Class
DomainLoader Overload
W.Domains Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

W
DomainLoader Properties

The DomainLoader type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DomainName</td>
<td>The name of the new AppDomain</td>
</tr>
</tbody>
</table>

See Also

Reference

- DomainLoader Class
- W.Domains Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
DomainLoaderDomainName Property

The name of the new AppDomain

**Namespace:** W.Domains  
**Assembly:** Tungsten.Domains (in Tungsten.Domains.dll)  
**Version:** 2.0.0

### Syntax

```csharp
public string DomainName { get; }
```

### Property Value

**Type:** String

### See Also

**Reference**
- DomainLoader Class
- W.Domains Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
## DomainLoader Methods

The **DomainLoader** type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Create" /> Create(String)</td>
<td>Instantiates a class and returns a handle to it. This handle must be cast to an interface in order to work across AppDomains.</td>
</tr>
<tr>
<td><img src="image" alt="CreateTInterfaceType" /> CreateTInterfaceType(String)</td>
<td>Instantiates a class and returns a handle to it. This handle must be cast to an interface in order to work across AppDomains.</td>
</tr>
<tr>
<td><img src="image" alt="Dispose" /> Dispose</td>
<td>Performs application-defined tasks associated with freeing,</td>
</tr>
</tbody>
</table>
releasing, or resetting unmanaged resources.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DoCallback</td>
<td>Executes an action in the context of the hosted AppDomain</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object. (Inherited from Object.)</td>
</tr>
<tr>
<td>Execute(String, String, Object)</td>
<td>Instantiates a class and calls a method exposed by it.</td>
</tr>
<tr>
<td>ExecuteTResult(String, String, Object)</td>
<td>Instantiates a class and calls a method exposed by it.</td>
</tr>
<tr>
<td>ExecuteStaticMethod(String, String, Object)</td>
<td>Executes a static method on the specified type across the AppDomain</td>
</tr>
<tr>
<td>ExecuteStaticMethodTResult(String, String, Object)</td>
<td>Executes a static method</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Finalize</td>
<td>Destroys the DomainLoader instance. Calls Dispose. (Overrides ObjectFinalize.)</td>
</tr>
<tr>
<td>GetDataTData</td>
<td>Gets the value stored in the current application domain for the specified name</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>Load</td>
<td>Loads the dlls into the new AppDomain</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current</td>
</tr>
</tbody>
</table>
Object.
(Inherited from Object.)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SetData</strong></td>
<td>Sets the value of the specified application domain property</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Returns a string that represents the current object. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>Unload</strong></td>
<td>Unloads the AppDomain and deletes files in the cache folder. The cache folder is where dlls are copied, and run, when using shadow copying.</td>
</tr>
</tbody>
</table>

Top

### Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AsTType</strong></td>
<td>Use Generic syntax for the as operator. (Defined by</td>
</tr>
</tbody>
</table>
AsExtensions.)

InitializeProperties  Scans the fields and properties of "owner" and sets the member's Owner property to "owner" This method should be called in the constructor of any class which has IOwnedProperty members (Defined by PropertyHostExtensions.)

InLock(Action)  Overloaded. Performs the action in a Monitor lock (Defined by MonitorExtensions.)

InLockTType(Func<TType>)  Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.)

InLockAsync(Action)  Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)

InLockAsyncTType(Func<TType>)  Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty</td>
</tr>
<tr>
<td></td>
<td>(Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets it's IsDirty flag to false</td>
</tr>
<tr>
<td></td>
<td>(Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>

See Also

Reference
- DomainLoader Class
- W.Domains Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
DomainLoader Create Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateTInterfaceType(String)</td>
<td>Instantiates a class and returns a handle to it. This handle must be cast to an interface in order to work across AppDomains.</td>
</tr>
<tr>
<td>Create(String)</td>
<td>Instantiates a class and returns a handle to it. This handle must be cast to an interface in order to work across AppDomains.</td>
</tr>
</tbody>
</table>

See Also

Reference
DomainLoader Class
W.Domains Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
DomainLoaderCreate<TInterfaceType> Method (String)

Instantiates a class and returns a handle to it. This handle must be cast to an interface in order to work across AppDomains.

**Namespace:** W.Domains  
**Assembly:** Tungsten.Domains (in Tungsten.Domains.dll)  
**Version:** 2.0.0

### Syntax

```csharp
public TInterfaceType Create<TInterfaceType>(
    string typeName
)
```

### Parameters

- **typeName**  
  - **Type:** System.String  
  - The name of the type which is to be instantiated

### Type Parameters

- **TInterfaceType**  
  - The handle to the class is automatically cast to the interface as TInterfaceType

### Return Value

- **Type:** TInterfaceType  
  - A handle to the instantiated object. This value should be cast to an interface as only interfaces will work across AppDomains.
Implements
IDomainLoader
Create TInterfaceType(String)

See Also
Reference
DomainLoader Class
Create Overload
W.Domains Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W^+$
DomainLoader Create Method (String)

Instantiates a class and returns a handle to it. This handle must be cast to an interface in order to work across AppDomains.

**Namespace:**  W.Domains  
**Assembly:**  Tungsten.Domains (in Tungsten.Domains.dll)  
**Version:**  2.0.0

### Syntax

```
public Object Create(
    string typeName
)
```

### Parameters

**typeName**
- Type: `System.String`
- The name of the type which is to be instantiated

### Return Value

- **Type:** `Object`
- A handle to the instantiated object. This value should be cast to an interface as only interfaces will work across AppDomains.

### Implements

`IDomainLoader.Create(String)`

### See Also

- Reference
DomainLoader Class
Create Overload
W.Domains Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
DomainLoaderDispose Method

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

**Namespace:** W.Domains  
**Assembly:** Tungsten.Domains (in Tungsten.Domains.dll)  
**Version:** 2.0.0

### Syntax

```csharp
public void Dispose()
```

**Implements**
- IDomainLoaderDispose  
- IDisposableDispose

### See Also

**Reference**
- DomainLoader Class  
- W.Domains Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
DomainLoaderDoCallback Method

Executes an action in the context of the hosted AppDomain

**Namespace**: W.Domains
**Assembly**: Tungsten.Domains (in Tungsten.Domains.dll) Version: 2.0.0

**Syntax**

```csharp
public void DoCallback(
    Action action
)
```

**Parameters**

**action**
Type: SystemAction
The action to be called in the context of the hosted AppDomain

**See Also**

Reference
DomainLoader Class
W.Domains Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
# DomainLoaderExecute Method

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="https://example.com/exclamation.png" alt="Exclamation" /> <strong>Execute</strong>(String, String, Object)</td>
<td>Instantiates a class and calls a method exposed by it.</td>
</tr>
<tr>
<td><img src="https://example.com/exclamation.png" alt="Exclamation" /> <strong>Execute</strong>(String, String, Object)</td>
<td>Instantiates a class and calls a method exposed by it.</td>
</tr>
</tbody>
</table>

## See Also

- Reference
  - DomainLoader Class
  - W.Domains Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
DomainLoader

**Execute**

Method (String, String, Object)

Instantiates a class and calls a method exposed by it.

**Namespace:** W.Domains

**Assembly:** Tungsten.Domains (in Tungsten.Domains.dll) Version: 2.0.0

### Syntax

```csharp
public TResult Execute<TResult>(
    string typeName,
    string methodName,
    params Object[] args
)
```

### Parameters

**typeName**
- **Type:** System.String
- The name of the type which exposes the static method

**methodName**
- **Type:** System.String
- The name of the static method

**args**
- **Type:** System.Object
- Any arguments to be passed to the static method

### Type Parameters

**TResult**
- The result of the function call is cast to TResult
Return Value
Type: TResult
The return value from the function, casted to TResult

Implements
IDomainLoader
Execute TResult(String, String, Object)

See Also

Reference
DomainLoader Class
Execute Overload
W.Domains Namespace

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
</tr>
</tbody>
</table>
DomainLoaderExecute Method (String, String, Object)

Instantiates a class and calls a method exposed by it.

**Namespace:** W.Domains  
**Assembly:** Tungsten.Domains (in Tungsten.Domains.dll)  
**Version:** 2.0.0

### Syntax

```csharp
public void Execute(
    string typeName,
    string methodName,
    params Object[] args
)
```

### Parameters

- **typeName**
  - Type: **System.String**  
  - The name of the type which exposes the static method

- **methodName**
  - Type: **System.String**  
  - The name of the static method

- **args**
  - Type: **System.Object**  
  - Any arguments to be passed to the static method

### Implements

- **IDomainLoaderExecute(String, String, Object)**
See Also

Reference
DomainLoader Class
Execute Overload
W.Domains Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
## DomainLoader_ExecuteStaticMethod Method

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExecuteStaticMethodTResult(String, String, Object)</td>
<td>Executes a static method on the specified type across the AppDomain</td>
</tr>
<tr>
<td>ExecuteStaticMethod(String, String, Object)</td>
<td>Executes a static method on the specified type across the AppDomain</td>
</tr>
</tbody>
</table>

### See Also

Reference
- DomainLoader Class
- W.Domains Namespace
Tungsten

$W$
DomainLoader_ExecuteStaticMethod Method (String, String, Object)

Executes a static method on the specified type across the AppDomain.

**Namespace:** W.Domains  
**Assembly:** Tungsten.Domains (in Tungsten.Domains.dll) Version: 2.0.0

## Syntax

```csharp
public TResult ExecuteStaticMethod<TResult>(
    string typeName,
    string staticMethodName,
    params Object[] args
)
```

### Parameters

- **typeName**
  - **Type:** System.String
  - The name of the type which exposes the static method.

- **staticMethodName**
  - **Type:** System.String
  - The name of the static method.

- **args**
  - **Type:** System.Object
  - Any parameters to be passed to the static method.

### Type Parameters

- **TResult**
  - The result of the function call is cast to TResult.
Return Value
Type: **TResult**
The return value from the function, casted to TResult.

Implements
**IDomainLoader**
**ExecuteStaticMethod**
**TResult(String, String, Object)**

See Also

Reference
- DomainLoader Class
- ExecuteStaticMethod Overload
- W.Domains Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
DomainLoaderExecuteStaticMethod Method (String, String, Object)

Executes a static method on the specified type across the AppDomain

**Namespace:** W.Domains  
**Assembly:** Tungsten.Domains (in Tungsten.Domains.dll)  
**Version:** 2.0.0

**Syntax**

```csharp
public void ExecuteStaticMethod(
    string typeName,
    string staticMethodName,
    params Object[] args
)
```

**Parameters**

- **typeName**
  - Type: System.String
  - The name of the type which exposes the static method

- **staticMethodName**
  - Type: System.String
  - The name of the static method

- **args**
  - Type: System.Object
  - Any arguments to be passed to the static method

**Implements**

IDomainLoaderExecuteStaticMethod(String, String, Object)
See Also

Reference

DomainLoader Class
ExecuteStaticMethod Overload
W.Domains Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

\( W \)
DomainLoaderFinalize Method

Destructs the DomainLoader instance. Calls Dispose.

Namespace: W.Domains
Assembly: Tungsten.Domains (in Tungsten.Domains.dll) Version: 2.0.0

Syntax

```c#
protected override void Finalize()
```

Implements
ObjectFinalize

See Also

Reference
DomainLoader Class
W.Domains Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
DomainLoaderGetData<TData> Method

Gets the value stored in the current application domain for the specified name

**Namespace:** W.Domains  
**Assembly:** Tungsten.Domains (in Tungsten.Domains.dll)  
**Version:** 2.0.0

### Syntax

```csharp
public TData GetData<TData>(
    string name
)
```

### Parameters

- **name**
  
  Type: **System.String**
  
  The name of a predefined or custom domain property

### Type Parameters

- **TData**
  
  The type of data to be returned

### Return Value

- **Type:** **TData**
  
  The data stored in the domain property as cast to T

### See Also
Tungsten

$W$
DomainLoaderLoad Method

Loads the dlls into the new AppDomain

**Namespace:** W.Domains
**Assembly:** Tungsten.Domains (in Tungsten.Domains.dll) **Version:** 2.0.0

### Syntax

```
public void Load()
```

Implements
**IDomainLoaderLoad**

### See Also

**Reference**
DomainLoader Class
W.Domains Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
DomainLoader SetData Method

Sets the value of the specified application domain property

**Namespace:**  W.Domains  
**Assembly:**  Tungsten.Domains (in Tungsten.Domains.dll)  
**Version:**  2.0.0

**Syntax**

```csharp
public void SetData(
    string name,
    Object data
)
```

**Parameters**

- **name**  
  Type: **SystemString**  
  The name of a predefined or custom domain property

- **data**  
  Type: **SystemObject**  
  The value to be assigned to the domain property

**See Also**

- Reference  
  [DomainLoader Class](#)  
  [W.Domains Namespace](#)

---

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
DomainLoaderUnload Method

Unloads the AppDomain and deletes files in the cache folder. The cache folder is where dlls are copied, and run, when using shadow copying.

**Namespace:** W.Domains  
**Assembly:** Tungsten.Domains (in Tungsten.Domains.dll)  
**Version:** 2.0.0

**Syntax**

```
public void Unload()
```

Implements  
IDomainLoaderUnload

**See Also**

Reference  
DomainLoader Class  
W.Domains Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
IDomainLoader Interface

Defines the interface for a DomainLoader

**Namespace:** W.Domains  
**Assembly:** Tungsten.Domains (in Tungsten.Domains.dll)  
**Version:** 2.0.0

## Syntax

```c#
public interface IDomainLoader
```

The **IDomainLoader** type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="https://example.com/icon.png" alt="icon" /></td>
<td><img src="https://example.com/icon.png" alt="icon" /> Create(String)</td>
</tr>
<tr>
<td><img src="https://example.com/icon.png" alt="icon" /></td>
<td>Instantiates a class and returns a handle to it. This handle must be cast to an interface in order to work across AppDomains.</td>
</tr>
<tr>
<td><img src="https://example.com/icon.png" alt="icon" /></td>
<td><img src="https://example.com/icon.png" alt="icon" /> CreateTInterfaceType(String)</td>
</tr>
<tr>
<td><img src="https://example.com/icon.png" alt="icon" /></td>
<td>Instantiates a class and returns a</td>
</tr>
</tbody>
</table>
This handle must be cast to an interface in order to work across AppDomains.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispose</td>
<td>Performs application-defined tasks associated with freeing, releasing, or resetting unmanaged resources.</td>
</tr>
<tr>
<td>Execute(String, String, Object)</td>
<td>Instantiates a class and calls a method exposed by it.</td>
</tr>
<tr>
<td>ExecuteTResult(String, String, Object)</td>
<td>Instantiates a class and calls a method exposed by it.</td>
</tr>
<tr>
<td>ExecuteStaticMethod(String, String, Object)</td>
<td>Executes a static method on the specified type across</td>
</tr>
</tbody>
</table>
the AppDomain

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExecuteStaticMethodTResult(String, String, Object)</td>
<td>Executes a static method on the specified type across the AppDomain</td>
</tr>
<tr>
<td>Load</td>
<td>Loads the dlls into the new AppDomain</td>
</tr>
<tr>
<td>Unload</td>
<td>Unloads the AppDomain and deletes files in the cache folder. The cache folder is where dlls are copied, and run, when using shadow copying.</td>
</tr>
</tbody>
</table>

See Also

Reference
W.Domains Namespace
Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
# IDomainLoader Methods

The **IDomainLoader** type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create(String)</td>
<td>Instantiates a class and returns a handle to it. This handle must be cast to an interface in order to work across AppDomains.</td>
</tr>
<tr>
<td>CreateTInterfaceType(String)</td>
<td>Instantiates a class and returns a handle to it. This handle must be cast to an interface in order to work across AppDomains.</td>
</tr>
<tr>
<td>Dispose</td>
<td>Performs application-defined tasks</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Execute(String, String, Object)</strong></td>
<td>Instantiates a class and calls a method exposed by it.</td>
</tr>
<tr>
<td><strong>ExecuteTResult(String, String, Object)</strong></td>
<td>Instantiates a class and calls a method exposed by it.</td>
</tr>
<tr>
<td><strong>ExecuteStaticMethod(String, String, Object)</strong></td>
<td>Executes a static method on the specified type across the AppDomain</td>
</tr>
<tr>
<td><strong>ExecuteStaticMethodTResult(String, String, Object)</strong></td>
<td>Executes a static method on the specified type across the AppDomain</td>
</tr>
<tr>
<td><strong>Load</strong></td>
<td>Loads the</td>
</tr>
</tbody>
</table>
dlls into the new AppDomain

unload

Unloads the AppDomain and deletes files in the cache folder. The cache folder is where dlls are copied, and run, when using shadow copying.

See Also

Reference
IDomainLoader Interface
W.Domains Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
IDomainLoaderCreate Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create(String)</td>
<td>Instantiates a class and returns a handle to it. This handle must be cast to an interface in order to work across AppDomains.</td>
</tr>
<tr>
<td>CreateTInterfaceType(String)</td>
<td>Instantiates a class and returns a handle to it. This handle must be cast to an interface in order to work across AppDomains.</td>
</tr>
</tbody>
</table>

See Also

Reference
IDomainLoader Interface
W.Domains Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
IDomainLoaderCreate Method (String)

Instantiates a class and returns a handle to it. This handle must be cast to an interface in order to work across AppDomains.

**Namespace:** W.Domains  
**Assembly:** Tungsten.Domains (in Tungsten.Domains.dll)  
**Version:** 2.0.0

### Syntax

```csharp
Object Create(
    string typeName
)
```

### Parameters

**typeName**
- Type: System.String  
  - The name of the type which is to be instantiated

### Return Value

- Type: Object  
  - A handle to the instantiated object. This value should be cast to an interface as only interfaces will work across AppDomains.

### See Also

- Reference  
  - IDomainLoader Interface  
  - Create Overload
Tungsten

$W$
IDomainLoader>CreateTInterfaceType

Method (String)

Instantiates a class and returns a handle to it. This handle must be cast to an interface in order to work across AppDomains.

**Namespace:** W.Domains  
**Assembly:** Tungsten.Domains (in Tungsten.Domains.dll)  
**Version:** 2.0.0

### Syntax

```csharp
TInterfaceType Create<TInterfaceType>(
    string typeName
)
```

### Parameters

- **typeName**
  - Type: System.String
  - The name of the type which is to be instantiated

### Type Parameters

- **TInterfaceType**
  - The handle to the class is automatically cast to the interface TInterfaceType

### Return Value

- Type: TInterfaceType
  - A handle to the instantiated object. This value should be cast to an interface as only interfaces will work across AppDomains.
See Also

Reference
IDomainLoader Interface
Create Overload
W.Domains Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
IDomainLoaderDispose Method

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

**Namespace:**  W.Domains  
**Assembly:**  Tungsten.Domains (in Tungsten.Domains.dll)  
**Version:**  2.0.0

### Syntax

```csharp
void Dispose()
```

### See Also

Reference  
IDomainLoader Interface  
W.Domains Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$\text{W}$
# IDomainLoaderExecute Method

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="ExecuteTResult" /> (String, String, Object)</td>
<td>Instantiates a class and calls a method exposed by it.</td>
</tr>
<tr>
<td><img src="image" alt="Execute" /> (String, String, Object)</td>
<td>Instantiates a class and calls a method exposed by it.</td>
</tr>
</tbody>
</table>

## See Also

Reference
- IDomainLoader Interface
- W.Domains Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
**IDomainLoader**

Execute **TResult** Method (String, String, Object)

Instantiates a class and calls a method exposed by it.

**Namespace:** W.Domains  
**Assembly:** Tungsten.Domains (in Tungsten.Domains.dll)  
**Version:** 2.0.0

### Syntax

```csharp
TResult Execute<TResult>(
    string typeName,
    string methodName,
    params Object[] args
)
```

### Parameters

- **typeName**
  - **Type:** System.String
  - The name of the type which exposes the static method

- **methodName**
  - **Type:** System.String
  - The name of the static method

- **args**
  - **Type:** System.Object
  - Any arguments to be passed to the static method

### Type Parameters

- **TResult**
  - The result of the function call is cast to TResult
Return Value
Type: TResult
The return value from the function, casted to TResult

See Also

Reference
IDomainLoader Interface
Execute Overload
W.Domains Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
IDomainLoaderExecute Method (String, String, Object)

Instantiates a class and calls a method exposed by it.

**Namespace:** W.Domains  
**Assembly:** Tungsten.Domains (in Tungsten.Domains.dll)  
**Version:** 2.0.0

### Syntax

```csharp
void Execute(
    string typeName,
    string methodName,
    params Object[] args
)
```

### Parameters

- **typeName**  
  Type: `System.String`  
  The name of the type which exposes the static method

- **methodName**  
  Type: `System.String`  
  The name of the static method

- **args**  
  Type: `System.Object`  
  Any arguments to be passed to the static method

### See Also

Reference
IDomainLoader Interface
Execute Overload
W.Domains Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
# IDomainLoaderExecuteStaticMethod Method

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExecuteStaticMethodTResult(String, String, Object)</td>
<td>Executes a static method on the specified type across the AppDomain</td>
</tr>
<tr>
<td>ExecuteStaticMethod(String, String, Object)</td>
<td>Executes a static method on the specified type across the AppDomain</td>
</tr>
</tbody>
</table>

## See Also

- IDomainLoader Interface
- W.Domains Namespace
Tungsten

W
IDomainLoader_ExecuteStaticMethod

Method (String, String, Object)

Executes a static method on the specified type across the AppDomain

**Namespace:** W.Domains
**Assembly:** Tungsten.Domains (in Tungsten.Domains.dll) Version: 2.0.0

**Syntax**

```csharp
TResult ExecuteStaticMethod<TResult>(
    string typeName,
    string staticMethodName,
    params Object[] args
)
```

**Parameters**

*typeName*
  - Type: System.String
  - The name of the type which exposes the static method

*staticMethodName*
  - Type: System.String
  - The name of the static method

*args*
  - Type: System.Object[]
  - Any parameters to be passed to the static method

**Type Parameters**

*TResult*
  - The result of the function call is cast to TResult
Return Value
Type: TResult
The return value from the function, casted to TResult.

See Also

Reference
IDomainLoader Interface
ExecuteStaticMethod Overload
W.Domains Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
IDomainLoader.ExecuteStaticMethod
Method (String, String, Object)

Executes a static method on the specified type across the AppDomain

Namespace: W.Domains
Assembly: Tungsten.Domains (in Tungsten.Domains.dll) Version: 2.0.0

Syntax

```c#
void ExecuteStaticMethod(
    string typeName, // Type: System.String
    string staticMethodName, // Type: System.String
    params Object[] args
)
```

Parameters

- **typeName**
  - Type: System.String
  - The name of the type which exposes the static method

- **staticMethodName**
  - Type: System.String
  - The name of the static method

- **args**
  - Type: System.Object
  - Any arguments to be passed to the static method

See Also

Reference
IDomainLoader Interface
ExecuteStaticMethod Overload
W.Domains Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
IDomainLoaderLoad Method

Loads the dlls into the new AppDomain

**Namespace:**  W.Domains  
**Assembly:**  Tungsten.Domains (in Tungsten.Domains.dll)  
**Version:**  2.0.0

### Syntax

```csharp
void Load()
```

### See Also

Reference
- IDomainLoader Interface
- W.Domains Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$^W$
IDomainLoaderUnload Method

Unloads the AppDomain and deletes files in the cache folder. The cache folder is where dlls are copied, and run, when using shadow copying.

**Namespace:**  W.Domains  
**Assembly:**  Tungsten.Domains (in Tungsten.Domains.dll)  
**Version:**  2.0.0

**Syntax**

```csharp
void Unload()
```

**See Also**

Reference  
IDomainLoader Interface  
W.Domains Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
## W.Encryption Namespace

[Missing <summary> documentation for "N:W.Encryption"]

### Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AssymetricEncryption</td>
<td>Facilitates two way (assymetric) encryption via RSA cryptography</td>
</tr>
<tr>
<td>MD5</td>
<td>Used to generate MD5 hashes and strings against them</td>
</tr>
<tr>
<td>RSA</td>
<td>Provides RSA encryption functionality</td>
</tr>
<tr>
<td>RSAMethods</td>
<td>Replaces RSA. This code was adapted for NetStandard from an article published on CodeProject by Mathew John Schlabaugh. It is less complicated but works more often than my initial RSA implementation. See: <a href="https://www.codeproject.com/Article">https://www.codeproject.com/Article</a> Key-RSA-Encryption-in-C-NET</td>
</tr>
</tbody>
</table>
Tungsten

W
AssymetricEncryption Class

Facilitates two way (assymmetric) encryption via RSA cryptography

Inheritance Hierarchy

SystemObject W.Encryption AssymetricEncryption

Namespace: W.Encryption

Syntax

```c#
public class AssymetricEncryption
```

The AssymetricEncryption type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AssymetricEncryption</td>
<td>Constructs a new TwoWayEncryption instance</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CurrentKeySize</td>
<td>The KeySize the object was</td>
</tr>
</tbody>
</table>
LegalKeySizes | The legal RSA key sizes supported by the platform
---|---
PublicKey | The local public key
RemotePublicKey | The remote's public key

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decrypt</td>
<td>Decrypts data with the local private key</td>
</tr>
<tr>
<td>Encrypt</td>
<td>Encrypts data with the remote public key</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object. (Inherited from Object.)</td>
</tr>
<tr>
<td>ExchangeKeys</td>
<td>Calls the function which completes the exchange and sets RemotePublicKey to the result. This function must be implemented by the developer and is contextual to his or her scenario. In all cases however, the return value must be the remote public key upon success, or null to specify a failure.</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it</td>
</tr>
</tbody>
</table>
is reclaimed by garbage collection.
(Inherited from Object.)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object.</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object.</td>
</tr>
</tbody>
</table>

**Top**

**Extension Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>InLock(Action)</strong></td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>InLockTType(FuncTType)</strong></td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>InLockAsync(Action)</strong></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>InLockAsyncTType(FuncTType)</strong></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>IsDirty</strong></td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td><strong>Lock</strong></td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>

**See Also**

Reference

W.Encryption Namespace

Copyright @ 2018 Jordan Duerksen
| Tungsten | $W$ |
AssymetricEncryption Constructor

Constructs a new TwoWayEncryption instance

**Namespace:** W.Encryption  
**Assembly:** Tungsten.Encryption (in Tungsten.Encryption.dll) Version: 2.0.2

## Syntax

```csharp
public AssymetricEncryption(int keySize = 2048)
```

**Parameters**

`keySize` *(Optional)*  
Type: `SystemInt32`  
The encryption key size

## See Also

Reference  
AssymetricEncryption Class  
W.Encryption Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W^n$
AssymetricEncryption Properties

The AssymetricEncryption type exposes the following members.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CurrentKeySize</td>
<td>The KeySize the object was created with</td>
</tr>
<tr>
<td>LegalKeySizes</td>
<td>The legal RSA key sizes supported by the platform</td>
</tr>
<tr>
<td>PublicKey</td>
<td>The local public key</td>
</tr>
<tr>
<td>RemotePublicKey</td>
<td>The remote's public key</td>
</tr>
</tbody>
</table>

See Also

Reference
AssymetricEncryption Class
W.Encryption Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
AssymetricEncryptionCurrentKeySize Property

The KeySize the object was created with

**Namespace:** W.Encryption  
**Assembly:** Tungsten.Encryption (in Tungsten.Encryption.dll) Version: 2.0.2

### Syntax

```csharp
public int CurrentKeySize { get; }
```

### Property Value

Type: **Int32**

### See Also

Reference
- AssymetricEncryption Class
- W.Encryption Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
AssymetricEncryptionLegalKeySizes

Property

The legal RSA key sizes supported by the platform

**Namespace:** W.Encryption

**Assembly:** Tungsten.Encryption (in Tungsten.Encryption.dll) Version: 2.0.2

**Syntax**

```c#
public KeySizes[] LegalKeySizes { get; }
```

**Property Value**

Type: KeySizes

**See Also**

Reference

* AssymetricEncryption Class
* W.Encryption Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
AssymetricEncryptionPublicKey Property

The local public key

**Namespace:**  W.Encryption
**Assembly:**  Tungsten.Encryption (in Tungsten.Encryption.dll) Version: 2.0.2

**Syntax**

```
public RSAPublicKey { get; }
```

Property Value
Type: **RSAPublicKey**

**See Also**

Reference
- AssymetricEncryption Class
- W.Encryption Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$\text{W}$
AssymetricEncryptionRemotePublicKey Property

The remote's public key

**Namespace:** W.Encryption  
**Assembly:** Tungsten.Encryption (in Tungsten.Encryption.dll)  
**Version:** 2.0.2

### Syntax

```csharp
public Nullable<RSAParameters> RemotePublicKey {
}
```

#### Property Value

**Type:** Nullable<RSAParameters>

### See Also

**Reference**  
AssymetricEncryption Class  
W.Encryption Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
AssymetricEncryption Methods

The AssymetricEncryption type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decrypt</td>
<td>Decrypts data with the local private key</td>
</tr>
<tr>
<td>Encrypt</td>
<td>Encrypts data with the remote public key</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>ExchangeKeys</td>
<td>Calls the function which completes the exchange and sets RemotePublicKey to</td>
</tr>
<tr>
<td></td>
<td>the result. This function must be implemented by the developer and is</td>
</tr>
<tr>
<td></td>
<td>contextual to his or her scenario. In all cases however, the return value</td>
</tr>
<tr>
<td></td>
<td>must be the remote public key upon success, or null to specify a failure.</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup</td>
</tr>
<tr>
<td></td>
<td>operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
</tbody>
</table>
GetHashCode  Serves as the default hash function.  
(Inherited from System.Object.)

GetType  Gets the Type of the current instance.  
(Inherited from System.Object.)

MemberwiseClone  Creates a shallow copy of the current Object.  
(Inherited from System.Object.)

ToString  Returns a string that represents the current object.  
(Inherited from System.Object.)

## Top

### Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| AsTType          | Use Generic syntax for the as operator.  
(Defined by AsExtensions.) |
| InitializeProperties | Scans the fields and properties of "owner" and sets the member's Owner property to "owner". This method should be called in the constructor of any class which has IOwnedProperty members.  
(Defined by PropertyHostExtensions.) |
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>InLock(Action)</strong></td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>InLockTType(FuncTType)</strong></td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>InLockAsync(Action)</strong></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>InLockAsyncTType(FuncTType)</strong></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>IsDirty</strong></td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td><strong>Lock</strong></td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>MarkAsClean</strong></td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to</td>
</tr>
</tbody>
</table>
| Unlock | Performs a Monitor unlock  
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>

See Also

Reference
- AssymmetricEncryption Class
- W.Encryption Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
AssymetricEncryptionDecrypt Method

Decrypts data with the local private key

**Namespace:** W.Encryption

**Assembly:** Tungsten.Encryption (in Tungsten.Encryption.dll) Version: 2.0.2

### Syntax

```csharp
public bool Decrypt(ref byte[] bytes)
```

### Parameters

**bytes**

Type: `System.Byte`

The data to decrypt

### Return Value

Type: `Boolean`

The decrypted data

### See Also

Reference

AssymetricEncryption Class

W.Encryption Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$^W$
AssymetricEncryptionEncrypt Method

Encrypts data with the remote public key

**Namespace:**  W.Encryption
**Assembly:**  Tungsten.Encryption (in Tungsten.Encryption.dll) Version: 2.0.2

### Syntax

#### C#

```csharp
public bool Encrypt(
    ref byte[] bytes
)
```

### Parameters

- **bytes**
  - Type: `System.Byte`
  - The data to encrypt with the remote public key

### Return Value

- **Type:** `Boolean`

### See Also

- **Reference**
  - AssymetricEncryption Class
  - W.Encryption Namespace
Tungsten

W
AssymetricEncryptionExchangeKeys Method

Calls the function which completes the exchange and sets RemotePublicKey to the result. This function must be implemented by the developer and is contextual to his or her scenario. In all cases however, the return value must be the remote public key upon success, or null to specify a failure.

**Namespace:** W.Encryption  
**Assembly:** Tungsten.Encryption (in Tungsten.Encryption.dll) **Version:** 2.0.2

**Syntax**

```csharp
public bool ExchangeKeys(
    Func<RSAParameters, Nullable<RSAParameters>>
)
```

**Parameters**

- `del`: Type: `System.Func<RSAParameters, Nullable<RSAParameters>>`  
The function to call

**Return Value**

- Type: `Boolean`  
  True if RemotePublicKey was assigned a non-null value, otherwise False

**See Also**
Reference

AssymetricEncryption Class
W.Encryption Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
MD5 Class

Used to generate MD5 hashes and verify input strings against them

Inheritance Hierarchy

- System
  - Object
    - W.Encryption
      - MD5

Namespace: W.Encryption

Syntax

```csharp
public class MD5
```

The **MD5** type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD5</td>
<td>Initializes a new instance of the <strong>MD5</strong> class</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to</td>
</tr>
<tr>
<td>Method Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetMd5Hash(String)</td>
<td>Generates an MD5 hash of the input string</td>
</tr>
<tr>
<td>GetMd5Hash(String, MD5)</td>
<td>Generates an MD5 hash of the input string</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object. (Inherited from Object.)</td>
</tr>
<tr>
<td>VerifyMd5Hash</td>
<td>Verifies a hash against a string</td>
</tr>
</tbody>
</table>
## Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AsTType</code></td>
<td>Use Generic syntax for the as operator. (Defined by <code>AsExtensions</code>.)</td>
</tr>
<tr>
<td><code>InitializeProperties</code></td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;! This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by <code>PropertyHostExtensions</code>.)</td>
</tr>
<tr>
<td><code>InLock(Action)</code></td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>InLockTType(FuncTType)</code></td>
<td>Overloaded. Performs the function in a Monitor lock. (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>InLockAsync(Action)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock. (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>InLockAsync&lt;TType&gt;(Func&lt;TType&gt;)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>

**See Also**

Reference

W.Encryption Namespace
| Tungsten | W |
MD5 Constructor

Initializes a new instance of the MD5 class

Namespace: W.Encryption

Syntax

```csharp
public MD5()
```

See Also

Reference
MD5 Class
W.Encryption Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
# MD5 Methods

The **MD5** type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>GetMd5Hash(String)</td>
<td>Generates an MD5 hash of the input string</td>
</tr>
<tr>
<td>GetMd5Hash(String, MD5)</td>
<td>Generates an MD5 hash of the input string</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <strong>Type</strong> of the current instance. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of</td>
</tr>
</tbody>
</table>


the current **Object**.
(Inherited from **Object**.)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ToString</strong></td>
<td>Returns a string that represents the current object. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>VerifyMd5Hash</strong></td>
<td>Verifies a hash against a string</td>
</tr>
</tbody>
</table>

## Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AsTType</strong></td>
<td>Use Generic syntax for the as operator. (Defined by <strong>AsExtensions</strong>.)</td>
</tr>
<tr>
<td><strong>InitializeProperties</strong></td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by <strong>PropertyHostExtensions</strong>.)</td>
</tr>
<tr>
<td><strong>InLock(Action)</strong></td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by <strong>MonitorExtensions</strong>.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>InLockTTType(Func&lt;TType&gt;)</code></td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>InLockAsync(Action)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>InLockAsyncTTType(Func&lt;TType&gt;)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>IsDirty</code></td>
<td>Scans the IsDirty value of each field and property of type <code>IProperty</code> (Defined by <code>PropertyHostExtensions</code>.)</td>
</tr>
<tr>
<td><code>Lock</code></td>
<td>Performs a Monitor lock (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>MarkAsClean</code></td>
<td>Scans each field and property of type <code>IProperty</code> and sets its IsDirty flag to false (Defined by <code>PropertyHostExtensions</code>.)</td>
</tr>
<tr>
<td><code>Unlock</code></td>
<td>Performs a Monitor unlock</td>
</tr>
</tbody>
</table>
See Also

Reference
MD5 Class
W.Encryption Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
MD5GetMd5Hash Method

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>GetMd5Hash(String)</code></td>
<td>Generates an MD5 hash of the input string</td>
</tr>
<tr>
<td><code>GetMd5Hash(String, MD5)</code></td>
<td>Generates an MD5 hash of the input string</td>
</tr>
</tbody>
</table>

See Also

Reference

MD5 Class
W.Encryption Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
MD5GetMd5Hash Method (String)

Generates an MD5 hash of the input string

**Namespace:**  W.Encryption  
**Assembly:**  Tungsten.Encryption (in Tungsten.Encryption.dll) Version: 2.0.2

## Syntax

```csharp
public static string GetMd5Hash(
    string input
)
```

**Parameters**

*input*
- Type: *System.String*
  An MD5 hash of this input will be created

**Return Value**
- Type: *String*
  An MD5 hash of the inputted value

## See Also

**Reference**
- MD5 Class
- GetMd5Hash Overload
- W.Encryption Namespace
Tungsten

$W$
MD5GetMd5Hash Method (String, MD5)

Generates an MD5 hash of the input string

**Namespace:** W.Encryption  
**Assembly:** Tungsten.Encryption (in Tungsten.Encryption.dll) Version: 2.0.2

### Syntax

```csharp
public static string GetMd5Hash(
    string input,
    MD5 md5
)
```

#### Parameters

- **input**  
  Type: `System.String`  
  An MD5 hash of this input will be created

- **md5**  
  The previously allocated MD5 object to use

#### Return Value

Type: `String`  
An MD5 hash of the inputted value

### See Also

Reference
<table>
<thead>
<tr>
<th>Tungsten</th>
<th>W</th>
</tr>
</thead>
</table>
**MD5VerifyMd5Hash Method**

Verifies a hash against a string

**Namespace:** W.Encryption  
**Assembly:** Tungsten.Encryption (in Tungsten.Encryption.dll) Version: 2.0.2

### Syntax

```csharp
public static bool VerifyMd5Hash(
    string input,
    string hash
)
```

### Parameters

- **input**  
  Type: `SystemString`  
  The string to verify

- **hash**  
  Type: `SystemString`  
  The MD5 hash used in the verification

### Return Value

Type: `Boolean`  
True if the input string is verified, otherwise False

### See Also

Reference  
**MD5 Class**  
**W.Encryption Namespace**
Tungsten

W
PasswordHash Class

Salted password hashing with PBKDF2-SHA1. Author: havoc AT defuse.ca www: http://crackstation.net/hashing-security.htm
Compatibility: .NET 3.0 and later.

Inheritance Hierarchy

- System
  - Object
    - W.Encryption
      - PasswordHash

Namespace: W.Encryption

Syntax

```csharp
public class PasswordHash
```

The PasswordHash type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PasswordHash</td>
<td>Initializes a new instance of the PasswordHash class</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy</td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CreateHash</td>
<td>Creates a salted PBKDF2 hash of the password.</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object. (Inherited from Object.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetLicenseInfo</td>
<td>License information related to the author of PasswordHash (Taylor Hornby)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object. (Inherited from Object.)</td>
</tr>
<tr>
<td>ValidatePassword</td>
<td>Validates a password given a hash of the correct one.</td>
</tr>
</tbody>
</table>
## Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HASH_BYTE_SIZE</td>
<td></td>
</tr>
<tr>
<td>ITERATION_INDEX</td>
<td></td>
</tr>
<tr>
<td>PBKDF2_INDEX</td>
<td></td>
</tr>
<tr>
<td>PBKDF2_ITERATIONS</td>
<td></td>
</tr>
<tr>
<td>SALT_BYTE_SIZE</td>
<td></td>
</tr>
<tr>
<td>SALT_INDEX</td>
<td></td>
</tr>
</tbody>
</table>

## Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>InLock(Action)</strong></td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>InLockTType(FuncTType)</strong></td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>InLockAsync(Action)</strong></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>InLockAsyncTType(FuncTType)</strong></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>IsDirty</strong></td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td><strong>Lock</strong></td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>IsDirty</strong></td>
<td>Scans each field and</td>
</tr>
<tr>
<td><strong>MarkAsClean</strong></td>
<td>property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td><strong>Unlock</strong></td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>

See Also

Reference

W.Encryption Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
PasswordHash Constructor

Initializes a new instance of the PasswordHash class

Namespace: W.Encryption

Syntax

```csharp
public PasswordHash()
```

See Also

Reference
PasswordHash Class
W.Encryption Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
# PasswordHash Methods

The **PasswordHash** type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![CreateHash]</td>
<td>Creates a salted PBKDF2 hash of the password.</td>
</tr>
<tr>
<td>![Equals]</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td>![Finalize]</td>
<td>Allows an object to try to free resources and perform other cleanup operations.</td>
</tr>
<tr>
<td>![GetHashCode]</td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td>![GetLicenseInfo]</td>
<td>License information related to the author of PasswordHash (Taylor Hornby)</td>
</tr>
<tr>
<td>![GetType]</td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
</tbody>
</table>
### Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Returns a string that represents the current object. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>ValidatePassword</strong></td>
<td>Validates a password given a hash of the correct one.</td>
</tr>
</tbody>
</table>

#### Top

**Extension Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AsTType</strong></td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td><strong>InitializeProperties</strong></td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td><strong>InLock(Action)</strong></td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>InLockTType(Func&lt;TType&gt;)</td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTType(Func&lt;TType&gt;)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets it's IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock</td>
</tr>
</tbody>
</table>
See Also

Reference
PasswordHash Class
W.Encryption Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
PasswordHashCreateHash Method

Creates a salted PBKDF2 hash of the password.

**Namespace:**  W.Encryption  
**Assembly:**  Tungsten.Encryption (in Tungsten.Encryption.dll)  
**Version:**  2.0.2

### Syntax

```csharp
public static string CreateHash(
    string password
)
```

### Parameters

- **password**  
  Type: **System.String**  
  The password to hash.

### Return Value

- **Type:** **String**  
  The hash of the password.

### See Also

- **Reference**  
  PasswordHash Class  
  W.Encryption Namespace

© 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
</tr>
</tbody>
</table>
PasswordHashGetLicenseInfo Method

License information related to the author of PasswordHash (Taylor Hornby)

**Namespace:** W.Encryption  
**Assembly:** Tungsten.Encryption (in Tungsten.Encryption.dll)  
**Version:** 2.0.2

### Syntax

```csharp
public static string GetLicenseInfo()
```

**Return Value**
**Type:** String  
License information related to the author of PasswordHash (Taylor Hornby)

### See Also

**Reference**
PasswordHash Class  
W.Encryption Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
PasswordHash\n
ValidatePassword

Method

Validates a password given a hash of the correct one.

**Namespace:**  W.Encryption  
**Assembly:**  Tungsten.Encryption (in Tungsten.Encryption.dll)  
**Version:**  2.0.2

### Syntax

```csharp
public static bool ValidatePassword(
    string password,
    string correctHash
)
```

### Parameters

**password**  
Type: **System.String**  
The password to check.

**correctHash**  
Type: **System.String**  
A hash of the correct password.

### Return Value

Type: **Boolean**  
True if the password is correct. False otherwise.

### See Also

Reference
PasswordHash Class
W.Encryption Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
PasswordHash Fields

The PasswordHash type exposes the following members.

## Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HASH_BYTE_SIZE</td>
<td></td>
</tr>
<tr>
<td>ITERATION_INDEX</td>
<td></td>
</tr>
<tr>
<td>PBKDF2_INDEX</td>
<td></td>
</tr>
<tr>
<td>PBKDF2_ITERATIONS</td>
<td></td>
</tr>
<tr>
<td>SALT_BYTE_SIZE</td>
<td></td>
</tr>
<tr>
<td>SALT_INDEX</td>
<td></td>
</tr>
</tbody>
</table>

### See Also

Reference
- PasswordHash Class
- W.Encryption Namespace

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
</tr>
</tbody>
</table>
PasswordHash

Field

[Missing <summary> documentation for "F:W.Encryption.PasswordHash.HASH_BYTE_SIZE"]

Namespace: W.Encryption

Syntax

C#

```csharp
public const int HASH_BYTE_SIZE = 24
```

Field Value
Type: Int32

See Also

Reference:
PasswordHash Class
W.Encryption Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
PasswordHash ITERATION_INDEX Field

[Missing <summary> documentation for "F:W.Encryption.PasswordHash.ITERATION_INDEX"]

Namespace:  W.Encryption  

Syntax

```
public const int ITERATION_INDEX = 0
```

Field Value
Type:  Int32

See Also

Reference
PasswordHash Class  
W.Encryption Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
PasswordHash

PBKDF2_INDEX Field

[Missing <summary> documentation for "F:W.Encryption.PasswordHash.PBKDF2_INDEX"]

Namespace: W.Encryption

Syntax

C#

    public const int PBKDF2_INDEX = 2

Field Value
Type: Int32

See Also

Reference
PasswordHash Class
W.Encryption Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
PasswordHash

PBKDF2_ITERATIONS

Field

[Missing <summary> documentation for "F:W.Encryption.PasswordHash.PBKDF2_ITERATIONS"]

Namespace: W.Encryption

⚠️ Syntax

```c#
public const int PBKDF2_ITERATIONS = 1000
```

Field Value
Type: Int32

⚠️ See Also

Reference
PasswordHash Class
W.Encryption Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
PasswordHashSALT_BYTE_SIZE Field

[Missing <summary> documentation for "F:W.Encryption.PasswordHash.SALT_BYTE_SIZE"]

Namespace: W.Encryption

Syntax

```csharp
public const int SALT_BYTE_SIZE = 24
```

Field Value
Type: Int32

See Also

Reference
PasswordHash Class
W.Encryption Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
PasswordHashSALT_INDEX Field

[Missing <summary> documentation for "F:W.Encryption.PasswordHash.SALT_INDEX"]

Namespace:  W.Encryption

⚠️ Syntax

```c#
public const int SALT_INDEX = 1
```

Field Value
Type:  Int32

⚠️ See Also

Reference
PasswordHash Class
W.Encryption Namespace
Tungsten

$W$
RSA Class

Provides RSA encryption functionality

- Inheritance Hierarchy
  - `SystemObject`  `W.EncryptionRSA`

Namespace:  `W.Encryption`

Syntax

```c#
public class RSA : IDisposable
```

The RSA type exposes the following members.

- Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSA</td>
<td>Constructs a new RSA</td>
</tr>
</tbody>
</table>

- Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KeySize</td>
<td>The encryption key size</td>
</tr>
<tr>
<td>LegalKeySizes</td>
<td>Gets the key sizes that are</td>
</tr>
</tbody>
</table>
supported by the asymmetric algorithm

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decrypt(Byte)</td>
<td>Decrypts a byte[] (previously encrypted with the Encrypt method)</td>
</tr>
<tr>
<td>Decrypt(Byte, RSAParameters)</td>
<td>Decrypts a byte[] (previously encrypted with the Encrypt method)</td>
</tr>
<tr>
<td>Dispose</td>
<td>Disposes the instance and releases resources</td>
</tr>
<tr>
<td>Encrypt(Byte)</td>
<td>Encrypts a byte[]</td>
</tr>
<tr>
<td>Encrypt(Byte, RSAParameters)</td>
<td>Encrypts a byte[]</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup</td>
</tr>
<tr>
<td></td>
<td>operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash</td>
</tr>
</tbody>
</table>
function.
(Inherited from Object.)

---

### Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PrivateKey</strong></td>
<td>The private key used to decrypt data (do not share)</td>
</tr>
<tr>
<td><strong>PublicKey</strong></td>
<td>The public key used to encrypt data (should be shared)</td>
</tr>
</tbody>
</table>

---

### Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AsTType</strong></td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTTType(FuncTTType)</td>
<td>Overloaded. Performs the function in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTTType(FuncTTType)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>

Top

See Also

Reference
W.Encryption Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\[ W \]
RSA Constructor

Constructs a new RSA

**Namespace:** W.Encryption  
**Assembly:** Tungsten.Encryption (in Tungsten.Encryption.dll) Version: 2.0.2

- **Syntax**

```csharp
public RSA(
    int keySize
)
```

- **Parameters**

  `keySize`
  
  Type: SystemInt32  
  [Missing `<param name="keySize"/>` documentation for "M:W.Encryption.RSA.#ctor(System.Int32)"]

- **See Also**

  Reference  
  RSA Class  
  W.Encryption Namespace

Copyright © 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
</tr>
</tbody>
</table>
RSA Properties

The RSA type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KeySize</td>
<td>The encryption key size</td>
</tr>
<tr>
<td>LegalKeySizes</td>
<td>Gets the key sizes that are supported by the asymmetric algorithm</td>
</tr>
</tbody>
</table>

See Also

Reference

RSA Class

W.Encryption Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$^6\text{W}$
RSAKeySize Property

The encryption key size

**Namespace:** W.Encryption

**Assembly:** Tungsten.Encryption (in Tungsten.Encryption.dll) Version: 2.0.2

**Syntax**

```csharp
public int KeySize { get; }
```

**Property Value**

Type: Int32

**See Also**

Reference
- RSA Class
- W.Encryption Namespace

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
</tr>
</tbody>
</table>
RSALegalKeySizes Property

Gets the key sizes that are supported by the asymmetric algorithm

**Namespace:** W.Encryption  
**Assembly:** Tungsten.Encryption (in Tungsten.Encryption.dll) Version: 2.0.2

### Syntax

```
public KeySizes[] LegalKeySizes { get; }
```

**Return Value**  
Type: `KeySizes`  
An enumeration of the supported key sizes

### See Also

- Reference  
  RSA Class  
  W.Encryption Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
## RSA Methods

The **RSA** type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decrypt(Byte)</td>
<td>Decrypts a byte[] (previously encrypted with the Encrypt method)</td>
</tr>
<tr>
<td>Decrypt(Byte, RSAParameters)</td>
<td>Decrypts a byte[] (previously encrypted with the Encrypt method)</td>
</tr>
<tr>
<td>Dispose</td>
<td>Disposes the instance and releases resources</td>
</tr>
<tr>
<td>Encrypt(Byte)</td>
<td>Encrypts a byte[]</td>
</tr>
<tr>
<td>Encrypt(Byte, RSAParameters)</td>
<td>Encrypts a byte[]</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup</td>
</tr>
<tr>
<td></td>
<td>operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>Object</strong>.)</td>
</tr>
</tbody>
</table>
GetHashCode  Serves as the default hash function.  
(Inherited from Object.)

GetType  Gets the Type of the current instance.  
(Inherited from Object.)

MemberwiseClone  Creates a shallow copy of the current Object.  
(Inherited from Object.)

ToString  Returns a string that represents the current object.  
(Inherited from Object.)

---

Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| AsTType       | Use Generic syntax for the as operator.  
(Defined by AsExtensions.) |
| InitializeProperties | Scans the fields and properties of "owner" and sets the member's Owner property to "owner" 
This method should be called in the constructor of any class which has IOwnedProperty members  
(Defined by PropertyHostExtensions.) |
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTType(FuncTType)</td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTType(FuncTType)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to...</td>
</tr>
</tbody>
</table>
false
(Defined by PropertyHostExtensions.)

Unlock
Performs a Monitor unlock
(Defined by MonitorExtensions.)

See Also

Reference
RSA Class
W.Encryption Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
## RSA Decrypt Method

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decrypt(Byte)</td>
<td>Decrypts a byte[] (previously encrypted with the Encrypt method)</td>
</tr>
<tr>
<td>Decrypt(Byte, RSAParameters)</td>
<td>Decrypts a byte[] (previously encrypted with the Encrypt method)</td>
</tr>
</tbody>
</table>

### See Also

- Reference
- RSA Class
- W.Encryption Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
RSADecrypt Method (Byte)

Decrypts a byte[] (previously encrypted with the Encrypt method)

Namespace:  W.Encryption

Syntax

```csharp
public byte[] Decrypt(
    byte[] bytes
)
```

Parameters

*bytes*

Type: SystemByte
The encrypted byte[]

Return Value

Type: Byte
A byte[] containing the decrypted value

See Also

Reference
RSA Class
Decrypt Overload
W.Encryption Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$^\text{W}$
RSADecrypt Method (Byte, RSAPublicKey)

Decrypts a byte[] (previously encrypted with the Encrypt method)

Namespace: W.Encryption

Syntax

```csharp
public byte[] Decrypt(
    byte[] bytes,
    RSAPublicKey privateKey
)
```

Parameters

- **bytes**
  - Type: System.Byte
  - The encrypted byte[]

- **privateKey**
  - Type: System.Security.Cryptography.RSAPublicKey
  - The private key used to decrypt the byte[]

Return Value

Type: Byte
A byte[] containing the decrypted value

See Also

Reference
Tungsten

$W$
RSADispose Method

Disposes the instance and releases resources

**Namespace:** W.Encryption  
**Assembly:** Tungsten.Encryption (in Tungsten.Encryption.dll) Version: 2.0.2

▲ **Syntax**

```csharp
public virtual void Dispose()
```

Implements  
IDisposableDispose

▲ **See Also**

Reference  
RSA Class  
W.Encryption Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$^W$
RSAEncrypt Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encrypt(Byte)</td>
<td>Encrypts a byte[]</td>
</tr>
<tr>
<td>Encrypt(Byte, RSAParameters)</td>
<td>Encrypts a byte[]</td>
</tr>
</tbody>
</table>

See Also

Reference
- RSA Class
- W.Encryption Namespace

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>$^\text{W}$</td>
</tr>
</tbody>
</table>
RSAEncrypt Method (Byte)

Encrypts a byte[]

**Namespace:** W.Encryption  
**Assembly:** Tungsten.Encryption (in Tungsten.Encryption.dll)  
**Version:** 2.0.2

### Syntax

```
public byte[] Encrypt(byte[] bytes)
```

### Parameters

`bytes`  
Type: `SystemByte`  
The byte[] to be encrypted

### Return Value

Type: `Byte`  
A byte[] containing the encrypted bytes

### See Also

Reference  
**RSA Class**  
**Encrypt Overload**  
**W.Encryption Namespace**

Copyright © 2018 Jordan Duerksen
Tungsten

W
RSAEncrypt Method (Byte, RSAParameters)

Encrypts a byte[]

**Namespace:**  W.Encryption  
**Assembly:**  Tungsten.Encryption (in Tungsten.Encryption.dll)  
**Version:**  2.0.2

### Syntax

**C#**

```csharp
public byte[] Encrypt(
    byte[] bytes,
    RSAParameters publicKey
)
```

### Parameters

- **bytes**  
  Type:  `System.Byte`  
  The bytes to encrypt

- **publicKey**  
  Type:  `System.Security.Cryptography.RSAParameters`  
  The public key used to encrypt the bytes

### Return Value

Type:  `Byte`  
A byte[] containing the encrypted bytes

### See Also

- Reference
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
</tr>
</tbody>
</table>
RSA Fields

The RSA type exposes the following members.

Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PrivateKey</td>
<td>The private key used to decrypt data (do not share)</td>
</tr>
<tr>
<td>PublicKey</td>
<td>The public key used to encrypt data (should be shared)</td>
</tr>
</tbody>
</table>

See Also

Reference
RSA Class
W.Encryption Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
RSAPrivateKey Field

The private key used to decrypt data (do not share)

**Namespace:** W.Encryption  
**Assembly:** Tungsten.Encryption (in Tungsten.Encryption.dll) Version: 2.0.2

### Syntax

```
public RSAParameters PrivateKey
```

**Field Value**  
Type: RSAParameters

### See Also

**Reference**  
RSA Class  
W.Encryption Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

W
RSAPublicKey Field

The public key used to encrypt data (should be shared)

**Namespace:** W.Encryption  
**Assembly:** Tungsten.Encryption (in Tungsten.Encryption.dll)  
**Version:** 2.0.2

### Syntax

```csharp
public RSAPublicKey
```

### Field Value

Type: `RSAPublicKey`

### See Also

Reference
- RSA Class
- W.Encryption Namespace
Tungsten

$W$
RSAMethods Class

Replaces RSA. This code was adapted for NetStandard from an article published on CodeProject by Mathew John Schlabaugh in 2007. It is less complicated but works more often than my initial RSA implementation. See: https://www.codeproject.com/Articles/10877/Public-Key-RSA-Encryption-in-C-NET

Inheritance Hierarchy

- System
- Object
- W.Encryption
- RSAMethods

Namespace: W.Encryption

Syntax

```c#
public static class RSAMethods
```

The RSAMethods type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateKeyPair</td>
<td>Generates a public/private key pair</td>
</tr>
<tr>
<td>Decrypt</td>
<td>Decrypts a byte array which was previously encrypted with the Encrypt method</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DecryptAsync</td>
<td>Asynchronously decrypts a string which was previously encrypted with the Encrypt method</td>
</tr>
<tr>
<td>Encrypt</td>
<td>Encrypts a byte array using the specified keysize and public key</td>
</tr>
<tr>
<td>EncryptAsync</td>
<td>Asynchronously encrypts a string using the specified keysize and public key</td>
</tr>
<tr>
<td>LegalKeySizes</td>
<td>Returns an array containing the supported key sizes</td>
</tr>
</tbody>
</table>

**See Also**

Reference

W.Encryption Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
RSAMethods Methods

The RSAMethods type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>CreateKeyPair</code></td>
<td>Generates a public/private key pair</td>
</tr>
<tr>
<td><code>Decrypt</code></td>
<td>Decrypts a byte array which was previously encrypted with the Encrypt method</td>
</tr>
<tr>
<td><code>DecryptAsync</code></td>
<td>Asynchronously decrypts a string which was previously encrypted with the Encrypt method</td>
</tr>
<tr>
<td><code>Encrypt</code></td>
<td>Encrypts a byte array using the specified keysize and public key</td>
</tr>
<tr>
<td><code>EncryptAsync</code></td>
<td>Asynchronously encrypts a string using the specified keysize and public key</td>
</tr>
<tr>
<td><code>LegalKeySizes</code></td>
<td>Returns an array containing the supported key sizes</td>
</tr>
</tbody>
</table>

See Also

Reference
RSAMethods Class
Tungsten

W
RSAMethods

CreateKeyPair Method

Generates a public/private key pair

**Namespace:** W.Encryption  
**Assembly:** Tungsten.Encryption (in Tungsten.Encryption.dll)  
**Version:** 2.0.2

**Syntax**

```csharp
public static void CreateKeyPair(
    int keySize,
    out RSAParameters privateKey,
    out RSAParameters publicKey
)
```

**Parameters**

- **keySize**
  - Type: System.Int32
  - The key size to use when creating the public and private keys

- **privateKey**
  - Type: System.Security.Cryptography.RSAPublicKey
  - The generated private key

- **publicKey**
  - Type: System.Security.Cryptography.RSAPublicKey
  - The generated public key

**Return Value**

Type: A newly created PublicPrivateKeyPair containing the public and...
private keys

See Also

Reference
RSAMethods Class
W.Encryption Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\(W\)
RSAMethodsDecrypt Method

Decrypts a byte array which was previously encrypted with the Encrypt method

**Namespace:** W.Encryption  
**Assembly:** Tungsten.Encryption (in Tungsten.Encryption.dll) Version: 2.0.2

### Syntax

```csharp
public static byte[] Decrypt(
    byte[] bytes,
    RSAParameters key
)
```

### Parameters

- **bytes**  
  Type: System.Byte  
  The encrypted data

- **key**  
  Type: System.Security.Cryptography.RSAPParameters  
  The key to decrypt the data

### Return Value

- **Type:** Byte  
  A byte array containing the decrypted value

### See Also

- Reference: RSAMethods Class
Tungsten

\( W \)
RSAMethodsDecryptAsync Method

Asynchronously decrypts a string which was previously encrypted with the Encrypt method

**Namespace:** W.Encryption  
**Assembly:** Tungsten.Encryption (in Tungsten.Encryption.dll)  
**Version:** 2.0.2

## Syntax

```csharp
public static Task<byte[]> DecryptAsync(byte[] bytes, RSAParameters key)
```

### Parameters

- **bytes**
  - Type: `System.Byte`
  - The encrypted byte array

- **key**
  - Type: `System.Security.Cryptography.RSAParameters`
  - The key to decrypt the data

### Return Value

- Type: `Task<Byte>`
  - A byte array containing the decrypted value

## See Also
Reference
RSAMethods Class
W.Encryption Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
RSAMethodsEncrypt Method

Encrypts a byte array using the specified keysize and public key

Namespace: W.Encryption

Syntax

```csharp
public static byte[] Encrypt(
    byte[] bytes,
    RSAPrimitives key
)
```

Parameters

*bytes*  
Type: `SystemByte`  
The data to encrypt

*key*  
Type: `System.Security.Cryptography.RSAPrimitives`  
The key used to encrypt the data

Return Value

Type: `Byte`  
A byte array containing the encrypted data

See Also

Reference  
RSAMethods Class  
W.Encryption Namespace
Tungsten

$W$
RSAMethodsEncryptAsync Method

Asynchronously encrypts a string using the specified keysize and public key

**Namespace:** W.Encryption  
**Assembly:** Tungsten.Encryption (in Tungsten.Encryption.dll) Version: 2.0.2

### Syntax

```csharp
public static Task<byte[]> EncryptAsync(
    byte[] bytes,
    RSAPublicKeyParameters key
)
```

### Parameters

- **bytes**
  - Type: `SystemByte`  
  - The data to encrypt

- **key**
  - Type: `System.Security.CryptographyRSAPublicKeyParameters`  
  - The key to encrypt the data

### Return Value

- Type: `Task.Byte`  
  - A byte array containing the encrypted data

### See Also
Reference
RSAMethods Class
W.Encryption Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
RSAMethodsLegalKeySizes Method

Returns an array containing the supported key sizes

**Namespace:** W.Encryption  
**Assembly:** Tungsten.Encryption (in Tungsten.Encryption.dll)  
**Version:** 2.0.2

**Syntax**

```csharp
public static KeySizes[] LegalKeySizes()
```

**Return Value**
- **Type:** KeySizes  
- An array of supported key sizes

**See Also**

Reference  
RSAMethods Class  
W.Encryption Namespace

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
</tr>
</tbody>
</table>
# W.Firewall Namespace

[Missing <summary> documentation for "N:W.Firewall"]

## Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rules</td>
<td>Provides static methods to add, remove and check the existence of, Windows firewall rules</td>
</tr>
</tbody>
</table>

## Enumerations

<table>
<thead>
<tr>
<th>Enumeration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RulesEFirewallProfiles</td>
<td>The firewall profile type</td>
</tr>
<tr>
<td>RulesEFirewallRuleAction</td>
<td>Firewall rule actions</td>
</tr>
</tbody>
</table>

Copyright @ 2018 Jordan Duerksen
Tungsten

\[ W \]
Rules Class

Provides static methods to add, remove and check the existence of, Windows firewall rules

Inheritance Hierarchy

- System
  - Object
    - W.Firewall
      - Rules

Namespace: W.Firewall
Assembly: Tungsten.Firewall (in Tungsten.Firewall.dll) Version: 2.0.0

Syntax

```csharp
public static class Rules
```

The Rules type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>☀️ Add</td>
<td>Adds a rule to the firewall</td>
</tr>
<tr>
<td>☀️ Exists</td>
<td>Checks if a particular rule exists</td>
</tr>
<tr>
<td>☀️ Remove</td>
<td>Removes a firewall rule</td>
</tr>
</tbody>
</table>

See Also

Reference
Tungsten

W
Rules Methods

The Rules type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Adds a rule to the firewall</td>
</tr>
<tr>
<td>Exists</td>
<td>Checks if a particular rule exists</td>
</tr>
<tr>
<td>Remove</td>
<td>Removes a firewall rule</td>
</tr>
</tbody>
</table>

See Also

Reference

Rules Class
W.Firewall Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
RulesAdd Method

Adds a rule to the firewall

**Namespace:** W.Firewall  
**Assembly:** Tungsten.Firewall (in Tungsten.Firewall.dll) Version: 2.0.0

### Syntax

```csharp
public static void Add(
    string ruleName,
    string ruleGroup,
    int protocol = 6,
    string localPorts = "80",
    RulesEFirewallRuleAction action = RulesEF
    RulesEFirewallProfiles profiles = RulesEF
)
```

### Parameters

- **ruleName**
  - Type: System.String
  - The name of the rule to add

- **ruleGroup**
  - Type: System.String
  - The group under which the rule is added

- **protocol** *(Optional)*
  - Type: System.Int32
  - The desired rule protocol

- **localPorts** *(Optional)*
  - Type: System.String
  - The desired rule port

- **action** *(Optional)*
Type: `W.FirewallRules.EFirewallRuleAction`
The desired rule action, to allow or block communications

profiles (Optional)
Type: `W.FirewallRules.EFirewallProfiles`
The desired rule profile

See Also

Reference
Rules Class
W.Firewall Namespace

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
</tr>
</tbody>
</table>
RulesExists Method

Checks if a particular rule exists

**Namespace:** W.Firewall  
**Assembly:** Tungsten.Firewall (in Tungsten.Firewall.dll) Version: 2.0.0

### Syntax

```csharp
public static bool Exists(
    string ruleName
)
```

#### Parameters

`ruleName`  
Type: `System.String`  
The name of the rule to check

#### Return Value

Type: `Boolean`  
True if the rule exists, otherwise false

### See Also

**Reference**  
Rules Class  
W.Firewall Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
Rules

Remove Method

Removes a firewall rule

**Namespace:** W.Firewall

**Assembly:** Tungsten.Firewall (in Tungsten.Firewall.dll) Version: 2.0.0

### Syntax

```csharp
public static void Remove(
    string ruleName
)
```

### Parameters

**ruleName**

- **Type:** `System.String`
- The name of the rule to remove

### See Also

- Reference
- Rules Class
- W.Firewall Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
RulesEFirewallProfiles Enumeration

The firewall profile type

**Namespace:**  W.Firewall
**Assembly:**  Tungsten.Firewall (in Tungsten.Firewall.dll) Version: 2.0.0

### Syntax

```
public enum EFirewallProfiles
```

### Members

<table>
<thead>
<tr>
<th>Member name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Public</td>
<td>4</td>
<td>Public</td>
</tr>
<tr>
<td>Private</td>
<td>2</td>
<td>Private</td>
</tr>
<tr>
<td>Domain</td>
<td>1</td>
<td>Domain</td>
</tr>
<tr>
<td>All</td>
<td>2147483647</td>
<td>All</td>
</tr>
</tbody>
</table>

### See Also

Reference
W.Firewall Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
Rules\texttt{EFirewallRuleAction} Enumeration

Firewall rule actions

\textbf{Namespace: } \texttt{W.Firewall}  \\
\textbf{Assembly: } Tungsten.Firewall (in Tungsten.Firewall.dll) Version: 2.0.0

\section*{Syntax}

\begin{verbatim}
public enum EFirewallRuleAction

\end{verbatim}

\section*{Members}

\begin{table}[h]
\centering
\begin{tabular}{|c|c|p{6cm}|}
\hline
Member name & Value & Description                              \\
\hline
\texttt{Allowed} & 1     & Allow communications                      \\
\hline
\texttt{Block}   & 2     & Block communications                      \\
\hline
\end{tabular}
\end{table}

\section*{See Also}

Reference  \\
\texttt{W.Firewall Namespace}

\textcopyright @ 2018 Jordan Duerksen
Tungsten

W
W.InterProcess Namespace

[Missing <summary> documentation for "N:W.InterProcess"]

 Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CopyData</td>
<td>Sends and receives data via WM_COPYDATA</td>
</tr>
<tr>
<td>CopyDataCopyDataStruct</td>
<td>Helper class which converts byte arrays and COPYDATASTRUCTs</td>
</tr>
<tr>
<td>CopyDataTMessage</td>
<td>Sends and receives Generics via WM_COPYDATA</td>
</tr>
<tr>
<td>CopyDataLogger</td>
<td>Logs messages to the specified window via WM_COPYDATA messages</td>
</tr>
</tbody>
</table>

 Structures

<table>
<thead>
<tr>
<th>Structure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COPYDATASTRUCT</td>
<td></td>
</tr>
</tbody>
</table>

Copyright © 2018 Jordan Duerksen
Tungsten

W
CopyData Class

Sends and receives data via WM_COPYDATA

Inheritance Hierarchy

System \to System\_Marshal\_By\_Ref\_Object
System.Windows.Forms\to NativeWindow
W.InterProcess\to CopyData
W.InterProcess\to CopyData\_TMessage

Namespace: W.InterProcess
Assembly: Tungsten.InterProcess (in Tungsten.InterProcess.dll)
Version: 2.0.0

Syntax

```csharp
public class CopyData : NativeWindow
```

The CopyData type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CopyData(String)</td>
<td>Constructs a new CopyData instance which can only listen for messages</td>
</tr>
<tr>
<td>CopyData(PredicateString, Boolean)</td>
<td>Constructs a new CopyData instance</td>
</tr>
</tbody>
</table>
CopyData(IntPtr, Predicate<String>, Boolean) Constructs a new CopyData instance

CopyData(Form, Predicate<String>, Boolean) Constructs a new CopyData instance

## Top

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Handle</td>
<td>Gets the handle for this window. (Inherited from NativeWindow.)</td>
</tr>
<tr>
<td>TargetWindows</td>
<td>The windows which should receive messages</td>
</tr>
</tbody>
</table>

## Top

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AssignHandle</td>
<td>Assigns a handle to this window. (Inherited from NativeWindow)</td>
</tr>
<tr>
<td>CreateHandle</td>
<td>Creates a window and its handle with the specified creation parameters.</td>
</tr>
<tr>
<td>CreateObjRef</td>
<td>Creates an object that contains the relevant information required to generate a proxy used to communicate with a remote object. (Inherited from MarshalByRefObject.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DefWndProc</td>
<td>Invokes the default window procedure associated with this window. (Inherited from NativeWindow)</td>
</tr>
<tr>
<td>DestroyHandle</td>
<td>Destroys the window and its handle. (Inherited from NativeWindow)</td>
</tr>
<tr>
<td>Enable</td>
<td>Enables a window to receive WM_COPYDATA messages</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.     (Inherited from Object.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Releases the resources associated with this window. (Inherited from NativeWindow)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetLifetimeService</td>
<td>Retrieves the current lifetime service object that controls the lifetime policy for this instance. (Inherited from MarshalByRefObject.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>InitializeLifetimeService</td>
<td>Obtains a lifetime service object to control the lifetime policy for this instance.</td>
</tr>
<tr>
<td>Method Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current <code>Object</code>. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>MemberwiseClone(Boolean)</td>
<td>Creates a shallow copy of the current <code>MarshalByRefObject</code> object.</td>
</tr>
<tr>
<td>OnHandleChange</td>
<td>Specifies a notification method that is called when the handle for the window is changed. (Inherited from <code>NativeWindow</code>.)</td>
</tr>
<tr>
<td>OnThreadException</td>
<td>When overridden in a derived class, manages an unhandled thread exception.</td>
</tr>
<tr>
<td>OnWM_COPYDATA</td>
<td>Called when the window receives a WM_COPYDATA message.</td>
</tr>
<tr>
<td>RaiseBytesReceived</td>
<td>Raises the BytesReceived event.</td>
</tr>
<tr>
<td>RaiseError</td>
<td>Raises the Error event</td>
</tr>
<tr>
<td>RefreshTargets</td>
<td>Refreshes the list of target windows.</td>
</tr>
<tr>
<td>ReleaseHandle</td>
<td>Releases the handle associated with this window.</td>
</tr>
<tr>
<td>Send(Byte)</td>
<td>Sends a message via WM_COPYDATA</td>
</tr>
</tbody>
</table>
### Send(IntPtr, Byte, IntPtr)
- Sends a byte array from the source window to windows matching the filter predicate.

### Send(IntPtr, Byte, IntPtr, ActionWin32Exception)
- Sends a byte array from the source window to windows matching the filter predicate.

### Send(IntPtr, Byte, IntPtr, PredicateString, ActionWin32Exception)
- Sends a byte array from the source window to windows matching the filter predicate.

### ToString
- Returns a string that represents the current object.
  (Inherited from `Object`.)

### WndProc
- The window procedure.
  (Overrides `NativeWindow.WndProc(Message)`.)

### Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="https://icons.iconarchive.com/icons/urbanagencies/open-clipart-vectors/256/windows-1.png" alt="vrolet" /></td>
<td>BytesReceived</td>
</tr>
<tr>
<td><img src="https://icons.iconarchive.com/icons/urbanagencies/open-clipart-vectors/256/windows-1.png" alt="vrolet" /></td>
<td>Error</td>
</tr>
</tbody>
</table>

### Extension Methods

<p>| Name | Description |
|------|-------------|-------------|
|      |             |             |</p>
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AsTType</strong></td>
<td>Use Generic syntax for the as operator. (Defined by <a href="#">AsExtensions</a>.)</td>
</tr>
<tr>
<td><strong>InitializeProperties</strong></td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by <a href="#">PropertyHostExtensions</a>.)</td>
</tr>
<tr>
<td><strong>InLock(Action)</strong></td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by <a href="#">MonitorExtensions</a>.)</td>
</tr>
<tr>
<td><strong>InLockTType(FuncTType)</strong></td>
<td>Overloaded. Performs the function in a Monitor lock. (Defined by <a href="#">MonitorExtensions</a>.)</td>
</tr>
<tr>
<td><strong>InLockAsync(Action)</strong></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock. (Defined by <a href="#">MonitorExtensions</a>.)</td>
</tr>
<tr>
<td><strong>InLockAsyncTType(FuncTType)</strong></td>
<td>Overloaded. Asynchronously performs...</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>

See Also

Reference
W.InterProcess Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$\text{W}$
# CopyData Constructor

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CopyData(String)</td>
<td>Constructs a new CopyData instance which can only listen for messages</td>
</tr>
<tr>
<td>CopyData(PredicateString, Boolean)</td>
<td>Constructs a new CopyData instance</td>
</tr>
<tr>
<td>CopyData(IntPtr, PredicateString, Boolean)</td>
<td>Constructs a new CopyData instance</td>
</tr>
<tr>
<td>CopyData(Form, PredicateString, Boolean)</td>
<td>Constructs a new CopyData instance</td>
</tr>
</tbody>
</table>

See Also

Reference
- CopyData Class
- W.InterProcess Namespace

Copyright © 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
<th>W</th>
</tr>
</thead>
</table>

CopyData Constructor (String)

Constructs a new CopyData instance which can only listen for messages

**Namespace:** W.InterProcess  
**Assembly:** Tungsten.InterProcess (in Tungsten.InterProcess.dll)  
**Version:** 2.0.0

## Syntax

```csharp
public CopyData(string windowText)
```

### Parameters

- **windowText**  
  Type: System.String  
  The Window Text for the underlying NativeWindow

## See Also

- Reference  
  CopyData Class  
  CopyData Overload  
  W.InterProcess Namespace

---

Copyright © 2018 Jordan Duerksen
Tungsten

$^7\text{W}$
CopyData Constructor
(Predicate<String, Boolean>)

Constructs a new CopyData instance

**Namespace:** W.InterProcess
**Assembly:** Tungsten.InterProcess (in Tungsten.InterProcess.dll)
**Version:** 2.0.0

### Syntax

```csharp
public CopyData(
    Predicate<string> filter,
    bool findAll
)
```

### Parameters

**filter**
Type: `System.Predicate<string>`
The predicate used to filter target windows by Window Text

**findAll**
Type: `System.Boolean`
If True, multiple windows can be targeted, otherwise only the first window found will be targeted

### See Also

Reference
- CopyData Class
- CopyData Overload
- W.InterProcess Namespace
Tungsten

$W$
CopyData Constructor (IntPtr, Predicate<String>, Boolean)

Constructs a new CopyData instance

**Namespace:** W.InterProcess  
**Assembly:** Tungsten.InterProcess (in Tungsten.InterProcess.dll) 
**Version:** 2.0.0

## Syntax

```csharp
public CopyData(
    IntPtr hSourceWnd,
    Predicate<string> filter,
    bool findAll
)
```

### Parameters

**hSourceWnd**
- Type: SystemIntPtr
- The window handle which will send and receive messages

**filter**
- Type: SystemPredicateString
- The predicate used to filter target windows by Window Text

**findAll**
- Type: SystemBoolean
- If True, multiple windows can be targeted, otherwise only the first window found will be targeted

## See Also
Reference
CopyData Class
CopyData Overload
W.InterProcess Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
CopyData Constructor (Form, Predicate<String>, Boolean)

Constructs a new CopyData instance

**Namespace:** W.InterProcess  
**Assembly:** Tungsten.InterProcess (in Tungsten.InterProcess.dll)  
**Version:** 2.0.0

### Syntax

```csharp
public CopyData(  
    Form parent,  
    Predicate<string> filter,  
    bool findAll
)
```

### Parameters

**parent**
- **Type:** System.Windows.Forms.Form
- The Form which will send and receive messages

**filter**
- **Type:** System.Predicate<String>
- The predicate used to filter target windows by Window Text

**findAll**
- **Type:** System.Boolean
- If True, multiple windows can be targeted, otherwise only the first window found will be targeted

### See Also
Reference
CopyData Class
CopyData Overload
W.InterProcess Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
CopyData Properties

The CopyData type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Handle</td>
<td>Gets the handle for this window. (Inherited from NativeWindow.)</td>
</tr>
<tr>
<td>TargetWindows</td>
<td>The windows which should receive messages</td>
</tr>
</tbody>
</table>

See Also

Reference

CopyData Class
W.InterProcess Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
CopyDataTargetWindows Property

The windows which should receive messages

**Namespace:**  W.InterProcess  
**Assembly:**  Tungsten.InterProcess (in Tungsten.InterProcess.dll)  
**Version:**  2.0.0

▲ Syntax

```csharp
protected IntPtr[] TargetWindows { get; }
```

Property Value
Type: IntPtr

▲ See Also

Reference
CopyData Class  
W.InterProcess Namespace

Copyright © 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
</tr>
</tbody>
</table>
CopyData Methods

The `CopyData` type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AssignHandle</code></td>
<td>Assigns a handle to this window.</td>
</tr>
<tr>
<td><code>CreateHandle</code></td>
<td>Creates a window and its handle with the specified creation parameters.</td>
</tr>
<tr>
<td><code>CreateObjRef</code></td>
<td>Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object.</td>
</tr>
<tr>
<td><code>DefWndProc</code></td>
<td>Invokes the default window procedure associated with this window.</td>
</tr>
<tr>
<td><code>DestroyHandle</code></td>
<td>Destroys the window and its handle.</td>
</tr>
<tr>
<td><code>Enable</code></td>
<td>Enables a window to receive <code>WM_COPYDATA</code> messages.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Releases the resources associated with this window.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>NativeWindow</strong>.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>GetLifetimeService</strong></td>
<td>Retrieves the current lifetime service object that controls the lifetime</td>
</tr>
<tr>
<td></td>
<td>policy for this instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>MarshalByRefObject</strong>.)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>InitializeLifetimeService</strong></td>
<td>Obtains a lifetime service object to control the lifetime policy for this</td>
</tr>
<tr>
<td></td>
<td>instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>MarshalByRefObject</strong>.)</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current <strong>Object</strong>.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>MemberwiseClone(Boolean)</strong></td>
<td>Creates a shallow copy of the current <strong>MarshalByRefObject</strong>.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>MarshalByRefObject</strong>.)</td>
</tr>
<tr>
<td><strong>OnHandleChange</strong></td>
<td>Specifies a notification method.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>OnThreadException</td>
<td>When overridden in a derived class, manages an unhandled thread exception. (Inherited from NativeWindow)</td>
</tr>
<tr>
<td>OnWM_COPYDATA</td>
<td>Called when the window receives a WM_COPYDATA message.</td>
</tr>
<tr>
<td>RaiseBytesReceived</td>
<td>Raises the BytesReceived event.</td>
</tr>
<tr>
<td>RaiseError</td>
<td>Raises the Error event.</td>
</tr>
<tr>
<td>RefreshTargets</td>
<td>Refreshes the list of target windows.</td>
</tr>
<tr>
<td>ReleaseHandle</td>
<td>Releases the handle associated with this window. (Inherited from NativeWindow)</td>
</tr>
<tr>
<td>Send(Byte)</td>
<td>Sends a message via WM_COPYDATA.</td>
</tr>
<tr>
<td>Send(IntPtr, Byte, IntPtr)</td>
<td>Sends a byte array from the source window to windows matching the filter predicate.</td>
</tr>
<tr>
<td>Send(IntPtr, Byte, IntPtr, ActionWin32Exception)</td>
<td>Sends a byte array from the source window to windows matching the filter predicate.</td>
</tr>
<tr>
<td>Send(IntPtr, Byte, PredicateString, ActionWin32Exception)</td>
<td>Sends a byte array from the source window to windows matching the filter predicate.</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the window.</td>
</tr>
</tbody>
</table>
current object. (Inherited from Object.)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WndProc</td>
<td>The window procedure (Overrides NativeWindowWndProc)</td>
</tr>
</tbody>
</table>

### Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot; This method should be called in the constructor of any class which has IOwnedProperty members (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTTType(FuncTTType)</td>
<td>Overloaded. Performs the function in a Monitor lock</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>InLockAsync(Decor)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>
See Also

Reference
CopyData Class
W.InterProcess Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
CopyDataEnable Method

Enables a window to receive WM_COPYDATA messages

Namespace: W.InterProcess
Assembly: Tungsten.InterProcess (in Tungsten.InterProcess.dll)
Version: 2.0.0

Syntax

```csharp
public static Win32Exception Enable(
    IntPtr hWnd
)
```

Parameters

**hWnd**
Type: `System.IntPtr`
The handle of the window to enable

Return Value
Type: `Win32Exception`
Null upon success, otherwise a Win32Exception containing exception information

See Also

Reference
- CopyData Class
- W.InterProcess Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
CopyDataOnWM_COPYDATA Method

Called when the window receives a WM_COPYDATA message

**Namespace:** W.InterProcess  
**Assembly:** Tungsten.InterProcess (in Tungsten.InterProcess.dll)  
Version: 2.0.0

## Syntax

```csharp
protected virtual void OnWM_COPYDATA(
    COPYDATASTRUCT cds
)
```

### Parameters

- **cds**

  Type: W.InterProcessCOPYDATASTRUCT  
The COPYDATASTRUCT associated with the WM_COPYDATA windows message

## See Also

- **Reference**  
  CopyData Class  
  W.InterProcess Namespace

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
</tr>
</tbody>
</table>
CopyDataRaiseBytesReceived Method

Raises the BytesReceived event

**Namespace:** W.InterProcess  
**Assembly:** Tungsten.InterProcess (in Tungsten.InterProcess.dll)  
**Version:** 2.0.0

## Syntax

```csharp
protected void RaiseBytesReceived(
    byte[] bytes
)
```

### Parameters

*bytes*

- **Type:** `System.Byte`  
  - The bytes received

## See Also

- **Reference**  
  - CopyData Class  
  - W.InterProcess Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
CopyDataRaiseError Method

Raises the Error event

**Namespace:** W.InterProcess  
**Assembly:** Tungsten.InterProcess (in Tungsten.InterProcess.dll)  
Version: 2.0.0

### Syntax

```csharp
protected void RaiseError(
    Win32Exception e
)
```

### Parameters

- **e**  
  Type: `System.ComponentModel.Win32Exception`  
  The exception

### See Also

**Reference**  
- CopyData Class  
- W.InterProcess Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
CopyDataRefreshTargets Method

Refreshes the list of target windows

**Namespace:**  W.InterProcess  
**Assembly:**  Tungsten.InterProcess (in Tungsten.InterProcess.dll)  
**Version:**  2.0.0

**Syntax**

```c#
public void RefreshTargets()
```

**See Also**

Reference  
CopyData Class  
W.InterProcess Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
## CopyDataSend Method

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>✪ Send(Byte)</td>
<td>Sends a message via WM_COPYDATA</td>
</tr>
<tr>
<td>✪ S Send(IntPtr, Byte, IntPtr)</td>
<td>Sends a byte array from the source window to windows matching the filter predicate</td>
</tr>
<tr>
<td>✪ S Send(IntPtr, Byte, IntPtr, Action Win32Exception)</td>
<td>Sends a byte array from the source window to windows matching the filter predicate</td>
</tr>
<tr>
<td>✪ S Send(IntPtr, Byte, Predicate String, Action Win32Exception)</td>
<td>Sends a byte array from the source window to windows matching the filter predicate</td>
</tr>
</tbody>
</table>

**Top**

### See Also

- **Reference**
  - CopyData Class
  - W.InterProcess Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
CopyDataSend Method (Byte)

Sends a message via WM_COPYDATA

Namespace: W.InterProcess
Assembly: Tungsten.InterProcess (in Tungsten.InterProcess.dll)
Version: 2.0.0

Syntax

C#

```csharp
public void Send(
    byte[] message
)
```

Parameters

message
  Type: System.Byte
  The message to send

See Also

Reference
CopyData Class
Send Overload
W.InterProcess Namespace
Tungsten

$W$
CopyDataSend Method (IntPtr, Byte, IntPtr)

Sends a byte array from the source window to windows matching the filter predicate

**Namespace:** W.InterProcess  
**Assembly:** Tungsten.InterProcess (in Tungsten.InterProcess.dll)  
**Version:** 2.0.0

**Syntax**

```csharp
public static Win32Exception Send(  
    IntPtr hSourceWnd,  
    byte[] message,  
    IntPtr hTargetWnd  
)
```

**Parameters**

*hSourceWnd*
- Type: SystemIntPtr  
  The window which is sending the message

*message*
- Type: SystemByte  
  The array of bytes to send

*hTargetWnd*
- Type: SystemIntPtr  
  The receiving window

**Return Value**
- Type: Win32Exception
An exception if one occurs while sending the message

See Also

Reference

CopyData Class
Send Overload
W.InterProcess Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$\text{W}$
CopyDataSend Method (IntPtr, Byte, IntPtr, Action<Win32Exception>)

Sends a byte array from the source window to windows matching the filter predicate

Namespace: W.InterProcess
Assembly: Tungsten.InterProcess (in Tungsten.InterProcess.dll)
Version: 2.0.0

Syntax

```csharp
public static void Send(
    IntPtr hSourceWnd,
    byte[] message,
    IntPtr hTargetWnd,
    Action<Win32Exception> onError = null
)
```

Parameters

*hSourceWnd*
Type: SystemIntPtr
The window which is sending the message

*message*
Type: SystemByte
The array of bytes to send

*hTargetWnd*
Type: SystemIntPtr
The receiving window
**onError (Optional)**
Type: System.Action\Win32Exception
Called if an error occurs

### See Also

Reference
CopyData Class
Send Overload
W.InterProcess Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
CopyDataSend Method (IntPtr, Byte, Predicate<String>, Action<Win32Exception>)

Sends a byte array from the source window to windows matching the filter predicate.

Namespace: W.InterProcess
Assembly: Tungsten.InterProcess (in Tungsten.InterProcess.dll)
Version: 2.0.0

Syntax

```csharp
public static void Send(
    IntPtr hSourceWnd,
    byte[] message,
    Predicate<string> filter,
    Action<Win32Exception> onError = null
)
```

Parameters

*hSourceWnd*
Type: System.IntPtr
The window which is sending the message.

*message*
Type: System.Byte
The array of bytes to send.

*filter*
Type: System.Predicate<String>
Used to target one or more windows based on Window Text.
**onError (Optional)**
Type: `SystemActionWin32Exception`
Called if an error occurs

See Also

Reference
- CopyData Class
- Send Overload
- W.InterProcess Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

W
CopyDataWndProc Method

The window procedure

**Namespace:** W.InterProcess  
**Assembly:** Tungsten.InterProcess (in Tungsten.InterProcess.dll)  
**Version:** 2.0.0

### Syntax

```csharp
protected override void WndProc(
    ref Message m
)
```

### Parameters

- **m**  
  - Type: `System.Windows.Forms.Message`  
  - The message received

### See Also

**Reference**  
- CopyData Class  
- W.InterProcess Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
CopyData Events

The `CopyData` type exposes the following members.

## Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BytesReceived</td>
<td>Raised when a message has been received</td>
</tr>
<tr>
<td>Error</td>
<td>Raised when an error occurs while sending or receiving messages</td>
</tr>
</tbody>
</table>

See Also

Reference

- `CopyData Class`
- `W.InterProcess Namespace`

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
CopyDataBytesReceived Event

Raised when a message has been received

**Namespace:** W.InterProcess  
**Assembly:** Tungsten.InterProcess (in Tungsten.InterProcess.dll)  
**Version:** 2.0.0

### Syntax

```csharp
public event Action<byte[]> BytesReceived
```

**Value**  
**Type:** System.Action<Byte>

### See Also

**Reference**  
CopyData Class  
W.InterProcess Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
CopyDataError Event

Raised when an error occurs while sending or receiving messages

**Namespace:** W.InterProcess  
**Assembly:** Tungsten.InterProcess (in Tungsten.InterProcess.dll)  
**Version:** 2.0.0

### Syntax

```
public event Action<Win32Exception> Error
```

**Value**  
**Type:** SystemAction<Win32Exception>

### See Also

- Reference  
  CopyData Class  
  W.InterProcess Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
CopyDataCopyDataStruct Class

Helper class which converts byte arrays and COPYDATASTRUCTs

Inheritance Hierarchy

System \< Object \< W.InterProcess \< CopyDataCopyDataStruct

Namespace: W.InterProcess
Assembly: Tungsten.InterProcess (in Tungsten.InterProcess.dll)
Version: 2.0.0

Syntax

```csharp
protected class CopyDataStruct : IDisposable
```

The CopyDataCopyDataStruct type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CopyDataCopyDataStruct</td>
<td>Constructs a new CopyDataStruct from the specified byte array</td>
</tr>
</tbody>
</table>

Methods
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsBytes(Message)</td>
<td>Converts the contents of the COPYDATASTRUCT in the lParam of the specified message m to a byte array</td>
</tr>
<tr>
<td>AsBytes(COPYDATASTRUCT)</td>
<td>Converts the contents of the COPYDATASTRUCT to a byte array</td>
</tr>
<tr>
<td>Dispose</td>
<td>Disposes the CopyDataStruct and releases resource</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object. (Inherited from Object.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetCopyDataStruct</td>
<td>Get the COPYDATASTRUCT representing the byte array</td>
</tr>
</tbody>
</table>
### GetHashCode
Serves as the default hash function. (Inherited from Object.)

### GetType
Gets the Type of the current instance. (Inherited from Object.)

### MemberwiseClone
Creates a shallow copy of the current Object. (Inherited from Object.)

### ToString
Returns a string that represents the current object. (Inherited from Object.)

---

### Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>constructor of any class which has IOwnedProperty members</td>
<td>(Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock</td>
</tr>
<tr>
<td>InLockTTType(FuncTTType)</td>
<td>Overloaded. Performs the function in a Monitor lock</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock</td>
</tr>
<tr>
<td>InLockAsyncTTType(FuncTTType)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IPROPERTY</td>
</tr>
</tbody>
</table>

(Defined by PropertyHostExtensions.)
<table>
<thead>
<tr>
<th></th>
<th>Lock</th>
<th>Performs a Monitor lock (Defined by MonitorExtensions.)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets it's IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td></td>
<td>Unlock</td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>

**See Also**

Reference

W.InterProcess Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W

74
CopyDataCopyDataStruct Constructor

Constructs a new CopyDataStruct from the specified byte array

**Namespace:** W.InterProcess  
**Assembly:** Tungsten.InterProcess (in Tungsten.InterProcess.dll)  
**Version:** 2.0.0

### Syntax

```csharp
public CopyDataStruct(
    byte[] itemBytes
)
```

### Parameters

- **itemBytes**
  - **Type:** System.Byte
  - The byte array used to create the COPYDATASTRUCT

### See Also

- Reference
  - CopyDataCopyDataStruct Class
  - W.InterProcess Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

\( W \)
The CopyDataCopyDataStruct type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![s] AsBytes(Message)</td>
<td>Converts the contents of the COPYDATASTRUCT in the lParam of the specified message m to a byte array</td>
</tr>
<tr>
<td>![s] AsBytes(COPYDATASTRUCT)</td>
<td>Converts the contents of the COPYDATASTRUCT to a byte array</td>
</tr>
<tr>
<td>![s] Dispose</td>
<td>Disposes the CopyDataStruct and releases resource</td>
</tr>
<tr>
<td>![s] Equals</td>
<td>Determines whether the specified object is equal to the current object. (Inherited from Object.)</td>
</tr>
<tr>
<td>![s] Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations</td>
</tr>
</tbody>
</table>
before it is reclaimed by garbage collection. (Inherited from Object.)

- **GetCopyDataStruct**
  
  Get the COPYDATASTRUCT representing the byte array.

- **GetHashCode**
  
  Serves as the default hash function. (Inherited from Object.)

- **GetType**
  
  Gets the Type of the current instance. (Inherited from Object.)

- **MemberwiseClone**
  
  Creates a shallow copy of the current Object. (Inherited from Object.)

- **ToString**
  
  Returns a string that represents the current object. (Inherited from Object.)

---

Top

## Extension Methods
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot; This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTType(FuncTType)</td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTType(FuncTType)</td>
<td>Overloaded.</td>
</tr>
</tbody>
</table>


Asynchronously performs the action in a Monitor lock
(Defined by MonitorExtensions.)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty</td>
<td>(Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock</td>
<td>(Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets it's IsDirty flag to false</td>
<td>(Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock</td>
<td>(Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>

**See Also**

Reference
CopyDataCopyDataStruct Class
W.InterProcess Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
# CopyDataStructAsBytes Method

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AsBytes(Message)</code></td>
<td>Converts the contents of the COPYDATASTRUCT in the IParam of the specified message m to a byte array</td>
</tr>
<tr>
<td><code>AsBytes(COPYDATASTRUCT)</code></td>
<td>Converts the contents of the COPYDATASTRUCT to a byte array</td>
</tr>
</tbody>
</table>

## See Also

**Reference**
- CopyData
- CopyDataStruct Class
- W.InterProcess Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
CopyDataCopyDataStructAsBytes
Method (Message)

Converts the contents of the COPYDATASTRUCT in the lParam of the specified message \texttt{m} to a byte array

\textbf{Namespace:} W.InterProcess  
\textbf{Assembly:} Tungsten.InterProcess (in Tungsten.InterProcess.dll)  
\textbf{Version:} 2.0.0

\section*{Syntax}

\begin{codeblock}[C#]
public static byte[] AsBytes(
    Message \texttt{m}
)
\end{codeblock}

\section*{Parameters}

\textit{m}

Type: \texttt{System.Windows.Forms.Message}  
The windows message

\section*{Return Value}

Type: \texttt{Byte}  
A contents of the COPYDATASTRUCT

\section*{See Also}

Reference

CopyDataCopyDataStruct Class  
AsBytes Overload  
W.InterProcess Namespace
Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
</tr>
</tbody>
</table>
CopyDataCopyDataStructAsBytes Method (COPYDATASTRUCT)

Converts the contents of the COPYDATASTRUCT to a byte array

**Namespace:** W.InterProcess
**Assembly:** Tungsten.InterProcess (in Tungsten.InterProcess.dll)
**Version:** 2.0.0

**Syntax**

```csharp
public static byte[] AsBytes(
    COPYDATASTRUCT cds
)
```

**Parameters**

*cds*

Type: W.InterProcessCOPYDATASTRUCT
The COPYDATASTRUCT containing a byte array

**Return Value**

Type: Byte
A contents of the COPYDATASTRUCT

**See Also**

Reference
CopyDataCopyDataStruct Class
AsBytes Overload
W.InterProcess Namespace
Tungsten

$\text{W}$
CopyDataCopyDataStructDispose

Method

Disposes the CopyDataStruct and releases resource

**Namespace:** W.InterProcess  
**Assembly:** Tungsten.InterProcess (in Tungsten.InterProcess.dll)  
**Version:** 2.0.0

▲ **Syntax**

```csharp
public void Dispose()
```

Implements  
IDisposable

▲ **See Also**

Reference  
CopyDataCopyDataStruct Class  
W.InterProcess Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
CopyDataCopyDataStructGetCopyDataStruct Method

Get the COPYDATASTRUCT representing the byte array

**Namespace:** W.InterProcess  
**Assembly:** Tungsten.InterProcess (in Tungsten.InterProcess.dll)  
**Version:** 2.0.0

### Syntax

```csharp
public COPYDATASTRUCT GetCopyDataStruct()
```

### Return Value

**Type:** COPYDATASTRUCT  
The COPYDATASTRUCT representing the byte array

### See Also

**Reference**  
CopyDataCopyDataStruct Class  
W.InterProcess Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
CopyDataTMessage Class

Sends and receives Generics via WM_COPYDATA

Inheritance Hierarchy

SystemObject System.MarshalByRefObject
  System.Windows.Forms.NativeWindow
  W.InterProcess.CopyData
  W.InterProcess.CopyDataTMessage

Namespace: W.InterProcess
Assembly: Tungsten.InterProcess (in Tungsten.InterProcess.dll)
Version: 2.0.0

Syntax

```csharp
public sealed class CopyData<TMessage> : CopyData:
```

Type Parameters

TMessage

[Missing <typeparam name="TMessage"/> documentation for
"T:W.InterProcess.CopyData`1"]

The CopyDataTMessage type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CopyDataTMessage(String)</td>
<td>Constructs a new</td>
</tr>
</tbody>
</table>
CopyData instance which can only listen for messages

<table>
<thead>
<tr>
<th>CopyDataTMessage(PredicateString, Boolean)</th>
<th>Constructs a new CopyData instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>CopyDataTMessage(IntPtr, PredicateString, Boolean)</td>
<td>Constructs a new CopyData instance</td>
</tr>
<tr>
<td>CopyDataTMessage(Form, PredicateString, Boolean)</td>
<td>Constructs a new CopyData instance</td>
</tr>
</tbody>
</table>

Top

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Handle</td>
<td>Gets the handle for this window. (Inherited from NativeWindow.)</td>
</tr>
</tbody>
</table>

Top

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AssignHandle</td>
<td>Assigns a handle to this window.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CreateHandle</td>
<td>Creates a window and its handle with the specified creation parameters.</td>
</tr>
<tr>
<td>CreateObjRef</td>
<td>Creates an object that contains all the relevant information required to</td>
</tr>
<tr>
<td></td>
<td>generate a proxy used to communicate with a remote object.</td>
</tr>
<tr>
<td>DefWndProc</td>
<td>Invokes the default window procedure associated with this window.</td>
</tr>
<tr>
<td>DestroyHandle</td>
<td>Destroys the window and its handle.</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>hash function.</td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>GetLifetimeService</td>
<td>Retrieves the current lifetime service object that controls the lifetime policy for this instance. (Inherited from MarshalByRefObject.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>InitializeLifetimeService</td>
<td>Obtains a lifetime service object to control the lifetime policy for this instance. (Inherited from MarshalByRefObject.)</td>
</tr>
<tr>
<td>RefreshTargets</td>
<td>Refreshes the list of target windows (Inherited from CopyData.)</td>
</tr>
<tr>
<td>ReleaseHandle</td>
<td>Releases the handle associated with this window. (Inherited from NativeWindow.)</td>
</tr>
<tr>
<td>Send(Byte)</td>
<td>Sends a message via WM_COPYDATA (Inherited from CopyData.)</td>
</tr>
</tbody>
</table>
Send(TMessage) Sends a message via WM_COPYDATA

ToString Returns a string that represents the current object. (Inherited from Object.)

## Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BytesReceived</td>
<td>Raised when a message has been received (Inherited from CopyData.)</td>
</tr>
<tr>
<td>Error</td>
<td>Raised when an error occurs while sending or receiving messages (Inherited from CopyData.)</td>
</tr>
<tr>
<td>MessageReceived</td>
<td>Raised when a message has been received</td>
</tr>
</tbody>
</table>

## Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and</td>
</tr>
</tbody>
</table>
properties of "owner" and sets the member's Owner property to "owner". This method should be called in the constructor of any class which has IOwnedProperty members (Defined by PropertyHostExtensions.)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTTType(FuncTTType)</td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTTType(FuncTTType)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of</td>
</tr>
<tr>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>IProperty</td>
<td>(Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>

See Also

Reference

W.InterProcess Namespace
Tungsten

$W$
## CopyData TMessage Constructor

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CopyDataTMessage(String)</td>
<td>Constructs a new CopyData instance which can only listen for messages</td>
</tr>
<tr>
<td>CopyDataTMessage(PredicateString, Boolean)</td>
<td>Constructs a new CopyData instance</td>
</tr>
<tr>
<td>CopyDataTMessage(IntPtr, PredicateString, Boolean)</td>
<td>Constructs a new CopyData instance</td>
</tr>
<tr>
<td>CopyDataTMessage(Form, PredicateString, Boolean)</td>
<td>Constructs a new CopyData instance</td>
</tr>
</tbody>
</table>

### See Also

Reference
CopyDataTMessage Class
W.InterProcess Namespace

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
<th>W</th>
</tr>
</thead>
</table>

CopyData\textit{TMessage} Constructor (String)

Constructs a new CopyData instance which can only listen for messages

**Namespace:** W.InterProcess  
**Assembly:** Tungsten.InterProcess (in Tungsten.InterProcess.dll)  
**Version:** 2.0.0

### Syntax

```csharp
public CopyData(
    string windowText
)
```

### Parameters

\textit{windowText}  
Type: System\textit{String}  
The Window Text for the underlying Native\textit{Window}

### See Also

**Reference**  
CopyData\textit{TMessage} Class  
CopyData\textit{TMessage} Overload  
W.InterProcess Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
CopyDataTMessage Constructor (Predicate<String, Boolean>)

Constructs a new CopyData instance

**Namespace**: W.InterProcess  
**Assembly**: Tungsten.InterProcess (in Tungsten.InterProcess.dll)  
**Version**: 2.0.0

#### Syntax

```csharp
public CopyData(
    Predicate<string> filter,
    bool findAll
)
```

#### Parameters

- **filter**
  - Type: `System.Predicate<String>`
  - The predicate used to filter target windows by Window Text

- **findAll**
  - Type: `System.Boolean`
  - If True, multiple windows can be targeted, otherwise only the first window found will be targeted

#### See Also

- Reference  
  - CopyDataTMessage Class  
  - CopyDataTMessage Overload  
  - W.InterProcess Namespace
Tungsten

\[ W \]
CopyDataTMessage Constructor
(IntPtr, Predicate<String>, Boolean)

Constructs a new CopyData instance

**Namespace:** W.InterProcess

**Assembly:** Tungsten.InterProcess (in Tungsten.InterProcess.dll)

**Version:** 2.0.0

### Syntax

```csharp
public CopyData(
    IntPtr hSourceWnd,
    Predicate<string> filter,
    bool findAll
)
```

### Parameters

**hSourceWnd**
- Type: SystemIntPtr
- The window handle which will send and receive messages

**filter**
- Type: SystemPredicateString
- The predicate used to filter target windows by Window Text

**findAll**
- Type: SystemBoolean
- If True, multiple windows can be targeted, otherwise only the first window found will be targeted

### See Also
Reference
CopyDataTMessage Class
CopyDataTMessage Overload
W.InterProcess Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
CopyData\textit{TMessage} Constructor (Form, Predicate\textit{String}, Boolean)

Constructs a new CopyData instance

\textbf{Namespace:} W.InterProcess  
\textbf{Assembly:} Tungsten.InterProcess (in Tungsten.InterProcess.dll)  
\textbf{Version:} 2.0.0

\section*{Syntax}

\begin{verbatim}
public CopyData(
    Form parent,
    Predicate<string> filter,
    bool findAll
)
\end{verbatim}

\section*{Parameters}

\textit{parent}  
Type: System.Windows.Forms.Form  
The form which will send and receive messages

\textit{filter}  
Type: System.Predicate<String>  
The predicate used to filter target windows by Window Text

\textit{findAll}  
Type: System.Boolean  
If True, multiple windows can be targeted, otherwise only the first window found will be targeted

\section*{See Also}
Reference
CopyDataTMessage Class
CopyDataTMessage Overload
W.InterProcess Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
CopyDataTMessage Properties

The CopyDataTMessage generic type exposes the following members.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Handle</td>
<td>Gets the handle for this window. (Inherited from NativeWindow.)</td>
</tr>
</tbody>
</table>

See Also

Reference
CopyDataTMessage Class
W.InterProcess Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

W
CopyData\textit{TMessage} Methods

The \textit{CopyDataTMessage} generic type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AssignHandle</td>
<td>Assigns a handle to this window. (Inherited from NativeWindow.)</td>
</tr>
<tr>
<td>CreateHandle</td>
<td>Creates a window and its handle with the specified creation parameters. (Inherited from NativeWindow.)</td>
</tr>
<tr>
<td>CreateObjRef</td>
<td>Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object. (Inherited from MarshalByRefObject.)</td>
</tr>
<tr>
<td>DefWndProc</td>
<td>Invokes the default window procedure associated with this window. (Inherited from NativeWindow.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DestroyHandle</td>
<td>Destroys the window and its handle.</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td>GetLifetimeService</td>
<td>Retrieves the current lifetime service object that controls the lifetime policy for this instance.</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td>InitializeLifetimeService</td>
<td>Obtains a lifetime service object to control the lifetime policy for this instance.</td>
</tr>
<tr>
<td>RefreshTargets</td>
<td>Refreshes the list of target windows (Inherited from CopyData.)</td>
</tr>
</tbody>
</table>
**ReleaseHandle**
Releases the handle associated with this window. (Inherited from NativeWindow.)

**Send(Byte)**
Sends a message via WM_COPYDATA (Inherited from CopyData.)

**Send(TMessage)**
Sends a message via WM_COPYDATA

**ToString**
Returns a string that represents the current object. (Inherited from Object.)

## Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![ ] AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>![ ] InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot; This method should be called in the constructor of any class which has IOwnedProperty members</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTTType(FuncTTType)</td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTTType(FuncTTType)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>

**See Also**

Reference
- CopyDataTMessage Class
- W.InterProcess Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$^\text{W}$
CopyData TMessage Send Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Send(Byte)</td>
<td>Sends a message via WM_COPYDATA (Inherited from CopyData.)</td>
</tr>
<tr>
<td>Send(TMessage)</td>
<td>Sends a message via WM_COPYDATA</td>
</tr>
</tbody>
</table>

See Also

Reference
CopyData TMessage Class
W.InterProcess Namespace
Tungsten

$W$
CopyData\textit{TMessage} \textbf{Send} Method (\textit{TMessage})

Sends a message via WM\_COPYDATA

\textbf{Namespace:} W.InterProcess  
\textbf{Assembly:} Tungsten.InterProcess (in Tungsten.InterProcess.dll)  
\textbf{Version:} 2.0.0

\section*{Syntax}

\begin{verbatim}
public void Send(
    TMessage message
)
\end{verbatim}

\section*{Parameters}

\textit{message}  
Type: \textit{TMessage}  
The message to send

\section*{See Also}

Reference  
CopyData\textit{TMessage} Class  
Send Overload  
W.InterProcess Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
CopyDataTMessage Events

The CopyDataTMessage generic type exposes the following members.

## Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BytesReceived</td>
<td>Raised when a message has been received (Inherited from CopyData.)</td>
</tr>
<tr>
<td>Error</td>
<td>Raised when an error occurs while sending or receiving messages (Inherited from CopyData.)</td>
</tr>
<tr>
<td>MessageReceived</td>
<td>Raised when a message has been received</td>
</tr>
</tbody>
</table>

See Also

Reference
- CopyDataTMessage Class
- W.InterProcess Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
CopyDataTMessageMessageReceived Event

Raised when a message has been received

**Namespace:**  W.InterProcess  
**Assembly:**  Tungsten.InterProcess (in Tungsten.InterProcess.dll)  
**Version:**  2.0.0

### Syntax

```csharp
public event Action<TMessage> MessageReceived
```

### Value

**Type:**  SystemAction<TMessage>

### See Also

**Reference**  
CopyDataTMessage Class  
W.InterProcess Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
CopyDataLogger Class

Logs messages to the specified window via WM_COPYDATA messages

Inheritance Hierarchy

```
SystemObject  W.InterProcessCopyDataLogger
```

Namespace: W.InterProcess
Assembly: Tungsten.InterProcess (in Tungsten.InterProcess.dll)
Version: 2.0.0

Syntax

```c#
public static class CopyDataLogger
```

The CopyDataLogger type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LogTheMessage</td>
<td>Log a message to the specified window via WM_COPYDATA messaging</td>
</tr>
</tbody>
</table>

Examples

```
Log.LogTheMessage += (category, message) =>
```
true, category, message);

See Also

Reference
W.InterProcess Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

\( W \)
CopyDataLogger Methods

The CopyDataLogger type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LogTheMessage</td>
<td>Log a message to the specified window via WM_COPYDATA messaging</td>
</tr>
</tbody>
</table>

See Also

Reference
CopyDataLogger Class
W.InterProcess Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
CopyDataLoggerLogTheMessage Method

Log a message to the specified window via WM_COPYDATA messaging

**Namespace:** W.InterProcess  
**Assembly:** Tungsten.InterProcess (in Tungsten.InterProcess.dll)  
**Version:** 2.0.0

### Syntax

```csharp
public static void LogTheMessage(
    string windowCaption,
    string message
)
```

### Parameters

- **windowCaption**  
  Type: System.String  

- **message**  
  Type: System.String  

### See Also

Reference  
CopyDataLogger Class
Tungsten

$W$
COPYDATASTRUCT Structure

[Missing <summary> documentation for "T:W.InterProcess.COPYDATASTRUCT"]

Namespace: W.InterProcess
Assembly: Tungsten.InterProcess (in Tungsten.InterProcess.dll)
Version: 2.0.0

Syntax

```csharp
public struct COPYDATASTRUCT
```

The COPYDATASTRUCT type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Indicates whether this instance and a specified object are equal. (Inherited from ValueType.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Returns the hash code for this instance. (Inherited from ValueType.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns the fully qualified type name of this instance. (Inherited from ValueType.)</td>
</tr>
</tbody>
</table>
## Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cbData</td>
<td></td>
</tr>
<tr>
<td>dwData</td>
<td></td>
</tr>
<tr>
<td>lpData</td>
<td></td>
</tr>
</tbody>
</table>

## Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>InLockTTType(Func TTType)</code></td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><code>InLockAsync(Action)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><code>InLockAsyncTTType(Func TTType)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><code>IsDirty</code></td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td><code>Lock</code></td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><code>MarkAsClean</code></td>
<td>Scans each field and property of type IProperty and sets it’s IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
</tbody>
</table>
Unlock

Performs a Monitor unlock
(Defined by MonitorExtensions.)

See Also

Reference
W.InterProcess Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
COPYDATASTRUCT Methods

The COPYDATASTRUCT type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Indicates whether this instance and a specified object are equal. (Inherited from ValueType.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Returns the hash code for this instance. (Inherited from ValueType.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns the fully qualified type name of this instance. (Inherited from ValueType.)</td>
</tr>
</tbody>
</table>

Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
</tbody>
</table>
InitializeProperties

Scans the fields and properties of "owner" and sets the member's Owner property to "owner". This method should be called in the constructor of any class which has IOwnedProperty members (Defined by PropertyHostExtensions.)

InLock(Action)

Overloaded. Performs the action in a Monitor lock (Defined by MonitorExtensions.)

InLockTTType(FuncTTType)

Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.)

InLockAsync(Action)

Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)

InLockAsyncTTType(FuncTTType)

Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)

IsDirty

Scans the IsDirty value of
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>

See Also

Reference
COPYDATASTRUCT Structure
W.InterProcess Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
COPYDATASTRUCT Fields

The COPYDATASTRUCT type exposes the following members.

Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cbData</td>
<td></td>
</tr>
<tr>
<td>dwData</td>
<td></td>
</tr>
<tr>
<td>lpData</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Reference
COPYDATASTRUCT Structure
W.InterProcess Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
COPYDATASTRUCT cbData Field

[Missing <summary> documentation for "F:W.InterProcess.COPYDATASTRUCT.cbData"]

Namespace:  W.InterProcess
Assembly:  Tungsten.InterProcess (in Tungsten.InterProcess.dll)
Version:  2.0.0

● Syntax

C#

```csharp
public int cbData
```

Field Value
Type:  Int32

● See Also

Reference
COPYDATASTRUCT Structure
W.InterProcess Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
COPYDATASTRUCT.dwData Field

[Missing <summary> documentation for "F:W.InterProcess.COPYDATASTRUCT.dwData"]

Namespace: W.InterProcess
Assembly: Tungsten.InterProcess (in Tungsten.InterProcess.dll)
Version: 2.0.0

➤ Syntax

C#

```csharp
public IntPtr dwData
```

Field Value
Type: IntPtr

➤ See Also

Reference
COPYDATASTRUCT Structure
W.InterProcess Namespace

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>$^\text{W}$</td>
</tr>
</tbody>
</table>


COPYDATASTRUCT lpData Field

[Missing <summary> documentation for "F:W.InterProcess.COPYDATASTRUCT.lpData"]

**Namespace:** W.InterProcess  
**Assembly:** Tungsten.InterProcess (in Tungsten.InterProcess.dll)  
**Version:** 2.0.0

### Syntax

```csharp
public IntPtr lpData
```

**Field Value**  
**Type:** IntPtr

### See Also

Reference  
COPYDATASTRUCT Structure  
W.InterProcess Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
## W.IO.Pipes Namespace

[Missing <summary> documentation for "N:W.IO.Pipes"]

### Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pipe</td>
<td>The untyped base Pipe class</td>
</tr>
<tr>
<td>PipeTMessage</td>
<td>The base generic Pipe class</td>
</tr>
<tr>
<td>PipeClient</td>
<td>A pipe client. This class sends and receives byte arrays.</td>
</tr>
<tr>
<td>PipeClientTMessage</td>
<td>The generic version of PipeClient. This class expects all messages to be of the specified type.</td>
</tr>
<tr>
<td>PipeHost</td>
<td>Hosts a number of PipeServers. This class sends and receives byte arrays.</td>
</tr>
<tr>
<td>PipeHostTMessage</td>
<td>The generic version of PipeHost. This class expects all messages to be of the specified type.</td>
</tr>
<tr>
<td>Class</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>PipeLogger</strong></td>
<td>Sends log messages to a remote server via a named pipe</td>
</tr>
<tr>
<td><strong>PipeReadWriteExtensions</strong></td>
<td>Read/Write functionality for Pipe</td>
</tr>
<tr>
<td><strong>PipeServer</strong></td>
<td>A Pipe server. This class sends and receives only byte arrays.</td>
</tr>
<tr>
<td><strong>PipeServerTMessage</strong></td>
<td>The generic version of PipeServer. This class expects all messages to be of the specified type.</td>
</tr>
</tbody>
</table>

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
Pipe Class

The untyped base Pipe class

Inheritance Hierarchy

- `System.Object`
- `W.IO.PipesPipe`
- `W.IO.PipesPipeTMessage`

Namespace: `W.IO.Pipes`
Assembly: `Tungsten.IO.Pipes` (in `Tungsten.IO.Pipes.dll`) Version: 2.0.3

Syntax

```
public abstract class Pipe : IDisposable
```

The `Pipe` type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Pipe]</td>
<td>Initializes a new instance of the <code>Pipe</code> class</td>
</tr>
</tbody>
</table>

Top

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Id]</td>
<td>A unique id for this Pipe</td>
</tr>
</tbody>
</table>
**Top**

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispose</td>
<td>Disconnects and disposes the pipe</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>Listen</td>
<td>Continuously waits for data from the pipe. The</td>
</tr>
</tbody>
</table>
BytesReceived event is raised when data arrives, then immediately waits for more data.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>OnDispose</strong></td>
<td>Disconnects and disposes the pipe</td>
</tr>
<tr>
<td><strong>OnListen</strong></td>
<td>Continuously waits for data from the pipe. The BytesReceived event is raised when data arrives, then immediately waits for more data.</td>
</tr>
<tr>
<td><strong>OnStopListening</strong></td>
<td>After the next bytes received, stops waiting for data</td>
</tr>
<tr>
<td><strong>RaiseDisconnection</strong></td>
<td>Raises the Disconnected event. Pass in an exception if desired.</td>
</tr>
<tr>
<td><strong>StopListening</strong></td>
<td>After the next bytes received, stops waiting for data</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Returns a string that represents the current object. (Inherited from Object.)</td>
</tr>
</tbody>
</table>
# Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>✨ Disconnected</td>
<td>Raised when the pipe has disconnected</td>
</tr>
</tbody>
</table>

# Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>✨ _shouldListen</td>
<td></td>
</tr>
</tbody>
</table>

# Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>✨ AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>✨ InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLock&lt;TType&gt;(Func&lt;TType&gt;)</td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync&lt;TType&gt;(Func&lt;TType&gt;)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to...</td>
</tr>
</tbody>
</table>
false
(Defined by PropertyHostExtensions.)

Unlock
Performs a Monitor unlock
(Defined by MonitorExtensions.)

See Also

Reference
W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
Pipe Constructor

Initializes a new instance of the Pipe class

Namespace: W.IO.Pipes
Assembly: Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll) Version: 2.0.3

Syntax

```c#
protected Pipe()
```

See Also

Reference
Pipe Class
W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
| Tungsten | W |
Pipe Properties

The **Pipe** type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Pipe" /> Id</td>
<td>A unique id for this Pipe</td>
</tr>
<tr>
<td><img src="image" alt="Pipe" /> InBufferSize</td>
<td></td>
</tr>
<tr>
<td><img src="image" alt="Pipe" /> OutBufferSize</td>
<td></td>
</tr>
<tr>
<td><img src="image" alt="Pipe" /> Stream</td>
<td>The PipeStream on which to send and receive data</td>
</tr>
</tbody>
</table>

**See Also**

**Reference**
- Pipe Class
- W.IO.Pipes Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
PipeId Property

A unique id for this Pipe

Namespace: W.IO.Pipes
Assembly: Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll) Version: 2.0.3

Syntax

```csharp
public string Id { get; }
```

Property Value
Type: String

See Also

Reference
Pipe Class
W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
PipeInBufferSize Property


Namespace:  W.IO.Pipes
Assembly:  Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll) Version: 2.0.3

Syntax

```csharp
public static Nullable<int> InBufferTime { get; }
```

Property Value
Type:  NullableInt32

See Also

Reference
Pipe Class
W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
PipeOutBufferSize Property


Namespace:  W.IO.Pipes  
Assembly:  Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll)  Version:  2.0.3

Syntax

```csharp
public static Nullable<int> OutBufferSize { get; }
```

Property Value  
Type:  NullableInt32

See Also

Reference  
Pipe Class  
W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
PipeStream Property

The PipeStream on which to send and receive data

**Namespace:**  W.IO.Pipes  
**Assembly:**  Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll)  
**Version:**  2.0.3

### Syntax

```csharp
public PipeStream Stream { get; protected set; }
```

### Property Value

- **Type:**  PipeStream

### See Also

- **Reference**
  - Pipe Class
  - W.IO.Pipes Namespace

---

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
The **Pipe** type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispose</td>
<td>Disconnects and disposes the pipe</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td><em>(Inherited from Object.)</em></td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. <em>(Inherited from Object.)</em></td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. <em>(Inherited from Object.)</em></td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. <em>(Inherited from Object.)</em></td>
</tr>
<tr>
<td>Listen</td>
<td>Continuously waits for data from the pipe. The BytesReceived event is raised when data arrives,</td>
</tr>
</tbody>
</table>
then immediately waits for more data.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current <strong>Object</strong>. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>OnDispose</strong></td>
<td>Disconnects and disposes the pipe</td>
</tr>
<tr>
<td><strong>OnListen</strong></td>
<td>Continuously waits for data from the pipe. The BytesReceived event is raised when data arrives, then immediately waits for more data.</td>
</tr>
<tr>
<td><strong>OnStopListening</strong></td>
<td>After the next bytes received, stops waiting for data</td>
</tr>
<tr>
<td><strong>RaiseDisconnection</strong></td>
<td>Raises the Disconnected event. Pass in an exception if desired.</td>
</tr>
<tr>
<td><strong>StopListening</strong></td>
<td>After the next bytes received, stops waiting for data</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Returns a string that represents the current object. (Inherited from <strong>Object</strong>.)</td>
</tr>
</tbody>
</table>

**Extension Methods**
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTTType(FuncTTType)</td>
<td>Overloaded. Performs the function in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTTType(FuncTTType)</td>
<td>Overloaded.</td>
</tr>
</tbody>
</table>
Asynchronously performs the action in a Monitor lock
(Defined by MonitorExtensions.)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty</td>
</tr>
<tr>
<td></td>
<td>(Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock</td>
</tr>
<tr>
<td></td>
<td>(Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets it's IsDirty flag to false</td>
</tr>
<tr>
<td></td>
<td>(Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock</td>
</tr>
<tr>
<td></td>
<td>(Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>

See Also

Reference
Pipe Class
W.IO.Pipes Namespace
Tungsten

\( W \)
PipeDispose Method

Disconnects and disposes the pipe

**Namespace:** W.IO.Pipes  
**Assembly:** Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll)  
**Version:** 2.0.3

### Syntax

```csharp
public void Dispose()
```

Implements  
IDisposableDispose

### See Also

Reference  
Pipe Class  
W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
PipeListen Method

Continuously waits for data from the pipe. The BytesReceived event is raised when data arrives, then immediately waits for more data.

Namespace: W.IO.Pipes
Assembly: Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll) Version: 2.0.3

Syntax

```csharp
public void Listen()
```

See Also

Reference
Pipe Class
W.IO.Pipes Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
PipeOnDispose Method

Disconnects and disposes the pipe

**Namespace:** W.IO.Pipes  
**Assembly:** Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll)  
**Version:** 2.0.3

### Syntax

C#  
```csharp
protected virtual void OnDispose()
```

### See Also

Reference  
Pipe Class  
W.IO.Pipes Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

\( W \)
PipeOnListen Method

Continuously waits for data from the pipe. The BytesReceived event is raised when data arrives, then immediately waits for more data.

Namespace:  W.IO.Pipes
Assembly:  Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll) Version: 2.0.3

⚠ Syntax

```c#
protected virtual void OnListen()
```

⚠ See Also

Reference
Pipe Class
W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
<th>W</th>
</tr>
</thead>
</table>

PipeOnStopListening Method

After the next bytes received, stops waiting for data

**Namespace:** W.IO.Pipes  
**Assembly:** Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll)  
**Version:** 2.0.3

### Syntax

```c#
protected virtual void OnStopListening()
```

### See Also

Reference  
Pipe Class  
W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
PipeRaiseDisconnection Method

Raises the Disconnected event. Pass in an exception if desired.

**Namespace:** W.IO.Pipes
**Assembly:** Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll) Version: 2.0.3

### Syntax

```csharp
protected void RaiseDisconnection(
    Pipe sender,
    Exception e
)
```

### Parameters

**sender**
Type: W.IO.PipesPipe
A reference to the caller

**e**
Type: SystemException
An exception if one was captured

### See Also

Reference
Pipe Class
W.IO.Pipes Namespace

Copyright © 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
</tr>
</tbody>
</table>
PipeStopListening Method

After the next bytes received, stops waiting for data

**Namespace:** W.IO.Pipes
**Assembly:** Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll) Version: 2.0.3

---

**Syntax**

```csharp
public void StopListening()
```

---

**See Also**

Reference
Pipe Class
W.IO.Pipes Namespace

---

Copyright @ 2018 Jordan Duerksen
Tungsten

$W^+$
Pipe Events

The Pipe type exposes the following members.

### Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="no_icon" alt="Disconnected" /></td>
<td>Raised when the pipe has disconnected</td>
</tr>
</tbody>
</table>

See Also

Reference

Pipe Class

W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
PipeDisconnected Event

Raised when the pipe has disconnected

**Namespace:** W.IO.Pipes  
**Assembly:** Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll)  
**Version:** 2.0.3

### Syntax

```csharp
public event Action<Pipe, Exception> Disconnected;
```

**Value**
Type: SystemActionPipe, Exception

### See Also

Reference
Pipe Class  
W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
Pipe Fields

The Pipe type exposes the following members.

Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_shouldListen</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Reference
Pipe Class
W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
Pipe_shouldListen Field

[Missing <summary> documentation for "F:W.IO.Pipes.Pipe._shouldListen"]

Namespace:  W.IO.Pipes
Assembly:  Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll) Version: 2.0.3

Syntax

```csharp
protected bool _shouldListen
```

Field Value
Type: Boolean

See Also

Reference
Pipe Class
W.IO.Pipes Namespace
Tungsten

W
PipeTMessage Class

The base generic Pipe class

Inheritance Hierarchy

- System
  - Object
  - W.IO.PipesPipe
    - W.IO.PipesPipeTMessage
      - W.IO.PipesPipeClientTMessage
      - W.IO.PipesPipeServerTMessage

Namespace: W.IO.Pipes
Assembly: Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll) Version: 2.0.3

Syntax

```csharp
public abstract class Pipe<TMessage> : Pipe
```

Type Parameters

- `TMessage`
  
  [Missing <typeparam name="TMessage"/> documentation for "T:W.IO.Pipes.Pipe`1"]

The PipeTMessage type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PipeTMessage</td>
<td>Initializes a new instance of the PipeTMessage class</td>
</tr>
</tbody>
</table>
### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Image]</td>
<td><strong>Id</strong></td>
</tr>
<tr>
<td>![Image]</td>
<td><strong>Stream</strong></td>
</tr>
</tbody>
</table>

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Image]</td>
<td><strong>Dispose</strong></td>
</tr>
<tr>
<td>![Image]</td>
<td><strong>Equals</strong></td>
</tr>
<tr>
<td>![Image]</td>
<td><strong>Finalize</strong></td>
</tr>
<tr>
<td>![Image]</td>
<td><strong>GetHashCode</strong></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>Get Type</strong></td>
<td>Gets the <strong>Type</strong> of the current instance. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>Listen</strong></td>
<td>Continuously waits for data from the pipe. The BytesReceived event is raised when data arrives, then immediately waits for more data. (Inherited from <strong>Pipe</strong>.)</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current <strong>Object</strong>. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>OnDispose</strong></td>
<td>Disconnects and disposes the pipe (Inherited from <strong>Pipe</strong>.)</td>
</tr>
<tr>
<td><strong>OnListen</strong></td>
<td>Continuously waits for data from the pipe. The BytesReceived event is raised when data arrives, then immediately waits for more data. (Overrides <strong>PipeOnListen</strong>.)</td>
</tr>
<tr>
<td><strong>OnStopListening</strong></td>
<td>After the next bytes received, stops waiting for data (Inherited from <strong>Pipe</strong>.)</td>
</tr>
</tbody>
</table>
**RaiseDisconnection**

Raises the Disconnected event. Pass in an exception if desired. (Inherited from Pipe.)

**RaiseMessageReceived**

Called by the PipeExtensions extension methods to raise the MessageReceived event

**StopListening**

After the next bytes received, stops waiting for data (Inherited from Pipe.)

**ToString**

Returns a string that represents the current object. (Inherited from Object.)

---

### Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>⚡️ <strong>Disconnected</strong></td>
<td>Raised when the pipe has disconnected (Inherited from Pipe.)</td>
</tr>
<tr>
<td>⚡️ <strong>MessageReceived</strong></td>
<td>Raised when a message has been received</td>
</tr>
</tbody>
</table>
## Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_shouldListen</td>
<td>(Inherited from Pipe.)</td>
</tr>
</tbody>
</table>

## Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the <code>as</code> operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTTType(FuncTTType)</td>
<td>Overloaded. Performs the function in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTTType(FuncTTType)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>ReadTMessage</td>
<td>Waits for a message to be read from the pipe (Defined by PipeReadWriteExtensions.)</td>
</tr>
<tr>
<td>ReadAsyncTMessage</td>
<td>Waits for a message to be read from the pipe (Defined by PipeReadWriteExtensions.)</td>
</tr>
<tr>
<td><strong>Unlock</strong></td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>-----------</td>
<td>---------------------------------------------------------</td>
</tr>
<tr>
<td><strong>WriteTMessage</strong></td>
<td>Write a message to the pipe (Defined by PipeReadWriteExtensions)</td>
</tr>
<tr>
<td><strong>WriteAsyncTMessage</strong></td>
<td>Asynchronously write a message to the pipe (Defined by PipeReadWriteExtensions)</td>
</tr>
</tbody>
</table>

See Also

Reference

W.IO.Pipes Namespace
Tungsten

\( W \)
PipeTMessage Constructor

Initializes a new instance of the PipeTMessage class

**Namespace:** W.IO.Pipes  
**Assembly:** Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll)  
**Version:** 2.0.3

### Syntax

```csharp
public Pipe()
```

### See Also

Reference  
PipeTMessage Class  
W.IO.Pipes Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$\text{W}$
PipeTMessage Properties

The PipeTMessage generic type exposes the following members.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>A unique id for this Pipe (Inherited from Pipe.)</td>
</tr>
<tr>
<td>Stream</td>
<td>The PipeStream on which to send and receive data (Inherited from Pipe.)</td>
</tr>
</tbody>
</table>

See Also

Reference
PipeTMessage Class
W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
PipeTMessage Methods

The PipeTMessage generic type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispose</td>
<td>Disconnects and disposes the pipe (Inherited from Pipe.)</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup</td>
</tr>
<tr>
<td></td>
<td>operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>Listen</td>
<td>Continuously waits for data from the pipe.</td>
</tr>
<tr>
<td>Event/Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>BytesReceived</td>
<td>The event is raised when data arrives, then immediately waits for more data. (Inherited from Pipe.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>OnDispose</td>
<td>Disconnects and disposes the pipe. (Inherited from Pipe.)</td>
</tr>
<tr>
<td>OnListen</td>
<td>Continuously waits for data from the pipe. The BytesReceived event is raised when data arrives, then immediately waits for more data. (Overrides PipeOnListen.)</td>
</tr>
<tr>
<td>OnStopListening</td>
<td>After the next bytes received, stops waiting for data. (Inherited from Pipe.)</td>
</tr>
<tr>
<td>RaiseDisconnection</td>
<td>Raises the Disconnected event. Pass in an exception if desired. (Inherited from Pipe.)</td>
</tr>
<tr>
<td>RaiseMessageReceived</td>
<td>Called by the</td>
</tr>
</tbody>
</table>
PipeExtensions extension methods to raise the MessageReceived event

- **StopListening**
  After the next bytes received, stops waiting for data
  (Inherited from Pipe.)

- **ToString**
  Returns a string that represents the current object.
  (Inherited from Object.)

---

### Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![AsTType]</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions)</td>
</tr>
<tr>
<td>![InitializeProperties]</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot; This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>![InLock(Action)]</td>
<td>Overloaded. Performs the action in a</td>
</tr>
</tbody>
</table>

---
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monitor lock (Defined by MonitorExtensions.)</td>
<td>InLockTType(FuncTType) Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTType(FuncTType)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>ReadTMessage</strong></td>
<td>Waits for a message to be read from the pipe (Defined by PipeReadWriteExtensions)</td>
</tr>
<tr>
<td><strong>ReadAsyncTMessage</strong></td>
<td>Waits for a message to be read from the pipe (Defined by PipeReadWriteExtensions)</td>
</tr>
<tr>
<td><strong>Unlock</strong></td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>WriteTMessage</strong></td>
<td>Write a message to the pipe (Defined by PipeReadWriteExtensions)</td>
</tr>
<tr>
<td><strong>WriteAsyncTMessage</strong></td>
<td>Asynchronously write a message to the pipe (Defined by PipeReadWriteExtensions)</td>
</tr>
</tbody>
</table>

### See Also

- **Reference**
  - PipeTMessage Class
  - W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
PipeTMessageOnListen Method

Continuously waits for data from the pipe. The BytesReceived event is raised when data arrives, then immediately waits for more data.

**Namespace:** W.IO.Pipes  
**Assembly:** Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll)  
**Version:** 2.0.3

### Syntax

```
C#  Copy

protected override void OnListen()
```

### See Also

Reference  
PipeTMessage Class  
W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
PipeTMessageRaiseMessageReceived Method

Called by the PipeExtensions extension methods to raise the MessageReceived event

**Namespace:** W.IO.Pipes  
**Assembly:** Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll)  
**Version:** 2.0.3

### Syntax

```csharp
protected void RaiseMessageReceived(
    Pipe<TMessage> pipe,
    TMessage message
)
```

### Parameters

**pipe**
- Type: W.IO.PipesPipeTMessage  
  [Missing <param name="pipe"/> documentation for "M:W.IO.Pipes.Pipe`1.RaiseMessageReceived(W.IO.Pipes.Pipe{`0},`0)"

**message**
- Type: TMessage  
  [Missing <param name="message"/> documentation for "M:W.IO.Pipes.Pipe`1.RaiseMessageReceived(W.IO.Pipes.Pipe{`0},`0)"

### See Also

Reference  
PipeTMessage Class
Tungsten

$W$
PipeTMessage Events

The PipeTMessage generic type exposes the following members.

## Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disconnected</td>
<td>Raised when the pipe has disconnected (Inherited from Pipe.)</td>
</tr>
<tr>
<td>MessageReceived</td>
<td>Raised when a message has been received</td>
</tr>
</tbody>
</table>

See Also

Reference

PipeTMessage Class

W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\[ W \]
Pipe `TMessage` MessageReceived Event

Raised when a message has been received

**Namespace:** W.IO.Pipes  
**Assembly:** Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll)  
**Version:** 2.0.3

### Syntax

```csharp
public event Action<Pipe<TMessage>, TMessage> MessageReceived
```

### Value

Type: `System.IAction<Pipe<TMessage>, TMessage>`

### Remarks

This event will only be raised if `TMessage != byte[]`

### See Also

**Reference**  
Pipe`TMessage` Class  
W.IO.Pipes Namespace

---

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
PipeTMessage Fields

The PipeTMessage generic type exposes the following members.

### Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_shouldListen</td>
<td>(Inherited from Pipe.)</td>
</tr>
</tbody>
</table>

See Also

Reference

PipeTMessage Class
W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
PipeClient Class

A pipe client. This class sends and receives byte arrays.

Inheritance Hierarchy

- System
  - Object
  - W.IO.PipesPipe
    - W.IO.PipesPipeByte
      - W.IO.PipesPipeClientByte
        - W.IO.PipesPipeClient

Namespace: W.IO.Pipes
Assembly: Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll) Version: 2.0.3

Syntax

```c#
public class PipeClient : PipeClient<byte[]>
```

The `PipeClient` type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="pipe" /> PipeClient</td>
<td>Initializes a new instance of the PipeClient class</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
### Id
A unique id for this Pipe (Inherited from **Pipe**.)

### Stream
The PipeStream on which to send and receive data (Inherited from **Pipe**.)

---

**Top**

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![icon] Connect</td>
<td>Attempts to connect the pipe to a pipe server (Inherited from <strong>PipeClientTMessage</strong>.)</td>
</tr>
<tr>
<td>![icon] ConnectAsync</td>
<td>Attempts to asynchronously connect the pipe to a pipe server (Inherited from <strong>PipeClientTMessage</strong>.)</td>
</tr>
<tr>
<td>![icon] Dispose</td>
<td>Disconnects and disposes the pipe (Inherited from <strong>Pipe</strong>.)</td>
</tr>
<tr>
<td>![icon] Equals</td>
<td>Determines whether the specified object is equal to the current object. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>![icon] Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <code>Type</code> of the current instance. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>Listen</td>
<td>Continuously waits for data from the pipe. The <code>BytesReceived</code> event is raised when data arrives, then immediately waits for more data. (Inherited from <code>Pipe</code>.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current <code>Object</code>. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>OnDispose</td>
<td>Disconnects and disposes the pipe (Inherited from <code>Pipe</code>.)</td>
</tr>
<tr>
<td>OnListen</td>
<td>Continuously waits for data from the pipe. The <code>BytesReceived</code> event is raised when data arrives, then immediately waits for more data. (Inherited from <code>PipeTMessage</code>.)</td>
</tr>
</tbody>
</table>
Nullable in .NET 6.0.

Events:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connected</td>
<td>Raised when a connection is established.</td>
</tr>
</tbody>
</table>

**Methods**

- **OnStopListening**
  - After the next bytes received, stops waiting for data.
  - (Inherited from Pipe.)

- **RaiseDisconnection**
  - Raises the Disconnected event.
  - Pass in an exception if desired.
  - (Inherited from Pipe.)

- **RaiseMessageReceived**
  - Called by the PipeExtensions extension methods to raise the MessageReceived event.
  - (Inherited from PipeTMessage.)

- **StopListening**
  - After the next bytes received, stops waiting for data.
  - (Inherited from Pipe.)

- **ToString**
  - Returns a string that represents the current object.
  - (Inherited from Object.)
ConnectionFailed
Raised when a connection attempt fails
(Inherited from PipeClientTMessage.)

Disconnected
Raised when the pipe has disconnected
(Inherited from Pipe.)

MessageReceived
Raised when a message has been received
(Inherited from PipeTMessage.)

Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_shouldListen</td>
<td>(Inherited from Pipe.)</td>
</tr>
</tbody>
</table>

Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>sets the member's Owner property to &quot;owner&quot;</td>
<td>This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTTType(Func&lt;TType&gt;)</td>
<td>Overloaded. Performs the function in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTTType(Func&lt;TType&gt;)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets it's IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>ReadByte</td>
<td>Waits for a message to be read from the pipe (Defined by PipeReadWriteExtensions:)</td>
</tr>
<tr>
<td>ReadAsyncByte</td>
<td>Waits for a message to be read from the pipe (Defined by PipeReadWriteExtensions:)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>WriteByte</td>
<td>Write a message to the pipe (Defined by PipeReadWriteExtensions:)</td>
</tr>
<tr>
<td>WriteAsyncByte</td>
<td>Asynchronously write a message to the pipe (Defined by PipeReadWriteExtensions:)</td>
</tr>
</tbody>
</table>

See Also
Tungsten

$W$
PipeClient Constructor

Initializes a new instance of the PipeClient class

Namespace: W.IO.Pipes
Assembly: Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll) Version: 2.0.3

Syntax

```csharp
public PipeClient()
```

See Also

Reference
PipeClient Class
W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
PipeClient Properties

The PipeClient type exposes the following members.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>A unique id for this Pipe (Inherited from Pipe.)</td>
</tr>
<tr>
<td>Stream</td>
<td>The PipeStream on which to send and receive data (Inherited from Pipe.)</td>
</tr>
</tbody>
</table>

See Also

Reference
PipeClient Class
W.IO.Pipes Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
PipeClient Methods

The **PipeClient** type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect</td>
<td>Attempts to connect the pipe to a pipe server (Inherited from <strong>PipeClientTMessage</strong>.)</td>
</tr>
<tr>
<td>ConnectAsync</td>
<td>Attempts to asynchronously connect the pipe to a pipe server (Inherited from <strong>PipeClientTMessage</strong>.)</td>
</tr>
<tr>
<td>Dispose</td>
<td>Disconnects and disposes the pipe (Inherited from <strong>Pipe</strong>.)</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>Listen</td>
<td>Continuously waits for data from the pipe. The BytesReceived event is raised when data arrives, then immediately waits for more data.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Pipe.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>OnDispose</td>
<td>Disconnects and disposes the pipe (Inherited from Pipe.)</td>
</tr>
<tr>
<td>OnListen</td>
<td>Continuously waits for data from the pipe. The BytesReceived event is raised when data arrives, then immediately waits for more data.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from PipeTMessage.)</td>
</tr>
</tbody>
</table>
**OnStopListening**
After the next bytes received, stops waiting for data
(Inherited from Pipe.)

**RaiseDisconnection**
Raises the Disconnected event. Pass in an exception if desired.
(Inherited from Pipe.)

**RaiseMessageReceived**
Called by the PipeExtensions extension methods to raise the MessageReceived event
(Inherited from PipeTMessage.)

**StopListening**
After the next bytes received, stops waiting for data
(Inherited from Pipe.)

**ToString**
Returns a string that represents the current object.
(Inherited from Object.)

### Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTTType(FuncTTType)</td>
<td>Overloaded. Performs the function in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTTType(FuncTTType)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty.</td>
</tr>
</tbody>
</table>
- **Lock**: Performs a Monitor lock.
  (Defined by MonitorExtensions.)

- **MarkAsClean**: Scans each field and property of type IProperty and sets its IsDirty flag to false.
  (Defined by PropertyHostExtensions.)

- **ReadByte**: Waits for a message to be read from the pipe.
  (Defined by PipeReadWriteExtensions.)

- **ReadAsyncByte**: Waits for a message to be read from the pipe.
  (Defined by PipeReadWriteExtensions.)

- **Unlock**: Performs a Monitor unlock.
  (Defined by MonitorExtensions.)

- **WriteByte**: Write a message to the pipe.
  (Defined by PipeReadWriteExtensions.)

- **WriteAsyncByte**: Asynchronously write a message to the pipe.
  (Defined by PipeReadWriteExtensions.)
See Also

Reference
PipeClient Class
W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
PipeClient Events

The `PipeClient` type exposes the following members.

## Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connected</td>
<td>Raised when a connection attempt succeeds (Inherited from <code>PipeClientTMessage</code>.)</td>
</tr>
<tr>
<td>ConnectionFailed</td>
<td>Raised when a connection attempt fails (Inherited from <code>PipeClientTMessage</code>.)</td>
</tr>
<tr>
<td>Disconnected</td>
<td>Raised when the pipe has disconnected (Inherited from <code>Pipe</code>.)</td>
</tr>
<tr>
<td>MessageReceived</td>
<td>Raised when a message has been received (Inherited from <code>PipeTMessage</code>.)</td>
</tr>
</tbody>
</table>

See Also

Reference

- `PipeClient Class`
- `W.IO.Pipes Namespace`
Tungsten

$W$
PipeClient Fields

The PipeClient type exposes the following members.

Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_shouldListen</td>
<td>(Inherited from Pipe.)</td>
</tr>
</tbody>
</table>

See Also

Reference
PipeClient Class
W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>$^7W$</td>
</tr>
</tbody>
</table>
PipeClient<TMessage> Class

The generic version of PipeClient. This class expects all messages to be of the specified type.

Inheritance Hierarchy

System
   object
   W.IO.PipesPipe
       W.IO.PipesPipe TMessage
           W.IO.PipesPipePipeClientTMessage
               W.IO.PipesPipePipeClient

Namespace: W.IO.Pipes
Assembly: Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll) Version: 2.0.3

Syntax

```c#
public class PipeClient<TMessage> : Pipe<TMessage>
```

Type Parameters

`TMessage`

The message type to send and receive

The PipeClientTMessageType type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PipeClientTMessageType</td>
<td>Initializes a new instance of the PipeClientTMessageType</td>
</tr>
</tbody>
</table>
### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>A unique id for this Pipe (Inherited from <a href="#">Pipe</a>).</td>
</tr>
<tr>
<td>Stream</td>
<td>The PipeStream on which to send and receive data (Inherited from <a href="#">Pipe</a>).</td>
</tr>
</tbody>
</table>

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect</td>
<td>Attempts to connect the pipe to a pipe server</td>
</tr>
<tr>
<td>ConnectAsync</td>
<td>Attempts to asynchronously connect the pipe to a pipe server</td>
</tr>
<tr>
<td>Create</td>
<td>Creates a new PipeClient and attempts to connect the pipe to a pipe server</td>
</tr>
<tr>
<td>CreateAsync</td>
<td>Creates a new PipeClient and attempts to asynchronously connect the pipe to a pipe server</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Dispose</td>
<td>Disconnects and disposes the pipe (Inherited from Pipe.)</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object. (Inherited from Object.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>Listen</td>
<td>Continuously waits for data from the pipe. The BytesReceived event is raised when data arrives, then immediately waits for more data. (Inherited from Pipe.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy</td>
</tr>
<tr>
<td>Event</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>OnDispose</strong></td>
<td>Disconnects and disposes the pipe (Inherited from Pipe.)</td>
</tr>
<tr>
<td><strong>OnListen</strong></td>
<td>Continuously waits for data from the pipe. The BytesReceived event is raised when data arrives, then immediately waits for more data. (Inherited from PipeTMessage.)</td>
</tr>
<tr>
<td><strong>OnStopListening</strong></td>
<td>After the next bytes received, stops waiting for data (Inherited from Pipe.)</td>
</tr>
<tr>
<td><strong>RaiseDisconnection</strong></td>
<td>Raises the Disconnected event. Pass in an exception if desired. (Inherited from Pipe.)</td>
</tr>
<tr>
<td><strong>RaiseMessageReceived</strong></td>
<td>Called by the PipeExtensions extension methods to raise the MessageReceived event (Inherited from PipeTMessage.)</td>
</tr>
</tbody>
</table>
StopListening

After the next bytes received, stops waiting for data
(Inherited from Pipe.)

ToString

Returns a string that represents the current object.
(Inherited from Object.)

Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connected</td>
<td>Raised when a connection attempt succeeds</td>
</tr>
<tr>
<td>ConnectionFailed</td>
<td>Raised when a connection attempt fails</td>
</tr>
<tr>
<td>Disconnect</td>
<td>Raised when the pipe has disconnected</td>
</tr>
<tr>
<td>MessageReceived</td>
<td>Raised when a message has been received</td>
</tr>
</tbody>
</table>

Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_shouldListen</td>
<td>(Inherited from Pipe.)</td>
</tr>
</tbody>
</table>
## Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AsTType</code></td>
<td>Use Generic syntax for the <code>as</code> operator. (Defined by <code>AsExtensions</code>)</td>
</tr>
<tr>
<td><code>InitializeProperties</code></td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's <code>Owner</code> property to &quot;owner&quot;. This method should be called in the constructor of any class which has <code>IOwnedProperty</code> members. (Defined by <code>PropertyHostExtensions</code>.)</td>
</tr>
<tr>
<td><code>InLock(Action)</code></td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>InLockTType(FuncTType)</code></td>
<td>Overloaded. Performs the function in a Monitor lock. (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>InLockAsync(Action)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock. (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><strong>Method</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------</td>
</tr>
<tr>
<td><strong>InLockAsyncTType(Func&lt;TType&gt;)</strong></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>IsDirty</strong></td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td><strong>Lock</strong></td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>MarkAsClean</strong></td>
<td>Scans each field and property of type IProperty and sets it's IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td><strong>ReadTMessage</strong></td>
<td>Waits for a message to be read from the pipe (Defined by PipeReadWriteExtensions.)</td>
</tr>
<tr>
<td><strong>ReadAsyncTMessage</strong></td>
<td>Waits for a message to be read from the pipe (Defined by PipeReadWriteExtensions.)</td>
</tr>
<tr>
<td><strong>Unlock</strong></td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>WriteTMessage</strong></td>
<td>Write a message to the pipe</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td><code>WriteAsyncTMessage</code></td>
<td>Asynchronously write a message to the pipe</td>
</tr>
</tbody>
</table>

(Defined by `PipeReadWriteExtensions`)

See Also

Reference

W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
PipeClientTMessage Constructor

Initializes a new instance of the PipeClientTMessage class

Namespace: W.IO.Pipes
Assembly: Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll) Version: 2.0.3

Syntax

```csharp
public PipeClient()
```

See Also

Reference
PipeClientTMessage Class
W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
<th>W</th>
</tr>
</thead>
</table>


PipeClientTMessage Properties

The PipeClientTMessage generic type exposes the following members.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>A unique id for this Pipe (Inherited from Pipe.)</td>
</tr>
<tr>
<td>Stream</td>
<td>The PipeStream on which to send and receive data (Inherited from Pipe.)</td>
</tr>
</tbody>
</table>

See Also

Reference
PipeClientTMessage Class
W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
PipeClient{TMessage} Methods

The PipeClient{TMessage} generic type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Pipe icon]</td>
<td>![Connect icon] Connect</td>
</tr>
<tr>
<td>![Pipe icon]</td>
<td>![ConnectAsync icon] ConnectAsync</td>
</tr>
<tr>
<td>![Pipe icon]</td>
<td>![Create icon] Create</td>
</tr>
<tr>
<td>![Pipe icon]</td>
<td>![CreateAsync icon] CreateAsync</td>
</tr>
<tr>
<td>![Pipe icon]</td>
<td>![Dispose icon] Dispose</td>
</tr>
<tr>
<td>![Pipe icon]</td>
<td>![Equals icon] Equals</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as the default hash function. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <strong>Type</strong> of the current instance. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>Listen</strong></td>
<td>Continuously waits for data from the pipe. The BytesReceived event is raised when data arrives, then immediately waits for more data. (Inherited from <strong>Pipe</strong>.)</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current <strong>Object</strong>. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>OnDispose</strong></td>
<td>Disconnects and disposes the pipe (Inherited from <strong>Pipe</strong>.)</td>
</tr>
<tr>
<td><strong>OnListen</strong></td>
<td>Continuously waits for data from the pipe. The BytesReceived event is</td>
</tr>
</tbody>
</table>
raised when data arrives, then immediately waits for more data. (Inherited from PipeTMessage.)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnStopListening</td>
<td>After the next bytes received, stops waiting for data (Inherited from Pipe.)</td>
</tr>
<tr>
<td>RaiseDisconnection</td>
<td>Raises the Disconnected event. Pass in an exception if desired. (Inherited from Pipe.)</td>
</tr>
<tr>
<td>RaiseMessageReceived</td>
<td>Called by the PipeExtensions extension methods to raise the MessageReceived event (Inherited from PipeTMessage.)</td>
</tr>
<tr>
<td>StopListening</td>
<td>After the next bytes received, stops waiting for data (Inherited from Pipe.)</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object. (Inherited from Object.)</td>
</tr>
</tbody>
</table>
## Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the <code>as</code> operator. (Defined by AsExtensions)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTType(FuncTType)</td>
<td>Overloaded. Performs the function in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTType(FuncTType)</td>
<td>Overloaded. Asynchronously performs</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>the action in a Monitor lock</td>
<td>(Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false</td>
</tr>
<tr>
<td>ReadTMessage</td>
<td>Waits for a message to be read from the pipe</td>
</tr>
<tr>
<td>ReadAsyncTMessage</td>
<td>Waits for a message to be read from the pipe</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock</td>
</tr>
<tr>
<td>WriteTMessage</td>
<td>Write a message to the pipe</td>
</tr>
</tbody>
</table>
WriteAsyncTMessage

Asynchronously write a message to the pipe (Defined by PipeReadWriteExtensions)

Top

See Also

Reference
PipeClientTMessage Class
W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
PipeClient TMessage Connect Method

Attempts to connect the pipe to a pipe server

**Namespace:** W.IO.Pipes  
**Assembly:** Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll)  
**Version:** 2.0.3

Syntax

```csharp
public bool Connect(  
    string serverName,  
    string pipeName,  
    TokenImpersonationLevel tokenImpersonationLevel  
    int msTimeout
)
```

Parameters

**serverName**  
Type: System.String  
The name or ip of the machine hosting the server pipe

**pipeName**  
Type: System.String  
The name of the pipe

**tokenImpersonationLevel**  
Type: System.Security.Principal.TokenImpersonationLevel  
The impersonation type for the pipe to use

**msTimeout**  
Type: System.Int32  
The maximum amount of time, in milliseconds, to wait for the
server to connect

Return Value
Type: **Boolean**
True if a connection was established, otherwise False

See Also

Reference
PipeClientTMessage Class
W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
PipeClient

TMessage

ConnectAsync

Method

Attempts to asynchronously connect the pipe to a pipe server

**Namespace:** W.IO.Pipes
**Assembly:** Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll) Version: 2.0.3

**Syntax**

```csharp
public Task<bool> ConnectAsync(
    string serverName,
    string pipeName,
    TokenImpersonationLevel tokenImpersonationLevel
    int msTimeout
)
```

**Parameters**

*serverName*
  Type: **System.String**
  The name or ip of the machine hosting the server pipe

*pipeName*
  Type: **System.String**
  The name of the pipe

*tokenImpersonationLevel*
  Type: **System.Security.Principal.TokenImpersonationLevel**
  The impersonation type for the pipe to use

*msTimeout*
  Type: **System.Int32**
  The maximum amount of time, in milliseconds, to wait for the
server to connect

Return Value
Type: TaskBoolean
True if a connection was established, otherwise False

See Also

Reference
PipeClientTMessage Class
W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
PipeClient

PipeClient Create Method

Creates a new PipeClient and attempts to connect the pipe to a pipe server

**Namespace:**  W.IO.Pipes  
**Assembly:**  Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll)  
**Version:**  2.0.3

### Syntax

```csharp
public static PipeClient Create(
    string serverName,
    string pipeName,
    TokenImpersonationLevel tokenImpersonationLevel,
    int msTimeout
)
```

### Parameters

- **servername**
  - Type: **SystemString**  
  - The name or ip of the machine hosting the server pipe

- **pipeName**
  - Type: **SystemString**  
  - The name of the pipe

- **tokenImpersonationLevel**
  - Type: **System.Security.Principal.TokenImpersonationLevel**  
  - The impersonation type for the pipe to use

- **msTimeout**
  - Type: **SystemInt32**
The maximum amount of time, in milliseconds, to wait for the server to connect

Return Value
Type: PipeClient
True if a connection was established, otherwise False

See Also

Reference
PipeClientTMessage Class
W.IO.Pipes Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

W
PipeClient

CreateAsync Method

Creates a new PipeClient and attempts to asynchronously connect the pipe to a pipe server

Namespace: W.IO.Pipes
Assembly: Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll) Version: 2.0.3

Syntax

```csharp
public static Task<Pipe> CreateAsync(
    string serverName,
    string pipeName,
    TokenImpersonationLevel tokenImpersonationLevel
    int msTimeout
)
```

Parameters

*serverName*
  Type: System.String
  The name or ip of the machine hosting the server pipe

*pipeName*
  Type: System.String
  The name of the pipe

*tokenImpersonationLevel*
  Type: System.Security.Principal.TokenImpersonationLevel
  The impersonation type for the pipe to use

*msTimeout*
  Type: System.Int32
The maximum amount of time, in milliseconds, to wait for the server to connect

Return Value
Type: TaskPipe
True if a connection was established, otherwise False

See Also

Reference
PipeClientTMessage Class
W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
PipeClientTMessage Events

The PipeClientTMessage generic type exposes the following members.

## Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connected</td>
<td>Raised when a connection attempt succeeds</td>
</tr>
<tr>
<td>ConnectionFailed</td>
<td>Raised when a connection attempt fails</td>
</tr>
<tr>
<td>Disconnected</td>
<td>Raised when the pipe has disconnected (Inherited from Pipe.)</td>
</tr>
<tr>
<td>MessageReceived</td>
<td>Raised when a message has been received (Inherited from PipeTMessage.)</td>
</tr>
</tbody>
</table>

See Also

Reference
PipeClientTMessage Class
W.IO.Pipes Namespace

Copyright © 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
</tr>
</tbody>
</table>
PipeClient\textit{TMMessage} Connected Event

Raised when a connection attempt succeeds

\textbf{Namespace: } W.IO.Pipes  
\textbf{Assembly: } Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll)  
\textbf{Version: } 2.0.3

\section*{Syntax}

\begin{verbatim}
public event Action<Pipe> Connected
\end{verbatim}

\section*{Value}

\textbf{Type: } SystemActionPipe

\section*{See Also}

\textbf{Reference}

PipeClient\textit{TMMessage Class}  
W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
| Tungsten | $W$ |
PipeClient

**TMessage**

**ConnectionFailed Event**

Raised when a connection attempt fails

**Namespace:** W.IO.Pipes

**Assembly:** Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll) **Version:** 2.0.3

### Syntax

```csharp
public event Action<Pipe, Exception> ConnectionFailed;
```

### Value

Type: `System.Action<Pipe, Exception>`

### See Also

**Reference**

- PipeClient
- TMessage Class
- W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
# PipeClientTMessage Fields

The `PipeClientTMessage` generic type exposes the following members.

## Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>_shouldListen</code></td>
<td>(Inherited from <code>Pipe</code>.)</td>
</tr>
</tbody>
</table>

## See Also

- Reference
  - `PipeClientTMessage Class`
  - `W.IO.Pipes Namespace`

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
PipeHost Class

Hosts a number of PipeServers. This class sends and receives byte arrays.

Inheritance Hierarchy

- System
  - Object
    - W.IO.Pipes
      - PipeHost

Namespace: W.IO.Pipes
Assembly: Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll) Version: 2.0.3

Syntax

```csharp
public class PipeHost : PipeHost<byte[]>
```

The **PipeHost** type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PipeHost</td>
<td>Initializes a new instance of the <strong>PipeHost</strong> class</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Dispose</td>
<td>Stops the host and releases resources (Inherited from PipeHostTMessage.)</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object. (Inherited from Object.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>Start</td>
<td>Creates the specified number of pipe servers and starts listening for clients (Inherited from PipeHostTMessage.)</td>
</tr>
<tr>
<td>Stop</td>
<td>Disconnects and disposes all of the pipe servers (Inherited from</td>
</tr>
</tbody>
</table>
PipeHostTMessage.)

**ToString**

Returns a string that represents the current object.
(Inherited from Object.)

---

### Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MessageReceived</td>
<td>Raised when a pipe server has received data from a client (Inherited from PipeHostTMessage.)</td>
</tr>
</tbody>
</table>

---

### Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot; This method should be called in the constructor of any class which has IOwnedProperty members (Defined by</td>
</tr>
</tbody>
</table>

---
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTType(Func&lt;TType&gt;)</td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTType(Func&lt;TType&gt;)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td></td>
<td>Scans each field and</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>property of type IProperty and sets it's IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>

See Also

Reference

W.IO.Pipes Namespace
Tungsten

$W$
PipeHost Constructor

Initializes a new instance of the PipeHost class

Namespace:  W.IO.Pipes
Assembly:  Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll) Version: 2.0.3

Syntax

```
public PipeHost()
```

See Also

Reference
PipeHost Class
W.IO.Pipes Namespace
Tungsten

$W$
PipeHost Methods

The **PipeHost** type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispose</td>
<td>Stops the host and releases resources (Inherited from <code>PipeHostTMessage</code>.)</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup</td>
</tr>
<tr>
<td></td>
<td>operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <code>Type</code> of the current instance. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current <code>Object</code>.</td>
</tr>
</tbody>
</table>
Start

Creates the specified number of pipe servers and starts listening for clients
(Inherited from PipeHostTMessage.)

Stop

Disconnects and disposes all of the pipe servers
(Inherited from PipeHostTMessage.)

ToString

Returns a string that represents the current object.
(Inherited from Object.)

Top

Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>InLock(Action)</code></td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by <code>MonitorExtensions</code>).</td>
</tr>
<tr>
<td><code>InLockTTType(FuncTTType)</code></td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by <code>MonitorExtensions</code>).</td>
</tr>
<tr>
<td><code>InLockAsync(Action)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by <code>MonitorExtensions</code>).</td>
</tr>
<tr>
<td><code>InLockAsyncTTType(FuncTTType)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by <code>MonitorExtensions</code>).</td>
</tr>
<tr>
<td><code>IsDirty</code></td>
<td>Scans the IsDirty value of each field and property of type <code>IProperty</code> (Defined by <code>PropertyHostExtensions</code>).</td>
</tr>
<tr>
<td><code>Lock</code></td>
<td>Performs a Monitor lock (Defined by <code>MonitorExtensions</code>).</td>
</tr>
<tr>
<td><code>MarkAsClean</code></td>
<td>Scans each field and property of type <code>IProperty</code> and sets its IsDirty flag to</td>
</tr>
</tbody>
</table>
Unlock

Performs a Monitor unlock
(Defined by MonitorExtensions.)

See Also

Reference
PipeHost Class
W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
PipeHost Events

The PipeHost type exposes the following members.

## Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MessageReceived</td>
<td>Raised when a pipe server has received data from a client (Inherited from PipeHostTMessage.)</td>
</tr>
</tbody>
</table>

See Also

Reference
PipeHost Class
W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
PipeHost\textit{TMessage} Class

The generic version of PipeHost. This class expects all messages to be of the specified type.

\section*{Inheritance Hierarchy}

\begin{center}
\begin{tabular}{|c|c|}
\hline
\texttt{SystemObject} & \texttt{W.IO.PipesPipeHostTMessage} \\
\texttt{W.IO.PipesPipeHost} &  \\
\hline
\end{tabular}
\end{center}

\textbf{Namespace:} \texttt{W.IO.Pipes}

\textbf{Assembly:} Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll) Version: 2.0.3

\section*{Syntax}

\begin{verbatim}
public class PipeHost<TMessage> : IDisposable

Type Parameters

\textit{TMessage}  \\
\textit{The message type to send and receive}

The \texttt{PipeHostTMessage} type exposes the following members.

\section*{Constructors}

\begin{center}
\begin{tabular}{|c|c|}
\hline
\textbf{Name} & \textbf{Description} \\
\hline
\texttt{PipeHostTMessage} & Initializes a new instance of the \texttt{PipeHostTMessage} class \\
\hline
\end{tabular}
\end{center}
# Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispose</td>
<td>Stops the host and releases resources</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <a href="#">Object</a>.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <a href="#">Object</a>.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <a href="#">Object</a>.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <a href="#">Object</a>.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current <strong>Object</strong>.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <a href="#">Object</a>.)</td>
</tr>
<tr>
<td>Start</td>
<td>Creates the specified number of pipe servers and starts listening for clients</td>
</tr>
<tr>
<td>Stop</td>
<td>Disconnects and disposes all</td>
</tr>
</tbody>
</table>
of the pipe servers

- **ToString**
  - Returns a string that represents the current object. (Inherited from **Object**.)

### Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MessageReceived</strong></td>
<td>Raised when a pipe server has received data from a client</td>
</tr>
</tbody>
</table>

### Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AsTType</strong></td>
<td>Use Generic syntax for the as operator. (Defined by <strong>AsExtensions</strong>.)</td>
</tr>
<tr>
<td><strong>InitializeProperties</strong></td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot; This method should be called in the constructor of any class which has IOwnedProperty members (Defined by <strong>PropertyHostExtensions</strong>.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>InLock(Action)</code></td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><code>InLockTType(FuncTType)</code></td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><code>InLockAsync(Action)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><code>InLockAsyncTType(FuncTType)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><code>IsDirty</code></td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td><code>Lock</code></td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><code>MarkAsClean</code></td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to</td>
</tr>
</tbody>
</table>
false
(Defined by PropertyHostExtensions.)

Unlock
Performs a Monitor unlock
(Defined by MonitorExtensions.)

See Also

Reference
W.IO.Pipes Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$\text{W}$
PipeHostTMessage Constructor

Initializes a new instance of the PipeHostTMessage class

**Namespace:**  W.IO.Pipes  
**Assembly:**  Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll)  
**Version:**  2.0.3

### Syntax

```
public PipeHost()
```

### See Also

Reference  
PipeHostTMessage Class  
W.IO.Pipes Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
PipeHostTMessage Methods

The PipeHostTMessage generic type exposes the following members.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispose</td>
<td>Stops the host and releases resources</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations</td>
</tr>
<tr>
<td></td>
<td>before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
</tbody>
</table>
Start
Creates the specified number of pipe servers and starts listening for clients

Stop
Disconnects and disposes all of the pipe servers

ToString
Returns a string that represents the current object. (Inherited from Object.)

Top

Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![AsTType]</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>![InitializeProperties]</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot; This method should be called in the constructor of any class which has IOwnedProperty members (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>![InLock(Action)]</td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>InLockTType(FuncTType)</td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTType(FuncTType)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets it's IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock</td>
</tr>
</tbody>
</table>
See Also

Reference
PipeHostTMessage Class
W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$^\text{W}$
PipeHost\n\nMethod

Stops the host and releases resources

**Namespace:** W.IO.Pipes  
**Assembly:** Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll)  
**Version:** 2.0.3

### Syntax

```
public void Dispose()
```

**Implements**  
IDisposable Dispose

### See Also

**Reference**  
PipeHost\n\nTMessage Class  
W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$^W$
PipeHost TMessage Start Method

Creates the specified number of pipe servers and starts listening for clients

Namespace: W.IO.Pipes
Assembly: Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll) Version: 2.0.3

Syntax

```csharp
public uint Start(
    string pipeName,
    int maxConnections
)
```

Parameters

pipeName
Type: System.String
The name of the pipe

maxConnections
Type: System.Int32
The maximum number of pipe servers to create.

Return Value
Type: UInt32

Remarks

Because PipeHost creates a PipeServer for each possible
connection, this value cannot be negative. This breaks the standard paradigm for pipes. If this does not work for you, use PipeServer instead.

**See Also**

Reference

PipeHostTMessage Class
W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
PipeHost TMessage Stop Method

Disconnects and disposes all of the pipe servers

Namespace: W.IO.Pipes
Assembly: Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll) Version: 2.0.3

Syntax

```csharp
public void Stop()
```

See Also

Reference
PipeHostTMessage Class
W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
</tr>
</tbody>
</table>
PipeHostTMessage Events

The PipeHostTMessage generic type exposes the following members.

Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MessageReceived</td>
<td>Raised when a pipe server has received data from a client</td>
</tr>
</tbody>
</table>

See Also

Reference
PipeHostTMessage Class
W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
PipeHost TMessage MessageReceived Event

Raised when a pipe server has received data from a client

**Namespace:** W.IO.Pipes  
**Assembly:** Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll)  
**Version:** 2.0.3

### Syntax

```csharp
public event Action<PipeHost<TMessage>, Pipe<TMessage>, TMessage>
```

### Value

Type: SystemActionPipeHost<TMessage>, Pipe<TMessage>, TMessage

### See Also

Reference  
PipeHostTMessage Class  
W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
PipeLogger Class

Sends log messages to a remote server via a named pipe

Inheritance Hierarchy

```
System
  Object
  W.IO.Pipes
  PipeLogger
```

Namespace: W.IO.Pipes
Assembly: Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll) Version: 2.0.3

Syntax

```
public static class PipeLogger
```

The PipeLogger type exposes the following members.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![ ]</td>
<td>PipeClient</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![ ]</td>
<td>LogTheMessage</td>
</tr>
</tbody>
</table>
message is slower due to parsing the remoteIP with each call

Top

Examples

Log.LogTheMessage += (category, message) =>
W.IO.Pipes.PipeLogger.LogTheMessage(“.”, “PipeLogger”, true, category, message);

See Also

Reference
W.IO.Pipes Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

W
PipeLogger Properties

The PipeLogger type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![s]</td>
<td>PipeClient</td>
</tr>
<tr>
<td></td>
<td>The named pipe client used to send log messages</td>
</tr>
</tbody>
</table>

### See Also

Reference

PipeLogger Class
W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
PipeLoggerPipeClient Property

The named pipe client used to send log messages

**Namespace**: W.IO.Pipes  
**Assembly**: Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll) Version: 2.0.3

## Syntax

```csharp
public static PipeClient PipeClient { get; }
```

**Property Value**  
Type: PipeClient

## See Also

**Reference**  
PipeLogger Class  
W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$^W$
PipeLogger Methods

The `PipeLogger` type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LogTheMessage</td>
<td>Log a message to the remote machine. Note that this message is slower due to parsing the remote IP with each call</td>
</tr>
</tbody>
</table>

See Also

Reference

- `PipeLogger Class`
- `W.IO.Pipes Namespace`

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
PipeLoggerLogTheMessage Method

Log a message to the remote machine. Note that this message is slower due to parsing the remote IP with each call.

**Namespace:** W.IO.Pipes  
**Assembly:** Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll)  
**Version:** 2.0.3

### Syntax

```
    public static void LogTheMessage(  
        string server,  
        string pipeName,  
        string message  
    )
```

### Parameters

- **server**
  - Type: `System.String`
  - The name of the server hosting the named pipe

- **pipeName**
  - Type: `System.String`
  - The name of the named pipe

- **message**
  - Type: `System.String`
  - The log message

### See Also
Reference
PipeLogger Class
W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$\text{W}$
PipeReadWriteExtensions Class

Read/Write functionality for Pipe

Inheritance Hierarchy

- System
  - Object
    - W.IO.Pipes
      - PipeReadWriteExtensions

Namespace: W.IO.Pipes
Assembly: Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll) Version: 2.0.3

Syntax

```csharp
public static class PipeReadWriteExtensions
```

The `PipeReadWriteExtensions` type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ReadTMessage</code></td>
<td>Waits for a message to be read from the pipe</td>
</tr>
<tr>
<td><code>ReadAsyncTMessage</code></td>
<td>Waits for a message to be read from the pipe</td>
</tr>
<tr>
<td><code>WriteTMessage</code></td>
<td>Write a message to the pipe</td>
</tr>
<tr>
<td><code>WriteAsyncTMessage</code></td>
<td>Asynchronously write a message to the pipe</td>
</tr>
</tbody>
</table>
See Also

Reference

W.IO.Pipes Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
PipeReadWriteExtensions

Methods

The **PipeReadWriteExtensions** type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReadTMessage</td>
<td>Waits for a message to be read from the pipe</td>
</tr>
<tr>
<td>ReadAsyncTMessage</td>
<td>Waits for a message to be read from the pipe</td>
</tr>
<tr>
<td>WriteTMessage</td>
<td>Write a message to the pipe</td>
</tr>
<tr>
<td>WriteAsyncTMessage</td>
<td>Asynchronously write a message to the pipe</td>
</tr>
</tbody>
</table>

See Also

Reference
- PipeReadWriteExtensions Class
- W.IO.Pipes Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
PipeReadWriteExtensions.Read<TMessage> Method

Waits for a message to be read from the pipe

**Namespace:** W.IO.Pipes  
**Assembly:** Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll)  
**Version:** 2.0.3

## Syntax

```csharp
public static TMessage Read<TMessage>(
    this Pipe<TMessage> pipe
)
```

### Parameters

- **pipe**  
  Type: W.IO.PipesPipe<TMessage>  
  The pipe from which to read data

### Type Parameters

- **TMessage**  
  [Missing <typeparam name="TMessage"/> documentation for "M:W.IO.Pipes.PipeReadWriteExtensions.Read`1(W.IO.Pipes.Pipe`0)""]

### Return Value

Type: **TMessage**  
The message received or null if the read failed (the pipe was closed)

### Usage Note

In Visual Basic and C#, you can call this method as an instance
method on any object of type PipeTMessage. When you use instance method syntax to call this method, omit the first parameter. For more information, see Extension Methods (Visual Basic) or Extension Methods (C# Programming Guide).

See Also

Reference
PipeReadWriteExtensions Class
W.IO.Pipes Namespace
Tungsten

$W$
PipeReadWriteExtensions.ReadAsync Method

Waits for a message to be read from the pipe

**Namespace:** W.IO.Pipes  
**Assembly:** Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll)  
**Version:** 2.0.3

### Syntax

```csharp
public static Task<TMessage> ReadAsync<TMessage>(
    this Pipe<TMessage> pipe
)
```

### Parameters

- **pipe**
  - Type: W.IO.PipesPipe
  - The pipe from which to read data

### Type Parameters

- **TMessage**
  - [Missing <typeparam name="TMessage"/> documentation for "M:W.IO.Pipes.PipeReadWriteExtensions.ReadAsync`1(W.IO.Pipes.Pipe`{0})"]

### Return Value

- **Type:** Task
  - The message received or null if the read failed (the pipe was closed)

### Usage Note
In Visual Basic and C#, you can call this method as an instance method on any object of type `PipeTMessage`. When you use instance method syntax to call this method, omit the first parameter. For more information, see Extension Methods (Visual Basic) or Extension Methods (C# Programming Guide).

See Also

Reference
PipeReadWriteExtensions Class
W.IO.Pipes Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
PipeReadWriteExtensions.Write<TMessage> Method

Write a message to the pipe

**Namespace:**  W.IO.Pipes  
**Assembly:**  Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll)  
**Version:**  2.0.3

### Syntax

```csharp
public static bool Write<TMessage>(
    this Pipe<TMessage> pipe,
    TMessage message
)
```

### Parameters

- **pipe**
  - Type:  W.IO.PipesPipe<TMessage>
  - The pipe on which to write data

- **message**
  - Type:  TMessage
  - The message to write

### Type Parameters

- **TMessage**
  - [Missing <typeparam name="TMessage"/> documentation for "M:W.IO.Pipes.PipeReadWriteExtensions.Write``1(W.IO.Pipes.Pipe{``0},``0)""]

### Return Value
Type: **Boolean**
True if the message was sent successfully, otherwise false

**Usage Note**
In Visual Basic and C#, you can call this method as an instance method on any object of type `PipeTMessage`. When you use instance method syntax to call this method, omit the first parameter. For more information, see [Extension Methods (Visual Basic)](Extension Methods (Visual Basic)) or [Extension Methods (C# Programming Guide)](Extension Methods (C# Programming Guide)).

**See Also**

**Reference**
PipeReadWriteExtensions Class
W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>$W$</td>
</tr>
</tbody>
</table>
PipeReadWriteExtensions\WriteAsync Method

Asynchronously write a message to the pipe

**Namespace:**  W.IO.Pipes
**Assembly:**  Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll) Version: 2.0.3

**Syntax**

```csharp
public static Task<bool> WriteAsync<TMessage>(
    this Pipe<TMessage> pipe,
    TMessage message
)
```

**Parameters**

- **pipe**
  Type:  W.IO.PipesPipe\TMessage
  The pipe on which to write data

- **message**
  Type:  TMessage
  The message to write

**Type Parameters**

- **TMessage**
  [Missing <typeparam name="TMessage"/> documentation for "M:W.IO.Pipes.PipeReadWriteExtensions.WriteAsync`1(W.IO.Pipes.Pipe`0,`0)"

**Return Value**
Type: TaskBoolean
True if the message was sent successfully, otherwise false

Usage Note
In Visual Basic and C#, you can call this method as an instance method on any object of type PipeTMessage. When you use instance method syntax to call this method, omit the first parameter. For more information, see Extension Methods (Visual Basic) or Extension Methods (C# Programming Guide).

See Also

Reference
PipeReadWriteExtensions Class
W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
PipeServer Class

A Pipe server. This class sends and receives only byte arrays.

Inheritance Hierarchy

- System
  - Object
- W.IO.Pipes
  - Pipe
    - W.IO.PipesPipe
      - W.IO.PipesPipeByte
    - W.IO.PipesPipeServerByte
- W.IO.PipesPipeServer

Namespace: W.IO.Pipes
Assembly: Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll) Version: 2.0.3

Syntax

```
public class PipeServer : PipeServer<byte[]>
```

The `PipeServer` type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>PipeServer</code></td>
<td>Initializes a new instance of the <code>PipeServer</code> class</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Id</strong></td>
<td>A unique id for this Pipe (Inherited from Pipe.)</td>
</tr>
<tr>
<td><strong>Stream</strong></td>
<td>The PipeStream on which to send and receive data (Inherited from Pipe.)</td>
</tr>
<tr>
<td><strong>Dispose</strong></td>
<td>Disconnects and disposes the pipe (Inherited from Pipe.)</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup</td>
</tr>
<tr>
<td></td>
<td>operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as the default hash function. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Listen</td>
<td>Continuously waits for data from the pipe. The BytesReceived event is raised when data arrives, then immediately waits for more data. (Inherited from Pipe.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>OnDispose</td>
<td>Disposes the PipeServer and release resources (Inherited from PipeServerTMessage.)</td>
</tr>
<tr>
<td>OnListen</td>
<td>Continuously waits for data from the pipe. The BytesReceived event is raised when data arrives, then immediately waits for more data. (Inherited from PipeTMessage.)</td>
</tr>
<tr>
<td>OnStopListening</td>
<td>After the next bytes received, stops waiting for data (Inherited from Pipe.)</td>
</tr>
<tr>
<td>RaiseDisconnection</td>
<td>Raises the Disconnected event. Pass in an exception if desired. (Inherited from Pipe.)</td>
</tr>
<tr>
<td>RaiseMessageReceived</td>
<td>Called by the PipeExtensions</td>
</tr>
</tbody>
</table>


extension methods to raise the MessageReceived event (Inherited from PipeTMessage.)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StopListening</td>
<td>After the next bytes received, stops waiting for data (Inherited from Pipe.)</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object. (Inherited from Object.)</td>
</tr>
<tr>
<td>WaitForConnection</td>
<td>Creates a new NamedPipeServerStream and waits for a client to connect (Inherited from PipeServerTMessage.)</td>
</tr>
</tbody>
</table>

**Top**

**Events**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connected</td>
<td>Raised when a client has connected to the server (Inherited from PipeServerTMessage.)</td>
</tr>
<tr>
<td>Disconnected</td>
<td>Raised when the pipe has disconnected (Inherited from Pipe.)</td>
</tr>
</tbody>
</table>
### MessageReceived
Raised when a message has been received
(Inherited from PipeTMessage.)

### StartException
Raised if an exception occurs while creating the NamedPipeServerStream
(Inherited from PipeServerTMessage.)

---

### Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_shouldListen</td>
<td>(Inherited from Pipe.)</td>
</tr>
</tbody>
</table>

---

### Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot; This method should be called in the constructor of any class which has IOwnedProperty members (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTTType(FuncTTType)</td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTTType(FuncTTType)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ReadByte</td>
<td>Waits for a message to be read from the pipe</td>
</tr>
<tr>
<td>ReadAsyncByte</td>
<td>Waits for a message to be read from the pipe</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock</td>
</tr>
<tr>
<td>WriteByte</td>
<td>Write a message to the pipe</td>
</tr>
<tr>
<td>WriteAsyncByte</td>
<td>Asynchronously write a message to the pipe</td>
</tr>
</tbody>
</table>

**See Also**

Reference

W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
PipeServer Constructor

Initializes a new instance of the PipeServer class

**Namespace:** W.IO.Pipes  
**Assembly:** Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll)  
**Version:** 2.0.3

### Syntax

C#

```csharp
public PipeServer()
```

### See Also

Reference  
PipeServer Class  
W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
PipeServer Properties

The PipeServer type exposes the following members.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>A unique id for this Pipe (Inherited from Pipe.)</td>
</tr>
<tr>
<td>Stream</td>
<td>The PipeStream on which to send and receive data (Inherited from Pipe.)</td>
</tr>
</tbody>
</table>

See Also

Reference
PipeServer Class
W.IO.Pipes Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
PipeServer Methods

The PipeServer type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispose</td>
<td>Disconnects and disposes the pipe (Inherited from Pipe.)</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object. (Inherited from Object.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>Listen</td>
<td>Continuously waits for data from the pipe. The</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>BytesReceived</td>
<td>The BytesReceived event is raised when data arrives, then immediately waits for more data. (Inherited from Pipe.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>OnDispose</td>
<td>Disposes the PipeServer and release resources (Inherited from PipeServerTMessage.)</td>
</tr>
<tr>
<td>OnListen</td>
<td>Continuously waits for data from the pipe. The BytesReceived event is raised when data arrives, then immediately waits for more data. (Inherited from PipeTMessage.)</td>
</tr>
<tr>
<td>OnStopListening</td>
<td>After the next bytes received, stops waiting for data (Inherited from Pipe.)</td>
</tr>
<tr>
<td>RaiseDisconnection</td>
<td>Raises the Disconnected event. Pass in an exception if desired. (Inherited from Pipe.)</td>
</tr>
<tr>
<td>RaiseMessageReceived</td>
<td>Called by the PipeExtensions extension methods to raise the</td>
</tr>
</tbody>
</table>
MessageReceived event (Inherited from PipeTMessage.)

- **StopListening**
  - After the next bytes received, stops waiting for data
  - (Inherited from Pipe.)

- **ToString**
  - Returns a string that represents the current object.
  - (Inherited from Object.)

- **WaitForConnection**
  - Creates a new NamedPipeServerStream and Waits for a client to connect
  - (Inherited from PipeServerTMessage.)

**Extension Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![ ] AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions)</td>
</tr>
<tr>
<td>![ ] InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLock&lt;TType(Func&lt;TType)&gt;</td>
<td>Overloaded. Performs the function in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync&lt;TType(Func&lt;TType)&gt;</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty.</td>
</tr>
</tbody>
</table>
and sets its IsDirty flag to false
(Defined by PropertyHostExtensions.)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ReadStream</strong></td>
<td>Waits for a message to be read from the pipe (Defined by PipeReadWriteExtensions).</td>
</tr>
<tr>
<td><strong>ReadStreamAsync</strong></td>
<td>Waits for a message to be read from the pipe (Defined by PipeReadWriteExtensions).</td>
</tr>
<tr>
<td><strong>Unlock</strong></td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>WriteByte</strong></td>
<td>Write a message to the pipe (Defined by PipeReadWriteExtensions).</td>
</tr>
<tr>
<td><strong>WriteAsyncByte</strong></td>
<td>Asynchronously write a message to the pipe (Defined by PipeReadWriteExtensions).</td>
</tr>
</tbody>
</table>

See Also

Reference
PipeServer Class
W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
PipeServer Events

The PipeServer type exposes the following members.

## Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connected</td>
<td>Raised when a client has connected to the server (Inherited from PipeServerTMessage.)</td>
</tr>
<tr>
<td>Disconnected</td>
<td>Raised when the pipe has disconnected (Inherited from Pipe.)</td>
</tr>
<tr>
<td>MessageReceived</td>
<td>Raised when a message has been received (Inherited from PipeTMessage.)</td>
</tr>
<tr>
<td>StartException</td>
<td>Raised if an exception occurs while creating the NamedPipeServerStream (Inherited from PipeServerTMessage.)</td>
</tr>
</tbody>
</table>

[^Top]

## See Also

Reference
PipeServer Class
W.IO.Pipes Namespace
Tungsten

$W$
PipeServer Fields

The PipeServer type exposes the following members.

Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_shouldListen</td>
<td>(Inherited from Pipe.)</td>
</tr>
</tbody>
</table>

See Also

Reference

PipeServer Class
W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
PipeServer <TMessage> Class

The generic version of PipeServer. This class expects all messages to be of the specified type.

Inheritance Hierarchy

- System
- Object
- W.IO.PipesPipe
- W.IO.PipesPipe <TMessage>
- W.IO.PipesPipeServer <TMessage>
- W.IO.PipesPipeServer

Namespace: W.IO.Pipes
Assembly: Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll) Version: 2.0.3

Syntax

```csharp
public class PipeServer<TMessage> : Pipe<TMessage>
```

Type Parameters

- <TMessage>
  - The message type to send and receive

The PipeServer<TMessage> type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PipeServer&lt;TMessage&gt;</td>
<td>Initializes a new instance of the</td>
</tr>
</tbody>
</table>
PipeServerTMessage
class

Top

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>A unique id for this Pipe (Inherited from Pipe.)</td>
</tr>
<tr>
<td>Stream</td>
<td>The PipeStream on which to send and receive data (Inherited from Pipe.)</td>
</tr>
</tbody>
</table>

Top

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispose</td>
<td>Disconnects and disposes the pipe (Inherited from Pipe.)</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object. (Inherited from Object.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>Listen</td>
<td>Continuously waits for data from the pipe. The BytesReceived event is raised when data arrives, then immediately waits for more data. (Inherited from Pipe.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>OnDispose</td>
<td>Disposes the PipeServer and release resources (Overrides PipeOnDispose.)</td>
</tr>
<tr>
<td>OnListen</td>
<td>Continuously waits for data from the pipe. The BytesReceived event is raised when data arrives, then immediately waits for more data. (Inherited from PipeTMessage.)</td>
</tr>
<tr>
<td>OnStopListening</td>
<td>After the next bytes received, stops waiting for data</td>
</tr>
</tbody>
</table>
RaiseDisconnection

Hooks the Disconnected event. Pass in an exception if desired.
(Inherited from Pipe.)

RaiseMessageReceived

Called by the PipeExtensions extension methods to raise the MessageReceived event
(Inherited from PipeTMessage.)

StopListening

After the next bytes received, stops waiting for data
(Inherited from Pipe.)

ToString

Returns a string that represents the current object.
(Inherited from Object.)

WaitForConnection

Creates a new NamedPipeServerStream and Waits for a client to connect

---

**Events**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connected</td>
<td>Raised when a client has connected to the server</td>
</tr>
</tbody>
</table>
Disconnected
Raised when the pipe has disconnected
(Inherited from Pipe.)

MessageReceived
Raised when a message has been received
(Inherited from PipeTMessage.)

StartException
Raised if an exception occurs while creating the
NamedPipeServerStream

Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_shouldListen</td>
<td>(Inherited from Pipe.)</td>
</tr>
</tbody>
</table>

Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot; This method should be called in the constructor of any class which has IOwnedProperty members</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>InLock(Action)</code></td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><code>InLockTType(Func&lt;TType&gt;)</code></td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><code>InLockAsync(Action)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><code>InLockAsyncTType(Func&lt;TType&gt;)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><code>IsDirty</code></td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td><code>Lock</code></td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><code>MarkAsClean</code></td>
<td>Scans each field and property of type IProperty</td>
</tr>
</tbody>
</table>
and sets its IsDirty flag to false
(Defined by PropertyHostExtensions.)

- **ReadTMessage**
  Waits for a message to be read from the pipe
  (Defined by PipeReadWriteExtensions.)

- **ReadAsyncTMessage**
  Waits for a message to be read from the pipe
  (Defined by PipeReadWriteExtensions.)

- **Unlock**
  Performs a Monitor unlock
  (Defined by MonitorExtensions.)

- **WriteTMessage**
  Write a message to the pipe
  (Defined by PipeReadWriteExtensions.)

- **WriteAsyncTMessage**
  Asynchronously write a message to the pipe
  (Defined by PipeReadWriteExtensions.)

See Also

Reference
W.IO.Pipes Namespace
PipeServer\textit{TMessage} Constructor

Initializes a new instance of the \textit{PipeServerTMessage} class

\textbf{Namespace:} W.IO.Pipes
\textbf{Assembly:} Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll) Version: 2.0.3

\section*{Syntax}

\begin{verbatim}
public PipeServer()
\end{verbatim}

\section*{See Also}

Reference
\textit{PipeServerTMessage Class}
\textit{W.IO.Pipes Namespace}

Copyright @ 2018 Jordan Duerksen
Tungsten

$W^n$
PipeServerTMessage Properties

The PipeServerTMessage generic type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>A unique id for this Pipe (Inherited from Pipe.)</td>
</tr>
<tr>
<td>Stream</td>
<td>The PipeStream on which to send and receive data (Inherited from Pipe.)</td>
</tr>
</tbody>
</table>

See Also

Reference

PipeServerTMessage Class

W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
The `PipeServerTMessage` generic type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Dispose icon] Dispose</td>
<td>Disconnects and disposes the pipe (Inherited from <code>Pipe</code>.)</td>
</tr>
<tr>
<td>![Equals icon] Equals</td>
<td>Determines whether the specified object is equal to the current object. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>![Finalize icon] Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>![GetHashCode icon] GetHashCode</td>
<td>Serves as the default hash function. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>![GetType icon] GetType</td>
<td>Gets the <code>Type</code> of the current instance. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>![Listen icon] Listen</td>
<td>Continuously waits for</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>OnDispose</td>
<td>Disposes the PipeServer and release resources ( Overrides PipeOnDispose.)</td>
</tr>
<tr>
<td>OnListen</td>
<td>Continuously waits for data from the pipe. The BytesReceived event is raised when data arrives, then immediately waits for more data.</td>
</tr>
<tr>
<td>OnStopListening</td>
<td>After the next bytes received, stops waiting for data (Inherited from Pipe.)</td>
</tr>
<tr>
<td>RaiseDisconnection</td>
<td>Raises the Disconnected event. Pass in an exception if desired. (Inherited from Pipe.)</td>
</tr>
<tr>
<td>RaiseMessageReceived</td>
<td>Called by the PipeExtensions extension methods to</td>
</tr>
</tbody>
</table>
raise the MessageReceived event (Inherited from PipeTMessage.)

- **StopListening**
  - After the next bytes received, stops waiting for data
  - (Inherited from Pipe.)

- **ToString**
  - Returns a string that represents the current object.
  - (Inherited from Object.)

- **WaitForConnection**
  - Creates a new NamedPipeServerStream and Waits for a client to connect

---

### Top

#### Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>✽ ‡ <strong>AsTType</strong></td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions)</td>
</tr>
<tr>
<td>✽ ‡ <strong>InitializeProperties</strong></td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>InLock(Action)</strong></td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by MonitorExtensions.).</td>
</tr>
<tr>
<td><strong>InLockTType(FuncTType)</strong></td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.).</td>
</tr>
<tr>
<td><strong>InLockAsync(Action)</strong></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.).</td>
</tr>
<tr>
<td><strong>InLockAsyncTType(FuncTType)</strong></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.).</td>
</tr>
<tr>
<td><strong>IsDirty</strong></td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.).</td>
</tr>
<tr>
<td><strong>Lock</strong></td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.).</td>
</tr>
<tr>
<td><strong>MarkAsClean</strong></td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ReadTMessage</td>
<td>Waits for a message to be read from the pipe</td>
</tr>
<tr>
<td>ReadAsyncTMessage</td>
<td>Waits for a message to be read from the pipe</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock</td>
</tr>
<tr>
<td>WriteTMessage</td>
<td>Write a message to the pipe</td>
</tr>
<tr>
<td>WriteAsyncTMessage</td>
<td>Asynchronously write a message to the pipe</td>
</tr>
</tbody>
</table>

See Also

Reference
PipeServerTMessage Class
W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
PipeServer TMessage OnDispose Method

Disposes the PipeServer and release resources

**Namespace:** W.IO.Pipes  
**Assembly:** Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll)  
**Version:** 2.0.3

### Syntax

```csharp
protected override void OnDispose()
```

### See Also

Reference
- PipeServer TMessage Class
- W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$^7\text{W}$
PipeServer \texttt{TMessage WaitForConnection} Method

Creates a new NamedPipeServerStream and Waits for a client to connect

\textbf{Namespace:} W.IO.Pipes  
\textbf{Assembly:} Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll)  
\textbf{Version:} 2.0.3

\section*{Syntax}

\begin{verbatim}
public bool WaitForConnection(
    string pipeName,
    int maxConnections = -1
)
\end{verbatim}

\section*{Parameters}

\textit{pipeName}  
\begin{itemize}
  \item Type: \texttt{System.String}
  \item The name of the pipe
\end{itemize}

\textit{maxConnections (Optional)}  
\begin{itemize}
  \item Type: \texttt{System.Int32}
  \item The maximum number of pipes with this name
\end{itemize}

\section*{Return Value}

\begin{itemize}
  \item Type: \texttt{Boolean}
  \item True if the server was created and is waiting, otherwise False
\end{itemize}

\section*{See Also
Reference

PipeServerTMessage Class
W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
PipeServerTMessage Events

The PipeServerTMessage generic type exposes the following members.

Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connected</td>
<td>Raised when a client has connected to the server</td>
</tr>
<tr>
<td>Disconnected</td>
<td>Raised when the pipe has disconnected (Inherited from Pipe. )</td>
</tr>
<tr>
<td>MessageReceived</td>
<td>Raised when a message has been received (Inherited from PipeTMessage.)</td>
</tr>
<tr>
<td>StartException</td>
<td>Raised if an exception occurs while creating the NamedPipeServerStream</td>
</tr>
</tbody>
</table>

See Also

Reference
PipeServerTMessage Class
W.IO.Pipes Namespace
PipeServer

TMessage

Connected

Event

Raised when a client has connected to the server

**Namespace:** W.IO.Pipes

**Assembly:** Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll)  Version: 2.0.3

**Syntax**

```csharp
public event Action<Pipe> Connected
```

**Value**

Type: SystemActionPipe

**See Also**

Reference

PipeServerTMessage Class

W.IO.Pipes Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

W
PipeServer

TMessage

StartException

Event

Raised if an exception occurs while creating the NamedPipeServerStream

Namespace: W.IO.Pipes
Assembly: Tungsten.IO.Pipes (in Tungsten.IO.Pipes.dll) Version: 2.0.3

Syntax

C#

```csharp
public event Action<Pipe, Exception> StartException;
```

Value

Type: System.Action<Pipe, Exception>

See Also

Reference

PipeServer TMessage Class
W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
PipeServerTMessage Fields

The PipeServerTMessage generic type exposes the following members.

Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_shouldListen</td>
<td>(Inherited from Pipe.)</td>
</tr>
</tbody>
</table>

See Also

Reference
PipeServerTMessage Class
W.IO.Pipes Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
# W.Logging Namespace

[Missing `<summary>` documentation for "N:W.Logging"]

## Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log</td>
<td>Exposes static methods for logging. LogTheMessage can be assigned a new value for customized logging.</td>
</tr>
<tr>
<td>LogMessageHistory</td>
<td>Maintains a history of Log information</td>
</tr>
<tr>
<td>LogMessageHistoryLogMessage</td>
<td>An individual log message</td>
</tr>
<tr>
<td>LogPropertyHost</td>
<td>Provides</td>
</tr>
</tbody>
</table>

## Enumerations

<table>
<thead>
<tr>
<th>Enumeration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LogLogMessageCategory</td>
<td>The log message type</td>
</tr>
<tr>
<td>Tungsten</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td></td>
</tr>
<tr>
<td>$W$</td>
<td></td>
</tr>
</tbody>
</table>
Log Class

Exposes static methods for logging. LogTheMessage can be assigned a new value for customized logging.

Inheritance Hierarchy

- System
  - Object
  - W.Logging

Namespace: W.Logging
Assembly: Tungsten.Logging (in Tungsten.Logging.dll) Version: 2.0.0

Syntax

```
public static class Log
```

The Log type exposes the following members.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddTimestamp</td>
<td>If True, log messages will be prefixed with a timestamp</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>e(Exception)</td>
<td>Log an Exception</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>e(String, Object)</td>
<td>Log a formatted exception message. This method uses <code>string.Format</code> to format the message.</td>
</tr>
<tr>
<td>i(String, Object)</td>
<td>Log a formatted informational message. This method uses <code>string.Format</code> to format the message.</td>
</tr>
<tr>
<td>i(String, String, Int32)</td>
<td>Log a formatted informational message. This method uses <code>string.Format</code> to format the message.</td>
</tr>
<tr>
<td>v(String, Object)</td>
<td>Log a formatted verbose message. This method uses <code>string.Format</code> to format the message.</td>
</tr>
<tr>
<td>v(String, String, Int32)</td>
<td>Log a formatted verbose message. This method uses <code>string.Format</code> to format the message.</td>
</tr>
<tr>
<td>w(String, Object)</td>
<td>Log a formatted warning message. This method uses <code>string.Format</code> to format the message.</td>
</tr>
<tr>
<td>w(String, String, Int32)</td>
<td>Log a formatted warning message. This method uses <code>string.Format</code> to format the message.</td>
</tr>
</tbody>
</table>

Top

**Events**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>✨ LogTheMessage</td>
<td>Configure this Action to log</td>
</tr>
</tbody>
</table>
messages the way you like to

See Also

Reference
W.Logging Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
Log Properties

The Log type exposes the following members.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddTimestamp</td>
<td>If True, log messages will be prefixed with a timestamp</td>
</tr>
</tbody>
</table>

See Also

Reference
Log Class
W.Logging Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
LogAddTimestamp Property

If True, log messages will be prefixed with a timestamp

Namespace: W.Logging
Assembly: Tungsten.Logging (in Tungsten.Logging.dll) Version: 2.0.0

Syntax

```c#
public static bool AddTimestamp { get; set; }
```

Property Value
Type: Boolean

See Also

Reference
Log Class
W.Logging Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
# Log Methods

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>e( )</td>
<td>Log an Exception</td>
</tr>
<tr>
<td>e(String, Object)</td>
<td>Log a formatted exception message. This method uses string.Format to format the message.</td>
</tr>
<tr>
<td>i(String, Object)</td>
<td>Log a formatted informational message. This method uses string.Format to format the message.</td>
</tr>
<tr>
<td>i(String, String, Int32)</td>
<td>Log a formatted informational message. This method uses string.Format to format the message.</td>
</tr>
<tr>
<td>v(String, Object)</td>
<td>Log a formatted verbose message. This method uses string.Format to format the message.</td>
</tr>
<tr>
<td>v(String, String, Int32)</td>
<td>Log a formatted verbose message. This method uses string.Format to format the message.</td>
</tr>
<tr>
<td>w(String, Object)</td>
<td>Log a formatted warning message. This method uses string.Format to format the message.</td>
</tr>
</tbody>
</table>
Log a formatted warning message. This method uses `string.Format` to format the message.

**See Also**

Reference
Log Class
W.Logging Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
Log Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>e(Exception)</td>
<td>Log an Exception</td>
</tr>
<tr>
<td>e(String, Object)</td>
<td>Log a formatted exception message. This method uses string.Format to format the message.</td>
</tr>
</tbody>
</table>

See Also

Reference
- Log Class
- W.Logging Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
Log Method (Exception)

Log an Exception

**Namespace:** W.Logging  
**Assembly:** Tungsten.Logging (in Tungsten.Logging.dll) Version: 2.0.0

### Syntax

```csharp
public static void Log(Exception e)
```

### Parameters

- `e`  
  Type: `SystemException`  
  The exception to log. This will be boxed with `ToString()`.

### See Also

- Reference  
  - Log Class  
  - `Log` Overload  
  - W.Logging Namespace

---

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
Log a formatted exception message. This method uses string.Format to format the message.

**Namespace:** W.Logging  
**Assembly:** Tungsten.Logging (in Tungsten.Logging.dll) Version: 2.0.0

### Syntax

```csharp
public static void e(
    string format,
    params Object[] args
)
```

### Parameters

- **format**  
  *Type: System.String*  
  Format of the message

- **args**  
  *Type: System.Object*  
  Parameters to be passed during message formatting

### See Also

- Reference
  - Log Class
  - e Overload
  - W.Logging Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
Logi Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>i(String, Object)</td>
<td>Log a formatted informational message. This method uses string.Format to format the message.</td>
</tr>
<tr>
<td>i(String, String, Int32)</td>
<td>Log a formatted informational message. This method uses string.Format to format the message.</td>
</tr>
</tbody>
</table>

See Also

Reference
Log Class
W.Logging Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
Logi Method (String, Object)

Log a formatted informational message. This method uses string.Format to format the message.

**Namespace:** W.Logging  
**Assembly:** Tungsten.Logging (in Tungsten.Logging.dll) Version: 2.0.0

### Syntax

```
public static void i(
    string format,
    params Object[] args
)
```

### Parameters

- **format**
  
  *Type:* System.String  
  *Format of the message*

- **args**
  
  *Type:* System.Object  
  *Parameters to be passed during message formatting*

### See Also

- **Reference**  
  *Log Class*  
  *i Overload*  
  *W.Logging Namespace*

---

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
</tr>
</tbody>
</table>
Logi Method (String, String, Int32)

Log a formatted informational message. This method uses string.Format to format the message.

Namespace: W.Logging  
Assembly: Tungsten.Logging (in Tungsten.Logging.dll) Version: 2.0.0

Syntax

```csharp
public static void i(
    string message,
    string callerName = "",
    int callerLineNumber = 0
)
```

Parameters

- `message`  
  Type: System.String
  Format of the message

- `callerName (Optional)`
  Type: System.String
  The name of the caller

- `callerLineNumber (Optional)`
  Type: System.Int32
  The line number of the caller

See Also

Reference
Tungsten

$\text{W}$
Logv Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>v(String, Object)</strong></td>
<td>Log a formatted verbose message. This method uses string.Format to format the message.</td>
</tr>
<tr>
<td><strong>v(String, String, Int32)</strong></td>
<td>Log a formatted verbose message. This method uses string.Format to format the message.</td>
</tr>
</tbody>
</table>

See Also

Reference
Log Class
W.Logging Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
Logv Method (String, Object)

Log a formatted verbose message. This method uses string.Format to format the message.

**Namespace:** W.Logging  
**Assembly:** Tungsten.Logging (in Tungsten.Logging.dll) Version: 2.0.0

### Syntax

```
public static void v(
    string format,
    params Object[] args
)
```

### Parameters

- **format**
  - Type: `System.String`
  - Format of the message

- **args**
  - Type: `System.Object`
  - Parameters to be passed during message formatting

### See Also

- Reference
  - Log Class
  - v Overload
  - W.Logging Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
Logv Method (String, String, Int32)

Log a formatted verbose message. This method uses string.Format to format the message.

**Namespace:** W.Logging  
**Assembly:** Tungsten.Logging (in Tungsten.Logging.dll) Version: 2.0.0

### Syntax

```
public static void v(
    string message,
    string callerName = "",
    int callerLineNumber = 0
)
```

### Parameters

- **message**
  - Type: System.String
  - Format of the message

- **callerName (Optional)**
  - Type: System.String
  - The name of the caller

- **callerLineNumber (Optional)**
  - Type: System.Int32
  - The line number of the caller

### See Also

Reference
Log Class
v Overload
W.Logging Namespace

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>$^\text{W}$</td>
</tr>
</tbody>
</table>
Logw Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>w(String, Object)</td>
<td>Log a formatted warning message. This method uses string.Format to format the message.</td>
</tr>
<tr>
<td>w(String, String, Int32)</td>
<td>Log a formatted warning message. This method uses string.Format to format the message.</td>
</tr>
</tbody>
</table>

See Also

Reference
Log Class
W.Logging Namespace
Tungsten

$W$
Logw Method (String, Object)

Log a formatted warning message. This method uses string.Format to format the message.

Namespace: W.Logging
Assembly: Tungsten.Logging (in Tungsten.Logging.dll) Version: 2.0.0

Syntax

```csharp
public static void w(
    string format,
    params Object[] args
)
```

Parameters

- **format**
  - Type: SystemString
  - Format of the message

- **args**
  - Type: SystemObject
  - Parameters to be passed during message formatting

See Also

Reference
- Log Class
- w Overload
- W.Logging Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
Logw Method (String, String, Int32)

Log a formatted warning message. This method uses string.Format to format the message.

Namespace: W.Logging
Assembly: Tungsten.Logging (in Tungsten.Logging.dll) Version: 2.0.0

Syntax

```csharp
public static void w(
    string message,
    string callerName = "",
    int callerLineNumber = 0
)
```

Parameters

message
Type: System.String
Format of the message
callerName (Optional)
Type: System.String
The name of the caller
callerLineNumber (Optional)
Type: System.Int32
The line number of the caller

See Also

Reference
Tungsten

$W$
Log Events

The Log type exposes the following members.

### Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LogTheMessage</td>
<td>Configure this Action to log messages the way you like to</td>
</tr>
</tbody>
</table>

See Also

Reference

Log Class
W.Logging Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
LogTheMessage Event

Configure this Action to log messages the way you like to

**Namespace:** W.Logging  
**Assembly:** Tungsten.Logging (in Tungsten.Logging.dll) Version: 2.0.0

**Syntax**

```csharp
public static event Action<string> LogTheMessage
```

**Value**  
Type: SystemAction<string>

**Remarks**  
This method needs to be replaced with something useful to you.

**See Also**

Reference  
Log Class  
W.Logging Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
LogMessageCategory Enumeration

The log message type

Namespace: W.Logging
Assembly: Tungsten.Logging (in Tungsten.Logging.dll) Version: 2.0.0

Syntax

```csharp
public enum LogMessageCategory
```

Members

<table>
<thead>
<tr>
<th>Member name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verbose</td>
<td>0</td>
<td>Denotes verbose message</td>
</tr>
<tr>
<td>Information</td>
<td>1</td>
<td>Denotes a informational message</td>
</tr>
<tr>
<td>Warning</td>
<td>2</td>
<td>Denotes a warning message</td>
</tr>
<tr>
<td>Error</td>
<td>3</td>
<td>Denotes an error message</td>
</tr>
</tbody>
</table>

See Also

Reference
W.Logging Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
LogMessageHistory Class

Maintains a history of Log information

Inheritance Hierarchy

- System
- Object
- W.Logging
- LogPropertyHost
- W.Logging.LogMessageHistory

Namespace: W.Logging
Assembly: Tungsten.Logging (in Tungsten.Logging.dll) Version: 2.0.0

Syntax

```csharp
public class MessageHistory : LogPropertyHost
```

The LogMessageHistory type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LogMessageHistory</td>
<td>Constructs a new LogMessageHistory</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enabled</td>
<td>If True, log messages will be</td>
</tr>
</tbody>
</table>
added to the history. If False, no history is maintained.

<table>
<thead>
<tr>
<th><strong>MaximumNumberOfMessages</strong></th>
<th>The maximum number of historical messages to maintain. When the maximum is reached, the oldest messages are removed as needed.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Messages</strong></td>
<td>The history of log messages</td>
</tr>
</tbody>
</table>

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object. (Inherited from Object.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>RaisePropertyChanged</td>
<td>Raises the PropertyChanged event. (Inherited from LogPropertyHost.)</td>
</tr>
<tr>
<td>SetValue</td>
<td>(Inherited from LogPropertyHost.)</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object. (Inherited from Object.)</td>
</tr>
</tbody>
</table>

**Events**
### Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PropertyChanged</td>
<td>Raised when the value of a property has changed (Inherited from LogPropertyHost.)</td>
</tr>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnerProperty members (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTType(FuncTType)</td>
<td>Overloaded. Performs the function in a Monitor lock</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>InLockAsync(Action)</strong></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>InLockAsyncTTType(FuncTTType)</strong></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>IsDirty</strong></td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td><strong>Lock</strong></td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>MarkAsClean</strong></td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td><strong>Unlock</strong></td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>Tungsten</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td></td>
</tr>
<tr>
<td>W</td>
<td></td>
</tr>
</tbody>
</table>
LogMessageHistory Constructor

Constructs a new LogMessageHistory

**Namespace:** W.Logging

**Assembly:** Tungsten.Logging (in Tungsten.Logging.dll) Version: 2.0.0

### Syntax

```csharp
public MessageHistory(
    bool newestFirst = true
)
```

### Parameters

**newestFirst (Optional)**

*Type: System.Boolean*  
If True, log messages are inserted at the start of the collection rather than appended to the end

### See Also

**Reference**  
LogMessageHistory Class  
W.Logging Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
# MessageHistory Properties

The `LogMessageHistory` type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Enabled</strong></td>
<td>If True, log messages will be added to the history. If False, no history is maintained.</td>
</tr>
<tr>
<td><strong>MaximumNumberOfMessages</strong></td>
<td>The maximum number of historical messages to maintain. When the maximum is reached, the oldest messages are removed as needed.</td>
</tr>
<tr>
<td><strong>Messages</strong></td>
<td>The history of log messages</td>
</tr>
</tbody>
</table>

## See Also

Reference
LogMessageHistory Class
W.Logging Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
LogMessageHistoryEnabled Property

If True, log messages will be added to the history. If False, no history is maintained.

Namespace: W.Logging
Assembly: Tungsten.Logging (in Tungsten.Logging.dll) Version: 2.0.0

Syntax

```c#
public bool Enabled { get; set; }
```

Property Value
Type: Boolean

See Also

Reference
LogMessageHistory Class
W.Logging Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
LogMessageHistoryMaximumNumberOfMessages Property

The maximum number of historical messages to maintain. When the maximum is reached, the oldest messages are removed as needed.

**Namespace:**  W.Logging  
**Assembly:**  Tungsten.Logging (in Tungsten.Logging.dll) Version: 2.0.0

### Syntax

```csharp
public int MaximumNumberOfMessages { get; }
```

### Property Value

Type:  Int32

### See Also

Reference

LogMessageHistory Class

W.Logging Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
LogMessageHistoryMessages Property

The history of log messages

**Namespace:** W.Logging  
**Assembly:** Tungsten.Logging (in Tungsten.Logging.dll) Version: 2.0.0

### Syntax

```
public ObservableCollection<LogMessageHistoryLogMessage> LogMessageHistory
```

**Property Value**

Type: `ObservableCollection<LogMessageHistoryLogMessage>`

### See Also

Reference

- LogMessageHistory Class
- W.Logging Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
# MessageHistory Methods

The `LogMessageHistory` type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td><em>(Inherited from <code>Object</code>)</em></td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td><em>(Inherited from <code>Object</code>)</em></td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td></td>
<td><em>(Inherited from <code>Object</code>)</em></td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <code>Type</code> of the current instance.</td>
</tr>
<tr>
<td></td>
<td><em>(Inherited from <code>Object</code>)</em></td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow</td>
</tr>
</tbody>
</table>
RaiseOnPropertyChanged

Raises the PropertyChanged event
(Inherited from LogPropertyHost.)

SetValue

(Inherited from LogPropertyHost.)

ToString

Returns a string that represents the current object.
(Inherited from Object.)

---

Top

Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot; This method should be called in the constructor of any class which has IOwnedProperty members</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTTType(FuncTTType)</td>
<td>Overloaded. Performs the function in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTTType(FuncTTType)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>

**See Also**

Reference
- LogMessageHistory Class
- W.Logging Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
MessageHistory Events

The LogMessageHistory type exposes the following members.

Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PropertyChanged</td>
<td>Raised when the value of a property has changed (Inherited from LogPropertyHost.)</td>
</tr>
</tbody>
</table>

See Also

Reference

- LogMessageHistory Class
- W.Logging Namespace

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
</tr>
</tbody>
</table>
LogMessageHistoryLogMessage Class

An individual log message

Inheritance Hierarchy

System\Object  W.Logging\LogPropertyHost
W.Logging\LogMessageHistory\LogMessage

Namespace: W.Logging
Assembly: Tungsten.Logging (in Tungsten.Logging.dll) Version: 2.0.0

Syntax

```c#
public class LogMessage : LogPropertyHost
```

The LogMessageHistoryLogMessage type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![logmessagehistorylogmessage]</td>
<td>Constructs a new LogMessage</td>
</tr>
</tbody>
</table>

Properties
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Message</strong></td>
<td>The logged message</td>
</tr>
</tbody>
</table>

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current object.</td>
</tr>
</tbody>
</table>
RaiseOnPropertyChanged

Raises the PropertyChanged event (Inherited from LogPropertyHost.)

SetValue

(Inherited from LogPropertyHost.)

ToString

Returns a string that represents the current object. (Inherited from Object.)

Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PropertyChanged</td>
<td>Raised when the value of a property has changed (Inherited from LogPropertyHost.)</td>
</tr>
</tbody>
</table>

Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTTType(FuncTTType)</td>
<td>Overloaded. Performs the function in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTTType(FuncTTType)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty</td>
</tr>
<tr>
<td></td>
<td>(Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock</td>
</tr>
<tr>
<td></td>
<td>(Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to</td>
</tr>
<tr>
<td></td>
<td>false</td>
</tr>
<tr>
<td></td>
<td>(Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock</td>
</tr>
<tr>
<td></td>
<td>(Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>

**Top**

**See Also**

Reference

W.Logging Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
LogMessageHistory

LogMessage Constructor

Constructs a new LogMessage

**Namespace:** W.Logging  
**Assembly:** Tungsten.Logging (in Tungsten.Logging.dll) Version: 2.0.0

## Syntax

```c#
public LogMessage()
```

## See Also

Reference

LogMessageHistory

LogMessage Class

W.Logging Namespace
Tungsten

$W$
LogMessage Properties

The LogMessageHistoryLogMessage type exposes the following members.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message</td>
<td>The logged message</td>
</tr>
</tbody>
</table>

See Also

Reference
LogMessageHistoryLogMessage Class
W.Logging Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
LogMessageHistoryLogMessageMessage

The logged message

**Namespace:** W.Logging
**Assembly:** Tungsten.Logging (in Tungsten.Logging.dll) Version: 2.0.0

⚠️ Syntax

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

Property Value
Type: String

⚠️ See Also

Reference
LogMessageHistoryLogMessage Class
W.Logging Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
LogMessage Methods

The LogMessageHistoryLogMessage type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
</tbody>
</table>
MemberwiseClone

Creates a shallow copy of the current Object. (Inherited from Object.)

RaiseOnPropertyChanged

Raises the PropertyChanged event (Inherited from LogPropertyHost.)

SetValue

(Inherited from LogPropertyHost.)

ToString

Returns a string that represents the current object. (Inherited from Object.)

---

Top

Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTType(FuncTType)</td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTType(FuncTType)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock</td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock. (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>

See Also

Reference

- LogMessageHistory
- LogMessage Class
- W.Logging Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

\( W \)
LogMessage Events

The LogMessageHistoryLogMessage type exposes the following members.

## Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PropertyChanged</td>
<td>Raised when the value of a property has changed (Inherited from LogPropertyHost.)</td>
</tr>
</tbody>
</table>

See Also

Reference
- LogMessageHistoryLogMessage Class
- W.Logging Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
LogPropertyHost Class

Provides

Inheritance Hierarchy

SystemObject  W.LoggingLogPropertyHost
  W.LoggingLogMessageHistory
  W.LoggingLogMessageHistoryLogMessage

Namespace:  W.Logging
Assembly:  Tungsten.Logging (in Tungsten.Logging.dll) Version: 2.0.0

Syntax

```csharp
public class PropertyHost : INotifyPropertyChanged
```

The LogPropertyHost type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LogPropertyHost</td>
<td>Initializes a new instance of the LogPropertyHost class</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as the default hash function. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>RaiseOnPropertyChanged</strong></td>
<td>Raises the PropertyChanged event</td>
</tr>
<tr>
<td><strong>SetValue</strong></td>
<td></td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Returns a string that</td>
</tr>
</tbody>
</table>
represents the current object. (Inherited from Object.)

Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PropertyChanged</td>
<td>Raised when the value of a property has changed</td>
</tr>
</tbody>
</table>

Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a</td>
</tr>
</tbody>
</table>
Monitor lock
(Defined by MonitorExtensions.)

- **InLock<TType(Func<TType)>**
  Overloaded. Performs the function in a Monitor lock
  (Defined by MonitorExtensions.)

- **InLockAsync(Action)**
  Overloaded. Asynchronously performs the action in a Monitor lock
  (Defined by MonitorExtensions.)

- **InLockAsync<TType(Func<TType)>**
  Overloaded. Asynchronously performs the action in a Monitor lock
  (Defined by MonitorExtensions.)

- **IsDirty**
  Scans the IsDirty value of each field and property of type IProperty
  (Defined by PropertyHostExtensions.)

- **Lock**
  Performs a Monitor lock
  (Defined by MonitorExtensions.)

- **MarkAsClean**
  Scans each field and property of type IProperty and sets its IsDirty flag to false
  (Defined by
Unlock

Performs a Monitor unlock (Defined by MonitorExtensions.)

See Also

Reference
W.Logging Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
LogPropertyHost Constructor

Initializes a new instance of the LogPropertyHost class

**Namespace:** W.Logging  
**Assembly:** Tungsten.Logging (in Tungsten.Logging.dll) Version: 2.0.0

### Syntax

```csharp
public PropertyHost()
```

### See Also

**Reference**  
LogPropertyHost Class  
W.Logging Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$\text{W}$
## PropertyHost Methods

The `LogPropertyHost` type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Equals</code></td>
<td>Determines whether the specified object is equal to the current object. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><code>Finalize</code></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><code>GetHashCode</code></td>
<td>Serves as the default hash function. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><code>GetType</code></td>
<td>Gets the <code>Type</code> of the current instance. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><code>MemberwiseClone</code></td>
<td>Creates a shallow</td>
</tr>
</tbody>
</table>
copy of the current `Object`.
(Inherited from `Object`.)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RaiseOnPropertyChanged</td>
<td>Raises the PropertyChanged event</td>
</tr>
<tr>
<td>SetValue</td>
<td></td>
</tr>
</tbody>
</table>
| ToString                  | Returns a string that represents the current object.
  (Inherited from `Object`.) |

**Extension Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by <code>AsExtensions</code>.)</td>
</tr>
<tr>
<td>InitializeProps</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot; This method should be called in the constructor of any class which has IOwnedProperty members (Defined by <code>PropertyHostExtensions</code>.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Performed the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTTTType(FuncTTTType)</td>
<td>Performed the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Asynchronously performed the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTTTType(FuncTTTType)</td>
<td>Asynchronously performed the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scanned the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performed a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scanned each field and property of type IProperty and set its IsDirty flag to</td>
</tr>
</tbody>
</table>
See Also

Reference
LogPropertyHost Class
W.Logging Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
LogPropertyHostRaiseOnPropertyChanged Method

Raises the PropertyChanged event

**Namespace:**  W.Logging  
**Assembly:**  Tungsten.Logging (in Tungsten.Logging.dll) Version: 2.0.0

## Syntax

```csharp
protected void RaisePropertyChanged(
    Object sender,
    string propertyName
)
```

### Parameters

**sender**
- Type: `System.Object`
- The sender is the owner of the property

**propertyName**
- Type: `System.String`
- The name of the caller (the property which changed)

## See Also

Reference
- LogPropertyHost Class
- W.Logging Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
LogPropertyHostSetValue Method

[Missing <summary> documentation for
]

Namespace: W.Logging
Assembly: Tungsten.Logging (in Tungsten.Logging.dll) Version: 2.0.0

Syntax

```c#
protected virtual void SetValue(
    Object owner,
    Action assignValue,
    string callerMemberName = ""
)
```

Parameters

owner
Type: SystemObject
[Missing <param name="owner"/> documentation for

assignValue
Type: SystemAction
[Missing <param name="assignValue"/> documentation for

callerMemberName (Optional)
Type: SystemString
[Missing <param name="callerMemberName"/> documentation for


See Also

Reference

LogPropertyHost Class
W.Logging Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$\text{W}$
PropertyHost Events

The LogPropertyHost type exposes the following members.

**Events**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PropertyChangedEventArgs</td>
<td>Raised when the value of a property has changed</td>
</tr>
</tbody>
</table>

See Also

Reference
LogPropertyHost Class
W.Logging Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
LogPropertyHostPropertyChanged Event

Raised when the value of a property has changed

**Namespace:** W.Logging  
**Assembly:** Tungsten.Logging (in Tungsten.Logging.dll) Version: 2.0.0

### Syntax

```csharp
public event PropertyChangedEventHandler PropertyChanged
```

**Value**
Type: `System.ComponentModel.PropertyChangedEventHandler`

**Implements**
`INotifyPropertyChanged`, `PropertyChanged`

### See Also

**Reference**
- LogPropertyHost Class
- W.Logging Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
# W.Net Namespace

[Missing `<summary>` documentation for "N:W.Net"]

## Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="ExtensionMethods" /></td>
<td>Extension methods for W.Net</td>
</tr>
<tr>
<td><img src="image" alt="Tcp" /></td>
<td></td>
</tr>
<tr>
<td><img src="image" alt="TcpGeneric" /></td>
<td></td>
</tr>
<tr>
<td><img src="image" alt="TcpGenericSecureTcpClientTMessage" /></td>
<td>Sends log messages to a remote server via secure TCP</td>
</tr>
<tr>
<td><img src="image" alt="TcpGenericSecureTcpHostTMessage" /></td>
<td></td>
</tr>
<tr>
<td><img src="image" alt="TcpGenericTcpClientTMessage" /></td>
<td></td>
</tr>
<tr>
<td><img src="image" alt="TcpGenericTcpHostTMessage" /></td>
<td></td>
</tr>
<tr>
<td><img src="image" alt="TcpSecureTcpClient" /></td>
<td></td>
</tr>
<tr>
<td><img src="image" alt="TcpSecureTcpHost" /></td>
<td></td>
</tr>
<tr>
<td><img src="image" alt="TcpSecureTcpLogger" /></td>
<td></td>
</tr>
<tr>
<td><img src="image" alt="TcpTcpClient" /></td>
<td></td>
</tr>
<tr>
<td><img src="image" alt="TcpTcpHost" /></td>
<td></td>
</tr>
</tbody>
</table>
### Interfaces

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TcpTcpLogger</strong></td>
<td>Sends log messages to a remote server via Tcp</td>
</tr>
<tr>
<td><strong>Udp</strong></td>
<td>Provides simple UDP reading and writing</td>
</tr>
<tr>
<td><strong>UdpGeneric</strong></td>
<td>Contains the generic implementation of W.Net.UdpPeer</td>
</tr>
<tr>
<td><strong>UdpGenericUdpPeerTTType</strong></td>
<td>A generic Udp peer</td>
</tr>
<tr>
<td><strong>UdpUdpLogger</strong></td>
<td>Sends log messages to a remote server via Udp</td>
</tr>
<tr>
<td><strong>UdpUdpPeer</strong></td>
<td>A Udp peer</td>
</tr>
<tr>
<td><strong>WakeOnLAN</strong></td>
<td>Methods to broadcast a magic packet to wake up a machine with the given MAC address</td>
</tr>
</tbody>
</table>
TcpIClient

TcpIInitialize

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
ExtensionMethods Class

Extension methods for W.Net

Inheritance Hierarchy

* System
  * Object
  * W.Net
  * ExtensionMethods

Namespace: W.Net

Syntax

```c#
public static class ExtensionMethods
```

The `ExtensionMethods` type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetBroadcastAddress</td>
</tr>
<tr>
<td>RequestAsync(TcpSecureTcpClient, Byte, Int32)</td>
</tr>
</tbody>
</table>
RequestAsync(TcpTcpClient, Byte, Int32)

RequestAsync(UdpUdpPeer, Byte, IPEndPoint, Int32)
RequestAsync(TcpGenericSecureTcpClientTMessage, TMessage, Int32)

RequestAsync(TcpGenericTcpClientTMessage, TMessage, Int32)

RequestAsync(UdpGenericUdpPeerTMessage, TMessage, IPEndPoint, Int32)
Tungsten

$W$
The **ExtensionMethods** type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GetBroadcastAddress</strong></td>
</tr>
<tr>
<td><strong>RequestAsync(TcpSecureTcpClient, Byte, Int32)</strong></td>
</tr>
<tr>
<td><strong>RequestAsync(TcpTcpClient, Byte, Int32)</strong></td>
</tr>
</tbody>
</table>
RequestAsync(UdpUdpPeer, Byte, IPEndPoint, Int32)

RequestAsyncTMessage(TcpGenericSecureTcpClientTMessage, Int32)
RequestAsyncTMessage(TcpGenericTcpClientTMessage, TMessage, Int32)

RequestAsyncTMessage(UdpGenericUdpPeerTMessage, TMessage, IPEndPoint, Int32)
Reference
ExtensionMethods Class
W.Net Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
ExtensionMethods

GetBroadcastAddress Method

Determines the broadcast address from an ip address and subnet mask

Namespace: W.Net

Syntax

C#

```csharp
public static IPAddress GetBroadcastAddress(
    this IPAddress address,
    IPAddress subnetMask
)
```

Parameters

address
Type: System.Net.IPAddress
The IP address

subnetMask
Type: System.Net.IPAddress
The subnet mask

Return Value

Type: IPAddress
The broadcast IP address associated with the given IP address and subnet mask

Usage Note

In Visual Basic and C#, you can call this method as an instance method on any object of type IPAddress. When you use instance
method syntax to call this method, omit the first parameter. For more information, see Extension Methods (Visual Basic) or Extension Methods (C# Programming Guide).

Remarks

Taken from: https://blogs.msdn.microsoft.com/knom/2008/12/31/ip-address-calculations-with-c-subnetmasks-networks/

See Also

Reference
ExtensionMethods Class
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W^+$
## RequestAsync Method

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![ vids ] RequestAsyncTMessage(TcpGenericSecureTcpClientTMessage, Int32)</td>
<td></td>
</tr>
<tr>
<td>![ vids ] RequestAsyncTMessage(TcpGenericTcpClientTMessage, TMessage, Int32)</td>
<td></td>
</tr>
</tbody>
</table>
RequestAsync(TcpSecureTcpClient, Byte, Int32)

RequestAsync(TcpTcpClient, Byte, Int32)

RequestAsyncTMessage(UdpGenericUdpPeerTMessage, TMessage, IPEndPoint, Int32)
RequestAsync(UdpUdpPeer, Byte, IPEndPoint, Int32)

See Also

Reference

ExtensionMethods Class
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
</tr>
</tbody>
</table>
ExtensionMethods

**RequestAsync**

**(TcpGenericSecureTcpClient<TMessage>, Int32)**

Sends a request to the server and waits for a response. Can be used along with, or in lieu of, the regular method of calling Write and handling the BytesReceived event.

**Namespace:** W.Net  
**Assembly:** Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

### Syntax

```csharp
public static Task<TMessage> RequestAsync<TMessage>(
    this TcpGenericSecureTcpClient<TMessage> client
    TMessage request,
    int msTimeout
)
```

### Parameters

- **client**  
  Type: `W.NetTcpGenericSecureTcpClient<TMessage>`  
  The W.Net.Tcp.Generic.SecureTcpClient<TMessage> instance  

- **request**  
  Type: `TMessage`  
  The request to be made to the server  

- **msTimeout**  
  Type: `SystemInt32`
The time to wait for a response

Type Parameters

TMessage
The message Type

Return Value
Type: TaskTMessage
default(TMessage) if a timeout occurs, otherwise the server's response

Usage Note
In Visual Basic and C#, you can call this method as an instance method on any object of type TcpGenericSecureTcpClientTMessage. When you use instance method syntax to call this method, omit the first parameter. For more information, see Extension Methods (Visual Basic) or Extension Methods (C# Programming Guide).

See Also

Reference
ExtensionMethods Class
RequestAsync Overload
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$^W$
ExtensionMethodsRequestAsync(TM)
Method (TcpGenericTcpClient<TMessage>, Int32)

Sends a request to the server and waits for a response. Can be used along with, or in lieu of, the regular method of calling Write and handling the BytesReceived event.

**Namespace:** W.Net  
**Assembly:** Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

### Syntax

```c#
public static Task<TMessage> RequestAsync<TMessage>(
    this TcpGenericTcpClient<TMessage> client,
    TMessage request,
    int msTimeout
)
```

### Parameters

**client**
- **Type:** `W.NetTcpGenericTcpClient<TMessage>`
- The W.Net.Tcp.Generic.TcpClient<TMessage> instance

**request**
- **Type:** `TMessage`
- The request to be made to the server

**msTimeout**
- **Type:** `System.Int32`
- The time to wait for a response
Type Parameters

_TMessage_

  The message Type

Return Value
Type: Task_Task_TMessage_
default(TMessage) if a timeout occurs, otherwise the server's response

Usage Note
In Visual Basic and C#, you can call this method as an instance method on any object of type TcpGenericTcpClient_TMessage_. When you use instance method syntax to call this method, omit the first parameter. For more information, see Extension Methods (Visual Basic) or Extension Methods (C# Programming Guide).

See Also

Reference
  ExtensionMethods Class
  RequestAsync Overload
  W.Net Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
ExtensionMethodsRequestAsync Method (TcpSecureTcpClient, Byte, Int32)

Sends a request to the server and waits for a response. Can be used along with, or in lieu of, the regular method of calling Write and handling the BytesReceived event.

**Namespace:** W.Net  
**Assembly:** Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

## Syntax

```csharp
public static Task<byte[]> RequestAsync(
    this TcpSecureTcpClient client,
    byte[] request,
    int msTimeout
)
```

### Parameters

**client**
- Type: `W.NetTcpSecureTcpClient`
- The W.Net.Tcp.SecureTcpClient instance

**request**
- Type: `SystemByte`
- The request to be made to the server

**msTimeout**
- Type: `SystemInt32`
- The time to wait for a response
Return Value
Type: TaskByte
Null if a timeout occurs, otherwise the server's response

Usage Note
In Visual Basic and C#, you can call this method as an instance method on any object of type TcpSecureTcpClient. When you use instance method syntax to call this method, omit the first parameter. For more information, see Extension Methods (Visual Basic) or Extension Methods (C# Programming Guide).

See Also
Reference
ExtensionMethods Class
RequestAsync Overload
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
ExtensionMethodsRequestAsync Method (TcpTcpClient, Byte, Int32)

Sends a request to the server and waits for a response. Can be used along with, or in lieu of, the regular method of calling Write and handling the BytesReceived event.

**Namespace:** W.Net  
**Assembly:** Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

### Syntax

```csharp
public static Task<byte[]> RequestAsync(
    this TcpTcpClient client,
    byte[] request,
    int msTimeout
)
```

#### Parameters

- **client**
  - Type: W.NetTcpTcpClient
  - The W.Net.Tcp.TcpClient instance

- **request**
  - Type: SystemByte
  - The request to be made to the server

- **msTimeout**
  - Type: SystemInt32
  - The time to wait for a response
Return Value
Type: TaskByte
Null if a timeout occurs, otherwise the server's response

Usage Note
In Visual Basic and C#, you can call this method as an instance method on any object of type TcpClient. When you use instance method syntax to call this method, omit the first parameter. For more information, see Extension Methods (Visual Basic) or Extension Methods (C# Programming Guide).

See Also

Reference
ExtensionMethods Class
RequestAsync Overload
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
ExtensionMethods RequestAsync<TMessage> Method (UdpGenericUdpPeer<TMessage>, IPEndPoint, Int32)

Sends a request to a generic Udp peer and waits for a response. Can be used along with, or in lieu of, the regular method of calling Write and handling the BytesReceived event.

**Namespace:** W.Net  
**Assembly:** Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

### Syntax

```c#
public static Task<TMessage> RequestAsync<TMessage>(
    this UdpGenericUdpPeer<TMessage> peer,
    TMessage request,
    IPEndPoint remoteEndPoint,
    int msTimeout
)
```

### Parameters

**peer**
- Type: `W.NetUdpGenericUdpPeer<TMessage>`

**request**
- Type: `TMessage`
- The request to be made to the server

**remoteEndPoint**
- Type: `System.NetIPEndPoint`
- The IPEndPoint of the remote Udp peer
**msTimeout**

Type: **SystemInt32**
The time to wait for a response

**Type Parameters**

**TMessage**


**Return Value**

Type: **Task** *TMessage*
default(TMessage) if a timeout occurs, otherwise the server's response

**Usage Note**

In Visual Basic and C#, you can call this method as an instance method on any object of type **UdpGenericUdpPeer** *TMessage*. When you use instance method syntax to call this method, omit the first parameter. For more information, see Extension Methods (Visual Basic) or Extension Methods (C# Programming Guide).

**See Also**

- Reference
  - **ExtensionMethods Class**
  - **RequestAsync Overload**
  - **W.Net Namespace**

Copyright © 2018 Jordan Duerksen
Tungsten

$\text{W}$
ExtensionMethodsRequestAsync Method (UdpUdpPeer, Byte, IPEndPoint, Int32)

Sends a request to a Udp peer and waits for a response. Can be used along with, or in lieu of, the regular method of calling Write and handling the BytesReceived event.

**Namespace:** W.Net  
**Assembly:** Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

### Syntax

```
public static Task<byte[]> RequestAsync(
    this UdpUdpPeer peer,
    byte[] request,
    IPEndPoint remoteEndPoint,
    int mstTimeout
)
```

### Parameters

- **peer**
  - Type: W.NetUdpUdpPeer
  - The W.Net.Udp.UdpPeer instance

- **request**
  - Type: SystemByte
  - The request to be made to the server

- **remoteEndPoint**
  - Type: System.NetIPEndPoint
  - The IPEndPoint of the remote Udp peer
**msTimeout**
Type: `SystemInt32`
The time to wait for a response

**Return Value**
Type: `TaskByte`
Null if a timeout occurs, otherwise the server's response

**Usage Note**
In Visual Basic and C#, you can call this method as an instance method on any object of type `UdpUdpPeer`. When you use instance method syntax to call this method, omit the first parameter. For more information, see [Extension Methods (Visual Basic)](ExtensionMethods.html) or [Extension Methods (C# Programming Guide)](ExtensionMethods.html).

**See Also**

**Reference**
- [ExtensionMethods Class](ExtensionMethods.html)
- [RequestAsync Overload](RequestAsync.html)
- [W.Net Namespace](W.Net.html)

Copyright © 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
</tr>
</tbody>
</table>
Tcp Class

[Missing <summary> documentation for "T:W.Net.Tcp"]

Inheritance Hierarchy

- System
  - Object
    - W.NetTcp

Namespace: W.Net

Syntax

```c# public static class Tcp
```

See Also

Reference

W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
TcpGeneric Class

[Missing <summary> documentation for "T:W.Net.Tcp.Generic"]

▲ Inheritance Hierarchy

- System
- Object
- W.Net
- Tcp
- Generic

Namespace:  W.Net

▲ Syntax

```
public static class Generic
```

▲ See Also

Reference

W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
TcpGenericSecureTcpClientTMessage Class


▲ Inheritance Hierarchy

System
  Object
  W.NetTcpTcpClient
    W.NetTcpSecureTcpClient
      W.NetTcpGenericSecureTcpClientTMessage

Namespace: W.Net

▲ Syntax

C#

```csharp
public class SecureTcpClient<TMessage> : TcpSecure...
```

Type Parameters

TMessage


The TcpGenericSecureTcpClientTMessage type exposes the following members.

▲ Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
TcpGenericSecureTcpClientTMessage Initializes a new instance of the TcpGenericSecure class.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encryption</td>
<td>(Inherited from TcpSecureTcpClient.)</td>
</tr>
<tr>
<td>IsSecure</td>
<td>(Inherited from TcpSecureTcpClient.)</td>
</tr>
<tr>
<td>IsServerSide</td>
<td>(Inherited from TcpTcpClient.)</td>
</tr>
<tr>
<td>Socket</td>
<td>(Inherited from TcpTcpClient.)</td>
</tr>
</tbody>
</table>

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect</td>
<td>(Inherited from TcpSecureTcpClient.)</td>
</tr>
<tr>
<td>Disconnect</td>
<td>(Inherited from TcpTcpClient.)</td>
</tr>
<tr>
<td>Disconnect(Boolean, Boolean)</td>
<td>(Inherited from TcpTcpClient.)</td>
</tr>
<tr>
<td>Dispose</td>
<td>(Inherited from TcpTcpClient.)</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
</tbody>
</table>
**Finalize**

Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from `Object`.)

**GetHashCode**

Serves as the default hash function. (Inherited from `Object`.)

**GetType**

Gets the `Type` of the current instance. (Inherited from `Object`.)

**MemberwiseClone**

Creates a shallow copy of the current `Object`. (Inherited from `Object`.)

**OnConnect**

(Inherited from `TcpTcpClient`.)

**OnInitialize**

(Inherited from `TcpSecureTcpClient`.)

**OnReceived**

(Overrides `TcpSecureTcpClient.OnReceived(Byte)`)

**OnSend**

(Inherited from `TcpSecureTcpClient`.)

**ToString**

Returns a string that represents the current object. (Inherited from `Object`.)

**Write(TMessage)**

**Write(Byte)**

(Inherited from `TcpTcpClient`.)

---

**Events**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top</td>
<td></td>
</tr>
</tbody>
</table>

---
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BytesReceived</td>
<td>(Inherited from TcpTcpClient.)</td>
</tr>
<tr>
<td>Connected</td>
<td>(Inherited from TcpTcpClient.)</td>
</tr>
<tr>
<td>Disconnected</td>
<td>(Inherited from TcpTcpClient.)</td>
</tr>
<tr>
<td>MessageReceived</td>
<td></td>
</tr>
<tr>
<td>SecureFailed</td>
<td>(Inherited from TcpSecureTcpClient.)</td>
</tr>
</tbody>
</table>

**Extension Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AsTType</code></td>
<td>Use Generic syntax the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td><code>InitializeProperties</code></td>
<td>Scans the fields and properties of &quot;owner&quot; sets the member's Owner property to &quot;owner&quot; This method should be called in the constructor of any class which has IOwnedProperty members (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td><code>InLock(Action)</code></td>
<td>Overloaded. Performs the action Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>InLockTType(Func&lt;TType&gt;)</code></td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><code>InLockAsync(Action)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><code>InLockAsync&lt;TType&gt;(Func&lt;TType&gt;)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><code>IsDirty</code></td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions)</td>
</tr>
<tr>
<td><code>Lock</code></td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><code>MarkAsClean</code></td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions)</td>
</tr>
</tbody>
</table>
| `RequestAsync(Byte, Int32)` | Overloaded. Sends a request to t
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RequestAsync(Byte, Int32)</td>
<td>Sends a request to the server and waits for a response. Can be used along with, or in lieu of, the regular method of calling Write and handling the BytesReceived event. (Defined by ExtensionMethods.)</td>
</tr>
<tr>
<td>RequestAsyncTMessage(TMessage, Int32)</td>
<td>Overloaded. Sends a request to the server and waits for a response. Can be used along with, or in lieu of, the regular method of calling Write and handling the BytesReceived event. (Defined by ExtensionMethods.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock.</td>
</tr>
</tbody>
</table>
See Also

Reference
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>$W$</td>
</tr>
</tbody>
</table>
TcpGenericSecureTcpClient TMessage Constructor

Initializes a new instance of the TcpGenericSecureTcpClient TMessage class

**Namespace:** W.Net  
**Assembly:** Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

### Syntax

```csharp
public SecureTcpClient(
    int keySize
)
```

### Parameters

**keySize**

Type: System.Int32


### See Also

**Reference**

TcpGenericSecureTcpClient TMessage Class  
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
SecureTcpClient

Properties

The `TcpGenericSecureTcpClientTMessage` generic type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encryption</td>
<td>(Inherited from <code>TcpSecureTcpClient</code>.)</td>
</tr>
<tr>
<td>IsSecure</td>
<td>(Inherited from <code>TcpSecureTcpClient</code>.)</td>
</tr>
<tr>
<td>IsServerSide</td>
<td>(Inherited from <code>TcpTcpClient</code>.)</td>
</tr>
<tr>
<td>Socket</td>
<td>(Inherited from <code>TcpTcpClient</code>.)</td>
</tr>
</tbody>
</table>

### See Also

Reference

`TcpGenericSecureTcpClientTMessage Class`

W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
SecureTcpClient TMessage

Methods

The TcpGenericSecureTcpClient TMessage generic type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect</td>
<td>(Inherited from TcpSecureTcpClient.)</td>
</tr>
<tr>
<td>Disconnect</td>
<td>(Inherited from TcpTcpClient.)</td>
</tr>
<tr>
<td>Disconnect(Boolean, Boolean)</td>
<td>(Inherited from TcpTcpClient.)</td>
</tr>
<tr>
<td>Dispose</td>
<td>(Inherited from TcpTcpClient.)</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance (Inherited from Object.)</td>
</tr>
</tbody>
</table>
### MemberwiseClone
- **MemberwiseClone**: Creates a shallow copy of the current `Object`.  
  (Inherited from `Object`.)

### OnConnect
- **OnConnect**: (Inherited from `TcpTcpClient`.)

### OnInitialize
- **OnInitialize**: (Inherited from `TcpSecureTcpClient`.)

### OnReceived
- **OnReceived**: (Overrides `TcpSecureTcpClientOnReceived(Byte)`)

### OnSend
- **OnSend**: (Inherited from `TcpSecureTcpClient`.)

### ToString
- **ToString**: Returns a string that represents the current object.  
  (Inherited from `Object`.)

### Write(TMessage)

### Write(Byte)
- **Write(ByteArray)**: (Inherited from `TcpTcpClient`.)

---

### Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AsTType</code></td>
<td>Use Generic syntax the as operator.</td>
</tr>
<tr>
<td></td>
<td>(Defined by <code>AsExtensions</code>.)</td>
</tr>
<tr>
<td></td>
<td>Scans the fields and properties of &quot;owner&quot; sets the member's Owner property to &quot;owner&quot; This</td>
</tr>
<tr>
<td></td>
<td>method should be called in the constructor of any class which has</td>
</tr>
<tr>
<td><code>InitializeProperties</code></td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>InLock(Action)</code></td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by <code>MonitorExtensions</code>).</td>
</tr>
<tr>
<td><code>InLockTTType(Func&lt;TType&gt;)</code></td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by <code>MonitorExtensions</code>).</td>
</tr>
<tr>
<td><code>InLockAsync(Action)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by <code>MonitorExtensions</code>).</td>
</tr>
<tr>
<td><code>InLockAsyncTTType(Func&lt;TType&gt;)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by <code>MonitorExtensions</code>).</td>
</tr>
<tr>
<td><code>IsDirty</code></td>
<td>Scans the <code>IsDirty</code> value of each field and property of type <code>IProperty</code> (Defined by <code>PropertyHostExtensions</code>).</td>
</tr>
<tr>
<td><code>Lock</code></td>
<td>Performs a Monitor lock.</td>
</tr>
</tbody>
</table>
MarkAsClean

Scans each field and property of type IProperty and sets its IsDirty flag to false

RequestAsync(Byte, Int32)

Overloaded. Sends a request to the server and waits for a response. Can be used along with, or in lieu of, the regular method of calling Write and handling the BytesReceived event.

RequestAsyncTMessage(TMessage, Int32)

Overloaded. Sends a request to the server and waits for a response. Can be used along with, or in lieu of, the regular method of calling Write and handling the BytesReceived event.
along with, or in lieu of, the regular method of calling Write and handling the BytesReceived event (Defined by ExtensionMethods.)

Unlock

Performs a Monitor unlock (Defined by MonitorExtensions.)

See Also

Reference
TcpGenericSecureTcpClientTMessage Class
W.Net Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
TcpGenericSecureTcpClientTMessage Method


Namespace: W.Net

Syntax

```csharp
protected override void OnReceived(
    ref byte[] bytes
)
```

Parameters

*bytes*

Type: SystemByte


See Also

Reference

TcpGenericSecureTcpClientTMessage Class
W.Net Namespace

Copyright © 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>$W$</td>
</tr>
</tbody>
</table>
SecureTcpClient TMessage Write Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write(TMessage)</td>
<td></td>
</tr>
<tr>
<td>Write(Byte)</td>
<td>(Inherited from TcpTcpClient.)</td>
</tr>
</tbody>
</table>

See Also

Reference
TcpGenericSecureTcpClientTMessage Class
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
TcpGenericSecureTcpClient TMessage Method (TMessage)


Namespace: W.Net

Syntax

```csharp
public void Write(
    TMessage message
)
```

Parameters

*message*

Type: TMessage

[Missing <param name="message"/> documentation for "M:W.Net.Tcp.Generic.SecureTcpClient`1.Write(`0)"

See Also

Reference
TcpGenericSecureTcpClient TMessage Class
Write Overload
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
SecureTcpClient

Events

The `TcpGenericSecureTcpClientTMessage` generic type exposes the following members.

### Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BytesReceived</td>
<td>(Inherited from <code>TcpTcpClient</code>.)</td>
</tr>
<tr>
<td>Connected</td>
<td>(Inherited from <code>TcpTcpClient</code>.)</td>
</tr>
<tr>
<td>Disconnected</td>
<td>(Inherited from <code>TcpTcpClient</code>.)</td>
</tr>
<tr>
<td>MessageReceived</td>
<td></td>
</tr>
<tr>
<td>SecureFailed</td>
<td>(Inherited from <code>TcpSecureTcpClient</code>.)</td>
</tr>
</tbody>
</table>

See Also

Reference

- `TcpGenericSecureTcpClientTMessage Class`
- `W.Net Namespace`

Copyright © 2018 Jordan Duerksen
Tungsten

\( \text{W} \)
TcpGenericSecureTcpClientTMessage Event

[Missing <summary> documentation for "E:W.Net.Tcp.Generic.SecureTcpClient\1\MessageReceived"]

Namespace: W.Net

Syntax

```csharp
public event Action<TcpGenericSecureTcpClient<TMessage>, TMessage>
```

Value
Type: SystemActionTcpGenericSecureTcpClientTMessage, TMessage

See Also

Reference
TcpGenericSecureTcpClientTMessage Class
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
TcpGenericSecureTcpHost\texttt{TMessage} Class

[Missing \textless summary\textgreater documentation for "T:W.Net.Tcp.Generic.SecureTcpHost`1"]

.parents
\begin{itemize}
\item \texttt{System}
\item \texttt{Object}
\item \texttt{W.NetTcpTcpHost}
\item \texttt{W.NetTcpGenericSecureTcpHostTMessage}
\end{itemize}

\textbf{Namespace:} W.Net

 Syntax

\begin{lstlisting}[language=C#]
public class SecureTcpHost\textless TMessage\textgreater : TcpTcpHost
\end{lstlisting}

Type Parameters

\textbf{TMessage}

[Missing \textless typeparam name="TMessage"/> documentation for "T:W.Net.Tcp.Generic.SecureTcpHost`1"]

The \texttt{TcpGenericSecureTcpHostTMessage} type exposes the following members.

 Constructors

\begin{itemize}
\item \texttt{TcpGenericSecureTcpHostTMessage} Initializes a new inst
TcpGenericSecureTcpHost class

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsListening</td>
<td>(Inherited from TcpTcpHost.)</td>
</tr>
<tr>
<td>OnCreateServer</td>
<td>(Inherited from TcpTcpHost.)</td>
</tr>
</tbody>
</table>

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispose</td>
<td>(Inherited from TcpTcpHost.)</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations</td>
</tr>
<tr>
<td></td>
<td>before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>Listen(IPEndPoint)</td>
<td>(Inherited from TcpTcpHost.)</td>
</tr>
</tbody>
</table>
Listen(IPEndPoint, Int32)  (Inherited from TcpTcpHost.)

MemberwiseClone  Creates a shallow copy of the current Object.  (Inherited from Object.)

OnBytesReceived  (Overrides TcpTcpHostOnBytesReceived(TcpIClient, Byte).)

ToString  Returns a string that represents the current object.  (Inherited from Object.)

<table>
<thead>
<tr>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>BytesReceived</td>
</tr>
<tr>
<td>MessageReceived</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Extension Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>AsTType</td>
</tr>
<tr>
<td>InitializeProperties</td>
</tr>
</tbody>
</table>
This method should be called in the constructor of any class which has IOwnedProperty members (Defined by PropertyHostExtensions.)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>InLock(Action)</code></td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><code>InLockTTType(FuncTTType)</code></td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><code>InLockAsync(Action)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><code>InLockAsyncTTType(FuncTTType)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><code>IsDirty</code></td>
<td>Scans the IsDirty value of each field and property of type IProperty</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>

See Also

Reference

W.Net Namespace
Tungsten

\( W \)
TcpGenericSecureTcpHostTMessage Constructor

Initializes a new instance of the TcpGenericSecureTcpHostTMessage class

Namespace: W.Net

Syntax

```csharp
public SecureTcpHost(int keySize)
```

Parameters

`keySize`
Type: `System.Int32`

See Also

Reference
TcpGenericSecureTcpHostTMessage Class
W.Net Namespace

Copyright © 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
</tr>
</tbody>
</table>
SecureTcpHost*TMessage Properties

The *TcpGenericSecureTcpHostTMessage* generic type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsListening</td>
<td>(Inherited from <em>TcpTcpHost.</em>)</td>
</tr>
<tr>
<td>OnCreateServer</td>
<td>(Inherited from <em>TcpTcpHost.</em>)</td>
</tr>
</tbody>
</table>

### See Also

Reference

[TcpGenericSecureTcpHostTMessage Class](#)

[W.Net Namespace](#)

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
## SecureTcpHostTMessage Methods

The `TcpGenericSecureTcpHostTMessage` generic type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispose</td>
<td>(Inherited from <code>TcpTcpHost</code>.)</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resource and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <code>Type</code> of the current instance. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>Listen(IPEndPoint)</td>
<td>(Inherited from <code>TcpTcpHost</code>.)</td>
</tr>
<tr>
<td>Listen(IPEndPoint, Int32)</td>
<td>(Inherited from <code>TcpTcpHost</code>.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current</td>
</tr>
</tbody>
</table>
Object.  
(Inherited from Object.)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnBytesReceived</td>
<td>(Overrides TcpTcpHostOnBytesReceived(TcpIClient, Byte).)</td>
</tr>
</tbody>
</table>
| ToString              | Returns a string that represents the current object.  
(Inherited from Object.) |

**Extension Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| AsTType            | Use Generic syntax for the as operator.  
(Defined by AsExtensions.) |
| InitializeProperties | Scans the fields and properties of "owner" and sets the member's Owner property to "owner".  
This method should be called in the constructor of any class which has IOwnedProperty members  
(Defined by PropertyHostExtensions.) |
| InLock(Action)      | Overloaded.  
Performs the action in a Monitor lock  
(Defined by MonitorExtensions.) |
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLockTType(Func&lt;TType&gt;)</td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTType(Func&lt;TType&gt;)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock</td>
</tr>
</tbody>
</table>
See Also

Reference
TcpGenericSecureTcpHostTMessage Class
W.Net Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
TcpGenericSecureTcpHost TMessage Method

[Missing <summary> documentation for
]

Namespace: W.Net

Syntax

C#

```csharp
protected override void OnBytesReceived(
    TcpIClient client,
    byte[] bytes
)
```

Parameters

client
Type: W.NetTcpIClient
[Missing <param name="client"/> documentation for
]

bytes
Type: SystemByte
[Missing <param name="bytes"/> documentation for
]

See Also

Reference
TcpGenericSecureTcpHostTMessage Class
W.Net Namespace
| Tungsten | W |
SecureTcpHost TMessage

Events

The TcpGenericSecureTcpHostTMessage generic type exposes the following members.

Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BytesReceived</td>
<td>(Inherited from TcpTcpHost.)</td>
</tr>
<tr>
<td>MessageReceived</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Reference
TcpGenericSecureTcpHostTMessage Class
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
TcpGenericSecureTcpHostTMessage

Event


Namespace: W.Net

Syntax

```csharp
public event Action<TcpGenericSecureTcpHost<TMessage>,
TcpGenericSecureTcpClient<TMessage>, TMessage>
```

Value

Type: SystemAction<TcpGenericSecureTcpHost<TMessage>,
TcpGenericSecureTcpClient<TMessage>, TMessage>

See Also

Reference
TcpGenericSecureTcpHostTMessage Class
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
TcpGenericTcpClientTMessage Class


Inheritance Hierarchy

- System
  - Object
  - W.Net.Tcp.TcpClient
  - W.Net.Tcp.Generic.TcpClientTMessage

Namespace: W.Net

Syntax

```csharp
public class TcpClient<TMessage> : TcpTcpClient
```

Type Parameters

- **TMessage**

The TcpGenericTcpClientTMessage type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcpGenericTcpClientTMessage</td>
<td>Initializes a new instance of TcpGenericTcpClientTMessage</td>
</tr>
</tbody>
</table>
class

Top

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsServerSide</td>
<td>(Inherited from <code>TcpTcpClient</code>.)</td>
</tr>
<tr>
<td>Socket</td>
<td>(Inherited from <code>TcpTcpClient</code>.)</td>
</tr>
</tbody>
</table>

Top

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect</td>
<td>(Inherited from <code>TcpTcpClient</code>.)</td>
</tr>
<tr>
<td>Disconnect</td>
<td>(Inherited from <code>TcpTcpClient</code>.)</td>
</tr>
<tr>
<td>Disconnect(Boolean, Boolean)</td>
<td>(Inherited from <code>TcpTcpClient</code>.)</td>
</tr>
<tr>
<td>Dispose</td>
<td>(Inherited from <code>TcpTcpClient</code>.)</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
</tbody>
</table>
**GetHashCode**
Serves as the default hash function. (Inherited from **Object**.)

**GetType**
Gets the **Type** of the current instance. (Inherited from **Object**.)

**MemberwiseClone**
Creates a shallow copy of the current **Object**. (Inherited from **Object**.)

**OnConnect**
(Inherited from **TcpTcpClient**.)

**OnInitialize**
(Inherited from **TcpTcpClient**.)

**OnReceived**
(Overrides **TcpTcpClientOnReceived(Byte)**.)

**OnSend**
(Inherited from **TcpTcpClient**.)

**ToString**
Returns a string that represents the current object. (Inherited from **Object**.)

**Write(TMessage)**

**Write(Byte)**
(Inherited from **TcpTcpClient**.)

---

**Events**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BytesReceived</td>
<td>(Inherited from <strong>TcpTcpClient</strong>.)</td>
</tr>
<tr>
<td>Connected</td>
<td>(Inherited from <strong>TcpTcpClient</strong>.)</td>
</tr>
</tbody>
</table>
Disconnected  (Inherited from TcpTcpClient.)

Disconnected

MessageReceived

Top

## Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AsTType</code></td>
<td>Use Generic syntax the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td><code>InitializeProperties</code></td>
<td>Scans the fields and properties of &quot;owner&quot; sets the member's Owner property to &quot;owner&quot; This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td><code>InLock(Action)</code></td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><code>InLockTTType(FuncTTType)</code></td>
<td>Overloaded. Performs the function in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>InLockAsync(Action)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by <code>MonitorExtensions</code>).</td>
</tr>
<tr>
<td><code>InLockAsyncTType(Func&lt;TType&gt;)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by <code>MonitorExtensions</code>).</td>
</tr>
<tr>
<td><code>IsDirty</code></td>
<td>Scans the IsDirty value of each field and property of type <code>IProperty</code> (Defined by <code>PropertyHostExtensions</code>).</td>
</tr>
<tr>
<td><code>Lock()</code></td>
<td>Performs a Monitor lock (Defined by <code>MonitorExtensions</code>).</td>
</tr>
<tr>
<td><code>MarkAsClean</code></td>
<td>Scans each field and property of type <code>IProperty</code> and sets its IsDirty flag to false (Defined by <code>PropertyHostExtensions</code>).</td>
</tr>
<tr>
<td><code>RequestAsync(Byte, Int32)</code></td>
<td>Overloaded. Sends a request to the server and waits for a response. Can be used along with, or in lieu of, the regular method of calling Write and handling the response.</td>
</tr>
<tr>
<td>RequestAsyncTMessage(TMessage, Int32)</td>
<td>Overloaded. Sends a request to the server and waits for a response. Can be used along with, or in lieu of, the regular method of calling Write and handling the BytesReceived event. (Defined by ExtensionMethods.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock. (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>

**See Also**

Reference

W.Net Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
TcpGenericTcpClientTMessage Constructor

Initializes a new instance of the TcpGenericTcpClientTMessage class

**Namespace:** W.Net  
**Assembly:** Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

### Syntax

```csharp
public TcpClient()
```

### See Also

Reference  
TcpGenericTcpClientTMessage Class  
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
TcpClient\textit{TM}essage Properties

The \textit{TcpGenericTcpClientTMessage} generic type exposes the following members.

\begin{table}
\begin{tabular}{|l|l|}
\hline
Name & Description \\
\hline
IsServerSide & (Inherited from \textit{TcpTcpClient}.)
\hline
Socket & (Inherited from \textit{TcpTcpClient}.)
\hline
\end{tabular}
\end{table}

\section*{See Also}

Reference

\textit{TcpGenericTcpClientTMessage} Class

W.Net Namespace

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
</tr>
</tbody>
</table>
TcpClientTMessage Methods

The `TcpGenericTcpClientTMessage` generic type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect</td>
<td>(Inherited from <code>TcpTcpClient</code>.)</td>
</tr>
<tr>
<td>Disconnect</td>
<td>(Inherited from <code>TcpTcpClient</code>.)</td>
</tr>
<tr>
<td>Disconnect(Boolean, Boolean)</td>
<td>(Inherited from <code>TcpTcpClient</code>.)</td>
</tr>
<tr>
<td>Dispose</td>
<td>(Inherited from <code>TcpTcpClient</code>.)</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <code>Type</code> of the current instance.</td>
</tr>
</tbody>
</table>
MemberwiseClone | Creates a shallow copy of the current Object. (Inherited from Object.)
OnConnect | (Inherited from TcpTcpClient.)
OnInitialize | (Inherited from TcpTcpClient.)
OnReceived | (Overrides TcpTcpClientOnReceived(Byte).)
OnSend | (Inherited from TcpTcpClient.)
ToString | Returns a string that represents the current object. (Inherited from Object.)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner' sets the member's Owner property to &quot;owner&quot; This method should be called in tl</td>
</tr>
</tbody>
</table>
constructor of any class which has IOwnedProperty members
(Defined by PropertyHostExtensions)

- InLock(Action)
  Overloaded.
  Performs the action in a Monitor lock
  (Defined by MonitorExtensions.)

- InLockTType(Func<TType>)
  Overloaded.
  Performs the function in a Monitor lock
  (Defined by MonitorExtensions.)

- InLockAsync(Action)
  Overloaded.
  Asynchronously performs the action in a Monitor lock
  (Defined by MonitorExtensions.)

- InLockAsyncTType(Func<TType>)
  Overloaded.
  Asynchronously performs the action in a Monitor lock
  (Defined by MonitorExtensions.)

- IsDirty
  Scans the IsDirty value of each field and property of type IProperty
  (Defined by PropertyHostExtensions)
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lock</td>
<td>Performs a Monitor I lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets it's IsDirty flag false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>RequestAsync(Byte, Int32)</td>
<td>Overloaded. Sends a request to the server and waits for a response. Can be used along with, or in lieu of, the regular method of calling Write and handling the BytesReceived event. (Defined by ExtensionMethods.)</td>
</tr>
<tr>
<td>RequestAsyncTMessage(TMessage, Int32)</td>
<td>Overloaded. Sends a request to the server and waits for a response. Can be used along with, or in lieu of, the regular method of calling Write and handling the BytesReceived event. (Defined by ExtensionMethods.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock (Defined by</td>
</tr>
</tbody>
</table>
See Also

Reference
TcpGenericTcpClientTMessage Class
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
</tr>
</tbody>
</table>
TcpGenericTcpClientTMessageOnReceived Method


Namespace: W.Net

Syntax

C# Copy

protected override void OnReceived(
    ref byte[] bytes
)

Parameters

bytes
    Type: System.Byte
[Missing <param name="bytes"/> documentation for "M:W.Net.Tcp.Generic.TcpClient`1.OnReceived(System.Byte[]@)"

See Also

Reference
TcpGenericTcpClientTMessage Class
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
TcpClient TMessage Write Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write(TMessage)</td>
<td></td>
</tr>
<tr>
<td>Write(Bytes)</td>
<td>(Inherited from TcpTcpClient.)</td>
</tr>
</tbody>
</table>

See Also

Reference
TcpGenericTcpClientTMessage Class
W.Net Namespace
Tungsten

$W$
TcpGenericTcpClient\textit{TMessage} Write Method (\textit{TMessage})


\textbf{Namespace:} W.Net  

\section*{Syntax}

\begin{verbatim}
public void Write(
    TMessage message
)
\end{verbatim}

\section*{Parameters}

\textit{message}  
Type: \textit{TMessage}  

\section*{See Also}

\textbf{Reference}  
TcpGenericTcpClient\textit{TMessage} Class  
Write Overload  
W.Net Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$^\text{W}$
TcpClient TMessage Events

The TcpGenericTcpClientTMessage generic type exposes the following members.

### Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BytesReceived</td>
<td>(Inherited from TcpTcpClient.)</td>
</tr>
<tr>
<td>Connected</td>
<td>(Inherited from TcpTcpClient.)</td>
</tr>
<tr>
<td>Disconnected</td>
<td>(Inherited from TcpTcpClient.)</td>
</tr>
<tr>
<td>MessageReceived</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Reference
TcpGenericTcpClientTMessage Class
W.Net Namespace

Copyright © 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>$W$</td>
</tr>
</tbody>
</table>
TcpGenericTcpClient TMessage Message Event

[Missing <summary> documentation for "E:W.Net.Tcp.Generic.TcpClient`1.MessageReceived"]

Namespace: W.Net

Syntax

```csharp
public event Action<TcpGenericTcpClient<TMessage>, TMessage>
```

Value
Type: System.Action<TcpGenericTcpClient<TMessage>, TMessage>

See Also

Reference
TcpGenericTcpClient TMessage Class
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
TcpGenericTcpHostTMessage Class

[Missing <summary> documentation for "T:W.Net.Tcp.Generic.TcpHost`1"]

▲ Inheritance Hierarchy

  System
    Object
      W.NetTcpTcpHost
        W.NetTcpGenericTcpHostTMessage

Namespace: W.Net

▲ Syntax

```
public class TcpHost<TMessage> : TcpTcpHost
```

Type Parameters

**TMessage**

[Missing <typeparam name="TMessage"/> documentation for "T:W.Net.Tcp.Generic.TcpHost`1"]

The TcpGenericTcpHostTMessage type exposes the following members.

▲ Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>TcpGenericTcpHostTMessage</td>
</tr>
</tbody>
</table>
### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsListening</td>
<td>(Inherited from <code>TcpTcpHost</code>.)</td>
</tr>
<tr>
<td>OnCreateServer</td>
<td>(Inherited from <code>TcpTcpHost</code>.)</td>
</tr>
</tbody>
</table>

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispose</td>
<td>(Inherited from <code>TcpTcpHost</code>.)</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <code>Type</code> of the current instance. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>Listen(IPEndPoint)</td>
<td>(Inherited from <code>TcpTcpHost</code>.)</td>
</tr>
</tbody>
</table>
Listen(IPEndPoint, Int32)  (Inherited from TcpTcpHost.)

MemberwiseClone  Creates a shallow copy of the current Object.  (Inherited from Object.)

OnBytesReceived  (Overrides TcpTcpHostOnBytesReceived(TcpIClient, Byte).)

ToString  Returns a string that represents the current object.  (Inherited from Object.)

Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BytesReceived</td>
<td>(Inherited from TcpTcpHost.)</td>
</tr>
<tr>
<td>MessageReceived</td>
<td></td>
</tr>
</tbody>
</table>

Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and</td>
</tr>
</tbody>
</table>
sets the member's Owner property to "owner". This method should be called in the constructor of any class which has IOwnedProperty members (Defined by PropertyHostExtensions.)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLock&lt;TType&gt;(Func&lt;TType&gt;)</td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync&lt;TType&gt;(Func&lt;TType&gt;)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets it's IsDirty flag to false. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>

**See Also**

Reference

W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\[ \text{W} \]
TcpGenericTcpHostTMessage

Constructor

Initializes a new instance of the TcpGenericTcpHostTMessage class

**Namespace:** W.Net  
**Assembly:** Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

### Syntax

```
public TcpHost()
```

### See Also

Reference

TcpGenericTcpHostTMessage Class  
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
TcpHost TMessage Properties

The TcpGenericTcpHostTMessage generic type exposes the following members.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsListening</td>
<td>(Inherited from TcpTcpHost.)</td>
</tr>
<tr>
<td>OnCreateServer</td>
<td>(Inherited from TcpTcpHost.)</td>
</tr>
</tbody>
</table>

See Also

Reference
TcpGenericTcpHostTMessage Class
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
# TcpHost TMessage Methods

The `TcpGenericTcpHostTMessage` generic type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dispose</strong></td>
<td>(Inherited from <code>TcpTcpHost</code>.)</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified object is equal to the current object. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as the default hash function. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <code>Type</code> of the current instance. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>Listen(IPEndPoint)</strong></td>
<td>(Inherited from <code>TcpTcpHost</code>.)</td>
</tr>
<tr>
<td><strong>Listen(IPEndPoint, Int32)</strong></td>
<td>(Inherited from <code>TcpTcpHost</code>.)</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current <code>Object</code>. (Inherited from <code>Object</code>.)</td>
</tr>
</tbody>
</table>
OnBytesReceived  (Overrides TcpTcpHostOnBytesReceived(TcpClient Byte).)

ToString  Returns a string that represents the current object. (Inherited from Object.)

---

### Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTType(FuncTType)</td>
<td>Overloaded. Performs the function in</td>
</tr>
<tr>
<td>Method/Extension</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>Monitor</strong></td>
<td>a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>![MonitorLock]</td>
<td><strong>InLockAsync(Action)</strong> Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>![MonitorLock]</td>
<td><strong>InLockAsyncTT&lt;T&lt;TType&gt;</strong> Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>![MonitorLock]</td>
<td><strong>IsDirty</strong> Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>![MonitorLock]</td>
<td><strong>Lock</strong> Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>![MonitorLock]</td>
<td><strong>MarkAsClean</strong> Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>![MonitorLock]</td>
<td><strong>Unlock</strong> Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>
See Also

Reference
TcpGenericTcpHostTMessage Class
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>$^7W$</td>
</tr>
</tbody>
</table>
TcpGenericTcpHost TMessage OnBytesReceived Method

[Missing <summary> documentation for
]

Namespace: W.Net

Syntax

```csharp
protected override void OnBytesReceived(
    TcpIClient client,
    byte[] bytes
)
```

Parameters

**client**
- Type: W.NetTcpIClient
  [Missing <param name="client"/> documentation for
]

**bytes**
- Type: System.Byte
  [Missing <param name="bytes"/> documentation for
]

See Also

Reference
TcpGenericTcpHostTMessage Class
W.Net Namespace
Tungsten

W
TcpHost *TMessage* Events

The *TcpGenericTcpHostTMessage* generic type exposes the following members.

### Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>🔄 BytesReceived</td>
<td>(Inherited from <em>TcpTcpHost.</em>)</td>
</tr>
<tr>
<td>🔄 MessageReceived</td>
<td></td>
</tr>
</tbody>
</table>

### See Also

Reference
- *TcpGenericTcpHostTMessage Class*
- *W.Net Namespace*

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
TcpGenericTcpHost TMessage MessageEvent

[Missing <summary> documentation for "E:W.Net.Tcp.Generic.TcpHost`1.MessageReceived"]

Namespace: W.Net

Syntax

```
public event Action<TcpGenericTcpHost<TMessage>,
TMessage>
```

Value

Type: SystemAction<TcpGenericTcpHost<TMessage>,
TcpGenericTcpClient<TMessage>, TMessage>

See Also

Reference
TcpGenericTcpHostTMessage Class
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
TcpIIClient Interface

[Missing <summary> documentation for "T:W.Net.Tcp.IClient"]

Namespace:  W.Net

Syntax

```c#
public interface IClient : IDisposable
```

The TcpIIClient type exposes the following members.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="" alt="Socket" /></td>
<td>Socket</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="" alt="Connect" /></td>
<td>Connect</td>
</tr>
<tr>
<td><img src="" alt="Dispose" /></td>
<td>Dispose</td>
</tr>
<tr>
<td><img src="" alt="Write" /></td>
<td>Write</td>
</tr>
</tbody>
</table>
### Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BytesReceived</td>
<td></td>
</tr>
<tr>
<td>Connected</td>
<td></td>
</tr>
<tr>
<td>Disconnected</td>
<td></td>
</tr>
</tbody>
</table>

### See Also

**Reference**
- W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
# IClient Properties

The `TcpIClient` type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="icon" /> Socket</td>
<td>Top</td>
</tr>
</tbody>
</table>

## See Also

- Reference
  - `TcpIClient Interface`
  - `W.Net Namespace`

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
TcpIClientSocket Property

[Missing <summary> documentation for "P:W.Net.Tcp.IClient.Socket"]

Namespace:  W.Net  

Syntax

```csharp
Socket Socket { get; }
```

Property Value
Type:  Socket

See Also

Reference
TcpIClient Interface
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
IClient Methods

The TcpIClient type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect</td>
<td></td>
</tr>
<tr>
<td>Dispose</td>
<td>Performs application-defined tasks associated with freeing, releasing, or resetting unmanaged resources. (Inherited from IDisposable.)</td>
</tr>
<tr>
<td>Write</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Reference
TcpIClient Interface
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
</tr>
</tbody>
</table>
TcpIClientConnect Method


**Namespace:** W.Net  
**Assembly:** Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

## Syntax

```csharp
    void Connect(
        IPEndPoint ep
    )
```

### Parameters

- **ep**
  - Type: System.Net.IPEndPoint  
  [Missing <param name="ep"/> documentation for "M:W.Net.Tcp.IClient.Connect(System.Net.IPEndPoint)"

## See Also

Reference
- TcpIClient Interface  
- W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
TcpIClient Write Method

[Missing <summary> documentation for "M:W.Net.Tcp.IClient.Write(System.Byte[])"

Namespace: W.Net  

Syntax

```c#
void Write(
    byte[] bytes
)
```

Parameters

- `bytes`  
  Type: System.Byte  
  [Missing <param name="bytes"/> documentation for "M:W.Net.Tcp.IClient.Write(System.Byte[])"

See Also

- Reference  
  TcpIClient Interface  
  W.Net Namespace

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>$W^3$</td>
</tr>
</tbody>
</table>
IClient Events

The `TcpIClient` type exposes the following members.

### Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BytesReceived</td>
<td></td>
</tr>
<tr>
<td>Connected</td>
<td></td>
</tr>
<tr>
<td>Disconnected</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Reference

`TcpClient Interface`

`W.Net Namespace`

Copyright © 2018 Jordan Duerksen
Tungsten

\(W\)
TcpIClient.BytesReceived Event

[Missing <summary> documentation for "E:W.Net.TcpIClient.BytesReceived"]

Namespace: W.Net

Syntax

```csharp
event Action<TcpIClient, byte[]> BytesReceived
```

Value
Type: SystemAction<TcpIClient, Byte>

See Also

Reference
TcpIClient Interface
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
TcpIClientConnected Event

[Missing <summary> documentation for "E:W.Net.Tcp.IClient.Connected"]

Namespace: W.Net

Syntax

```csharp
event Action<TcpIClient> Connected
```

Value
Type: SystemAction<TcpIClient>

See Also

Reference
TcpIClient Interface
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
TcpIClientDisconnected Event

[Missing <summary> documentation for "E:W.Net.Tcp.IClient.Disconnected"]

Namespace: W.Net

Syntax

```csharp
event Action<TcpIClient> Disconnected
```

Value
Type: SystemAction<TcpIClient>

See Also

Reference
TcpIClient Interface
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
TcpIInitialize Interface

[Missing <summary> documentation for "T:W.Net.Tcp.IInitialize"]

Namespace: W.Net

Syntax

```csharp
public interface IInitialize
```

The TcpIInitialize type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialize</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Reference

W.Net Namespace

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
</tr>
</tbody>
</table>
Initialize Methods

The TcpIInitialize type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialize</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Reference
TcpIInitialize Interface
W.Net Namespace

Copyright © 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
</tr>
</tbody>
</table>
TcpIInitializeInitialize Method

[Missing <summary> documentation for "M:W.Net.Tcp.IInitialize.Initialize(System.Object[])"]

Namespace: W.Net

Syntax

```csharp
bool Initialize(
    params Object[] args
)
```

Parameters

args
Type: SystemObject
[Missing <param name="args"/> documentation for "M:W.Net.Tcp.IInitialize.Initialize(System.Object[])"]

Return Value

Type: Boolean
[Missing <returns> documentation for "M:W.Net.Tcp.IInitialize.Initialize(System.Object[])"]

See Also

Reference
TcpIInitialize Interface
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
TcpSecureTcpClient Class

[Missing <summary> documentation for "T:W.Net.Tcp.SecureTcpClient"]

▲ Inheritance Hierarchy

System \rightarrow Object \rightarrow W.NetTcpTcpClient \rightarrow W.NetTcpSecureTcpClient \rightarrow W.NetTcpGenericSecureTcpClientTMessage

Namespace: W.Net

▲ Syntax

```c#
public class SecureTcpClient : TcpTcpClient
```

The TcpSecureTcpClient type exposes the following members.

▲ Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcpSecureTcpClient</td>
<td>Initializes a new instance of the TcpSecureTcpClient class</td>
</tr>
</tbody>
</table>

▲ Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
### Encryption

### IsSecure

### IsServerSide  (Inherited from TcpTcpClient.)

### Socket  (Inherited from TcpTcpClient.)

## Top

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect</td>
<td></td>
</tr>
<tr>
<td>Disconnect</td>
<td>(Inherited from TcpTcpClient.)</td>
</tr>
<tr>
<td>Disconnect(Boolean,</td>
<td>(Inherited from TcpTcpClient.)</td>
</tr>
<tr>
<td>Boolean)</td>
<td></td>
</tr>
<tr>
<td>Dispose</td>
<td>(Inherited from TcpTcpClient.)</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup</td>
</tr>
<tr>
<td></td>
<td>operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
</tbody>
</table>
GetType

Gets the Type of the current instance. (Inherited from Object.)

MemberwiseClone

Creates a shallow copy of the current Object. (Inherited from Object.)

OnConnect (Inherited from TcpTcpClient.)

OnInitialize (Overrides TcpTcpClientOnInitialize(Object).)

OnReceived (Overrides TcpTcpClientOnReceived(Byte).)

OnSend (Overrides TcpTcpClientOnSend(Byte).)

ToString

Returns a string that represents the current object. (Inherited from Object.)

Write (Inherited from TcpTcpClient.)

Top

Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BytesReceived</td>
<td>(Inherited from TcpTcpClient.)</td>
</tr>
<tr>
<td>Connected</td>
<td>(Inherited from TcpTcpClient.)</td>
</tr>
<tr>
<td>Disconnected</td>
<td>(Inherited from TcpTcpClient.)</td>
</tr>
<tr>
<td>SecureFailed</td>
<td></td>
</tr>
</tbody>
</table>
## Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| AsTType                | Use Generic syntax for the as operator.  
(Defined by AsExtensions.)                                                                       |
| InitializeProperties   | Scans the fields and properties of "owner" and sets the member's Owner property to "owner". This method should be called in the constructor of any class which has IOwnedProperty members.
(Defined by PropertyHostExtensions.)                                                               |
| InLock(Action)         | Overloaded. Performs the action in a Monitor lock.  
(Defined by MonitorExtensions.)                                                                  |
| InLockTType(FuncTType) | Overloaded. Performs the function in a Monitor lock.  
(Defined by MonitorExtensions.)                                                                  |
<p>| InLockAsync(Action)    | Overloaded. Asynchronously performs the action in a Monitor lock.                                                                         |</p>
<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLockAsync&lt;TType&gt;(Func&lt;TType&gt;)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>RequestAsync(Byte, Int32)</td>
<td>Overloaded. Sends a request to the server and waits for a response. Can be used along with, or in lieu of, the regular method of calling Write and handling the BytesReceived event. (Defined by ExtensionMethods.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>RequestAsync(Byte, Int32)</td>
<td>Overloaded. Sends a request to the server and waits for a response. Can be used along with, or in lieu of, the regular method of calling Write and handling the BytesReceived event. (Defined by ExtensionMethods.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>

**See Also**

**Reference**

W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
TcpSecureTcpClient Constructor

Initializes a new instance of theTcpSecureTcpClient class

**Namespace:** W.Net  
**Assembly:** Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

### Syntax

```csharp
public SecureTcpClient(
    int keySize
)
```

### Parameters

**keySize**

- Type: System.Int32
- [Missing <param name="keySize"/> documentation for "M:W.Net.Tcp.SecureTcpClient.#ctor(System.Int32)"

### See Also

- **Reference**
  - TcpSecureTcpClient Class
  - W.Net Namespace

Copyright © 2018 Jordan Duerksen
SecureTcpClient Properties

The **TcpSecureTcpClient** type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encryption</td>
<td></td>
</tr>
<tr>
<td>IsSecure</td>
<td></td>
</tr>
<tr>
<td>IsServerSide</td>
<td>(Inherited from <strong>TcpTcpClient</strong>)</td>
</tr>
<tr>
<td>Socket</td>
<td>(Inherited from <strong>TcpTcpClient</strong>)</td>
</tr>
</tbody>
</table>

See Also

Reference

**TcpSecureTcpClient Class**

**W.Net Namespace**

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
TcpSecureTcpClientEncryption Property


Namespace: W.Net

Syntax

```csharp
protected AssymetricEncryption Encryption { get; }
```

Property Value
Type: AssymetricEncryption

See Also

Reference
TcpSecureTcpClient Class
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
TcpSecureTcpClient.IsSecure Property


Namespace: W.Net

Syntax

```csharp
protected bool IsSecure { get; }
```

Property Value
Type: Boolean

See Also

Reference
TcpSecureTcpClient Class
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
SecureTcpClient Methods

The `TcpSecureTcpClient` type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Connect</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Disconnect</strong></td>
<td>(Inherited from <code>TcpTcpClient</code>.)</td>
</tr>
<tr>
<td><strong>Disconnect(Boolean, Boolean)</strong></td>
<td>(Inherited from <code>TcpTcpClient</code>.)</td>
</tr>
<tr>
<td><strong>Dispose</strong></td>
<td>(Inherited from <code>TcpTcpClient</code>.)</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <code>Type</code> of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>OnConnect</td>
<td>(Inherited from TcpTcpClient.)</td>
</tr>
<tr>
<td>OnInitialize</td>
<td>(Overides TcpTcpClientOnInitialize(Object).)</td>
</tr>
<tr>
<td>OnReceived</td>
<td>(Overides TcpTcpClientOnReceived(Byte).)</td>
</tr>
<tr>
<td>OnSend</td>
<td>(Overides TcpTcpClientOnSend(Byte).)</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object. (Inherited from Object.)</td>
</tr>
<tr>
<td>Write</td>
<td>(Inherited from TcpTcpClient.)</td>
</tr>
</tbody>
</table>

## Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot; This method should be called in the constructor of any class which has</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTTType(Func TTType)</td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTTType(Func TTType)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>RequestAsync(Byte, Int32)</td>
<td>Overloaded. Sends a request to the server and waits for a response. Can be used along with, or in lieu of, the regular method of calling Write and handling the BytesReceived event. (Defined by ExtensionMethods.)</td>
</tr>
<tr>
<td>RequestAsync(Byte, Int32)</td>
<td>Overloaded. Sends a request to the server and waits for a response. Can be used along with, or in lieu of, the regular method of calling Write and handling the BytesReceived event. (Defined by ExtensionMethods.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock. (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>
See Also

Reference
TcpSecureTcpClient Class
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>$W$</td>
</tr>
</tbody>
</table>
TcpSecureTcpClient
Connect Method

Namespace: W.Net

Syntax

```csharp
public void Connect( 
    IPEndPoint ep
)
```

Parameters

* **ep**
  Type: System.Net.IPEndPoint

See Also

Reference
 TcpSecureTcpClient Class
 W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
TcpSecureTcpClientOnInitialize Method


Namespace: W.Net

Syntax

C#

```csharp
protected override bool OnInitialize(
    params Object[] args
)
```

Parameters

args

Type: SystemObject


Return Value

Type: Boolean


See Also

Reference
TcpSecureTcpClient Class
W.Net Namespace
Tungsten

\( W \)
TcpSecureTcpClientOnReceived Method

[Missing <summary> documentation for "M:W.Net.Tcp.SecureTcpClient.OnReceived(System.Byte[]@)"

Namespace:  W.Net

Syntax

```csharp
protected override void OnReceived(
    ref byte[] bytes
)
```

Parameters

`bytes`
Type:  SystemByte

[Missing <param name="bytes"/> documentation for "M:W.Net.Tcp.SecureTcpClient.OnReceived(System.Byte[]@)"

See Also

Reference

TcpSecureTcpClient Class
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
TcpSecureTcpClientOnSend Method


**Namespace:** W.Net  
**Assembly:** Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

### Syntax

```csharp
protected override void OnSend(
    ref byte[] bytes
)
```

### Parameters

`bytes`  
Type: SystemByte  

### See Also

Reference  
TcpSecureTcpClient Class  
W.Net Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
SecureTcpClient Events

The `TcpSecureTcpClient` type exposes the following members.

### Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt=" " /> BytesReceived</td>
<td>(Inherited from <code>TcpTcpClient</code>.)</td>
</tr>
<tr>
<td><img src="image" alt=" " /> Connected</td>
<td>(Inherited from <code>TcpTcpClient</code>.)</td>
</tr>
<tr>
<td><img src="image" alt=" " /> Disconnected</td>
<td>(Inherited from <code>TcpTcpClient</code>.)</td>
</tr>
<tr>
<td><img src="image" alt=" " /> SecureFailed</td>
<td></td>
</tr>
</tbody>
</table>

### See Also

- Reference
- `TcpSecureTcpClient Class`
- `W.Net Namespace`

Copyright © 2018 Jordan Duerksen
Tungsten

W
TcpSecureTcpClientSecureFailed Event


Namespace:  W.Net

Syntax

```
public event Action<TcpSecureTcpClient> SecureFailed
```

Value
Type:  SystemActionTcpSecureTcpClient

See Also

Reference
TcpSecureTcpClient Class
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
TcpSecureTcpHost Class

[Missing <summary> documentation for "T:W.Net.Tcp.SecureTcpHost"]

Inheritance Hierarchy

System
  Object
  W.NetTcpTcpHost
  W.NetTcpSecureTcpHost

Namespace:  W.Net

Syntax

```csharp
public class SecureTcpHost : TcpTcpHost
```

The TcpSecureTcpHost type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="TcpSecureTcpHost" /></td>
<td>Initializes a new instance of the TcpSecureTcpHost class</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="IsListening" /></td>
<td>(Inherited from TcpTcpHost.)</td>
</tr>
</tbody>
</table>
### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispose</td>
<td>(Inherited from <strong>TcpTcpHost</strong>.)</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>Object</strong>)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>Object</strong>)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>Object</strong>)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>Object</strong>)</td>
</tr>
<tr>
<td>Listen(IPEndPoint)</td>
<td>(Inherited from <strong>TcpTcpHost</strong>.)</td>
</tr>
<tr>
<td>Listen(IPEndPoint, Int32)</td>
<td>(Inherited from <strong>TcpTcpHost</strong>.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current <strong>Object</strong>.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>Object</strong>)</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>OnBytesReceived</td>
<td>(Inherited from TcpTcpHost.)</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
</tbody>
</table>

**Events**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BytesReceived</td>
<td>(Inherited from TcpTcpHost.)</td>
</tr>
</tbody>
</table>

**Extension Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator.</td>
</tr>
<tr>
<td></td>
<td>(Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;</td>
</tr>
<tr>
<td></td>
<td>This method should be called in the constructor of any class which has IOwnedPropertyParams members</td>
</tr>
<tr>
<td></td>
<td>(Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>PerformstheactioninaMonitorlock</code> (Defined by <code>MonitorExtensions</code>.)</td>
<td>Performs the action in a Monitor lock (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>InLockTType(Func&lt;TType&gt;)</code></td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>InLockAsync(Action)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>InLockAsyncTType(Func&lt;TType&gt;)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>IsDirty</code></td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by <code>PropertyHostExtensions</code>.)</td>
</tr>
<tr>
<td><code>Lock</code></td>
<td>Performs a Monitor lock (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>MarkAsClean</code></td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false</td>
</tr>
</tbody>
</table>
Unlock
(Defined by PropertyHostExtensions.)

Performs a Monitor unlock
(Defined by MonitorExtensions.)

See Also
Reference
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
TcpSecureTcpHost Constructor

Initializes a new instance of the `TcpSecureTcpHost` class

**Namespace:** W.Net  
**Assembly:** Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

### Syntax

```
public SecureTcpHost(int keySize)
```

### Parameters

- **keySize**
  - Type: `System.Int32`
  - [Missing <param name="keySize"/> documentation for "M:W.Net.Tcp.SecureTcpHost.#ctor(System.Int32)"]

### See Also

- Reference
  - `TcpSecureTcpHost Class`
  - `W.Net Namespace`

Copyright © 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
</tr>
</tbody>
</table>
SecureTcpHost Properties

The `TcpSecureTcpHost` type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsListening</td>
<td>(Inherited from <code>TcpTcpHost</code>.)</td>
</tr>
<tr>
<td>OnCreateServer</td>
<td>(Inherited from <code>TcpTcpHost</code>.)</td>
</tr>
</tbody>
</table>

### See Also

Reference  
`TcpSecureTcpHost Class`  
`W.Net Namespace`

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
SecureTcpHost Methods

The `TcpSecureTcpHost` type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispose</td>
<td>(Inherited from <code>TcpTcpHost</code>.)</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <code>Type</code> of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>Listen(IPEndPoint)</td>
<td>(Inherited from <code>TcpTcpHost</code>.)</td>
</tr>
<tr>
<td>Listen(IPEndPoint, Int32)</td>
<td>(Inherited from <code>TcpTcpHost</code>.)</td>
</tr>
</tbody>
</table>
MemberwiseClone  Creates a shallow copy of the current Object.  
(Inherited from Object.)

OnBytesReceived  (Inherited from TcpTcpHost.)

ToString  Returns a string that represents the current object.  
(Inherited from Object.)

---

**Extension Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| ![ ](image) AsTType | Use Generic syntax for the as operator.  
(Defined by AsExtensions.) |
| ![ ](image) InitializeProperties | Scans the fields and properties of "owner" and sets the member's Owner property to "owner". This method should be called in the constructor of any class which has IOwnedProperty members  
(Defined by PropertyHostExtensions.) |
| ![ ](image) InLock(Action) | Overloaded. Performs the action in a Monitor lock  
(Defined by MonitorExtensions.) |
<table>
<thead>
<tr>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>InLockTTyp</strong>e(FuncTTyp<strong>e</strong>)</td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>InLockAsync(Action)</strong></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>InLockAsyncTTyp</strong>e(FuncTTyp<strong>e</strong>)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>IsDirty</strong></td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td><strong>Lock</strong></td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>MarkAsClean</strong></td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td><strong>Unlock</strong></td>
<td>Performs a Monitor unlock</td>
</tr>
</tbody>
</table>
See Also

Reference
TcpSecureTcpHost Class
W.Net Namespace
Tungsten

W
SecureTcpHost Events

The `TcpSecureTcpHost` type exposes the following members.

### Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BytesReceived</td>
<td>(Inherited from <code>TcpTcpHost</code>.)</td>
</tr>
</tbody>
</table>

See Also

Reference

- `TcpSecureTcpHost Class`
- `W.Net Namespace`

Copyright @ 2018 Jordan Duerksen
Tungsten

W
## TcpSecureTcpLogger Class

Sends log messages to a remote server via secure Tcp

### Inheritance Hierarchy

```
SystemObject  W.NetTcpSecureTcpLogger
```

**Namespace:**  W.Net  
**Assembly:**  Tungsten.Net (in Tungsten.Net.dll)  
**Version:**  2.0.2

### Syntax

```c#
public static class SecureTcpLogger
```

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![S] LogTheMessage(IPEndPoint, Int32, String)</td>
<td>Log a message to the remote machine. Note that this message is slower due to parsing the remoteIP with each call</td>
</tr>
<tr>
<td>![S] LogTheMessage(String, Int32, Int32, String)</td>
<td>Log a message to the remote machine. Note that this message is slower due to</td>
</tr>
</tbody>
</table>
parsing the remoteIP with each call

**Examples**

```
Log.LogTheMessage += (category, message) =>
W.Net.Tcp.SecureTcpLogger.LogTheMessage("127.0.0.1", 5555, true, category, message);
```

**See Also**

Reference

W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
SecureTcpLogger Methods

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LogTheMessage(IPEndPoint, Int32, String)</td>
<td>Log a message to the remote machine. Note that this message is slower due to parsing the remoteIP with each call</td>
</tr>
<tr>
<td>LogTheMessage(String, Int32, Int32, String)</td>
<td>Log a message to the remote machine. Note that this message is slower due to parsing the remoteIP with each call</td>
</tr>
</tbody>
</table>

**See Also**

Reference
- TcpSecureTcpLogger Class
- W.Net Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
SecureTcpLogger LogTheMessage Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LogTheMessage(IPEndPoint, Int32, String)</td>
<td>Log a message to the remote machine. Note that this message is slower due to parsing the remote IP with each call</td>
</tr>
<tr>
<td>LogTheMessage(String, Int32, Int32, String)</td>
<td>Log a message to the remote machine. Note that this message is slower due to parsing the remote IP with each call</td>
</tr>
</tbody>
</table>

See Also

Reference
TcpSecureTcpLogger Class
W.Net Namespace
Tungsten

$W$
TcpSecureTcpLoggerLogTheMessage Method (IPEndPoint, Int32, String)

Log a message to the remote machine. Note that this message is slower due to parsing the remoteIP with each call.

**Namespace:** W.Net  
**Assembly:** Tungsten.Net (in Tungsten.Net.dll)  
**Version:** 2.0.2

### Syntax

```csharp
class TcpSecureTcpLogger
{
    public static void LogTheMessage(IPEndPoint remoteIPEndPoint, int keySize, string message);
}
```

### Parameters

- **remoteIPEndPoint**  
  *Type:* System.Net.IPEndPoint  
  *The IPEndPoint of the remote server*

- **keySize**  
  *Type:* System.Int32  
  *The encryption key size*

- **message**  
  *Type:* System.String  
  *The log message*

### See Also

Reference
TcpSecureTcpLogger Class
LogTheMessage Overload
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
TcpSecureTcpLoggerLogTheMessage Method (String, Int32, Int32, String)

Log a message to the remote machine. Note that this message is slower due to parsing the remoteIP with each call.

**Namespace:** W.Net  
**Assembly:** Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

### Syntax

```csharp
public static void LogTheMessage(
    string remoteIP,
    int remotePort,
    int keySize,
    string message
)
```

### Parameters

- **remoteIP**
  - Type: **System.String**
  - The IP address or name of the remote server

- **remotePort**
  - Type: **System.Int32**
  - The port of the remote Udp server

- **keySize**
  - Type: **System.Int32**
  - The encryption key size

- **message**
  - Type: **System.String**
  - The log message
See Also

Reference
TcpSecureTcpLogger Class
LogTheMessage Overload
W.Net Namespace

Copyright © 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>$W$</td>
</tr>
</tbody>
</table>
TcpTcpClient Class

[Missing <summary> documentation for "T:W.Net.Tcp.TcpClient"]

Inheritance Hierarchy

System
  Object
  W.NetTcpTcpClient
    W.NetTcpGenericTcpClient
    TMessage
    W.NetTcpSecureTcpClient

Namespace: W.Net

Syntax

```csharp
public class TcpClient : TcpIInitialize, TcpIClient, IDisposable
```

The TcpTcpClient type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcpTcpClient</td>
<td>Initializes a new instance of the TcpTcpClient class</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect</td>
<td></td>
</tr>
<tr>
<td>Disconnect</td>
<td></td>
</tr>
<tr>
<td>Disconnect(Boolean, Boolean)</td>
<td></td>
</tr>
<tr>
<td>Dispose</td>
<td>Releases all resources used by the <a href="#">TcpTcpClient</a></td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object. (Inherited from <a href="#">Object</a>)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <a href="#">Object</a>)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from <a href="#">Object</a>)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <a href="#">Type</a> of the current instance.</td>
</tr>
</tbody>
</table>
### Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object.</td>
</tr>
<tr>
<td>OnConnect</td>
<td></td>
</tr>
<tr>
<td>OnInitialize</td>
<td></td>
</tr>
<tr>
<td>OnReceived</td>
<td></td>
</tr>
<tr>
<td>OnSend</td>
<td></td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object.</td>
</tr>
<tr>
<td>Write</td>
<td></td>
</tr>
</tbody>
</table>

## Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BytesReceived</td>
<td></td>
</tr>
<tr>
<td>Connected</td>
<td></td>
</tr>
<tr>
<td>Disconnected</td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>AsTType</code></td>
<td>Use Generic syntax for the as operator. (Defined by <code>AsExtensions</code>.)</td>
</tr>
<tr>
<td><code>InitializeProperties</code></td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by <code>PropertyHostExtensions</code>.)</td>
</tr>
<tr>
<td><code>InLock(Action)</code></td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>InLockTType(Func&lt;TType&gt;)</code></td>
<td>Overloaded. Performs the function in a Monitor lock. (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>InLockAsync(Action)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock. (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>InLockAsyncTType(Func&lt;TType&gt;)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>lock</td>
<td>(Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty</td>
</tr>
<tr>
<td></td>
<td>(Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to</td>
</tr>
<tr>
<td></td>
<td>false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>RequestAsync</td>
<td>Sends a request to the server and waits for a response. Can be used along</td>
</tr>
<tr>
<td></td>
<td>with, or in lieu of, the regular method of calling Write and handling the</td>
</tr>
<tr>
<td></td>
<td>BytesReceived event. (Defined by ExtensionMethods.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>

Top
See Also

Reference

W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\[ W \]
# TcpTcpClient Constructor

Initializes a new instance of the `TcpTcpClient` class

**Namespace:** W.Net  
**Assembly:** Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

## Syntax

```c#
public TcpClient()
```

## See Also

Reference  
[TcpTcpClient Class](#)  
[W.Net Namespace](#)

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
TcpClient Properties

The `TcpTcpClient` type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>IsServerSide</code></td>
<td></td>
</tr>
<tr>
<td><code>Socket</code></td>
<td></td>
</tr>
</tbody>
</table>

**See Also**

Reference

[TcpTcpClient Class](#)

[W.Net Namespace](#)

Copyright © 2018 Jordan Duerksen
Tungsten

W
TcpTcpClient.IsServerSide Property


Namespace: W.Net

Syntax

```csharp
public bool IsServerSide { get; set; }
```

Property Value
Type: Boolean

See Also

Reference
TcpTcpClient Class
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
TcpTcpClient.Socket Property


**Namespace:**  W.Net  
**Assembly:**  Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

▶ Syntax

```c#
public Socket Socket { get; }
```

**Property Value**
- **Type:**  Socket
- **Implements**  TcpClientSocket

▶ See Also

**Reference**  
TcpTcpClient Class  
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
# TcpClient Methods

The `TcpTcpClient` type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Inherited From</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Disconnect</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Disconnect(Boolean, Boolean)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dispose</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
<td><code>Object</code></td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function.</td>
<td><code>Object</code></td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <code>Type</code> of the current instance.</td>
<td></td>
</tr>
</tbody>
</table>
(Inherited from Object.)

**MemberwiseClone**  Creates a shallow copy of the current Object. (Inherited from Object.)

**OnConnect**

**OnInitialize**

**OnReceived**

**OnSend**

**ToString**  Returns a string that represents the current object. (Inherited from Object.)

**Write**

---

### Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTType(FuncTType)</td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTType(FuncTType)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false</td>
</tr>
<tr>
<td>RequestAsync</td>
<td>Sends a request to the server and waits for a response. Can be used along with, or in lieu of, the regular method of calling Write and handling the BytesReceived event.</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock</td>
</tr>
</tbody>
</table>

**See Also**

**Reference**
- TcpTcpClient Class
- W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
TcpTcpClientConnect Method


Namespace: W.Net

Syntax

```csharp
public void Connect(IPEndPoint ep)
```

Parameters

`ep`
Type: System.Net.IPEndPoint

[Missing <param name="ep"/> documentation for "M:W.Net.Tcp.TcpClient.Connect(System.Net.IPEndPoint)"]

Implements
TcpClientConnect(IPEndPoint)

See Also

Reference
TcpTcpClient Class
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
## TcpClientDisconnect Method

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![disconnect_icon] Disconnect</td>
<td>![disconnect_icon] Disconnect(Boolean, Boolean)</td>
</tr>
</tbody>
</table>

### See Also

Reference
- TcpTcpClient Class
- W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
TcpTcpClientDisconnect Method


Namespace:  W.Net

**Syntax**

```csharp
public void Disconnect()
```

**See Also**

Reference
TcpTcpClient Class
Disconnect Overload
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
TcpTcpClientDisconnect Method (Boolean, Boolean)


Namespace: W.Net

Syntax

```csharp
protected void Disconnect(
    bool notifyRemote,
    bool waitForThreadToExit
)
```

Parameters

- **notifyRemote**
  - Type: SystemBoolean

- **waitForThreadToExit**
  - Type: SystemBoolean

See Also

Reference
TcpTcpClient Class
Disconnect Overload
Tungsten

$W$
TcpTcpClientDispose Method

Releases all resources used by theTcpTcpClient

**Namespace:** W.Net  
**Assembly:** Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

### Syntax

```csharp
public virtual void Dispose()
```

Implements  
IDisposableDispose

### See Also

Reference  
TcpTcpClient Class  
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
TcpTcpClientOnConnect Method

[Missing <summary> documentation for
]

Namespace: W.Net

Syntax

```csharp
protected virtual void OnConnect(
    params Object[] args
)
```

Parameters

args
Type: SystemObject
[Missing <param name="args"/> documentation for
]

See Also

Reference
TcpTcpClient Class
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
TcpTcpClientOnInitialize Method

[Missing <summary> documentation for
]

Namespace:  W.Net

Syntax

```csharp
protected virtual bool OnInitialize(
    params Object[] args
)
```

Parameters

args
Type: SystemObject
[Missing <param name="args"/> documentation for
]

Return Value
Type: Boolean
[Missing <returns> documentation for
]

See Also

Reference
TcpTcpClient Class
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
TcpTcpClientOnReceived Method

[Missing <summary> documentation for "M:W.Net.Tcp.TcpClient.OnReceived(System.Byte[])"

**Namespace:** W.Net  
**Assembly:** Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

### Syntax

```csharp
protected virtual void OnReceived(
    ref byte[] bytes
)
```

**Parameters**

*bytes*

  Type: System.Byte  
[Missing <param name="bytes"/> documentation for "M:W.Net.Tcp.TcpClient.OnReceived(System.Byte[])"

### See Also

**Reference**

TcpTcpClient Class  
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$^W$
TcpTcpClientOnSend Method


Namespace: W.Net

Syntax

```csharp
protected virtual void OnSend(
    ref byte[] bytes
)
```

Parameters

`bytes`
Type: System.Byte

[Missing <param name="bytes"/> documentation for "M:W.Net.Tcp.TcpClient.OnSend(System.Byte[])"

See Also

Reference
TcpTcpClient Class
W.Net Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
TcpTcpClient

TcpTcpClientWrite Method


Namespace: W.Net

Syntax

```c#
public void Write(
    byte[] bytes
)
```

Parameters

`bytes`

Type: SystemByte

[Missing <param name="bytes"/> documentation for "M:W.Net.Tcp.TcpClient.Write(System.Byte[])""]

Implements

TcpTcpClientWrite(Byte)

See Also

Reference
TcpTcpClient Class
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
TcpClient Events

The TcpTcpClient type exposes the following members.

## Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BytesReceived</td>
<td></td>
</tr>
<tr>
<td>Connected</td>
<td></td>
</tr>
<tr>
<td>Disconnected</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Reference
TcpTcpClient Class
W.Net Namespace
Tungsten

$W$
TcpTcpClient.BytesReceived Event

[Missing <summary> documentation for "E:W.Net.Tcp.TcpClient.BytesReceived"]

Namespace: W.Net

Syntax

```csharp
public event Action<TcpIClient, byte[]> BytesReceived
```

Value
Type: SystemAction<TcpIClient, Byte>
Implements
TcpClient.BytesReceived

See Also
Reference
TcpTcpClient Class
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
TcpTcpClientConnected Event

[Missing <summary> documentation for "E:W.Net.Tcp.TcpClientConnected"]

Namespace:  W.Net

Syntax

```csharp
public event Action<TcpIClient> Connected
```

Value
Type:  SystemAction<TcpIClient>

Implements
TcpIClientConnected

See Also

Reference
TcpTcpClient Class
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
| Tungsten | \( W \) |
TcpTcpClientDisconnected Event


Namespace:  W.Net

Syntax

C# Copy

```csharp
public event Action<TcpIClient> Disconnected
```

Value
Type:  SystemAction<TcpIClient>
Implements
TcpIClientDisconnected

See Also

Reference
TcpTcpClient Class
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
TcpTcpHost Class

[Missing <summary> documentation for "T:W.Net.Tcp.TcpHost"]

Inheritance Hierarchy

System
  Object
  W.Net
    Tcp
      TcpHost
        W.NetTcpGenericSecureTcpHostTMessage
        W.NetTcpGenericTcpHostTMessage
        W.NetTcpSecureTcpHost

Namespace:  W.Net

Syntax

```c#
public class TcpHost : IDisposable
```

The TcpTcpHost type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TcpTcpHost</td>
<td>Initializes a new instance of the TcpTcpHost class</td>
</tr>
</tbody>
</table>

Properties

| Name | Description |
|------|-------------|-------------|
|      |             |             |
### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispose</td>
<td>Releases all resources used by the <code>TcpTcpHost</code></td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup</td>
</tr>
<tr>
<td></td>
<td>operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <code>Type</code> of the current instance. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>Listen(IPEndPoint)</td>
<td></td>
</tr>
<tr>
<td>Listen(IPEndPoint, Int32)</td>
<td></td>
</tr>
</tbody>
</table>
MemberwiseClone  Creates a shallow copy of the current Object.  (Inherited from Object.)

OnBytesReceived

ToString  Returns a string that represents the current object.  (Inherited from Object.)

Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BytesReceived</td>
<td></td>
</tr>
</tbody>
</table>

Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;! This method should be called in the constructor of any class which has IOwnedProperty members (Defined by</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTType(FuncTType)</td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTType(FuncTType)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>

Scans each field and
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MarkAsClean</td>
<td>Sets the IsDirty flag to false. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock. (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>

**See Also**

- Reference
- W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
TcpTcpHost Constructor

Initializes a new instance of the TcpTcpHost class

**Namespace:** W.Net  
**Assembly:** Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

### Syntax

```
public TcpHost()
```

### See Also

- Reference
  - TcpTcpHost Class
  - W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
TcpHost Properties

The **TcpHost** type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsListening</td>
<td></td>
</tr>
<tr>
<td>OnCreateServer</td>
<td></td>
</tr>
</tbody>
</table>

## See Also

Reference

**TcpTcpHost Class**

**W.Net Namespace**

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
TcpTcpHost.IsListening Property


Namespace: W.Net

▸ Syntax

```csharp
public bool IsListening { get; }
```

Property Value
Type: Boolean

▸ See Also

Reference
TcpTcpHost Class
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
TcpTcpHostOnCreateServer

Property


Namespace: W.Net

Syntax

```
protected virtual Func<Socket, TcpIClient> OnCreateServer
```

Property Value

Type: FuncSocket, TcpIClient

See Also

Reference
TcpTcpHost Class
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$\text{W}$
TcpHost Methods

The TcpTcpHost type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispose</td>
<td></td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>Listen(IPEndPoint)</td>
<td></td>
</tr>
<tr>
<td>Listen(IPEndPoint, Int32)</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>OnBytesReceived</td>
<td></td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object. (Inherited from Object.)</td>
</tr>
</tbody>
</table>

### Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>❁ AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>❁ InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot; This method should be called in the constructor of any class which has IOwnedProperty members (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>❁ InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>InLockTTType(FuncTTType)</td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTTType(FuncTTType)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock</td>
</tr>
</tbody>
</table>
See Also

Reference
TcpTcpHost Class
W.Net Namespace
| Tungsten | W |
TcpTcpHostDispose Method

Releases all resources used by the TcpTcpHost

Namespace: W.Net

Syntax

```csharp
public void Dispose()
```

Implements
IDisposableDispose

See Also

Reference
TcpTcpHost Class
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\(W\)
TcpHostListen Method

▲ Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Listen(IPEndPoint)</td>
<td></td>
</tr>
<tr>
<td>Listen(IPEndPoint, Int32)</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Reference
TcpTcpHost Class
W.Net Namespace

Copyright © 2018 Jordan Duerksen
TcpTcpHostListen Method (IPEndPoint)


Namespace: W.Net

Syntax

C#

```csharp
public void Listen(IPEndPoint ep)
```

Parameters

- `ep` Type: System.Net.IPEndPoint

See Also

Reference
- TcpTcpHost Class
- Listen Overload
- W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
TcpTcpHostListen Method (IPEndPoint, Int32)

[Missing <summary> documentation for
]

Namespace: W.Net

Syntax

```c#
public void Listen(
    IPEndPoint ep,
    int backlog
)
```

Parameters

- **ep**
  - Type: System.Net.IPEndPoint
  - [Missing <param name="ep"/> documentation for
]

- **backlog**
  - Type: System.Int32
  - [Missing <param name="backlog"/> documentation for
]

See Also

Reference
TcpTcpHost Class
Listen Overload
Tungsten

W
TcpTcpHostOnBytesReceived Method


Namespace:  W.Net

Syntax

```csharp
protected virtual void OnBytesReceived(
    TcpIClient client,
    byte[] bytes
)
```

Parameters

client
Type:  W.NetTcpIClient

bytes
Type:  System.Byte

See Also

Reference
TcpTcpHost Class
W.Net Namespace
Tungsten

$W$
TcpHost Events

The TcpTcpHost type exposes the following members.

### Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BytesReceived</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Reference

TcpTcpHost Class

W.Net Namespace

Copyright © 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
<th>W</th>
</tr>
</thead>
</table>
TcpTcpHostBytesReceived Event

[Missing <summary> documentation for "E:W.Net.Tcp.TcpHost.BytesReceived"]

Namespace: W.Net

Syntax

```csharp
public event Action<TcpTcpHost, TcpIClient, byte[]>
```

Value
Type: SystemAction<TcpTcpHost, TcpIClient, Byte>

See Also

Reference
TcpTcpHost Class
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
TcpTcpLogger Class

Sends log messages to a remote server via Tcp

Inheritance Hierarchy

- System
  - Object
    - W.Net
      - Tcp
        - TcpLogger

Namespace: W.Net

Syntax

```csharp
public static class TcpLogger
```

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![S] LogTheMessage(IPEndPoint, String)</td>
<td>Log a message to the remote machine. Note that this message is slower due to parsing the remoteIP with each call</td>
</tr>
<tr>
<td>![S] LogTheMessage(String, Int32, String)</td>
<td>Log a message to the custom logger. Note that this message is slower due to parsing the</td>
</tr>
</tbody>
</table>
remoteIP with each call

Examples

Log.LogTheMessage += (category, message) =>
W.Net.Tcp.TcpLogger.LogTheMessage("127.0.0.1", 5555, true, category, message);

See Also

Reference
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
TcpLogger Methods

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LogTheMessage(IPEndPoint, String)</td>
<td>Log a message to the remote machine. Note that this message is slower due to parsing the remoteIP with each call</td>
</tr>
<tr>
<td>LogTheMessage(String, Int32, String)</td>
<td>Log a message to the custom logger. Note that this message is slower due to parsing the remoteIP with each call</td>
</tr>
</tbody>
</table>

See Also

Reference
TcpTcpLogger Class
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
TcpLoggerLogTheMessage Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LogTheMessage(IPEndPoint, String)</td>
<td>Log a message to the remote machine. Note that this message is slower due to parsing the remoteIP with each call</td>
</tr>
<tr>
<td>LogTheMessage(String, Int32, String)</td>
<td>Log a message to the custom logger. Note that this message is slower due to parsing the remoteIP with each call</td>
</tr>
</tbody>
</table>

See Also

Reference
TcpTcpLogger Class
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( \text{W} \)
TcpTcpLogger.LogTheMessage Method (IPEndPoint, String)

Log a message to the remote machine. Note that this message is slower due to parsing the remoteIP with each call.

**Namespace:** W.Net  
**Assembly:** Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

### Syntax

```c#
public static void LogTheMessage(IPEndPoint remoteIPEndPoint, string message)
```

### Parameters

- **remoteIPEndPoint**
  - Type: `System.Net.IPEndPoint`
  - The IPEndPoint of the remote server

- **message**
  - Type: `System.String`
  - The log message

### See Also

- Reference
  - TcpTcpLogger Class
  - LogTheMessage Overload
  - W.Net Namespace
Tungsten

$W$
TcpTcpLoggerLogTheMessage
Method (String, Int32, String)

Log a message to the custom logger. Note that this message is slower due to parsing the remoteIP with each call.

**Namespace:** W.Net
**Assembly:** Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

### Syntax

```csharp
public static void LogTheMessage(
    string remoteIP,
    int remotePort,
    string message
)
```

### Parameters

- **remoteIP**
  - Type: `SystemString`
  - The IP address or name of the remote server

- **remotePort**
  - Type: `SystemInt32`
  - The port of the remote Udp server

- **message**
  - Type: `SystemString`
  - The log message

### See Also

Reference
TcpTcpLogger Class
LogTheMessage Overload
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
| Tungsten | W |
Udp Class

Provides simple UDP reading and writing

Inheritance Hierarchy

- System
- Object
- W.Net
- Udp

Namespace: W.Net

Syntax

```c#
public static class Udp
```

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SendAsync(IPEndPoint, Byte)</td>
<td>Sends data to a remote via UDP</td>
</tr>
<tr>
<td>SendAsync(IPEndPoint, String, Object)</td>
<td>Sends data to a remote via UDP</td>
</tr>
<tr>
<td>SendAsyncTType(IPEndPoint, TType)</td>
<td>Sends a message to a remote via UDP</td>
</tr>
</tbody>
</table>

See Also
Tungsten

\( W \)
Udp Methods

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SendAsync(IPEndPoint, Byte)</td>
<td>Sends data to a remote via UDP</td>
</tr>
<tr>
<td>SendAsync(IPEndPoint, String, Object)</td>
<td>Sends data to a remote via UDP</td>
</tr>
<tr>
<td>SendAsyncTTType(IPEndPoint, TType)</td>
<td>Sends a message to a remote via UDP</td>
</tr>
</tbody>
</table>

See Also

Reference
Udp Class
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
# UdpSendAsync Method

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SendAsync(IPEndPoint, Byte)</td>
<td>Sends data to a remote via UDP</td>
</tr>
<tr>
<td>SendAsync(IPEndPoint, TType)</td>
<td>Sends a message to a remote via UDP</td>
</tr>
<tr>
<td>SendAsync(IPEndPoint, String, Object)</td>
<td>Sends data to a remote via UDP</td>
</tr>
</tbody>
</table>

## See Also

Reference
- Udp Class
- W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
UdpSendAsync Method (IPEndPoint, Byte)

Sends data to a remote via UDP

**Namespace:** W.Net  
**Assembly:** Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

### Syntax

```csharp
public static Task SendAsync(IPEndPoint ipEndPoint, byte[] bytes)
```

### Parameters

- **ipEndPoint**
  - Type: System.Net.IPEndPoint  
  - The remote machine's endpoint

- **bytes**
  - Type: System.Byte  
  - The data to send

### Return Value

- **Type:** Task  
- The Task associated with this action

### See Also

- Reference
  - Udp Class
SendAsync Overload
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W^-$
UdpSendAsync<\textit{TType}> Method (IPEndPoint, \textit{TType})

Sends a message to a remote via UDP

\textbf{Namespace:} W.Net  

\section*{Syntax}

```csharp
public static Task SendAsync<\textit{TType}>(
    IPEndPoint \textit{ipEndPoint},
    \textit{TType} \textit{message}
)
```

\section*{Parameters}

\textit{ipEndPoint}

Type: System.Net.IPEndPoint  
The remote machine's endpoint

\textit{message}

Type: \textit{TType}  
The message to send

\section*{Type Parameters}

\textit{TType}

[Missing \texttt{<typeparam name=\textit{TType}/>} documentation for \texttt{``M:W.Net.Udp.SendAsync``\textunderscore{}1(System.Net.IPEndPoint,``0)``}]

\section*{Return Value}

Type: Task
The Task associated with this action

See Also

Reference

Udp Class
SendAsync Overload
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
UdpSendAsync Method
(IPEndPoint, String, Object)

Sends data to a remote via UDP

Namespace: W.Net  

Syntax

```
public static Task SendAsync(
    IPEndPoint ipEndPoint,
    string format,
    params Object[] args
)
```

Parameters

`ipEndPoint`
Type: System.Net.IPEndPoint
The remote machine's endpoint

`format`
Type: System.String
The string format

`args`
Type: System.Object
String formatting arguments

Return Value
Type: Task
The Task associated with this action
See Also

Reference
Udp Class
SendAsync Overload
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$^\text{W}$
UdpGeneric Class

Contains the generic implementation of W.Net.UdpPeer

Inheritance Hierarchy

- System
  - Object
    - W.Net
      - Udp
        - Generic

Namespace: W.Net

Syntax

```csharp
public static class Generic
```

See Also

- Reference
  - W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
UdpGenericUdpPeer<TType> Class

A generic Udp peer

Inheritance Hierarchy

```
  System
    Object
      W.Net
        Udp
          Generic
            UdpPeer
```

Namespace: W.Net

Syntax

```
public class UdpPeer<TType> : IDisposable
```

Type Parameters

**TType**

[Missing <typeparam name="TType"/> documentation for "T:W.Net.Udp.Generic.UdpPeer`1"]

The UdpGenericUdpPeer<TType> type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="UdpGenericUdpPeerTType" /></td>
<td>Constructs a new UdpServer</td>
</tr>
</tbody>
</table>

Top
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispose</td>
<td>Disposes the UdpServer and release resources</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object. (Inherited from Object.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>SendAsync</td>
<td>Sends a message to the specified remote client</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object. (Inherited from Object.)</td>
</tr>
</tbody>
</table>
## Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MessageReceived</td>
<td>Raised when a message has been received from a client</td>
</tr>
</tbody>
</table>

## Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot; This method should be called in the constructor of any class which has IOwnedProperty members (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>Method / Property</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>InLockTType(FuncTType)</td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTType(FuncTType)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>RequestAsyncTMessage</td>
<td>Sends a request to a generic Udp peer and</td>
</tr>
</tbody>
</table>
waits for a response. Can be used along with, or in lieu of, the regular method of calling Write and handling the BytesReceived event. (Defined by ExtensionMethods.)

| Unlock | Performs a Monitor unlock (Defined by MonitorExtensions.) |

See Also

Reference

W.Net Namespace
Tungsten

$W$
UdpGenericUdpPeerTTypConstructor

Constructs a new UdpServer

**Namespace:** W.Net  
**Assembly:** Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

### Syntax

```csharp
public UdpPeer(IPEndPoint localEndPoint, bool useCompression)
```

### Parameters

**localEndPoint**
- Type: System.Net.IPEndPoint
- The local IPEndPoint on which to listen for data

**useCompression**
- Type: System.Boolean
- If True, messages will be compressed before sending and decompressed when received

### See Also

Reference
- UdpGenericUdpPeerTTypClass  
- W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$\text{W}$
## UdpPeer<em>T</em>Type Methods

The <em>UdpGenericUdpPeerT</em>Type generic type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Dispose" /></td>
<td>Disposes the UdpServer and release resources</td>
</tr>
<tr>
<td><img src="image" alt="Equals" /></td>
<td>Determines whether the specified object is equal to the current object. (Inherited from &lt;em&gt;Object&lt;/em&gt;.)</td>
</tr>
<tr>
<td><img src="image" alt="Finalize" /></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from &lt;em&gt;Object&lt;/em&gt;.)</td>
</tr>
<tr>
<td><img src="image" alt="GetHashCode" /></td>
<td>Serves as the default hash function. (Inherited from &lt;em&gt;Object&lt;/em&gt;.)</td>
</tr>
<tr>
<td><img src="image" alt="GetType" /></td>
<td>Gets the &lt;em&gt;Type&lt;/em&gt; of the current instance. (Inherited from &lt;em&gt;Object&lt;/em&gt;.)</td>
</tr>
<tr>
<td><img src="image" alt="MemberwiseClone" /></td>
<td>Creates a shallow copy of the current &lt;em&gt;Object&lt;/em&gt;. (Inherited from &lt;em&gt;Object&lt;/em&gt;.)</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SendAsync</td>
<td>Sends a message to the specified remote client</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object. (Inherited from Object.)</td>
</tr>
</tbody>
</table>

### Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTTType(FuncTTType)</td>
<td>Overloaded. Performs the function in a Monitor lock.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTType(FuncTType)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets it's IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>RequestAsyncTMessage</td>
<td>Sends a request to a generic Udp peer and waits for a response. Can be used along with, or in lieu of, the regular</td>
</tr>
</tbody>
</table>
method of calling Write and handling the BytesReceived event. (Defined by ExtensionMethods.)

Unlock

Performs a Monitor unlock (Defined by MonitorExtensions.)

See Also

Reference

UdpGenericUdpPeerTType Class
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
UdpGenericUdpPeer<TType>Dispose Method

Disposes the UdpServer and release resources

**Namespace:** W.Net

**Assembly:** Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

### Syntax

```csharp
public void Dispose()
```

Implements

IDisposableDispose

### See Also

Reference

UdpGenericUdpPeer<TType> Class

W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
UdpGenericUdpPeer<TType>SendAsync Method

Sends a message to the specified remote client

**Namespace:** W.Net  
**Assembly:** Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

### Syntax

```csharp
public Task SendAsync(TType message, IPEndPoint remoteEndPoint)
```

**Parameters**

- **message**
  - Type: `TType`
  - The message to send

- **remoteEndPoint**
  - Type: `System.Net.IPEndPoint`
  - The remote client which is listening for messages

**Return Value**

- Type: `Task`


### See Also

Reference
UdpGenericUdpPeerTType Class
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$^7_2 W$
UdpPeer\textit{TType} Events

The \texttt{UdpGenericUdpPeer\textit{TType}} generic type exposes the following members.

### Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{MessageReceived}</td>
<td>Raised when a message has been received from a client</td>
</tr>
</tbody>
</table>

See Also

Reference

\texttt{UdpGenericUdpPeer\textit{TType} Class}

\texttt{W.Net Namespace}

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
UdpGenericUdpPeerTTyPeMessage Event

Raised when a message has been received from a client

Namespace: W.Net

Syntax

C#

```csharp
public event Action<IPEndPoint, TType> MessageReceived
```

Value
Type: SystemAction<IPEndPoint, TType>

See Also

Reference
UdpGenericUdpPeerTTyPe Class
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
UdpUdpLogger Class

Sends log messages to a remote server via Udp

Inheritance Hierarchy

- System
- Object
- W.Net
- Udp
- UdpLogger

Namespace: W.Net

Syntax

```csharp
public static class UdpLogger
```

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="LogTheMessage(IPEndPoint, String)" /></td>
<td>Log a message to the remote machine</td>
</tr>
<tr>
<td><img src="image" alt="LogTheMessage(String, Int32, String)" /></td>
<td>Log a message to the remote machine. Note that this message is slower due to parsing the remoteIP with each call</td>
</tr>
</tbody>
</table>
Examples

Log.LogTheMessage += (category, message) =>
W.Net.Udp.UdpLogger.LogTheMessage("127.0.0.1", 5555, true, category, message);

See Also

Reference
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
</tr>
</tbody>
</table>
# UdpLogger Methods

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LogTheMessage(IPEndPoint, String)</td>
<td>Log a message to the remote machine</td>
</tr>
<tr>
<td>LogTheMessage(String, Int32, String)</td>
<td>Log a message to the remote machine. Note that this message is slower due to parsing the remoteIP with each call</td>
</tr>
</tbody>
</table>

**See Also**

Reference

UdpUdpLogger Class

W.Net Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
# UdpLogger Log The Message Method

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LogTheMessage(IPEndPoint, String)</td>
<td>Log a message to the remote machine</td>
</tr>
<tr>
<td>LogTheMessage(String, Int32, String)</td>
<td>Log a message to the remote machine. Note that this message is slower due to parsing the remoteIP with each call</td>
</tr>
</tbody>
</table>

## See Also

Reference

- Udp
- UdpLogger Class
- W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
UdpUdpLoggerLogTheMessage Method (IPEndPoint, String)

Log a message to the remote machine

**Namespace:** W.Net  
**Assembly:** Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

## Syntax

```csharp
public static void LogTheMessage(
    IPEndPoint remoteIPEndPoint,
    string message
)
```

### Parameters

- **remoteIPEndPoint**
  - Type: System.Net.IPEndPoint
  - The IPEndPoint of the remote log server

- **message**
  - Type: System.String
  - The log message

## See Also

- **Reference**
  - UdpUdpLogger Class
  - LogTheMessage Overload
  - W.Net Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

W
UdpUdpLoggerLogTheMessage Method (String, Int32, String)

Log a message to the remote machine. Note that this message is slower due to parsing the remoteIP with each call

Namespace: W.Net

Syntax

```csharp
public static void LogTheMessage(
    string remoteIP,
    int remotePort,
    string message
)
```

Parameters

remoteIP
Type: System.String
The IP address or name of the remote server

remotePort
Type: System.Int32
The port of the remote Udp server

message
Type: System.String
The log message

See Also

Reference
Tungsten

W
# UdpUdpPeer Class

A Udp peer

## Inheritance Hierarchy

- System
  - Object
  - W.NetUdpUdpPeer

**Namespace:** W.Net  
**Assembly:** Tungsten.Net (in Tungsten.Net.dll)  
**Version:** 2.0.2

## Syntax

```csharp
public class UdpPeer : IDisposable
```

The `UdpUdpPeer` type exposes the following members.

## Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>UdpUdpPeer</code></td>
<td>Constructs a new UdpServer</td>
</tr>
</tbody>
</table>

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Dispose</code></td>
<td>Disposes the UdpServer and release resources</td>
</tr>
<tr>
<td><code>Equals</code></td>
<td>Determines whether the</td>
</tr>
</tbody>
</table>
specified object is equal to the current object. (Inherited from \texttt{Object}.)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from \texttt{Object}.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as the default hash function. (Inherited from \texttt{Object}.)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the \texttt{Type} of the current instance. (Inherited from \texttt{Object}.)</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current \texttt{Object}. (Inherited from \texttt{Object}.)</td>
</tr>
<tr>
<td><strong>SendAsync</strong></td>
<td>Sends bytes to the specified remote client</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Returns a string that represents the current object. (Inherited from \texttt{Object}.)</td>
</tr>
</tbody>
</table>

**Top**

**Events**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BytesReceived</strong></td>
<td>Raised when bytes have been received from a client</td>
</tr>
</tbody>
</table>
## Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTType(Func&lt;TType&gt;)</td>
<td>Overloaded. Performs the function in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor</td>
</tr>
<tr>
<td>Method/Property</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>InLockAsync&lt;TType&gt;()</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets it's IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>RequestAsync</td>
<td>Sends a request to a Udp peer and waits for a response. Can be used along with, or in lieu of, the regular method of calling Write and handling the BytesReceived event. (Defined by ExtensionMethods.)</td>
</tr>
</tbody>
</table>
Unlock

Performs a Monitor unlock (Defined by MonitorExtensions.)

See Also

Reference
W.Net Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
UdpUdpPeer Constructor

Constructs a new UdpServer

**Namespace:** W.Net  
**Assembly:** Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

### Syntax

```csharp
public UdpPeer(IPEndPoint localEndPoint, bool useCompression)
```

### Parameters

- **localEndPoint**  
  Type: System.Net.IPEndPoint  
  The local IPEndPoint on which to listen for data

- **useCompression**  
  Type: System.Boolean  
  If True, data is compressed before sending and decompressed when received

### See Also

- Reference  
  UdpUdpPeer Class  
  W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\text{W}
# UdpPeer Methods

The `UdpUdpPeer` type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispose</td>
<td>Disposes the UdpServer and release resources</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <code>Type</code> of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current <code>Object</code>.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
</tbody>
</table>
SendAsync  Sends bytes to the specified remote client

ToString   Returns a string that represents the current object. (Inherited from Object.)

Top

Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTType(FuncTType)</td>
<td>Overloaded. Performs the function in a Monitor lock</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync&lt;TType&gt;(Func&lt;TType&gt;)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>RequestAsync</td>
<td>Sends a request to a Udp peer and waits for a response. Can be used along with, or in lieu of, the regular method of</td>
</tr>
</tbody>
</table>
calling Write and handling the BytesReceived event. (Defined by ExtensionMethods.)

Unlock

Performs a Monitor unlock (Defined by MonitorExtensions.)

See Also

Reference
UdpUdpPeer Class
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
UdpUdpPeerDispose Method

Disposes the UdpServer and release resources

**Namespace:** W.Net  
**Assembly:** Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

### Syntax

```csharp
public void Dispose()
```

Implements

IDisposableDispose

### See Also

Reference

UdpUdpPeer Class  
W.Net Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
UdpUdpPeerSendAsync Method

Sends bytes to the specified remote client

**Namespace:** W.Net  
**Assembly:** Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

### Syntax

```csharp
public Task SendAsync(
    byte[] bytes,
    IPEndPoint remoteEndPoint
)
```

### Parameters

- **bytes**  
  - Type: System.Byte  
  - The data to send

- **remoteEndPoint**  
  - Type: System.Net.IPEndPoint  
  - The remote client which is listening for data

### Return Value

- Type: Task


### See Also

- **Reference**  
  - UdpUdpPeer Class  
  - W.Net Namespace
Copyright © 2018 Jordan Duerksen
| Tungsten | W |
UdpPeer Events

The UdpUdpPeer type exposes the following members.

## Events

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt=" " /></td>
<td>BytesReceived: Raised when bytes have been received from a client</td>
</tr>
</tbody>
</table>

See Also

Reference

UdpUdpPeer Class

W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W^-$
UdpUdpPeerBytesReceived Event

Raised when bytes have been received from a client

Namespace: W.Net

Syntax

```csharp
public event Action<IPEndPoint, byte[]> BytesReceived
```

Value
Type: SystemAction<IPEndPoint, Byte

See Also

Reference
UdpUdpPeer Class
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
WakeOnLAN Class

Methods to broadcast a magic packet to wake up a machine with the given MAC address

Inheritance Hierarchy

- System
  - Object
    - W.Net
      - WakeOnLAN

Namespace: W.Net

Syntax

```
public class WakeOnLAN
```

The **WakeOnLAN** type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WakeOnLAN</td>
<td>Initializes a new instance of the <strong>WakeOnLAN</strong> class</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as the default hash function. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Returns a string that represents the current object. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>Wakeup(String, String, Int32)</strong></td>
<td>Broadcasts a magic packet to wake up the machine with the given MAC address</td>
</tr>
<tr>
<td><strong>Wakeup(String, String, String)</strong></td>
<td>Broadcasts a magic packet to wake up the machine with the given MAC address</td>
</tr>
<tr>
<td><strong>Wakeup_via_Socket</strong></td>
<td>Broadcasts a magic packet</td>
</tr>
</tbody>
</table>
to wake up the machine with the given MAC address

### Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTType(FuncTType)</td>
<td>Overloaded. Performs the function in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTTType(FuncTTType)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>
See Also

Reference

W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
WakeOnLAN Constructor

Initializes a new instance of the `WakeOnLAN` class

**Namespace:** W.Net  
**Assembly:** Tungsten.Net (in Tungsten.Net.dll)  
**Version:** 2.0.2

### Syntax

```csharp
public WakeOnLAN()
```

### See Also

Reference  
- `WakeOnLAN Class`  
- `W.Net Namespace`

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
WakeOnLAN Methods

The **WakeOnLAN** type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>```{Equals}</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>```{Finalize}</td>
<td>Allows an object to try to free resources and perform other cleanup</td>
</tr>
<tr>
<td></td>
<td>operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>```{GetHashCode}</td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>```{GetType}</td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>```{MemberwiseClone}</td>
<td>Creates a shallow copy of the current <strong>Object</strong>.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>```{ToString}</td>
<td>Returns a string that represents the current object.</td>
</tr>
</tbody>
</table>
WakeUp(String, String, Int32)
Broadcasts a magic packet to wake up the machine with the given MAC address

WakeUp(String, String, String)
Broadcasts a magic packet to wake up the machine with the given MAC address

WakeUp_via_Socket
Broadcasts a magic packet to wake up the machine with the given MAC address

---

**Extension Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot; This method should be called in the constructor of any class which has IOwnedProperty members  (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Method/Signature</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>InLock(Action)</code></td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by <code>MonitorExtensions</code>).</td>
</tr>
<tr>
<td><code>InLockTType(FuncTType)</code></td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by <code>MonitorExtensions</code>).</td>
</tr>
<tr>
<td><code>InLockAsync(Action)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by <code>MonitorExtensions</code>).</td>
</tr>
<tr>
<td><code>InLockAsyncTType(FuncTType)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by <code>MonitorExtensions</code>).</td>
</tr>
<tr>
<td><code>IsDirty</code></td>
<td>Scans the IsDirty value of each field and property of type <code>IProperty</code> (Defined by <code>PropertyHostExtensions</code>).</td>
</tr>
<tr>
<td><code>Lock</code></td>
<td>Performs a Monitor lock (Defined by <code>MonitorExtensions</code>).</td>
</tr>
<tr>
<td><code>MarkAsClean</code></td>
<td>Scans each field and property of type <code>IProperty</code> and sets its IsDirty flag to</td>
</tr>
</tbody>
</table>
false
(Defined by PropertyHostExtensions.)

_unlock
Performs a Monitor unlock
(Defined by MonitorExtensions.)

See Also

Reference
WakeOnLAN Class
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
WakeOnLAN WakeUp Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WakeUp(String, String, Int32)</td>
<td>Broadcasts a magic packet to wake up the machine with the given MAC address</td>
</tr>
<tr>
<td>WakeUp(String, String, String)</td>
<td>Broadcasts a magic packet to wake up the machine with the given MAC address</td>
</tr>
</tbody>
</table>

See Also

Reference
- WakeOnLAN Class
- W.Net Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

\( W \)
WakeOnLANWakeUp Method (String, String, Int32)

Broadcasts a magic packet to wake up the machine with the given Mac address

**Namespace:** W.Net  
**Assembly:** Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

**Syntax**

```csharp
public static bool WakeUp(
    string macAddress,
    string ipBroadcastAddress = "192.168.1.255",
    int port = 9
)
```

**Parameters**

- **macAddress**
  - Type: SystemString
  - The MAC address of the machine to wake up

- **ipBroadcastAddress** *(Optional)*
  - Type: SystemString
  - The broadcast address. This should be determined by the machine's IP address and the desired network mask.

- **port** *(Optional)*
  - Type: SystemInt32
  - The socket port

**Return Value**

Type: Boolean
True if the magic packet was successfully sent, otherwise False

Remarks

Adapted from: https://www.codeproject.com/Articles/5315/Wake-On-Lan-sample-for-C

See Also

Reference
WakeOnLAN Class
WakeUp Overload
W.Net Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

W
WakeOnLAN WakeUp Method
(String, String, String)

Broadcasts a magic packet to wake up the machine with the given MAC address

Namespace: W.Net

Syntax

C# language

```csharp
public static bool WakeUp(
    string macAddress,
    string ipAddress,
    string subnetMask
)
```

Parameters

* `macAddress`
  Type: `System.String`
  The MAC address of the machine to wake up

* `ipAddress`
  Type: `System.String`
  The IP address of the machine to wake

* `subnetMask`
  Type: `System.String`
  The subnet mask to determine the broadcast IP address

Return Value

Type: `Boolean`
True if the magic packet was successfully sent, otherwise False
Remarks

Taken from:

See Also

Reference
WakeOnLAN Class
WakeUp Overload
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
WakeOnLANWakeUp_via_Socket Method

Broadcasts a magic packet to wake up the machine with the given MAC address

**Namespace:** W.Net  
**Assembly:** Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

### Syntax

```csharp
public static bool WakeUp_via_Socket(
    string macAddress,
    string broadcastAddress = "192.168.1.255",
    int port = 9
)
```

### Parameters

- **macAddress**  
  - Type: System.String  
  - The MAC address of the machine to wake up

- **broadcastAddress** *(Optional)*  
  - Type: System.String  
  - The port on which to send the magic packet (typically 0, 7 or 9)

- **port** *(Optional)*  
  - Type: System.Int32  

### Return Value

- Type: Boolean
True if the magic packet was successfully sent, otherwise False

Remarks
Adapted from: https://www.codeproject.com/Articles/5315/Wake-On-Lan-sample-for-C

See Also
Reference
WakeOnLAN Class
W.Net Namespace

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>$W$</td>
</tr>
</tbody>
</table>
# W.Net.RPC Namespace

[Missing <summary> documentation for "N:W.Net.RPC"]

## Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client</td>
<td>Make calls into a Tungsten.Net.RPC.Server over Tcp</td>
</tr>
<tr>
<td>MethodDictionary</td>
<td>Used to store and call RPC methods on a Tungsten.Net.RPC Server</td>
</tr>
<tr>
<td>RPCClassAttribute</td>
<td>Add this attribute to a class if it contains static methods with the RPCMethod attribute</td>
</tr>
<tr>
<td>RPCMethodAttribute</td>
<td>Add this attribute to a static method and Tungsten.Net.RPC.Server can automatically add the method to it's dictionary of callable methods.</td>
</tr>
<tr>
<td>RPCResponse</td>
<td>Encapsulates information related to making the RPC call and the return value</td>
</tr>
<tr>
<td>RPCResponseTResponseType</td>
<td>Encapsulates information related to making the</td>
</tr>
<tr>
<td>Server</td>
<td>Allows remote instances of Tungsten.Net.RPC.Client to call local methods.</td>
</tr>
</tbody>
</table>

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
## Client Class

Make calls into a Tungsten.Net.RPC.Server over Tcp

### Inheritance Hierarchy

```
System
  Object
  W.Net.RPCClient
```

**Namespace:** W.Net.RPC  
**Assembly:** Tungsten.Net (in Tungsten.Net.dll)  
**Version:** 2.0.2

### Syntax

```csharp
public class Client : IDisposable
```

The `Client` type exposes the following members.

### Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client(Int32)</td>
<td>Constructs a new Client</td>
</tr>
<tr>
<td>Client(IPEndPoint, Int32, Int32)</td>
<td>Constructs a new Client, initialized with the specified values</td>
</tr>
</tbody>
</table>

### Properties
### CallTimeout
The maximum amount of time, in milliseconds, that a call should wait for a response

### RemoteEndPoint
The IPEndPoint of the server
(The server must be a valid instance of W.Net.RPC.Server)

# Top

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call(String, Object)</td>
<td>Calls a method on the server</td>
</tr>
<tr>
<td>CallResponseType(String, Object)</td>
<td>Calls a method on the server</td>
</tr>
<tr>
<td>CallAsync(String, Object)</td>
<td>Calls a method on the server</td>
</tr>
<tr>
<td>CallAsyncResponseType(String, Object)</td>
<td>Calls a method on the server</td>
</tr>
<tr>
<td>Dispose</td>
<td>Disposes the Client and release resources</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>object is equal to the current object. (Inherited from Object.)</td>
<td></td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as the default hash function. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
</tbody>
</table>
Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ToString</code></td>
<td>Returns a string that represents the current object. (Inherited from Object.)</td>
</tr>
<tr>
<td><code>AsTType</code></td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td><code>InitializeProperties</code></td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td><code>InLock(Action)</code></td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><code>InLockTType(FuncTType)</code></td>
<td>Overloaded. Performs the function in a Monitor lock</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTTType(FuncTTType)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>
Remarks

Assymetric encryption is used to discourage sniffing. The server must be a valid instance of W.Net.RPC.Server

See Also

Reference

W.Net.RPC Namespace
Tungsten

$W$
Client Constructor

**Overload List**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![ ] Client(Int32)</td>
<td>Constructs a new Client</td>
</tr>
<tr>
<td>![ ] Client(IPEndPoint, Int32, Int32)</td>
<td>Constructs a new Client, initialized with the specified values</td>
</tr>
</tbody>
</table>

**See Also**

Reference

Client Class

W.Net.RPC Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
Client Constructor (Int32)

Constructs a new Client

Namespace: W.Net.RPC

Syntax

```csharp
public Client(
    int encryptionKeySize
)
```

Parameters

`encryptionKeySize`
Type: System.Int32
The encryption key size (typically 2048 or 4096; 384 to 16384 in increments of 8)

See Also

Reference
Client Class
Client Overload
W.Net.RPC Namespace

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
<th>W</th>
</tr>
</thead>
</table>
Client Constructor (IPEndPoint, Int32, Int32)

Constructs a new Client, initialized with the specified values

Namespace: W.Net.RPC

Syntax

C#  
public Client
  (IPEndPoint remoteEndPoint,
   int encryptionKeySize,
   int msCallTimeout = -1
  )

Parameters

remoteEndPoint
  Type: System.Net.IPEndPoint
  The server's IP address and port

encryptionKeySize
  Type: System.Int32
  The encryption key size (typically 2048 or 4096; 384 to 16384 in
  increments of 8)

msCallTimeout (Optional)
  Type: System.Int32
  The maximum number of milliseconds to wait for a call to complete

See Also
Reference
Client Class
Client Overload
W.Net.RPC Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

\( W \)
Client Properties

The Client type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CallTimeout</td>
<td>The maximum amount of time, in milliseconds, that a call should wait for a response</td>
</tr>
<tr>
<td>RemoteEndPoint</td>
<td>The IPEndPoint of the server (The server must be a valid instance of W.Net.RPC.Server)</td>
</tr>
</tbody>
</table>

See Also

Reference

Client Class

W.Net.RPC Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
ClientCallTimeout Property

The maximum amount of time, in milliseconds, that a call should wait for a response

**Namespace:** W.Net.RPC  
**Assembly:** Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

### Syntax

```csharp
public int CallTimeout { get; set; }
```

### Property Value

Type: Int32

### See Also

Reference
- **Client Class**
- **W.Net.RPC Namespace**

Copyright © 2018 Jordan Duerksen
Tungsten

$^	ext{73}$
ClientRemoteEndPoint Property

The IPEndPoint of the server (The server must be a valid instance of W.Net.RPC.Server)

Namespace: W.Net.RPC

Syntax

```csharp
public IPEndPoint RemoteEndPoint { get; set; }
```

Property Value
Type: IPEndPoint

See Also
Reference
Client Class
W.Net.RPC Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
## Client Methods

The **Client** type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Call(String, Object)</strong></td>
<td>Calls a method on the server</td>
</tr>
<tr>
<td><strong>CallTResponseType(String, Object)</strong></td>
<td>Calls a method on the server</td>
</tr>
<tr>
<td><strong>CallAsync(String, Object)</strong></td>
<td>Calls a method on the server</td>
</tr>
<tr>
<td><strong>CallAsyncTResponseType(String, Object)</strong></td>
<td>Calls a method on the server</td>
</tr>
<tr>
<td><strong>Dispose</strong></td>
<td>Disposes the Client and release resources</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that</td>
</tr>
</tbody>
</table>
Top

## Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Icon] AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>![Icon] InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>![Icon] InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>![Icon] InLockTType(FuncTType)</td>
<td>Overloaded. Performs the function in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>InLockAsync(Action)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock. Defined by MonitorExtensions.</td>
</tr>
<tr>
<td><code>InLockAsync&lt;TType&gt;(Func&lt;TType&gt;)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock. Defined by MonitorExtensions.</td>
</tr>
<tr>
<td><code>IsDirty</code></td>
<td>Scans the IsDirty value of each field and property of type IProperty. Defined by PropertyHostExtensions.</td>
</tr>
<tr>
<td><code>Lock</code></td>
<td>Performs a Monitor lock. Defined by MonitorExtensions.</td>
</tr>
<tr>
<td><code>MarkAsClean</code></td>
<td>Scans each field and property of type IProperty and sets it's IsDirty flag to false. Defined by PropertyHostExtensions.</td>
</tr>
<tr>
<td><code>Unlock</code></td>
<td>Performs a Monitor unlock. Defined by MonitorExtensions.</td>
</tr>
</tbody>
</table>

Top
See Also

Reference

Client Class
W.Net.RPC Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
ClientCall Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call(String, Object)</td>
<td>Calls a method on the server</td>
</tr>
<tr>
<td>CallTResponseType(String, Object)</td>
<td>Calls a method on the server</td>
</tr>
</tbody>
</table>

See Also

Reference
Client Class
W.Net.RPC Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
ClientCall Method (String, Object)

Calls a method on the server

**Namespace:** W.Net.RPC  
**Assembly:** Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

### Syntax

```csharp
public RPCResponse Call(
    string methodName,
    params Object[] args
)
```

### Parameters

- **methodName**
  - Type: `System.String`
  - The name of the method to call

- **args**
  - Type: `System.Object`
  - Any arguments to pass into the method

### Return Value

- Type: `RPCResponse`
  - `RPCResponse` containing information related to the call and the return value

### See Also

- Reference
Client Class
Call Overload
W.Net.RPC Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
ClientCall\textit{TResponseType} Method (String, Object)

Calls a method on the server

**Namespace:** W.Net.RPC  
**Assembly:** Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

### Syntax

```c#
public RPCResponse\text{<TResponseType> Call\text{<TResponseType>(
    string methodName,
    params Object[] args
)}
```

### Parameters

- **methodName**
  - Type: \text{SystemString}
  - The name of the method to call

- **args**
  - Type: \text{SystemObject}
  - Any arguments to pass into the method

### Type Parameters

- **TResponseType**
  - [Missing <typeparam name="TResponseType"/> documentation for "M:W.Net.RPC.Client.Call`1(System.String, System.Object[])""]

### Return Value

- Type: \text{RPCResponse\text{<TResponseType>}}
RPCResponse containing information related to the call and the return value

See Also

Reference

Client Class
Call Overload
W.Net.RPC Namespace

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>$^7$W</td>
</tr>
</tbody>
</table>
# ClientCallAsync Method

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CallAsync(String, Object)</td>
<td>Calls a method on the server</td>
</tr>
<tr>
<td>CallAsyncTResponseType(String, Object)</td>
<td>Calls a method on the server</td>
</tr>
</tbody>
</table>

## See Also

- Reference
- Client Class
- W.Net.RPC Namespace

Copyright © 2018 Jordan Duerksen
| Tungsten | W |
ClientCallAsync Method (String, Object)

Calls a method on the server

**Namespace:** W.Net.RPC  
**Assembly:** Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

## Syntax

```csharp
public Task<RPCResponse> CallAsync(
    string methodName,
    params Object[] args
)
```

### Parameters

- **methodName**
  - Type: System.String
  - The name of the method to call

- **args**
  - Type: System.Object
  - Any arguments to pass into the method

### Return Value

- Type: Task<RPCResponse>
  - RPCResponse containing information related to the call and the return value

## See Also

Reference
Client Class
CallAsync Overload
W.Net.RPC Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
ClientCallAsync TResponseType Method (String, Object)

Calls a method on the server

**Namespace:** W.Net.RPC  
**Assembly:** Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

### Syntax

```csharp
public Task<RPCResponse<TResponseType>> CallAsync
    string methodName,
    params Object[] args
}
```

### Parameters

**methodName**
- **Type:** System.String
- The name of the method to call

**args**
- **Type:** System.Object
- Any arguments to pass into the method

### Type Parameters

**TResponseType**

[Missing <typeparam name="TResponseType"/> documentation for "M:W.Net.RPC.Client.CallAsync`1(System.String,System.Object[])"]

### Return Value

**Type:** Task<RPCResponse<TResponseType>>
RPCResponse containing information related to the call and the return value

See Also

Reference
Client Class
CallAsync Overload
W.Net.RPC Namespace
Tungsten

W
ClientDispose Method

Disposes the Client and release resources

**Namespace:**  W.Net.RPC  
**Assembly:**  Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

### Syntax

```
public void Dispose()
```

Implements

IDisposableDispose

### See Also

Reference

Client Class
W.Net.RPC Namespace
Tungsten

$W$
MethodDictionary Class

Used to store and call RPC methods on a Tungsten.Net.RPC Server

Inheritance Hierarchy

- System
- Object
- W.Net.RPCMethodDictionary

Namespace:  W.Net.RPC

Syntax

```csharp
public class MethodDictionary
```

The `MethodDictionary` type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MethodDictionary</td>
<td>Initializes a new instance of the <code>MethodDictionary</code> class</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Methods</td>
<td>All of the RPC methods found. The name of each method is the complete namespace.class.methodname</td>
</tr>
</tbody>
</table>
hierarchy (recursive classes are allowed).

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Call</strong> (Object, Exception, String, Object)</td>
<td>Call a method on the Tungsten.Net.RPC Server. This method s</td>
</tr>
<tr>
<td><strong>CallTResult</strong> (String, Object)</td>
<td>Call a method on the Tungsten.Net.RPC Server.</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified object is equal to the current object. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as the default hash function. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current Object.</td>
</tr>
</tbody>
</table>
Refresh
Scans the server process for RPC methods (static methods with the RPCMethod attribute in classes with the RPCClass attribute)

ToString
Returns a string that represents the current object.
(Inherited from Object.)

Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;! This method should be called in the constructor of any class which has IOwnedProperty members (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>InLockTType(Func&lt;TType&gt;)</td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTType(Func&lt;TType&gt;)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
</tbody>
</table>
Unlock

Performs a Monitor unlock (Defined by MonitorExtensions.)

See Also

Reference

W.Net.RPC Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
MethodDictionary Constructor

Initializes a new instance of the MethodDictionary class

Namespace: W.Net.RPC

Syntax

```csharp
public MethodDictionary()
```

See Also

Reference

MethodDictionary Class
W.Net.RPC Namespace

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
<th>W</th>
</tr>
</thead>
</table>

MethodDictionary Properties

The MethodDictionary type exposes the following members.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Methods</td>
<td>All of the RPC methods found. The name of each method is the complete namespace.class.methodname hierarchy (recursive classes are allowed).</td>
</tr>
</tbody>
</table>

See Also

Reference
- MethodDictionary Class
- W.Net.RPC Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
MethodDictionary

Methods

Property

All of the RPC methods found. The name of each method is the complete namespace.class.methodname hierarchy (recursive classes are allowed).

Namespace:  W.Net.RPC

Syntax

C#

```csharp
public Dictionary<string, MethodInfo> Methods { 
```

Property Value

Type:  Dictionary<String, MethodInfo>

See Also

Reference

MethodDictionary Class
W.Net.RPC Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
The `MethodDictionary` type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call(Object, Exception, String, Object)</td>
<td>Call a method on the Tungsten.Net.RPC Server. This method s</td>
</tr>
<tr>
<td>CallTResult(String, Object)</td>
<td>Call a method on the Tungsten.Net.RPC Server.</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <code>Type</code> of the current instance. (Inherited from <code>Object</code>.)</td>
</tr>
</tbody>
</table>
MemberwiseClone  Creates a shallow copy of the current Object. (Inherited from Object.)

Refresh  Scans the server process for RPC methods (static methods with the RPCMethod attribute in classes with the RPCClass attribute)

ToString  Returns a string that represents the current object. (Inherited from Object.)

### Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot; This method should be called in the constructor of any class which has IOwnedProperty members (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>InLockTTType(FuncTTType)</code></td>
<td>Performed the action in a Monitor lock (Defined by MonitorExtensions.) Overloaded.</td>
</tr>
<tr>
<td><code>InLockAsync(Action)</code></td>
<td>Performs the function in a Monitor lock (Defined by MonitorExtensions.) Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><code>InLockAsyncTTType(FuncTTType)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><code>IsDirty</code></td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td><code>Lock</code></td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><code>MarkAsClean</code></td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false</td>
</tr>
</tbody>
</table>
Unlock

Performs a Monitor unlock

(Defined by MonitorExtensions.)

See Also

Reference
MethodDictionary Class
W.Net.RPC Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
MethodDictionary

Call Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CallTResult(String, Object)</td>
<td>Call a method on the Tungsten.Net.RPC Server.</td>
</tr>
<tr>
<td>Call(Object, Exception, String, Object)</td>
<td>Call a method on the Tungsten.Net.RPC Server. This method is</td>
</tr>
</tbody>
</table>

See Also

Reference
MethodDictionary Class
W.Net.RPC Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
MethodDictionaryCall<TResult> Method (String, Object)


**Namespace:**  W.Net.RPC  
**Assembly:**  Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

## Syntax

```
C#  Copy

public TResult Call<TResult>(
    string method,
    params Object[] args
)
```

### Parameters

**method**
Type: `System.String`  
The namespace, class name and method name of the method to call (ie: MyNamespace.MyClass.Method1)

**args**
Type: `System.Object`  
Arguments, if any, to be passed into the remote method

## Type Parameters

**TResult**
The expected return type of the call

## Return Value

**TResult**
A result of type TResult
Remarks

If TResult does not match the return type of the method on the server, a return value cannot be expected and the call may time out.

See Also

Reference
- MethodDictionary Class
- Call Overload
- W.Net.RPC Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

\( W \)
MethodDictionary

**Call Method**

(Object, Exception, String, Object)

Call a method on the Tungsten.Net.RPC Server. This method is

**Namespace:**  W.Net.RPC
**Assembly:**  Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

## Syntax

```csharp
public bool Call(
    out Object result,
    out Exception exception,
    string method,
    params Object[] args
)
```

### Parameters

- **result**
  - Type: `SystemObject`
  - The value returned from the called method

- **exception**
  - Type: `SystemException`
  - The exception if one occurred

- **method**
  - Type: `SystemString`
  - The namespace, class name and method name of the method to call (ie: MyNamespace.MyClass.Method1)

- **args**
Type: **SystemObject**
Arguments, if any, to be passed into the remote method

Return Value
Type: **Boolean**
A CallResult object describing the result of the call. If the remote method does not have a return value, the value of CallResult.Result will be null.

**See Also**

Reference
**MethodDictionary Class**
**Call Overload**
**W.Net.RPC Namespace**

Copyright @ 2018 Jordan Duerksen
Tungsten

$\text{W}$
MethodDictionaryRefresh Method

Scans the server process for RPC methods (static methods with the RPCMethod attribute in classes with the RPCClass attribute)

**Namespace:** W.Net.RPC  
**Assembly:** Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

### Syntax

```csharp
public void Refresh(
    Assembly rootAssembly,
    bool recurse
)
```

### Parameters

- **rootAssembly**
  - Type: System.ReflectionAssembly  
  - The Assembly to scan for RPC methods

- **recurse**
  - Type: SystemBoolean  
  - If True, referenced assemblies will also be scanned

### Remarks

Any methods previously added manually will have to be re-added

### See Also

Reference
<table>
<thead>
<tr>
<th>Tungsten</th>
<th>W</th>
</tr>
</thead>
</table>

RPCClassAttribute Class

Add this attribute to a class if it contains static methods with the RPCMethod attribute

Inheritance Hierarchy

- System
  - SystemObject
    - SystemAttribute
      - W.Net.RPC
        - RPCClassAttribute

Namespace: W.Net.RPC

Syntax

```csharp
public class RPCClassAttribute : Attribute
```

The `RPCClassAttribute` type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>__RPCClassAttribute</td>
<td>Initializes a new instance of the <code>RPCClassAttribute</code> class</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TypeId</td>
<td>When implemented in a derived class,</td>
</tr>
</tbody>
</table>
gets a unique identifier for this Attribute.
(Inherited from Attribute.)

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Returns a value that indicates whether this instance is equal to a specified object. (Inherited from Attribute.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Returns the hash code for this instance. (Inherited from Attribute.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>IsDefaultAttribute</td>
<td>When overridden in a derived class, indicates whether the value of this instance is the default value for the derived class. (Inherited from Attribute.)</td>
</tr>
<tr>
<td>Match</td>
<td>When overridden in a derived class, returns a value that</td>
</tr>
</tbody>
</table>
indicates whether this instance equals a specified object.  (Inherited from **Attribute**.)

**MemberwiseClone**  
Creates a shallow copy of the current **Object**.  
(Inherited from **Object**.)

**ToString**  
Returns a string that represents the current object.  
(Inherited from **Object**.)

## Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| AsTType             | Use Generic syntax for the as operator.  
(Defined by **AsExtensions**.) |
| InitializeProperties | Scans the fields and properties of "owner" and sets the member's Owner property to "owner".  
This method should be called in the constructor of any class which has IOwnedProperty members  
(Defined by **PropertyHostExtensions**.) |
| InLock(Action)      | Overloaded.  
Performs the action in a Monitor lock |
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>InLockTType(Func&lt;TType&gt;)</strong></td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>InLockAsync(Action)</strong></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>InLockAsyncTType(Func&lt;TType&gt;</strong></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>IsDirty</strong></td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td><strong>Lock</strong></td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>MarkAsClean</strong></td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
</tbody>
</table>
Unlock

Performs a Monitor unlock (Defined by MonitorExtensions.)

See Also

Reference

W.Net.RPC Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
**RPCClassAttribute Constructor**

Initializes a new instance of the `RPCClassAttribute` class

**Namespace:** W.Net.RPC  
**Assembly:** Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

### Syntax

```csharp
public RPCClassAttribute()
```

### See Also

- Reference
  - `RPCClassAttribute Class`
  - `W.Net.RPC Namespace`

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
RPCClassAttribute Properties

The RPCClassAttribute type exposes the following members.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Typeld</td>
<td>When implemented in a derived class, gets a unique identifier for this Attribute. (Inherited from Attribute.)</td>
</tr>
</tbody>
</table>

See Also

Reference
RPCClassAttribute Class
W.Net.RPC Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W^n$
RPCClassAttribute Methods

The RPCClassAttribute type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong></td>
<td>Returns a value that indicates whether this instance is equal to a specified object. (Inherited from Attribute.)</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Returns the hash code for this instance. (Inherited from Attribute.)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>IsDefaultAttribute</strong></td>
<td>When overridden in a derived class, indicates whether the value of this instance is the default value for the derived class. (Inherited from Attribute.)</td>
</tr>
</tbody>
</table>
**Match**
When overridden in a derived class, returns a value that indicates whether this instance equals a specified object.
(Inherited from **Attribute**.)

**MemberwiseClone**
Creates a shallow copy of the current **Object**.
(Inherited from **Object**.)

**ToString**
Returns a string that represents the current object.
(Inherited from **Object**.)

---

**Extension Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![AsTType]</td>
<td>Use Generic syntax for the as operator. (Defined by <strong>AsExtensions</strong>.)</td>
</tr>
<tr>
<td>![InitializeProperties]</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnerProperty members. (Defined by <strong>PropertyHostExtensions</strong>.)</td>
</tr>
<tr>
<td>![InLock(Action)]</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>InLockTType(Func&lt;TType&gt;)</code></td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>InLockAsync(Action)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>InLockAsyncTType(Func&lt;TType&gt;)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>IsDirty</code></td>
<td>Scans the <code>IsDirty</code> value of each field and property of type <code>IProperty</code> (Defined by <code>PropertyHostExtensions</code>.)</td>
</tr>
<tr>
<td><code>Lock</code></td>
<td>Performs a Monitor lock (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>MarkAsClean</code></td>
<td>Scans each field and property of type <code>IProperty</code> and sets its <code>IsDirty</code> flag to false</td>
</tr>
</tbody>
</table>
Unlock

Performs a Monitor unlock
(Defined by MonitorExtensions.)

See Also

Reference
RPCClassAttribute Class
W.Net.RPC Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
RPCMethodAttribute Class

Add this attribute to a static method and Tungsten.Net.RPC.Server can automatically add the method to its dictionary of callable methods.

Inheritance Hierarchy

```
SystemObject  SystemAttribute  W.Net.RPC.RPCMethodAttribute
```

Namespace: W.Net.RPC  

Syntax

```c#
public class RPCMethodAttribute : Attribute
```

The **RPCMethodAttribute** type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RPCMethodAttribute</strong></td>
<td>Initializes a new instance of the <strong>RPCMethodAttribute</strong> class</td>
</tr>
</tbody>
</table>

Properties
Typeld  When implemented in a derived class, gets a unique identifier for this Attribute. (Inherited from Attribute.)

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equals</strong></td>
<td>Returns a value that indicates whether this instance is equal to a specified object. (Inherited from Attribute.)</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Returns the hash code for this instance. (Inherited from Attribute.)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>IsDefaultAttribute</strong></td>
<td>When overridden in a derived class, indicates whether the value of this instance is the default value for the derived class. (Inherited from Attribute.)</td>
</tr>
<tr>
<td><strong>Match</strong></td>
<td>When overridden in a derived</td>
</tr>
</tbody>
</table>
class, returns a value that indicates whether this instance equals a specified object. (Inherited from Attribute.)

MemberwiseClone

Creates a shallow copy of the current Object. (Inherited from Object.)

ToString

Returns a string that represents the current object. (Inherited from Object.)

Top

Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot; This method should be called in the constructor of any class which has IOwnedProperty members (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Monitor lock (Defined by MonitorExtensions.)</td>
<td></td>
</tr>
<tr>
<td>InLockTType(FuncTType)</td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTType(FuncTType)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by</td>
</tr>
</tbody>
</table>
Unlock
Performs a Monitor unlock
(Defined by MonitorExtensions.)

Remarks

Note: Due to the way Newtonsoft.Json deserializes integers, do NOT use int (Int32) in your RPC methods as parameters or return types; use longs instead.

See Also

Reference
W.Net.RPC Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
RPCMethodAttribute Constructor

Initializes a new instance of the RPCMethodAttribute class

**Namespace:** W.Net.RPC  
**Assembly:** Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

### Syntax

C#

```csharp
public RPCMethodAttribute()
```

### See Also

Reference  
RPCMethodAttribute Class  
W.Net.RPC Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
The **RPCMethodAttribute** type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TypeId</td>
<td>When implemented in a derived class, gets a unique identifier for this <strong>Attribute</strong>. (Inherited from <strong>Attribute</strong>.)</td>
</tr>
</tbody>
</table>

### See Also

- Reference
  - **RPCMethodAttribute Class**
  - **W.Net.RPC Namespace**

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
</tr>
</tbody>
</table>
RPCMethodAttribute Methods

The **RPCMethodAttribute** type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Returns a value that indicates whether this instance is equal to a specified object. (Inherited from <strong>Attribute</strong>.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Returns the hash code for this instance. (Inherited from <strong>Attribute</strong>.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <strong>Type</strong> of the current instance. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>IsDefaultAttribute</td>
<td>When overridden in a derived class, indicates whether the value of this instance is the default value for the derived class. (Inherited from <strong>Attribute</strong>.)</td>
</tr>
</tbody>
</table>
Match

When overridden in a derived class, returns a value that indicates whether this instance equals a specified object. (Inherited from Attribute.)

MemberwiseClone

Creates a shallow copy of the current Object. (Inherited from Object.)

ToString

Returns a string that represents the current object. (Inherited from Object.)

---

Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded.</td>
</tr>
</tbody>
</table>

---
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLockTTyType(FuncTTyType)</td>
<td>Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTTyType(FuncTTyType)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets it’s IsDirty flag to false</td>
</tr>
</tbody>
</table>
Unlock

Performs a Monitor unlock
(Defined by MonitorExtensions.)

See Also

Reference
RPCMethodAttribute Class
W.Net.RPC Namespace

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>$W$</td>
</tr>
</tbody>
</table>
RPCResponse Class

Encapsulates information related to making the RPC call and the return value.

Inheritance Hierarchy

- `SystemObject`
- `W.Net.RPCRPCResponse`
- `W.Net.RPCRPCResponseTResponseBaseType`

**Namespace:** `W.Net.RPC`
**Assembly:** Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

Syntax

```csharp
public class RPCResponse
```

The `RPCResponse` type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPCResponse</td>
<td>Initializes a new instance of the <code>RPCResponse</code> class</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exception</td>
<td>May contain exception information if</td>
</tr>
</tbody>
</table>
there was an exception making or as a result of the call

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>The name of the method called</td>
</tr>
<tr>
<td>Response</td>
<td>The return value from the method</td>
</tr>
<tr>
<td>Success</td>
<td>True if the call was successful, otherwise False</td>
</tr>
</tbody>
</table>

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object.</td>
</tr>
</tbody>
</table>
(Inherited from **Object**.)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ToString</strong></td>
<td>Useful for debugging or displaying information quickly. (Overrides <strong>ObjectToString</strong>.)</td>
</tr>
</tbody>
</table>

## Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AsTType</strong></td>
<td>Use Generic syntax for the as operator. (Defined by <strong>AsExtensions</strong>.)</td>
</tr>
<tr>
<td><strong>InitializeProperties</strong></td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by <strong>PropertyHostExtensions</strong>.)</td>
</tr>
<tr>
<td><strong>InLock(Action)</strong></td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by <strong>MonitorExtensions</strong>.)</td>
</tr>
<tr>
<td><strong>InLockTType(FuncTType)</strong></td>
<td>Overloaded. Performs the function in a Monitor lock. (Defined by <strong>MonitorExtensions</strong>.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTTType(FuncTTType)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>

Top
See Also

Reference

W.Net.RPC Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
RPCResponse Constructor

Initializes a new instance of the RPCResponse class

**Namespace:** W.Net.RPC  
**Assembly:** Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

- **Syntax**

```c#
public RPCResponse()
```

- **See Also**

  **Reference**
  
  - RPCResponse Class
  - W.Net.RPC Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$^\text{W}$
RPCResponse Properties

The **RPCResponse** type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exception</td>
<td>May contain exception information if there was an exception making or as a result of the call</td>
</tr>
<tr>
<td>Method</td>
<td>The name of the method called</td>
</tr>
<tr>
<td>Response</td>
<td>The return value from the method</td>
</tr>
<tr>
<td>Success</td>
<td>True if the call was successful, otherwise False</td>
</tr>
</tbody>
</table>

**See Also**

Reference
- **RPCResponse Class**
- **W.Net.RPC Namespace**

Copyright © 2018 Jordan Duerksen
Tungsten

W
RPCResponseException

Property

May contain exception information if there was an exception making or as a result of the call

**Namespace:**  W.Net.RPC  
**Assembly:**  Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

### Syntax

```c#
public string Exception { get; set; }
```

Property Value

Type:  **String**

### See Also

**Reference**
- RPCResponse Class
- W.Net.RPC Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
RPCResponseMethod Property

The name of the method called

**Namespace:** W.Net.RPC  
**Assembly:** Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

### Syntax

```csharp
public string Method { get; set; }
```

### Property Value

Type: **String**

### See Also

Reference  
RPCResponse Class  
W.Net.RPC Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
RPCResponse

Property

The return value from the method

**Namespace:** W.Net.RPC  
**Assembly:** Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

⚠️ Syntax

```c#
public Object Response { get; set; }
```

Property Value

Type: **Object**

⚠️ See Also

Reference

RPCResponse Class  
W.Net.RPC Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
RPCResponseSuccess Property

True if the call was successful, otherwise False

**Namespace:**  W.Net.RPC  
**Assembly:**  Tungsten.Net (in Tungsten.Net.dll)  Version: 2.0.2

### Syntax

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

### Property Value

Type: Boolean

### Remarks

Note that this is different than the return value from the method, which can be of any value or type

### See Also

Reference
- RPCResponse Class
- W.Net.RPC Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

W
# RPCResponse Methods

The `RPCResponse` type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <code>Type</code> of the current instance. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current <code>Object</code>. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>ToString</td>
<td>Useful for debugging or displaying information quickly. (Overrides <code>Object.ToString</code>.)</td>
</tr>
</tbody>
</table>
## Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTType(FuncTType)</td>
<td>Overloaded. Performs the function in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock.</td>
</tr>
<tr>
<td>Method/Property</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>InLockAsyncTType(Func&lt;TType&gt;)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock. (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>

**See Also**

Reference
- RPCResponse Class
W.Net.RPC Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
RPCResponseToString Method

Useful for debugging or displaying information quickly.

**Namespace:** W.Net.RPC  
**Assembly:** Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

**Syntax**

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

**Return Value**

Type: *String*

Returns a string representation of class members and their values

**See Also**

Reference

RPCResponse Class  
W.Net.RPC Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
**RPCResponse**<br><br>**TRResponseType**

Class

Encapsulates information related to making the RPC call and the return value

**Inheritance Hierarchy**

```
System
    Object
    W.Net.RPC
    W.Net.RPCRPCEntity
    W.Net.RPCRPCTResponseType
```

**Namespace:** W.Net.RPC  
**Assembly:** Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

**Syntax**

```csharp
public class RPCResponse<TResponseType> : RPCResponse
```

**Type Parameters**

**TResponseType**  
The Type expected as a return value from the method call

The `RPCResponse<TResponseType>` type exposes the following members.

**Constructors**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPCResponse&lt;TResponseType&gt;</td>
<td>Initializes a new instance of <code>RPCResponse&lt;TResponseType&gt;</code></td>
</tr>
</tbody>
</table>


## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exception</td>
<td>May contain exception information if there was an exception making or as a result of the call (Inherited from RPCResponse.)</td>
</tr>
<tr>
<td>Method</td>
<td>The name of the method called (Inherited from RPCResponse.)</td>
</tr>
<tr>
<td>Response</td>
<td>The return value from the method</td>
</tr>
<tr>
<td>Success</td>
<td>True if the call was successful, otherwise False (Inherited from RPCResponse.)</td>
</tr>
</tbody>
</table>

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object. (Inherited from Object.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection.</td>
</tr>
</tbody>
</table>
GetHashCode

Serves as the default hash function.
(Inherited from Object.)

GetType

Gets the Type of the current instance.
(Inherited from Object.)

MemberwiseClone

Creates a shallow copy of the current Object.
(Inherited from Object.)

ToString

Useful for debugging or displaying information quickly.
(Inherited from RPCResponse.)

---

Top

**Extension Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot; This method should be called in the constructor of any class which has IOwnedProperty members</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLock&lt;TType&gt;(Func&lt;TType&gt;)</td>
<td>Overloaded. Performs the function in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync&lt;TType&gt;(Func&lt;TType&gt;)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>

**See Also**

- Reference
  - W.Net.RPC Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$^\text{W}$
RPCResponseTResponseType Constructor

Initializes a new instance of the RPCResponseTResponseType class

Namespace: W.Net.RPC

Syntax

```c#
public RPCResponse()
```

See Also

Reference
RPCResponseTResponseType Class
W.Net.RPC Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
The `RPCResponseTResponseType` generic type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exception</td>
<td>May contain exception information if there was an exception making or as a result of the call (Inherited from <code>RPCResponse</code>.)</td>
</tr>
<tr>
<td>Method</td>
<td>The name of the method called (Inherited from <code>RPCResponse</code>.)</td>
</tr>
<tr>
<td>Response</td>
<td>The return value from the method</td>
</tr>
<tr>
<td>Success</td>
<td>True if the call was successful, otherwise False (Inherited from <code>RPCResponse</code>.)</td>
</tr>
</tbody>
</table>

## See Also

- Reference
- `RPCResponseTResponseType` Class
- `W.Net.RPC Namespace`
| Tungsten | W |
RPCResponse

The return value from the method

Namespace: W.Net.RPC

**Syntax**

```csharp
public TResponseType Response { get; set; }
```

Property Value
Type: TResponseType

**See Also**

Reference
RPCResponseTResponseType Class
W.Net.RPC Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
RPCResponse<TResponseType> Methods

The `RPCResponse<TResponseType>` generic type exposes the following members.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Equals</code></td>
<td>Determines whether the specified object is equal to the current object. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><code>Finalize</code></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><code>GetHashCode</code></td>
<td>Serves as the default hash function. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><code>GetType</code></td>
<td>Gets the <code>Type</code> of the current instance. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><code>MemberwiseClone</code></td>
<td>Creates a shallow copy of the current <code>Object</code>. (Inherited from <code>Object</code>.)</td>
</tr>
</tbody>
</table>
**Extension Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AsTType</code></td>
<td>Use Generic syntax for the as operator. (Defined by <code>AsExtensions</code>.)</td>
</tr>
<tr>
<td><code>InitializeProperties</code></td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot; This method should be called in the constructor of any class which has <code>IOwnedProperty</code> members. (Defined by <code>PropertyHostExtensions</code>.)</td>
</tr>
<tr>
<td><code>InLock(Action)</code></td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>InLockTType(FuncTType)</code></td>
<td>Overloaded. Performs the function in a Monitor lock. (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>InLockAsync(Action)</strong></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>InLockAsyncTTType(FuncTTType)</strong></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>IsDirty</strong></td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td><strong>Lock</strong></td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>MarkAsClean</strong></td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td><strong>Unlock</strong></td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>
See Also

Reference
RPCResponseTResponseType Class
W.Net.RPC Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

\( W \)
Server Class

Allows remote instances of Tungsten.Net.RPC.Client to call local methods.

Inheritance Hierarchy

- System
  - Object
  - W.Net.RPC.Server

Namespace: W.Net.RPC

Syntax

```csharp
public class Server : IDisposable
```

The `Server` type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="constructor" /> Server</td>
<td>Initializes the Tungsten.Net.RPC.Server and loads the RPC methods</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="property" /> API</td>
<td>Exposes the dictionary of methods. Custom, non-attributed methods may be</td>
</tr>
</tbody>
</table>
Top

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispose</td>
<td>Disposes the Tungsten.Net.RPC.Server and releases resources</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Calls Dispose and deconstructs the Tungsten.Net.RPC.Server</td>
</tr>
<tr>
<td></td>
<td>(Overrides ObjectFinalize.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>Start</td>
<td>Starts listening for client connections on the specified network interface</td>
</tr>
<tr>
<td></td>
<td>and port</td>
</tr>
</tbody>
</table>
## Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AsTType</code></td>
<td>Use Generic syntax for the as operator. (Defined by <code>AsExtensions</code>.)</td>
</tr>
<tr>
<td><code>InitializeProperties</code></td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot; This method should be called in the constructor of any class which has IOwnedProperty members (Defined by <code>PropertyHostExtensions</code>.)</td>
</tr>
<tr>
<td><code>InLock(Action)</code></td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>InLockTType(FuncTType)</code></td>
<td>Overloaded. Performs the function in a Monitor lock</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| **InLockAsync**<br>**InLockAsyncTTtype**<br>**IsDirty**<br>**Lock**<br>**MarkAsClean**<br>**Unlock** | Overloaded. Asynchronously performs the action in a Monitor lock. (Defined by MonitorExtensions.)
| | Scans the IsDirty value of each field and property of type IProperty. (Defined by PropertyHostExtensions.)
| | Performs a Monitor lock. (Defined by MonitorExtensions.)
| | Scans each field and property of type IProperty and sets its IsDirty flag to false. (Defined by PropertyHostExtensions.)
| | Performs a Monitor unlock. (Defined by MonitorExtensions.) |
Remarks

Note: Due to the way Newtonsoft.Json deserializes integers, do NOT use int (Int32) in your api's as parameters or return types; use longs instead.

See Also

Reference
W.Net.RPC Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
Server Constructor

Initializes the Tungsten.Net.RPC.Server and loads the RPC methods

Namespace: W.Net.RPC

Syntax

```c#
public Server(
    int encryptionKeySize
)
```

Parameters

- `encryptionKeySize`
  - Type: `System.Int32`
  - The encryption key size (typically 2048 or 4096; 384 to 16384 in increments of 8)

Remarks

The client must be declared with the same value.

See Also

Reference
- Server Class
- W.Net.RPC Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
Server Properties

The Server type exposes the following members.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Exposes the dictionary of methods. Custom, non-attributed methods may be added to this dictionary.</td>
</tr>
</tbody>
</table>

See Also

Reference
Server Class
W.Net.RPC Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
ServerAPI Property

Exposes the dictionary of methods. Custom, non-attributed methods may be added to this dictionary.

**Namespace:** W.Net.RPC  
**Assembly:** Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

**Syntax**

```
public MethodDictionary API { get; }
```

Property Value  
Type: MethodDictionary

**See Also**

Reference  
Server Class  
W.Net.RPC Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
Server Methods

The Server type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispose</td>
<td>Disposes the Tungsten.Net.RPC.Server and releases resources</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td>Finalize</td>
<td>Calls Dispose and deconstructs the Tungsten.Net.RPC.Server</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object.</td>
</tr>
<tr>
<td>Start</td>
<td>Starts listening for client</td>
</tr>
</tbody>
</table>
connections on the specified network interface and port

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stop</td>
<td>Stops listening for client connections</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object. (Inherited from Object.)</td>
</tr>
</tbody>
</table>

**Extension Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>InLock(Func&lt;T&gt;` TType)</strong></td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>InLockAsync(Action)</strong></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>InLockAsync(Func&lt;T&gt;` TType)</strong></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>IsDirty</strong></td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td><strong>Lock</strong></td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>MarkAsClean</strong></td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td><strong>Unlock</strong></td>
<td>Performs a Monitor unlock</td>
</tr>
</tbody>
</table>
See Also

Reference
Server Class
W.Net.RPC Namespace
Tungsten

$W$
ServerDispose Method

Disposes the Tungsten.Net.RPC.Server and releases resources

**Namespace:** W.Net.RPC  
**Assembly:** Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

### Syntax

```csharp
public void Dispose()
```

Implements

IDisposable

### See Also

Reference

**Server Class**  
**W.Net.RPC Namespace**

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
ServerFinalize Method

Calls Dispose and deconstructs the Tungsten.Net.RPC.Server

**Namespace:**  W.Net.RPC  
**Assembly:**  Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

### Syntax

```csharp
protected override void Finalize()
```

Implements

ObjectFinalize

### See Also

Reference

Server Class  
W.Net.RPC Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
ServerStart Method

Starts listening for client connections on the specified network interface and port

**Namespace:**  W.Net.RPC  
**Assembly:**  Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

### Syntax

```csharp
public void Start(
    IPEndPoint ep,
    Assembly rpcAssembly,
    bool scanReferences = false
)
```

### Parameters

- **ep**
  - Type: `System.Net.IPEndPoint`
  - The IPEndPoint on which to bind and listen for clients

- **rpcAssembly**
  - Type: `System.Reflection.Assembly`
  - The root assembly in which to scan for RPC methods

- **scanReferences (Optional)**
  - Type: `System.Boolean`
  - If True, referenced assemblies will also be scanned for RPC methods

### See Also

- Reference
- Server Class
Tungsten

$W$
ServerStop Method

Stops listening for client connections

**Namespace:** W.Net.RPC  
**Assembly:** Tungsten.Net (in Tungsten.Net.dll) Version: 2.0.2

### Syntax

```
public void Stop()
```

### See Also

Reference
- Server Class
- W.Net.RPC Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
## W.Threading Namespace

[Missing <summary> documentation for "N:W.Threading"]

### Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thread</td>
<td>Functionality related to multi-threading</td>
</tr>
<tr>
<td>ThreadMethod</td>
<td>Task-based multi-threading</td>
</tr>
</tbody>
</table>

### Delegates

<table>
<thead>
<tr>
<th>Delegate</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ThreadMethodThreadMethodDelegate</td>
<td>Delegate type used by ThreadMethod</td>
</tr>
</tbody>
</table>

### Enumerations

<table>
<thead>
<tr>
<th>Enumeration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPUPProfEnum</td>
<td>The preferred level of CPU usage</td>
</tr>
</tbody>
</table>

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
CPUProfileEnum Enumeration

The preferred level of CPU usage

Namespace:  W.Threading
Assembly:  Tungsten.Threading (in Tungsten.Threading.dll) Version: 2.0.1

Syntax

```c#
public enum CPUProfileEnum
```

Members

<table>
<thead>
<tr>
<th>Member name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SpinWait0</td>
<td>0</td>
<td>High CPU usage, but fastest execution. May be faster on single-core/cpu machines. May be slower on multi-core/cpu machines.</td>
</tr>
<tr>
<td>Sleep</td>
<td>1</td>
<td>Medium CPU usage. Uses Thread.Sleep or Task.Delay to block the current thread.</td>
</tr>
<tr>
<td>SpinWait1</td>
<td>2</td>
<td>Low CPU usage. Should be faster on multi-core/cpu machines as the load will be divided among cores/cpus. Slowest on single-core/cpu machines.</td>
</tr>
<tr>
<td>---------</td>
<td>---</td>
<td>--------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SpinUntil</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

**See Also**

Reference

[W.Threading Namespace](#)
Tungsten

W
Thread Class

Functionality related to multi-threading

ʼ Inheritance Hierarchy

- System
  - object
  - W.Threading

Namespace: W.Threading
Assembly: Tungsten.Threading (in Tungsten.Threading.dll) Version: 2.0.1

ʼ Syntax

```c#
public class Thread
```

The `Thread` type exposes the following members.

ʼ Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thread</td>
<td>Initializes a new instance of the <code>Thread</code> class</td>
</tr>
</tbody>
</table>

ʼ Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as the default hash function. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <strong>Type</strong> of the current instance. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current <strong>Object</strong>. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>Sleep(Int32)</strong></td>
<td>Blocks the calling thread for the specified time</td>
</tr>
<tr>
<td><strong>Sleep(CPUProfileEnum)</strong></td>
<td>Attempts to free the CPU for other processes, based on the desired level. Consequences will vary depending on your hardware architecture. The more processors/cores you have, the better performance you will have.</td>
</tr>
</tbody>
</table>
have by selecting LowCPU. Likewise, on a single-core processor, you may wish to select HighCPU.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sleep(Int32, Boolean)</strong></td>
<td>Blocks the calling thread for the specified time</td>
</tr>
<tr>
<td><strong>Sleep(CPUProfileEnum, Int32)</strong></td>
<td>Attempts to free the CPU for other processes, based on the desired level. Consequences will vary depending on your hardware architecture. The more processors/cores you have, the better performance you will have by selecting SpinWait1. Likewise, on a single-core processor, you may wish to select SpinWait0.</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Returns a string that represents the current object. (Inherited from Object.)</td>
</tr>
</tbody>
</table>

Top

**Extension Methods**
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot; This method should be called in the constructor of any class which has IOwnerProperty members (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTTType(FuncTTType)</td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTTType(FuncTTType)</td>
<td>Overloaded.</td>
</tr>
</tbody>
</table>
Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IsDirty</strong></td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td><strong>Lock</strong></td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>MarkAsClean</strong></td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td><strong>Unlock</strong></td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>

**See Also**

Reference

W.Threading Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W^+$
Thread Constructor

Initializes a new instance of the Thread class

**Namespace:** W.Threading
**Assembly:** Tungsten.Threading (in Tungsten.Threading.dll) Version: 2.0.1

### Syntax

```c#
public Thread()
```

### See Also

- Reference
- Thread Class
- W.Threading Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
Thread Methods

The **Thread** type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup</td>
</tr>
<tr>
<td></td>
<td>operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <strong>Type</strong> of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current <strong>Object</strong>.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>Sleep(Int32)</td>
<td>Blocks the calling thread for the specified</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Sleep(CPUProfileEnum)</td>
<td>Attempts to free the CPU for other processes, based on the desired level. Consequences will vary depending on your hardware architecture. The more processors/cores you have, the better performance you will have by selecting LowCPU. Likewise, on a single-core processor, you may wish to select HighCPU.</td>
</tr>
<tr>
<td>Sleep(Int32, Boolean)</td>
<td>Blocks the calling thread for the specified time</td>
</tr>
<tr>
<td>Sleep(CPUProfileEnum, Int32)</td>
<td>Attempts to free the CPU for other processes, based on the desired level. Consequences will vary depending on your hardware architecture. The more processors/cores you have, the better performance you will have by selecting SpinWait1. Likewise, on a single-core processor,</td>
</tr>
</tbody>
</table>
you may wish to select SpinWait0.

### `ToString`
Returns a string that represents the current object.
(Inherited from `Object`.)

---

**Top**

### Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AsTType</code></td>
<td>Use Generic syntax for the as operator. (Defined by <code>AsExtensions</code>.)</td>
</tr>
<tr>
<td><code>InitializeProperties</code></td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by <code>PropertyHostExtensions</code>.)</td>
</tr>
<tr>
<td><code>InLock(Action)</code></td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>InLockTType(FuncTType)</code></td>
<td>Overloaded. Performs the function in</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTType(Func&lt;TType&gt;)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>
See Also

Reference
Thread Class
W.Threading Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
# ThreadSleep Method

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Sleep(Int32)</code></td>
<td>Blocks the calling thread for the specified time</td>
</tr>
<tr>
<td><code>Sleep(CPUProfileEnum)</code></td>
<td>Attempts to free the CPU for other processes, based on the desired level.</td>
</tr>
<tr>
<td></td>
<td>Consequences will vary depending on your hardware architecture.</td>
</tr>
<tr>
<td></td>
<td>The more processors/cores you have, the better performance you will have by</td>
</tr>
<tr>
<td></td>
<td>selecting LowCPU. Likewise, on a single-core processor, you may wish to</td>
</tr>
<tr>
<td></td>
<td>select HighCPU.</td>
</tr>
<tr>
<td><code>Sleep(Int32, Boolean)</code></td>
<td>Blocks the calling thread for the specified time</td>
</tr>
<tr>
<td><code>Sleep(CPUProfileEnum, Int32)</code></td>
<td>Attempts to free the CPU for other processes, based on the desired level.</td>
</tr>
</tbody>
</table>
Consequences will vary depending on your hardware architecture. The more processors/cores you have, the better performance you will have by selecting SpinWait1. Likewise, on a single-core processor, you may wish to select SpinWait0.

See Also

Reference
Thread Class
W.Threading Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
ThreadSleep Method (Int32)

Blocks the calling thread for the specified time

**Namespace:** W.Threading  
**Assembly:** Tungsten.Threading (in Tungsten.Threading.dll) Version: 2.0.1

▶ Syntax

```csharp
public static void Sleep(
    int msDelay
)
```

**Parameters**

*msDelay*  
Type: SystemInt32  
The number of milliseconds to block the thread

▶ See Also

**Reference**  
Thread Class  
Sleep Overload  
W.Threading Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
ThreadSleep Method (CPUProfileEnum)

Attempts to free the CPU for other processes, based on the desired level. Consequences will vary depending on your hardware architecture. The more processors/cores you have, the better performance you will have by selecting LowCPU. Likewise, on a single-core processor, you may wish to select HighCPU.

**Namespace:** W.Threading  
**Assembly:** Tungsten.Threading (in Tungsten.Threading.dll) Version: 2.0.1

### Syntax

```csharp
public static void Sleep(
    CPUProfileEnum level
)
```

### Parameters

- **level**
  - Type: `W.ThreadingCPUProfileEnum`  
  - The desired level of CPU usage

### Remarks

Note results may vary. LowCPU will spread the load onto multiple cores and can actually yield faster results depending on your hardware architecture. This may not always be the case.

### See Also
Reference

Thread Class
Sleep Overload
W.Threading Namespace

Copyright © 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>$W$</td>
</tr>
</tbody>
</table>
ThreadSleep Method (Int32, Boolean)

Blocks the calling thread for the specified time

**Namespace:**  W.Threading  
**Assembly:**  Tungsten.Threading (in Tungsten.Threading.dll)  
**Version:**  2.0.1

### Syntax

```
public static void Sleep(
    int msDelay,
    bool useSpinWait
)
```

### Parameters

- **msDelay**
  - Type: SystemInt32
  - The number of milliseconds to block the thread

- **useSpinWait**
  - Type: SystemBoolean
  - If True, a SpinWait.SpinUntil will be used instead of a call to Thread.Sleep (or Task.Delay). Note that SpinWait should only be used on multi-core/cpu machines.

### See Also

Reference
- Thread Class
- Sleep Overload
Tungsten

\( W \)
ThreadSleep Method
(CPUProfileEnum, Int32)

Attempts to free the CPU for other processes, based on the desired level. Consequences will vary depending on your hardware architecture. The more processors/cores you have, the better performance you will have by selecting SpinWait1. Likewise, on a single-core processor, you may wish to select SpinWait0.

Namespace: W.Threading
Assembly: Tungsten.Threading (in Tungsten.Threading.dll) Version: 2.0.1

Syntax

```csharp
public static void Sleep(
    CPUProfileEnum level,
    int msTimeout = 1
)
```

Parameters

- **level**
  Type: W.ThreadingCPUProfileEnum
  The desired level of CPU usage

- **msTimeout (Optional)**
  Type: System.Int32
  Optional value for CPUProfileEnum.Sleep and CPUProfileEnum.SpinUntil. Ignored by other profiles.

Remarks
Note results may vary. SpinWait1 will spread the load onto multiple cores and can actually yield faster results depending on your hardware architecture. This may not always be the case.

See Also

Reference
Thread Class
Sleep Overload
W.Threading Namespace
Tungsten

W
ThreadMethod Class

Task-based multi-threading

Inheritance Hierarchy

- System
  - Object
- W.Threading

Namespace: W.Threading
Assembly: Tungsten.Threading (in Tungsten.Threading.dll) Version: 2.0.1

Syntax

```csharp
public class ThreadMethod : IDisposable
```

The `ThreadMethod` type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ThreadMethod(Action CancellationToken)</td>
<td>Constructor creates a new <code>ThreadMethod</code></td>
</tr>
<tr>
<td>ThreadMethod(ThreadMethodThreadMethodDelegate)</td>
<td>Constructor creates a new <code>ThreadMethod</code></td>
</tr>
</tbody>
</table>

Properties

Top
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsComplete</td>
<td>True if the thread has completed, otherwise False</td>
</tr>
<tr>
<td>Name</td>
<td>A user-defined name for this object</td>
</tr>
</tbody>
</table>

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cancel</td>
<td>Signals the thread method via the CancellationToken to stop running and waits for it to complete</td>
</tr>
<tr>
<td>Create(ActionCancellationToken)</td>
<td>Constructs a new ThreadMethod</td>
</tr>
<tr>
<td>Create(ThreadMethodThreadMethodDelegate)</td>
<td>Constructs a new ThreadMethod</td>
</tr>
<tr>
<td>Dispose</td>
<td>Disposes the ThreadMethod and releases resources</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object (Inherited from Object.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as the default hash function. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <code>Type</code> of the current instance. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current <code>Object</code>. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>Start</strong></td>
<td>Starts the thread <code>Start(Object)</code></td>
</tr>
<tr>
<td><strong>Start(Object)</strong></td>
<td>Starts the thread <code>Start(Object)</code></td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Returns a string that represents the current object.</td>
</tr>
</tbody>
</table>
Wait

Waits for a specified number of milliseconds for the thread to complete.

---

Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>As&lt;TType&gt;</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>InLock&lt;Action&gt;</td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLock&lt;TType(Func&lt;TType)&gt;</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>Method/Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>InLockAsync(Action)</code></td>
<td>Performs the function in a Monitor lock (Defined by <code>MonitorExtensions</code>). Overloaded. Asynchronously performs the action in a Monitor lock (Defined by <code>MonitorExtensions</code>).</td>
</tr>
<tr>
<td><code>InLockAsync&lt;TType(Func&lt;TType)&gt;</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by <code>MonitorExtensions</code>).</td>
</tr>
<tr>
<td><code>IsDirty</code></td>
<td>Scans the <code>IsDirty</code> value of each field and property of type <code>IProperty</code> (Defined by <code>PropertyHostExtensions</code>).</td>
</tr>
<tr>
<td><code>Lock</code></td>
<td>Performs a Monitor lock (Defined by <code>MonitorExtensions</code>).</td>
</tr>
<tr>
<td><code>MarkAsClean</code></td>
<td>Scans each field and property of type <code>IProperty</code> and sets its <code>IsDirty</code> flag to false (Defined by <code>PropertyHostExtensions</code>).</td>
</tr>
<tr>
<td><code>Unlock</code></td>
<td>Performs a Monitor unlock (Defined by <code>MonitorExtensions</code>).</td>
</tr>
</tbody>
</table>
See Also

Reference

W.Threading Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

W
## ThreadMethod Constructor

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ThreadMethod&lt;ActionCancellationToken&gt;()</code></td>
<td>Constructs a new <code>ThreadMethod</code></td>
</tr>
<tr>
<td><code>ThreadMethod(ThreadMethodThreadMethodDelegate)</code></td>
<td>Constructs a new <code>ThreadMethod</code></td>
</tr>
</tbody>
</table>

### See Also

- Reference
  - `ThreadMethod Class`
  - `W.Threading Namespace`

Copyright @ 2018 Jordan Duerksen
ThreadMethod Constructor
(PlaceholderToken)

Constructs a new ThreadMethod

**Namespace:** W.Threading

**Assembly:** Tungsten.Threading (in Tungsten.Threading.dll) Version: 2.0.1

### Syntax

```csharp
public ThreadMethod(
    Action<CancellationToken> threadProc
)
```

### Parameters

*threadProc*

- **Type:** `System.Action<CancellationToken>`
- The thread proc

### See Also

Reference
- ThreadMethod Class
- ThreadMethod Overload
- W.Threading Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$^n W$
ThreadMethod Constructor
(ThreadMethodThreadMethodDelegate)

Constructs a new ThreadMethod

**Namespace**: W.Threading
**Assembly**: Tungsten.Threading (in Tungsten.Threading.dll) Version: 2.0.1

### Syntax

```csharp
public ThreadMethod(
    ThreadMethodThreadMethodDelegate threadProc
)
```

### Parameters

*threadProc*
Type: W.ThreadingThreadMethodThreadMethodDelegate
The thread proc

### See Also

**Reference**
- ThreadMethod Class
- ThreadMethod Overload
- W.Threading Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
ThreadMethod Properties

The ThreadMethod type exposes the following members.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsComplete</td>
<td>True if the thread has completed, otherwise False</td>
</tr>
<tr>
<td>Name</td>
<td>A user-defined name for this object</td>
</tr>
</tbody>
</table>

See Also

Reference

ThreadMethod Class
W.Threading Namespace

Copyright @ 2018 Jordan Duerksen
ThreadMethod.IsComplete Property

True if the thread has completed, otherwise False

**Namespace:** W.Threading  
**Assembly:** Tungsten.Threading (in Tungsten.Threading.dll) Version: 2.0.1

### Syntax

```csharp
public bool IsComplete { get; }
```

### Property Value

Type: **Boolean**

### See Also

Reference
- ThreadMethod Class
- W.Threading Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
ThreadMethod.Name Property

A user-defined name for this object

**Namespace:** W.Threading  
**Assembly:** Tungsten.Threading (in Tungsten.Threading.dll)  
**Version:** 2.0.1

▶ Syntax

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

Property Value

Type: **String**

▶ See Also

Reference

- ThreadMethod Class
- W.Threading Namespace

Copyright © 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>$^\text{W}$</td>
</tr>
</tbody>
</table>
The `ThreadMethod` type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Method" /> Cancel</td>
<td>Signals the thread method via CancellationToken to stop running and waits for completion</td>
</tr>
<tr>
<td><img src="image" alt="Method" /> Create(ActionResult CancellationToken)</td>
<td>Constructs a new ThreadMethod</td>
</tr>
<tr>
<td><img src="image" alt="Method" /> Create(ThreadMethodThreadMethodDelegate)</td>
<td>Constructs a new ThreadMethod</td>
</tr>
<tr>
<td><img src="image" alt="Method" /> Dispose</td>
<td>Disposes the ThreadMethod and releases resources</td>
</tr>
<tr>
<td><img src="image" alt="Method" /> Equals</td>
<td>Determines whether the specified object is equal to the current object (Inherited from Object.)</td>
</tr>
<tr>
<td><img src="image" alt="Method" /> Finalize</td>
<td>Allows an object to perform finalization before it is reclaimed.</td>
</tr>
</tbody>
</table>
to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)

- **GetHashCode**
  Serves as the default hash function. (Inherited from Object.)

- **GetType**
  Gets the Type of the current instance. (Inherited from Object.)

- **MemberwiseClone**
  Creates a shallow copy of the current Object. (Inherited from Object.)

- **Start**
  Starts the thread.

- **Start(Object)**
  Starts the thread.

- **ToString**
  Returns a string that represents the current object. (Inherited from Object.)
### Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Wait</strong></td>
<td>Waits a specified number of milliseconds for the thread to complete</td>
</tr>
<tr>
<td><strong>AsTType</strong></td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td><strong>InitializeProperties</strong></td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td><strong>InLock(Action)</strong></td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>InLockTType(FuncTType)</strong></td>
<td>Overloaded. Performs the function in a Monitor lock</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>InLockAsync(Action)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><code>InLockAsyncTType(FuncTType)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><code>IsDirty</code></td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td><code>Lock</code></td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><code>MarkAsClean</code></td>
<td>Scans each field and property of type IProperty and sets it's IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td><code>Unlock</code></td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>
See Also

Reference
ThreadMethod Class
W.Threading Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
ThreadMethod Cancel Method

Signals the thread method via the CancellationToken to stop running and waits for it to complete

**Namespace:** W.Threading
**Assembly:** Tungsten.Threading (in Tungsten.Threading.dll) Version: 2.0.1

## Syntax

```csharp
public void Cancel()
```

## See Also

Reference
- ThreadMethod Class
- W.Threading Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
## ThreadMethod

### Create Method

#### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create(ActionCancellationToken)</td>
<td>Constructs new ThreadMethod</td>
</tr>
<tr>
<td>Create(ThreadMethodThreadMethodDelegate)</td>
<td>Constructs new ThreadMethod</td>
</tr>
</tbody>
</table>

### See Also

- Reference
  - ThreadMethod Class
  - W.Threading Namespace

---

Copyright © 2018 Jordan Duerksen
Tungsten

\(W\)
ThreadMethod

Create Method
>ActionCancellationToken

Constructs a new ThreadMethod

**Namespace:** W.Threading

**Assembly:** Tungsten.Threading (in Tungsten.Threading.dll) Version: 2.0.1

### Syntax

**C#**

```csharp
public static ThreadMethod Create(
    Action<CancellationToken> threadProc
)
```

**Parameters**

*threadProc*

Type: System.ActionCancellationToken

The thread proc

**Return Value**

Type: ThreadMethod

[Missing <returns> documentation for

### See Also

Reference

ThreadMethod Class

Create Overload

W.Threading Namespace
Tungsten

$W$
ThreadMethodCreate Method (ThreadMethodThreadMethodThreadMethodDelegate)

Constructs a new ThreadMethod

**Namespace:** W.Threading  
**Assembly:** Tungsten.Threading (in Tungsten.Threading.dll)  
**Version:** 2.0.1

## Syntax

```c#
public static ThreadMethod Create(
    ThreadMethodThreadMethodDelegate threadProc
)
```

### Parameters

**threadProc**

- **Type:** W.ThreadingThreadMethodThreadMethodDelegate
- The thread proc

### Return Value

- **Type:** ThreadMethod

### See Also

- **Reference**
  - ThreadMethod Class
  - Create Overload
  - W.Threading Namespace
Tungsten

$W$
ThreadMethodDispose Method

Disposes the ThreadMethod and releases resources

Namespace: W.Threading
Assembly: Tungsten.Threading (in Tungsten.Threading.dll) Version: 2.0.1

Syntax

```
public void Dispose()
```

Implements

IDisposableDispose

See Also

Reference

ThreadMethod Class
W.Threading Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
ThreadMethodStart Method

▲ Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>Starts the thread</td>
</tr>
<tr>
<td>Start(Object)</td>
<td>Starts the thread</td>
</tr>
</tbody>
</table>

Top

▲ See Also

Reference
ThreadMethod Class
W.Threading Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
ThreadMethodStart Method

Starts the thread

**Namespace:**  W.Threading  
**Assembly:**  Tungsten.Threading (in Tungsten.Threading.dll)  
**Version:**  2.0.1

### Syntax

```csharp
public void Start()
```

### See Also

Reference
- ThreadMethod Class
- Start Overload
- W.Threading Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
ThreadMethodStart Method (Object)

Starts the thread

**Namespace:** W.Threading  
**Assembly:** Tungsten.Threading (in Tungsten.Threading.dll)  
Version: 2.0.1

### Syntax

```
public void Start(  
    params Object[] args  
)
```

**Parameters**

*args*

Type: `SystemObject`  
The arguments to pass into the thread procedure

### See Also

Reference

- ThreadMethod Class
- Start Overload
- W.Threading Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
ThreadMethod

Wait Method

Waits a specified number of milliseconds for the thread to complete

Namespace: W.Threading
Assembly: Tungsten.Threading (in Tungsten.Threading.dll) Version: 2.0.1

Syntax

```csharp
public bool Wait(
    int msTimeout = -1
)
```

Parameters

`msTimeout (Optional)`
Type: System.Int32
The number of milliseconds to wait for the thread to complete. A value of -1 indicates an infinite wait period.

Return Value
Type: Boolean
True if the thread completes within the timeout period, otherwise False

See Also

Reference
ThreadMethod Class
W.Threading Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
ThreadMethodDelegate

Delegate type used by ThreadMethod

**Namespace:**  W.Threading  
**Assembly:**  Tungsten.Threading (in Tungsten.Threading.dll)  
**Version:**  2.0.1

## Syntax

```csharp
public delegate void ThreadMethodDelegate(
    CancellationToken token,
    params Object[] args
)
```

## Parameters

**token**
Type:  `System.Threading.CancellationToken`
A CancellationToken which can be used to signal the threaded method to stop

**args**
Type:  `System.Object`
Zero or more arguments to pass into the thread method

## See Also

Reference  
**W.Threading Namespace**

Copyright © 2018 Jordan Duerksen
Tungsten

W
# W.Threading.Lockers Namespace

[Missing `<summary>` documentation for "N:W.Threading.Lockers"]

## Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disposer</td>
<td>Aids in implementing a clean Dispose method. Supports re-entrancy but only calls the cleanup Action once.</td>
</tr>
<tr>
<td>MonitorExtensions</td>
<td>Extensions to object to simplify locking with Monitor</td>
</tr>
<tr>
<td>MonitorLocker</td>
<td>Uses Monitor to provide resource locking</td>
</tr>
<tr>
<td>MonitorLockerTState</td>
<td>Extends MonitorLocker with an internal state variable</td>
</tr>
<tr>
<td>ReaderWriterLocker</td>
<td>Uses ReaderWriterLockSlim to provide resource locking</td>
</tr>
<tr>
<td>ReaderWriterLockerTState</td>
<td>Extends ReaderWriterLocker with an internal state variable</td>
</tr>
<tr>
<td>Extension</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>ReaderWriterLockSlimExtensions</strong></td>
<td>Extensions to simplify locking with ReaderWriterLockSlim</td>
</tr>
<tr>
<td><strong>SemaphoreSlimExtensions</strong></td>
<td>Extensions to simplify locking with SemaphoreSlim</td>
</tr>
<tr>
<td><strong>SemaphoreSlimLocker</strong></td>
<td>Uses SemaphoreSlim to provide resource locking</td>
</tr>
<tr>
<td><strong>SemaphoreSlimLockerTState</strong></td>
<td>Extends SemaphoreSlimLocker with an internal state variable</td>
</tr>
<tr>
<td><strong>SpinLocker</strong></td>
<td>Uses SpinLock to provide resource locking</td>
</tr>
<tr>
<td><strong>SpinLockerTState</strong></td>
<td>Extends SpinLocker with an internal state variable</td>
</tr>
<tr>
<td><strong>SpinLockExtensions</strong></td>
<td>Extensions to simplify locking with SpinLock</td>
</tr>
<tr>
<td><strong>StateLockerTLocker, TState</strong></td>
<td>Extends a locker (SpinLocker, MonitorLocker, ReaderWriterLocker, SemaphoreSlimLocker) with an internal state value</td>
</tr>
</tbody>
</table>
## Interfaces

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ILocker</td>
<td>The required implementation for a locking object</td>
</tr>
<tr>
<td>ILockerTLocker</td>
<td>The required implementation for a locking object</td>
</tr>
<tr>
<td>IReaderWriterStateLockerTState</td>
<td>Interface definition for a ReaderWriterLocker with a State variable</td>
</tr>
<tr>
<td>IStateLockerTLocker, TState</td>
<td>The required implementation for a stateful locking object</td>
</tr>
</tbody>
</table>

## Delegates

<table>
<thead>
<tr>
<th>Delegate</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StateAssignmentDelegateTState</td>
<td>Delegate which can be used to assign a new value to the internal state</td>
</tr>
</tbody>
</table>

## Enumerations

<table>
<thead>
<tr>
<th>Enumeration</th>
<th>Description</th>
</tr>
</thead>
</table>
LockTypeEnum

Used by ReaderWriterLocker to specify the type of lock to obtain
Tungsten

$W$
Disposer Class

Aids in implementing a clean Dispose method. Supports re-entrancy but only calls the cleanup Action once.

Inheritance Hierarchy

```plaintext
System
Object
W.Threading.Lockers
Disposer
```

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

Syntax

C#

```csharp
public class Disposer
```

The **Disposer** type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disposer</td>
<td>Initializes a new instance of the <strong>Disposer</strong> class</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsDisposed</td>
<td>True if Cleanup has been called and</td>
</tr>
</tbody>
</table>
completed, otherwise False

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsDisposing</td>
<td>True if the Disposer is in the process of disposing, otherwise False</td>
</tr>
<tr>
<td>Dispose(Action)</td>
<td>Calls the action (should contain cleanup code)</td>
</tr>
<tr>
<td>Dispose(Object, Action)</td>
<td>Calls the action (should contain cleanup code)</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object. (Inherited from Object.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the</td>
</tr>
</tbody>
</table>
ToString

Returns a string that represents the current object.
(Inherited from Object.)

---

### Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
</tbody>
</table>
| InitializeProperties      | Scans the fields and properties of "owner" and sets the member's Owner property to "owner" This method should be called in the constructor of any class which has IOwnedProperty members  
(Defined by PropertyHostExtensions.) |
<p>| InLock(Action)             | Overloaded. Performs the action in a Monitor lock (Defined by MonitorExtensions.)                                                              |
| InLockTType(FuncTType)     | Overloaded. Performs the function in a Monitor lock                                                                                           |</p>
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLockAsync(Action) Overloaded.</td>
<td>Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTTType(FuncTTType) Overloaded.</td>
<td>Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>
See Also

Reference

W.Threading.Lockers Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

W
Disposer Constructor

Initializes a new instance of the Disposer class

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll)  
**Version:** 2.0.1

#### Syntax

```
public Disposer()
```

#### See Also

Reference  
Disposer Class  
W.Threading.Lockers Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
## Disposer Properties

The **Disposer** type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>IsDisposed</code></td>
<td>True if Cleanup has been called and completed, otherwise False</td>
</tr>
<tr>
<td><code>IsDisposing</code></td>
<td>True if the Disposer is in the process of disposing, otherwise False</td>
</tr>
</tbody>
</table>

### See Also

Reference
- Disposer Class
- W.Threading.Lockers Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$^W$
DisposerIsDisposed Property

True if Cleanup has been called and completed, otherwise False

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```
public bool IsDisposed { get; }
```

### Property Value

Type: Boolean

### See Also

**Reference**
- Disposer Class
- W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
Disposer.IsDisposing Property

True if the Disposer is in the process of disposing, otherwise False

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

**Syntax**

```csharp
public bool IsDisposing { get; }
```

**Property Value**  
Type: Boolean

**See Also**

Reference  
Disposer Class  
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
## Disposer Methods

The **Disposer** type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Disposer" /> Dispose(Action)</td>
<td>Calls the action (should contain cleanup code)</td>
</tr>
<tr>
<td><img src="image" alt="Disposer" /> Dispose(Object, Action)</td>
<td>Calls the action (should contain cleanup code)</td>
</tr>
<tr>
<td><img src="image" alt="Disposer" /> Equals</td>
<td>Determines whether the specified object is equal to the current object. (Inherited from <strong>Object</strong>)</td>
</tr>
<tr>
<td><img src="image" alt="Disposer" /> Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <strong>Object</strong>)</td>
</tr>
<tr>
<td><img src="image" alt="Disposer" /> GetHashCode</td>
<td>Serves as the default hash function. (Inherited from <strong>Object</strong>)</td>
</tr>
<tr>
<td><img src="image" alt="Disposer" /> GetType</td>
<td>Gets the <strong>Type</strong> of the current instance. (Inherited from <strong>Object</strong>)</td>
</tr>
<tr>
<td><img src="image" alt="Disposer" /> MemberwiseClone</td>
<td>Creates a shallow copy of the</td>
</tr>
</tbody>
</table>
current **Object**.  
(Inherited from **Object**.)

---

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>ToString</strong></td>
</tr>
</tbody>
</table>
| Returns a string that represents the current object.  
(Inherited from **Object**.) |

Top

### Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| **AsTType**         | Use Generic syntax for the as operator.  
(Defined by **AsExtensions**.) |
| **InitializeProperties** | Scans the fields and properties of "owner" and sets the member's Owner property to "owner"  
This method should be called in the constructor of any class which has IOwnedPropertyChanged members  
(Defined by **PropertyHostExtensions**.) |
| **InLock(Action)**  | Overloaded. Performs the action in a Monitor lock  
(Defined by **MonitorExtensions**.) |
| **InLockTType(FuncTType)** | Overloaded. Performs the function in a Monitor lock |
InLockAsync(Action) | Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)
---|---
InLockAsyncTType(FuncTType) | Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)
IsDirty | Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)
Lock | Performs a Monitor lock (Defined by MonitorExtensions.)
MarkAsClean | Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)
Unlock | Performs a Monitor unlock (Defined by MonitorExtensions.)
See Also

Reference
Disposer Class
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>$^78$W</td>
</tr>
</tbody>
</table>
DisposerDispose Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispose(Action)</td>
<td>Calls the action (should contain cleanup code)</td>
</tr>
<tr>
<td>Dispose(Object, Action)</td>
<td>Calls the action (should contain cleanup code)</td>
</tr>
</tbody>
</table>

See Also

Reference
- Disposer Class
- W.Threading.Lockers Namespace
Tungsten

\( W \)
DisposerDispose Method (Action)

Calls the action (should contain cleanup code)

**Namespace:**  W.Threading.Lockers  
**Assembly:**  Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

**Syntax**

```
public void Dispose(
    Action cleanupAction
)
```

**Parameters**

*cleanupAction*
  
  **Type:**  SystemAction  
  The action to call

**See Also**

Reference
  
  Disposer Class
  Dispose Overload
  W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>$W$</td>
</tr>
</tbody>
</table>
Disposer Dispose Method
(Object, Action)

Calls the action (should contain cleanup code)

**Namespace:** W.Threading.Lockers

**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```csharp
public void Dispose(
    Object objToSupressFinalize,
    Action cleanupAction
)
```

### Parameters

- **objToSupressFinalize**
  - Type: SystemObject
  - The object on which to suppress the finalizer call (usually the one currently being disposed)

- **cleanupAction**
  - Type: SystemAction
  - The action to call

### See Also

Reference
- Disposer Class
- Dispose Overload
- W.Threading.Lockers Namespace
Tungsten

$W$
**ILocker Interface**

The required implementation for a locking object

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```csharp
public interface ILocker
```

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLock(Action)</td>
<td>Perform some action in a lock</td>
</tr>
<tr>
<td>InLockTResult(FuncTResult)</td>
<td>Perform some function in a lock</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td></td>
</tr>
<tr>
<td>InLockAsyncTResult(FuncTResult)</td>
<td>Asynchronously perform some function in a lock</td>
</tr>
</tbody>
</table>

### See Also
Tungsten

\( W \)
ILocker Methods

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLock(Action)</td>
<td>Perform some action in a lock</td>
</tr>
<tr>
<td>InLockTResult(FuncTResult)</td>
<td>Perform some function in a lock</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Asynchronously perform some function in a lock</td>
</tr>
<tr>
<td>InLockAsyncTResult(FuncTResult)</td>
<td>Asynchronously perform some function in a lock</td>
</tr>
</tbody>
</table>

See Also

Reference
- ILocker Interface
- W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
## ILocker\n
### InLock Method

#### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLock(Action)</td>
<td>Perform some action in a lock</td>
</tr>
<tr>
<td>InLockTResult(FuncTResult)</td>
<td>Perform some function in a lock</td>
</tr>
</tbody>
</table>

### See Also

- Reference
  - ILocker Interface
  - W.Threading.Lockers Namespace
Tungsten

$W^-$
ILockerInLock Method (Action)

Perform some action in a lock

Namespace: W.Threading.Lockers

Syntax

```csharp
void InLock(
    Action action
)
```

Parameters

`action`
Type: SystemAction
The action to perform

See Also

Reference
ILocker Interface
InLock Overload
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
</tr>
</tbody>
</table>
ILockerInLock\textit{TResult} Method \texttt{(Func\textit{TResult})}

Perform some function in a lock

\textbf{Namespace:}  \texttt{W.Threading.Lockers}  \\
\textbf{Assembly:}  \texttt{Tungsten.Threading.Lockers} (in \texttt{Tungsten.Threading.Lockers.dll})  \texttt{Version: 2.0.1}

\section*{Syntax}

\begin{verbatim}
C#

TResult InLock<TResult>(
    Func<TResult> func
)

\end{verbatim}

\section*{Parameters}

\textit{func}

Type: \texttt{System.Func\textit{TResult}}  \\
The function to perform

\section*{Type Parameters}

\textit{TResult}

The result Type

\section*{Return Value}

Type: \textit{TResult}  \\
The result of the function

\section*{See Also}

Reference
Tungsten

$W$
ILocker

InLockAsync Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLockAsync(Action)</td>
<td>Asynchronously perform some function in a lock</td>
</tr>
</tbody>
</table>

See Also

Reference
ILocker Interface
W.Threading.Lockers Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

W
ILockerInLockAsync Method (Action)


**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll)  
**Version:** 2.0.1

### Syntax

C#  

```csharp
Task InLockAsync(
    Action action
)
```

### Parameters

**action**  
*Type: SystemAction*  
[Missing <param name="action"/> documentation for "M:W.Threading.Lockers.ILocker.InLockAsync(System.Action)"

### Return Value

*Type: Task*  

### See Also

Reference  
ILocker Interface
InLockAsync Overload
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
ILocker\_InLockAsync\_TResult Method (Func\_TResult)

Asynchonously perform some function in a lock

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

## Syntax

```csharp
Task\<TResult\> InLockAsync\<TResult\>(
    Func\<TResult\> func
)
```

### Parameters

**func**  
Type: System\_Func\_TResult  
The function to perform

### Type Parameters

**TResult**  
The result Type

### Return Value

**Type:** Task\_TResult  
The result of the function

---

## See Also

Reference
| ILocker Interface | InLockAsync Overload | W.Threading.Lockers Namespace |

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
ILockerType Locker Interface

The required implementation for a locking object

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```c#
public interface ILocker<TLocker> : ILocker
```

### Type Parameters

**TLocker**
The type of locker to use (SpinLock, Monitor, SemaphoreSlim, ReaderWriterLock)

The `ILocker<TLocker>` type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![ ]</td>
<td>Locker</td>
</tr>
<tr>
<td>![ ]</td>
<td>The object used for locking</td>
</tr>
</tbody>
</table>

**Top**

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![ ]</td>
<td>InLock(Action)</td>
</tr>
<tr>
<td>![ ]</td>
<td>Perform some</td>
</tr>
</tbody>
</table>
### InLockAsyncResult(Func<TResult>)

Asynchronously perform some function in a lock (Inherited from ILocker.)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLockAsyncResult(Func&lt;TResult&gt;)</td>
<td>Asynchronously perform some function in a lock (Inherited from ILocker.)</td>
</tr>
<tr>
<td>InLockAsyncResult(Func&lt;TResult&gt;)</td>
<td>InLockAsyncResult(Func&lt;TResult&gt;)</td>
</tr>
</tbody>
</table>

---

### See Also

Reference

W.Threading.Lockers Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

W
The `ILockerTLocker` generic type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locker</td>
<td>The object used for locking</td>
</tr>
</tbody>
</table>

### See Also

**Reference**
- `ILockerTLocker Interface`
- `W.Threading.Lockers Namespace`

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
ILocker

The object used for locking

**Namespace:** W.Threading.Lockers
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```c#
TLocker Locker { get; }
```

**Property Value**

Type: *TLocker*

### See Also

**Reference**
- ILocker TLocker Interface
- W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
## ILocker TLocker Methods

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLock(Action)</td>
<td>Perform some action in a lock (Inherited from ILocker.)</td>
</tr>
<tr>
<td>InLockTResult(FuncTResult)</td>
<td>Perform some function in a lock (Inherited from ILocker.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>(Inherited from ILocker.)</td>
</tr>
<tr>
<td>InLockAsyncTResult(FuncTResult)</td>
<td>Asynchronously perform some function in a lock (Inherited from ILocker.)</td>
</tr>
</tbody>
</table>

### See Also

Reference
- ILocker TLocker Interface
- W.Threading.Lockers Namespace
Tungsten

$W$
IReaderWriterStateLocker<TState>
Interface

Interface definition for a ReaderWriterLocker with a State variable

**Namespace:** W.Threading.Lockers
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```c#
public interface IReaderWriterStateLocker<TState>
```

Type Parameters

**TState**

The state Type

The `IReaderWriterStateLocker<TState>` type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>GetState</code></td>
<td>Retrieves the internal state from within a ReaderWriterLockSlim</td>
</tr>
<tr>
<td><code>InLock(LockTypeEnum, Action&lt;TState&gt;)</code></td>
<td>Executes an action from within a ReaderWriterLockSlim</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>InLockTValue(LockTypeEnum, FuncTState, TValue)</code></td>
<td>Executes a function from within a <code>ReaderWriterLockSlim</code></td>
</tr>
<tr>
<td><code>InLockAsync(LockTypeEnum, ActionTState)</code></td>
<td>Executes an action from within a <code>ReaderWriterLockSlim</code></td>
</tr>
<tr>
<td><code>InLockAsyncTValue(LockTypeEnum, FuncTState, TValue)</code></td>
<td>Executes a function from within a <code>ReaderWriterLockSlim</code></td>
</tr>
<tr>
<td><code>SetState</code></td>
<td>Sets the internal state from within a <code>ReaderWriterLockSlim</code></td>
</tr>
</tbody>
</table>

**See Also**

Reference

W.Threading.Lockers Namespace
Tungsten

W
The `IReaderWriterStateLocker<TState>` generic type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetState</td>
<td>Retrieves the internal state from within a ReaderWriterLockSlim.</td>
</tr>
<tr>
<td>InLock(LockTypeEnum, Action&lt;TState&gt;)</td>
<td>Executes an action from within a ReaderWriterLockSlim.</td>
</tr>
<tr>
<td>InLockTValue(LockTypeEnum, Func&lt;TState, TValue&gt;)</td>
<td>Executes a function from within a ReaderWriterLockSlim.</td>
</tr>
<tr>
<td>InLockAsync(LockTypeEnum, Action&lt;TState&gt;)</td>
<td>Executes an action from within a ReaderWriterLockSlim.</td>
</tr>
<tr>
<td>InLockAsyncTValue(LockTypeEnum, Func&lt;TState, TValue&gt;)</td>
<td>Executes a function from within a ReaderWriterLockSlim.</td>
</tr>
<tr>
<td>SetState</td>
<td>Sets the internal state from within a ReaderWriterLockSlim.</td>
</tr>
</tbody>
</table>
See Also

Reference
IReaderWriterStateLocker<TState> Interface
W.Threading.Lockers Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

W
IReaderWriterStateLocker TState Get Method

Retrieves the internal state from within a ReaderWriterLockSlim

**Namespace:** W.Threading.Lockers
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```csharp
TState GetState()
```

### Return Value

Type: **TState**
The current state

### See Also

Reference
- IReaderWriterStateLocker TState Interface
- W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
IReaderWriterStateLocker<TState> InLock Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLock(LockTypeEnum, Action&lt;TState&gt;)</td>
<td>Executes an action from within a ReaderWriterLockSlim</td>
</tr>
<tr>
<td>InLockTValue(LockTypeEnum, Func&lt;TState, TValue&gt;)</td>
<td>Executes a function from within a ReaderWriterLockSlim</td>
</tr>
</tbody>
</table>

See Also

Reference
IReaderWriterStateLocker<TState> Interface
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
IReaderWriterStateLocker<TState> method (LockTypeEnum, Action<TState>)

Executes an action from within a ReaderWriterLockSlim

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```csharp
void InLock(
    LockTypeEnum lockType,
    Action<TState> action
)
```

**Parameters**

- **lockType**
  - Type: W.Threading.LockersLockTypeEnum
  - Specifies whether to use a Read or Write lock

- **action**
  - Type: SystemAction<TState>
  - The action to run

### See Also

**Reference**

- IReaderWriterStateLocker<TState> Interface
- InLock Overload
- W.Threading.Lockers Namespace
Tungsten

$^\text{74}$W
IReaderWriterStateLocker<TState> InLock Method (LockTypeEnum, Func<TState, TValue>)

Executes a function from within a ReaderWriterLockSlim

**Namespace:**  W.Threading.Lockers  
**Assembly:**  Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```csharp
TValue InLock<TValue>(
    LockTypeEnum lockType,
    Func<TState, TValue> func
)
```

### Parameters

**lockType**
- Type:  `W.Threading.LockersLockTypeEnum`
- Specifies whether to use a Read or Write lock

**func**
- Type:  `System.Func<TState, TValue>`
- The function to run

### Type Parameters

**TValue**
- The type of return value

### Return Value

- Type:  `TValue`
- The result of the function call (a value of type TValue)
See Also

Reference
IReaderWriterStateLocker{TState Interface
InLock Overload
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
# IReaderWriterStateLocker<TState> InLockAsync Method

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLockAsync(LockTypeEnum, Action&lt;TState&gt;)</td>
<td>Executes an action from within a ReaderWriterLockSlim</td>
</tr>
<tr>
<td>InLockAsyncTValue(LockTypeEnum, Func&lt;TState, TValue&gt;)</td>
<td>Executes a function from within a ReaderWriterLockSlim</td>
</tr>
</tbody>
</table>

## See Also

- Reference
  - IReaderWriterStateLocker<TState> Interface
  - W.Threading.Lockers Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$^w$
IReaderWriterStateLocker<TState>\n
Method (LockTypeEnum, Action<TState>)

Executes an action from within a ReaderWriterLockSlim

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```c#
Task InLockAsync(
    LockTypeEnum lockType,
    Action<TState> action
)
```

### Parameters

- **lockType**
  - Type: W.Threading.Lockers.LockTypeEnum
  - Specifies whether to use a Read or Write lock

- **action**
  - Type: SystemAction<TState>
  - The action to run

### Return Value

Type: Task


### See Also
Tungsten

$W$
IReaderWriterStateLocker<TState>InLockAsync Method (LockTypeEnum, Func<TState, TValue>)

Executes a function from within a ReaderWriterLockSlim

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

## Syntax

```
C#  

Task<TValue> InLockAsync<TValue>(
    LockTypeEnum lockType,
    Func<TState, TValue> func
)
```

### Parameters

- **lockType**
  - Type: `W.Threading.Lockers.LockTypeEnum`
  - Specifies whether to use a Read or Write lock

- **func**
  - Type: `System.Func<TState, TValue>`
  - The function to run

### Type Parameters

- **TValue**
  - The type of return value

### Return Value

- Type: `Task<TValue>`
  - The result of the function call (a value of type TValue)
See Also

Reference
  IReaderWriterStateLockerTState Interface
  InLockAsync Overload
  W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
**IReaderWriterStateLocker<TState> Set State Method**

Sets the internal state from within a ReaderWriterLockSlim

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

**Syntax**

```csharp
void SetState(
    TState newState
)
```

**Parameters**

*newState*  
Type: `TState`  
The new value

**See Also**

Reference  
**IReaderWriterStateLocker<TState> Interface**  
**W.Threading.Lockers Namespace**

Copyright © 2018 Jordan Duerksen
Tungsten

\( W \)
IStateLocker\textit{TLocker, TState}

Interface

The required implementation for a stateful locking object

\textbf{Namespace:} W.Threading.Lockers

\textbf{Syntax}

\begin{verbatim}
public interface IStateLocker<TLocker, TState> : ILocker
\end{verbatim}

Type Parameters

\textit{TLocker}

[Missing <typeparam name="TLocker"/> documentation for "T.W.Threading.Lockers.IStateLocker`2"]

\textit{TState}

[Missing <typeparam name="TState"/> documentation for "T.W.Threading.Lockers.IStateLocker`2"]

The IStateLocker\textit{TLocker, TState} type exposes the following members.

\textbf{Properties}

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![ ]</td>
<td>\textit{Locker} The object used for locking</td>
</tr>
<tr>
<td></td>
<td>(Inherited from ILocker\textit{TLocker}.)</td>
</tr>
</tbody>
</table>
## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLock(Action)</td>
<td>Perform some action in a lock (Inherited from ILocker.)</td>
</tr>
<tr>
<td>InLock(ActionTState)</td>
<td>Perform some action in a lock</td>
</tr>
<tr>
<td>InLock(FuncTState, TState)</td>
<td>Perform some function in a lock</td>
</tr>
<tr>
<td>InLockTResult(FuncTResult)</td>
<td>Perform some function in a lock (Inherited from ILocker.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>(Inherited from ILocker.)</td>
</tr>
<tr>
<td>InLockAsync(ActionTState)</td>
<td></td>
</tr>
<tr>
<td>InLockAsync(FuncTState, TState)</td>
<td>Asynchronously perform some function in a lock</td>
</tr>
<tr>
<td>InLockAsyncTResult(FuncTResult)</td>
<td>Asynchronously perform some function in a lock</td>
</tr>
</tbody>
</table>
See Also

Reference

W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
IStateLocker

The `IStateLocker`, `TState` generic type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locker</td>
<td>The object used for locking (Inherited from <code>ILockerTLocker</code>.)</td>
</tr>
</tbody>
</table>

### See Also

- `IStateLockerTLocker, TState Interface`
- `W.Threading.Lockers Namespace`

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
## IStateLocker

**TLocker, TState**

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLock(Action)</td>
<td>Perform some action in a lock (Inherited from ILocker.)</td>
</tr>
<tr>
<td>InLock(Action&lt;&gt;TState)</td>
<td>Perform some action in a lock</td>
</tr>
<tr>
<td>InLock(Func&lt;&gt;TState, TState)</td>
<td>Perform some function in a lock</td>
</tr>
<tr>
<td>InLockTResult(Func&lt;&gt;TResult)</td>
<td>Perform some function in a lock (Inherited from ILocker.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>(Inherited from ILocker.)</td>
</tr>
<tr>
<td>InLockAsync(Action&lt;&gt;TState)</td>
<td></td>
</tr>
<tr>
<td>InLockAsync(Func&lt;&gt;TState, TState)</td>
<td>Asynchronously perform some function in a lock</td>
</tr>
</tbody>
</table>
InLockAsyncTResult(Func<TResult>)  Asynchrhonously perform some function in a lock
(Inherited from ILocker.)

See Also

Reference
IStateLocker TLocker, TState Interface
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
IStateLocker

\textit{TLocker, TState} InLock Method

\textbf{Overload List}

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLock(Action)</td>
<td>Perform some action in a lock (Inherited from ILocker.)</td>
</tr>
<tr>
<td>InLockTResult(FuncTResult)</td>
<td>Perform some function in a lock (Inherited from ILocker.)</td>
</tr>
<tr>
<td>InLock(ActionTState)</td>
<td>Perform some action in a lock</td>
</tr>
<tr>
<td>InLock(FuncTState, TState)</td>
<td>Perform some function in a lock</td>
</tr>
</tbody>
</table>

\textbf{See Also}

Reference
IStateLockerTLocker, TState Interface
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\(W\)
IStateLocker TLocker, TState InLock Method (Action TState)

Perform some action in a lock

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```
void InLock(
    Action<TState> action
)
```

### Parameters

*action*

Type: `SystemAction<TState>`  
The action to perform

### See Also

Reference  
IStateLocker TLocker, TState Interface  
InLock Overload  
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
IStateLocker TLocker, TState InLock Method (Func TState, TState)

Perform some function in a lock

**Namespace:** W.Threading.Lockers
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```c#
TState InLock(
    Func<TState, TState> func
)
```

**Parameters**

*func*

Type: `System.Func<TState, TState>`

The function to perform

**Return Value**

Type: `TState`

The result of the function

### See Also

Reference

IStateLocker TLocker, TState Interface

InLock Overload
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$^W$
IStateLocker\(TLocker,\) \(TState\) InLockAsync Method

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLockAsync((Action))</td>
<td>(Inherited from ILocker.)</td>
</tr>
<tr>
<td>InLockAsync(TResult((Func(TResult))))</td>
<td>Asynchronously perform some function in a lock (Inherited from ILocker.)</td>
</tr>
<tr>
<td>InLockAsync((Action(TState))</td>
<td></td>
</tr>
<tr>
<td>InLockAsync((Func(TState, \ TState))</td>
<td>Asynchronously perform some function in a lock</td>
</tr>
</tbody>
</table>

**See Also**

Reference

IStateLocker\(TLocker, \ TState\) Interface

W.Threading.Lockers Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

W
IStateLocker\(T\)Locker,
\(T\)StateInLockAsync Method
(Action\(T\)State)

[Missing <summary> documentation for 
"M:W.Threading.Lockers.IStateLocker`2.InLockAsync(System.Action\`1)"
]

Namespace:  W.Threading.Lockers

Syntax

C#

```csharp
Task InLockAsync(
    Action\(<TState>\) action
)
```

Parameters

\(action\)
Type:  System\(Action\(T\)State\)
[Missing <param name="action"/> documentation for 
"M:W.Threading.Lockers.IStateLocker`2.InLockAsync(System.Action\`1)"
]

Return Value
Type:  Task
[Missing <returns> documentation for 
"M:W.Threading.Lockers.IStateLocker`2.InLockAsync(System.Action\`1)"
]

See Also
Reference
IStateLocker, TLocker, TState Interface
InLockAsync Overload
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$\text{W}$
**IStateLocker**<br>
**TLocker**, **TState**<br>
**InLockAsync Method** (Func`TState, TState)<br>

Asynchrnonously perform some function in a lock<br>

**Namespace:** W.Threading.Lockers<br>
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

**Syntax**

```csharp
Task<TState> InLockAsync(
    Func<TState, TState> func
)
```

**Parameters**

*func*

Type: `System.Func<TState, TState>`<br>The function to perform

**Return Value**

Type: `Task<TState>`<br>The result of the function

**See Also**

Reference<br>
*IStateLocker TLocker, TState Interface<br>InLockAsync Overload*
Tungsten

\( W \)
LockTypeEnum Enumeration

Used by ReaderWriterLocker to specify the type of lock to obtain

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```csharp
public enum LockTypeEnum
```

### Members

<table>
<thead>
<tr>
<th>Member name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read</td>
<td>0</td>
<td>Obtain a read-lock</td>
</tr>
<tr>
<td>Write</td>
<td>1</td>
<td>Obtain a write-lock</td>
</tr>
</tbody>
</table>

### See Also

Reference

W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\[ W \]
MonitorExtensions Class

Extensions to object to simplify locking with Monitor

Inheritance Hierarchy

```csharp
SystemObject  W.Threading.LockersMonitorExtensions
```

Namespace:  W.Threading.Lockers

Syntax

```csharp
public static class MonitorExtensions
```

The `MonitorExtensions` type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLock(Object, Action)</td>
<td>Performs the action in a Monitor lock</td>
</tr>
<tr>
<td><code>InLockTType(Object, Func&lt;TType&gt;)</code></td>
<td>Performs the function in a Monitor lock</td>
</tr>
<tr>
<td>InLockAsync(Object, Action)</td>
<td>Asynchronously performs the action in a Monitor lock</td>
</tr>
<tr>
<td><code>InLockAsyncTType(Object, Func&lt;TType&gt;)</code></td>
<td>Asynchronously performs the action</td>
</tr>
<tr>
<td><strong>Lock</strong></td>
<td>Performs a Monitor lock</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td><strong>Unlock</strong></td>
<td>Performs a Monitor unlock</td>
</tr>
</tbody>
</table>

### See Also

Reference

`W.Threading.Lockers Namespace`
Tungsten

$W$
MonitorExtensions Methods

The **MonitorExtensions** type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="lock.png" alt="lock" /> <strong>InLock</strong>(<em>Object, Action</em>)</td>
<td>Performs the action in a Monitor lock</td>
</tr>
<tr>
<td><img src="lock.png" alt="lock" /> <strong>InLock</strong>(<em>TType, Func&lt;TType&gt;</em>)</td>
<td>Performs the function in a Monitor lock</td>
</tr>
<tr>
<td><img src="lock.png" alt="lock" /> <strong>InLockAsync</strong>(<em>Object, Action</em>)</td>
<td>Asynchronously performs the action in a Monitor lock</td>
</tr>
<tr>
<td><img src="lock.png" alt="lock" /> <strong>InLockAsync</strong>(<em>TType, Func&lt;TType&gt;</em>)</td>
<td>Asynchronously performs the action in a Monitor lock</td>
</tr>
<tr>
<td><img src="lock.png" alt="lock" /> <strong>Lock</strong></td>
<td>Performs a Monitor lock</td>
</tr>
<tr>
<td><img src="lock.png" alt="lock" /> <strong>Unlock</strong></td>
<td>Performs a Monitor unlock</td>
</tr>
</tbody>
</table>

**Top**

## See Also

**Reference**

*MonitorExtensions Class*

*W.Threading.Lockers Namespace*
MonitorExtensions.InLock Method

**Overload List**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLock(Object, Action)</td>
<td>Performs the action in a Monitor lock</td>
</tr>
<tr>
<td>InLockTTType(Object, FuncTTType)</td>
<td>Performs the function in a Monitor lock</td>
</tr>
</tbody>
</table>

**See Also**

Reference
- MonitorExtensions Class
- W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
MonitorExtensionsInLock Method (Object, Action)

Performs the action in a Monitor lock

**Namespace:** W.Threading.Lockers
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

## Syntax

```csharp
public static void InLock(
    this Object this,
    Action action
)
```

## Parameters

- **this**
  - Type: `System.Object`
  - The object to provide resource locking

- **action**
  - Type: `System.Action`
  - The action to perform

## Usage Note

In Visual Basic and C#, you can call this method as an instance method on any object of type `Object`. When you use instance method syntax to call this method, omit the first parameter. For more information, see Extension Methods (Visual Basic) or Extension Methods (C# Programming Guide).
See Also

Reference
MonitorExtensions Class
InLock Overload
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
MonitorExtensions.InLock<TType> Method (Object, Func<TType>)

Performs the function in a Monitor lock

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

**Syntax**

```csharp
public static TType InLock<TType>(
    this Object this,
    Func<TType> func
)
```

**Parameters**

`this`
- Type: `SystemObject`  
The object to provide resource locking

`func`
- Type: `SystemFunc<TType>`  
The function to perform

**Type Parameters**

`TType`


**Return Value**
Type: `TTtype`

[Missing <returns> documentation for

**Usage Note**

In Visual Basic and C#, you can call this method as an instance method on any object of type `Object`. When you use instance method syntax to call this method, omit the first parameter. For more information, see *Extension Methods (Visual Basic)* or *Extension Methods (C# Programming Guide)*.

**See Also**

**Reference**

MonitorExtensions Class
InLock Overload
W.Threading.Lockers Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$^\text{W}$
MonitorExtensionsInLockAsync Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLockAsync(Object, Action)</td>
<td>Asynchronously performs the action in a Monitor lock</td>
</tr>
<tr>
<td>InLockAsyncTType(Object, FuncTType)</td>
<td>Asynchronously performs the action in a Monitor lock</td>
</tr>
</tbody>
</table>

See Also

- Reference
  - MonitorExtensions Class
  - W.Threading.Lockers Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
MonitorExtensionsInLockAsync Method (Object, Action)

Asynchronously performs the action in a Monitor lock

**Namespace:** W.Threading.Lockers
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

**Syntax**

```
public static Task InLockAsync(
    this Object this, 
    Action action
)
```

**Parameters**

- **this**
  Type: **SystemObject**  
The object to provide resource locking

- **action**
  Type: **SystemAction**  
The action to perform

**Return Value**

Type: **Task**


**Usage Note**

In Visual Basic and C#, you can call this method as an instance method on any object of type **Object**. When you use instance
method syntax to call this method, omit the first parameter. For more
information, see Extension Methods (Visual Basic) or Extension
Methods (C# Programming Guide).

See Also

Reference
MonitorExtensions Class
InLockAsync Overload
W.Threading.Lockers Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

\( W \)
MonitorExtensionsInLockAsync<TType> Method (Object, Func<TType>)

Asynchronously performs the action in a Monitor lock

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```csharp
public static Task<TType> InLockAsync<TType>(
    this Object this,
    Func<TType> func
)
```

### Parameters

**this**
- Type: System.Object  
The object to provide resource locking

**func**
- Type: System.Func<TType>  
The function to perform

### Type Parameters

**TType**

Type: Task\texttt{TTtype}

Usage Note
In Visual Basic and C#, you can call this method as an instance method on any object of type \texttt{Object}. When you use instance method syntax to call this method, omit the first parameter. For more information, see Extension Methods (Visual Basic) or Extension Methods (C# Programming Guide).

See Also

Reference

- \texttt{MonitorExtensions Class}
- \texttt{InLockAsync Overload}
- \texttt{W.Threading.Lockers Namespace}

Copyright @ 2018 Jordan Duerksen
Tungsten

\( \text{W} \)
MonitorExtensionsLock Method

Performs a Monitor lock

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```csharp
public static void Lock(
    this Object this,
    Action action
)
```

### Parameters

**this**
- Type: `SystemObject`  
The object to provide resource locking

**action**
- Type: `SystemAction`  

### Usage Note

In Visual Basic and C#, you can call this method as an instance method on any object of type `Object`. When you use instance method syntax to call this method, omit the first parameter. For more information, see Extension Methods (Visual Basic) or Extension Methods (C# Programming Guide).

### See Also
Reference
MonitorExtensions Class
W.Threading.Lockers Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
MonitorExtensionsUnlock Method

Performs a Monitor unlock

Namespace: W.Threading.Lockers

Syntax

C#

```csharp
public static void Unlock(
    this Object this,
    Action action
)
```

Parameters

`this`
Type: System.Object
The object to provide resource locking

`action`
Type: System.Action

[Missing `<param name="action"/>` documentation for

Usage Note

In Visual Basic and C#, you can call this method as an instance method on any object of type `Object`. When you use instance method syntax to call this method, omit the first parameter. For more information, see Extension Methods (Visual Basic) or Extension Methods (C# Programming Guide).
Tungsten

W
MonitorLocker Class

Uses Monitor to provide resource locking

Inheritance Hierarchy

- `SystemObject
- `W.Threading.Lockers
- `MonitorLocker

**Namespace:** `W.Threading.Lockers
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

Syntax

```csharp
public class MonitorLocker : ILocker<Object>, ILocker
```

The `MonitorLocker` type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MonitorLocker</td>
<td>Initializes a new instance of the <code>MonitorLocker</code> class</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locker</td>
<td>The object used to perform locks</td>
</tr>
</tbody>
</table>
### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object. (Inherited from Object.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>InLock(Action)</code></td>
<td>Executes an action from within a Monitor</td>
</tr>
<tr>
<td><code>InLockTValue(FuncTValue)</code></td>
<td>Executes a function from within a Monitor</td>
</tr>
<tr>
<td><code>InLockAsync(Action)</code></td>
<td>Executes an action from within a Monitor</td>
</tr>
<tr>
<td><code>InLockAsyncTValue(FuncTValue)</code></td>
<td>Executes a function from within a Monitor</td>
</tr>
<tr>
<td><code>Lock</code></td>
<td>Locks the resource</td>
</tr>
<tr>
<td><code>MemberwiseClone</code></td>
<td>Creates a shallow copy of the current <code>Object</code>. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><code>ToString</code></td>
<td>Returns a string that represents the current object. (Inherited from <code>Object</code>.)</td>
</tr>
</tbody>
</table>
## Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AsTType</code></td>
<td>Use Generic syntax for the as operator. (Defined by <code>AsExtensions</code>.)</td>
</tr>
<tr>
<td><code>InitializeProperties</code></td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by <code>PropertyHostExtensions</code>.)</td>
</tr>
<tr>
<td><code>InLock(Action)</code></td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>InLockTType(FuncTTType)</code></td>
<td>Overloaded. Performs the function in a Monitor lock. (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTTType(FuncTTType)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets it's IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>
Remarks

Can be overridden to provide additional functionality

See Also

Reference

W.Threading.Lockers Namespace
Tungsten

$W$
MonitorLocker Constructor

Initializes a new instance of the MonitorLocker class

**Namespace:**  W.Threading.Lockers  
**Assembly:**  Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

## Syntax

```csharp
public MonitorLocker()
```

## See Also

Reference
- MonitorLocker Class
- W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$\text{W}$
MonitorLocker Properties

The `MonitorLocker` type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locker</td>
<td>The object used to perform locks</td>
</tr>
</tbody>
</table>

See Also

Reference

- `MonitorLocker Class`
- `W.Threading.Lockers Namespace`

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
MonitorLocker

The object used to perform locks

**Namespace:** W.Threading.Lockers

**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

**Syntax**

```csharp
public Object Locker { get; }
```

**Property Value**
- **Type:** Object
- Implements **ILocker**
- Implements **TLocker**

**See Also**

Reference
- **MonitorLocker Class**
- **W.Threading.Lockers Namespace**

Copyright © 2018 Jordan Duerksen
Tungsten

W
MonitorLocker Methods

The `MonitorLocker` type exposes the following members.

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Equals</code></td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td><em>(Inherited from <code>Object</code>)</em></td>
</tr>
<tr>
<td><code>Finalize</code></td>
<td>Allows an object to try to free resources and perform other cleanup</td>
</tr>
<tr>
<td></td>
<td>operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td><em>(Inherited from <code>Object</code>)</em></td>
</tr>
<tr>
<td><code>GetHashCode</code></td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td></td>
<td><em>(Inherited from <code>Object</code>)</em></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Executes an action from within a Monitor</td>
</tr>
<tr>
<td>InLockTValue(FuncTValue)</td>
<td>Executes a function from within a Monitor</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Executes an action from within a Monitor</td>
</tr>
<tr>
<td>InLockAsyncTValue(FuncTValue)</td>
<td>Executes a function from within a Monitor</td>
</tr>
<tr>
<td>Lock</td>
<td>Locks the resource</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current <strong>Object</strong>. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a</td>
</tr>
</tbody>
</table>
string that represents the current object. (Inherited from Object.)

Unlock
Unlock the resource

**Top**

**Extension Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![AsTType] AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>![InitializeProperties] InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>![InLock(Action)] InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>![InLockTType(FuncTType)] InLockTType(FuncTType)</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTTType(Func&lt;TType&gt;)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock (Defined by</td>
</tr>
</tbody>
</table>
See Also

Reference
MonitorLocker Class
W.Threading.Lockers Namespace
Tungsten

$^W$
MonitorLocker

InLock Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLock(Action)</td>
<td>Executes an action from within a Monitor</td>
</tr>
<tr>
<td>InLockTValue(FuncTValue)</td>
<td>Executes a function from within a Monitor</td>
</tr>
</tbody>
</table>

See Also

Reference
MonitorLocker Class
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
MonitorLocker InLock Method (Action)

Executes an action from within a Monitor

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```csharp
public void InLock(Action action)
```

**Parameters**

*action*  
Type: System.Action  
The action to run

### Implements

ILockerInLock(Action)

### See Also

Reference  
MonitorLocker Class  
InLock Overload  
W.Threading.Lockers Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

W
MonitorLocker

**InLock**

Method **(Func<TValue>)**

Executes a function from within a Monitor

**Namespace:** W.Threading.Lockers

**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

**Syntax**

```csharp
public TValue InLock<TValue>(
    Func<TValue> func
)
```

**Parameters**

*func*

- **Type:** System.Func<TValue>
- The function to run

**Type Parameters**

*TValue*

- The type of return value

**Return Value**

- **Type:** TValue
- The result of the function call (a value of type TValue)

**See Also**

Reference
MonitorLocker Class
InLock Overload
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
</tr>
</tbody>
</table>
# MonitorLockerInLockAsync Method

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>InLockAsync(Action)</code></td>
<td>Executes an action from within a Monitor</td>
</tr>
<tr>
<td><code>InLockAsyncTValue(FuncTValue)</code></td>
<td>Executes a function from within a Monitor</td>
</tr>
</tbody>
</table>

## See Also

### Reference
- MonitorLocker Class
- W.Threading.Lockers Namespace

---

Copyright @ 2018 Jordan Duerksen
Tungsten

$^W$
MonitorLocker

InLockAsync Method (Action)

Executes an action from within a Monitor

**Namespace:** W.Threading.Lockers
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```
public Task InLockAsync(
    Action action
)
```

### Parameters

**action**
Type: SystemAction
The action to run

### Return Value
Type: Task


### Implements
ILockerInLockAsync(Action)

### See Also

Reference
MonitorLocker Class
MonitorLocker

InLockAsync<TValue>

Method (Func<TValue>)

Executes a function from within a Monitor

**Namespace:** W.Threading.Lockers

**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```
public Task<TValue> InLockAsync<TValue>(
    Func<TValue> func
)
```

### Parameters

*func*

Type: `System.Func<TValue>`

The function to run

### Type Parameters

**TValue**

The type of return value

### Return Value

Type: `Task<TValue>`

The result of the function call (a value of type TValue)

### See Also

Reference
MonitorLocker Class
InLockAsync Overload
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
MonitorLockerLock Method

Locks the resource

**Namespace:**  W.Threading.Lockers  
**Assembly:**  Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```csharp
public void Lock()
```

Return Value

Type:

- Returns True

### See Also

Reference

- MonitorLocker Class
- W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
MonitorLockerUnlock Method

Unlock the resource

**Namespace:** W.Threading.Lockers
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

## Syntax

```c#
public void Unlock()
```

## See Also

Reference

- MonitorLocker Class
- W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
MonitorLocker $TState$ Class

Extends MonitorLocker with an internal state variable

Inheritance Hierarchy

- System
  - Object
  - W.Threading.Lockers
    - StateLocker
    - MonitorLocker
    - $TState$
      - W.Threading.Lockers
      - MonitorLocker

Namespace: W.Threading.Lockers

Syntax

```csharp
public class MonitorLocker<TState> : StateLocker<!--
```

Type Parameters

$TState$

The state Type

The `MonitorLocker<TState>` type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MonitorLocker&lt;TState&gt;</td>
<td>Initializes a new instance of the <code>MonitorLocker&lt;TState&gt;</code> class</td>
</tr>
</tbody>
</table>
## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locker</td>
<td>The locking mechanism (SpinLock, Monitor, SemaphoreSlim, ReaderWriterLock) (Inherited from <code>StateLockerTLocker</code>, <code>TState</code>.)</td>
</tr>
</tbody>
</table>

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <code>Type</code> of the current instance. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>InLock(Action)</strong></td>
<td>Performs an action from within a lock (Inherited from <code>StateLockerTLocker</code>, <code>TState</code>.)</td>
</tr>
<tr>
<td><strong>InLock(Action&lt;TState&gt;)</strong></td>
<td>Performs an action from within a lock, passing in the current state (Inherited from <code>StateLockerTLocker</code>, <code>TState</code>.)</td>
</tr>
<tr>
<td><strong>InLock(Func&lt;TState, TState&gt;)</strong></td>
<td>Performs a function from within a lock, passing in the current state and assigning the state to the function result (Inherited from <code>StateLockerTLocker</code>, <code>TState</code>.)</td>
</tr>
<tr>
<td><strong>InLock&lt;TResult(Func&lt;TResult&gt;)</strong></td>
<td>Performs a function from within a lock (Inherited from <code>StateLockerTLocker</code>, <code>TState</code>.)</td>
</tr>
<tr>
<td>Method Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Asynchronously performs an action from within a lock (Inherited from StateLockerTLocker, TState.)</td>
</tr>
<tr>
<td>InLockAsync(Action&lt;TState&gt;)</td>
<td>Asynchronously performs an action from within a lock, passing in the current state (Inherited from StateLockerTLocker, TState.)</td>
</tr>
<tr>
<td>InLockAsync(Func&lt;TState, TState&gt;)</td>
<td>Asynchronously performs a function from within a lock, passing in the current state and assigning the state to the function result (Inherited from StateLockerTLocker, TState.)</td>
</tr>
<tr>
<td>InLockAsync&lt;TResult&gt;(Func&lt;TResult&gt;)</td>
<td>Asynchronously performs a function from within a lock (Inherited from StateLockerTLocker, TState.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from StateLockerTLocker, TState.)</td>
</tr>
</tbody>
</table>
Object.

ToString

Returns a string that represents the current object. (Inherited from Object.)

Top

Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
<td>The internal state (Inherited from StateLockerTLocker, TState.)</td>
</tr>
</tbody>
</table>

Top

Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;! This method should be called in the constructor of any class which has IOwnedProperty members (Defined by</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTTType(Func&lt;TType&gt;)</td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTTType(Func&lt;TType&gt;)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td></td>
<td>Scans each field and</td>
</tr>
</tbody>
</table>
| **MarkAsClean** | property of type IProperty and sets it's `IsDirty` flag to false  
(Defined by `PropertyHostExtensions`.) |
|----------------|--------------------------------------------------------------------------------------------------|
| **Unlock**     | Performs a Monitor unlock  
(Defined by `MonitorExtensions`.) |

### Remarks

Same as `StateLocker<MonitorLocker;, TState>`

### See Also

Reference

`W.Threading.Lockers Namespace`

Copyright © 2018 Jordan Duerksen
Tungsten

$\text{W}^*$
MonitorLockerTState Constructor

Initializes a new instance of the MonitorLockerTState class

Namespace:  W.Threading.Lockers

Syntax

```csharp
public MonitorLocker()
```

See Also

Reference
MonitorLockerTState Class
W.Threading.Lockers Namespace
Tungsten

$\text{W}$
MonitorLocker\textit{TState} Properties

The \texttt{MonitorLockerTState} generic type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Locker</td>
<td>The locking mechanism (SpinLock, Monitor, SemaphoreSlim, \n</td>
</tr>
</tbody>
</table>

### See Also

**Reference**
- \texttt{MonitorLockerTState Class}
- \texttt{W.Threading.Lockers Namespace}
Tungsten

$^\text{74}$ W
MonitorLocker{TState} Methods

The MonitorLocker{TState} generic type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Performs an action from within a lock (Inherited from StateLocker&lt;TLocker, TState&gt;.)</td>
</tr>
<tr>
<td>InLock(Action&lt;TState&gt;)</td>
<td>Performs an action from within a lock, passing in the current state (Inherited from StateLocker&lt;TLocker, TState&gt;.)</td>
</tr>
<tr>
<td>InLock(Func&lt;TState, TState&gt;)</td>
<td>Performs a function from within a lock, passing in the current state and assigning the state to the function result (Inherited from StateLocker&lt;TLocker, TState&gt;.)</td>
</tr>
<tr>
<td>InLockTResult(Func&lt;TResult&gt;)</td>
<td>Performs a function from within a lock (Inherited from StateLocker&lt;TLocker, TState&gt;.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Asynchronously performs an action from within a lock (Inherited from StateLocker&lt;TLocker, TState&gt;.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| `InLockAsync(Action<TState>)` | Asynchronously performs an action from within a lock, passing in the current state (Inherited from `StateLocker<TLocker, TState>`.)
| `InLockAsync(Func<TState, TState>)` | Asynchronously performs a function from within a lock, passing in the current state and assigning the state to the function result (Inherited from `StateLocker<TLocker, TState>`.)
| `InLockAsyncTResult(Func<TResult>)` | Asynchronously performs a function from within a lock (Inherited from `StateLocker<TLocker, TState>`.)
| `MemberwiseClone` | Creates a shallow copy of the current `Object`. (Inherited from `Object`.)
| `ToString` | Returns a string that represents the current object. (Inherited from `Object`.)
### Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>      AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>      InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>      InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>      InLockTType(FuncTType)</td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>      InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs</td>
</tr>
</tbody>
</table>
the action in a Monitor lock (Defined by `{MonitorExtensions}`.)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>InLockAsync&lt;TType&gt;(Func&lt;TType&gt;)</strong></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by <code>{MonitorExtensions}</code>.)</td>
</tr>
<tr>
<td><strong>IsDirty</strong></td>
<td>Scans the IsDirty value of each field and property of type <code>IProperty</code> (Defined by <code>{PropertyHostExtensions}</code>.)</td>
</tr>
<tr>
<td><strong>Lock</strong></td>
<td>Performs a Monitor lock (Defined by <code>{MonitorExtensions}</code>.)</td>
</tr>
<tr>
<td><strong>MarkAsClean</strong></td>
<td>Scans each field and property of type <code>IProperty</code> and sets it’s IsDirty flag to false (Defined by <code>{PropertyHostExtensions}</code>.)</td>
</tr>
<tr>
<td><strong>Unlock</strong></td>
<td>Performs a Monitor unlock (Defined by <code>{MonitorExtensions}</code>.)</td>
</tr>
</tbody>
</table>

See Also

Reference
MonitorLockerTState Class
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
MonitorLocker\texttt{TState} Fields

The \texttt{MonitorLockerTState} generic type exposes the following members.

### Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{State}</td>
<td>The internal state (Inherited from \texttt{StateLockerTLocke}, \texttt{TState}).</td>
</tr>
</tbody>
</table>

[See Also](#)

Reference

- \texttt{MonitorLockerTState Class}
- \texttt{W.Threading.Lockers Namespace}

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
ReaderWriterLocker Class

Uses ReaderWriterLockSlim to provide resource locking

Inheritance Hierarchy

System\Object \ W.Threading.Lockers\ReaderWriterLocker

Namespace: \ W.Threading.Lockers

Syntax

C#

```csharp
public class ReaderWriterLocker : IDisposable, ILocker<ReaderWriterLockSlim>, ILocker
```

The ReaderWriterLocker type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReaderWriterLocker</td>
<td>Constructs a n ReaderWriterL with a LockRecursion of NoRecursion</td>
</tr>
<tr>
<td>ReaderWriterLocker(LockRecursionPolicy)</td>
<td>Constructs a n ReaderWriterL using the spec LockRecursion</td>
</tr>
</tbody>
</table>
## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locker</td>
<td>The <code>ReaderWriterLockSlim</code> used to perform locks</td>
</tr>
</tbody>
</table>

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispose</td>
<td>Disposes the instance and releases resources</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>GetType</code></td>
<td>Gets the Type of the current instance. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><code>InLock(Action)</code></td>
<td>Performs the action in a read lock</td>
</tr>
<tr>
<td><code>InLock(LockTypeEnum, Action)</code></td>
<td>Executes an action from within a ReaderWriterLockSlim</td>
</tr>
<tr>
<td><code>InLock TResult(Func&lt;TResult&gt;)</code></td>
<td>Performs the function in a read lock</td>
</tr>
<tr>
<td><code>InLock TValue(LockTypeEnum, Func TValue)</code></td>
<td>Executes a function from within a ReaderWriterLockSlim</td>
</tr>
<tr>
<td><code>InLockAsync(Action)</code></td>
<td>Asynchronously performs the action in a read lock</td>
</tr>
<tr>
<td><code>InLockAsync(LockTypeEnum, Action)</code></td>
<td>Executes an action from within a ReaderWriterLockSlim</td>
</tr>
<tr>
<td><code>InLockAsync TResult(Func&lt;TResult&gt;)</code></td>
<td>Asynchronously performs the function in a read lock</td>
</tr>
<tr>
<td><code>InLockAsync TValue(LockTypeEnum, Func TValue)</code></td>
<td>Executes a function from within a ReaderWriterLockSlim</td>
</tr>
<tr>
<td><code>Lock</code></td>
<td>Enters a read or write lock</td>
</tr>
</tbody>
</table>
lock on the ReaderWriterLockSlim

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MemberwiseClone</strong></td>
</tr>
<tr>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ToString</strong></td>
</tr>
<tr>
<td>Returns a string that represents the current object. (Inherited from Object.)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Unlock</strong></td>
</tr>
<tr>
<td>Exits a read or write lock on the ReaderWriterLockSlim</td>
</tr>
</tbody>
</table>

Top

## Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![AsTType][1]</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>![InitializeProperties][2]</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot; This method should be called in the constructor of any class which has IOwnedProperty members</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>InLock(Action)</code></td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by <code>MonitorExtensions</code>).</td>
</tr>
<tr>
<td><code>InLockTType(Func&lt;TType&gt;)</code></td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by <code>MonitorExtensions</code>).</td>
</tr>
<tr>
<td><code>InLockAsync(Action)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by <code>MonitorExtensions</code>).</td>
</tr>
<tr>
<td><code>InLockAsyncTType(Func&lt;TType&gt;)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by <code>MonitorExtensions</code>).</td>
</tr>
<tr>
<td><code>IsDirty</code></td>
<td>Scans the IsDirty value of each field and property of type <code>IProperty</code> (Defined by <code>PropertyHostExtensions</code>).</td>
</tr>
<tr>
<td><code>Lock</code></td>
<td>Performs a Monitor lock (Defined by <code>MonitorExtensions</code>).</td>
</tr>
<tr>
<td>Remarks</td>
<td>See Also</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Can be overridden to provide additional functionality</td>
<td>Reference</td>
</tr>
<tr>
<td></td>
<td>W.Threading.Lockers Namespace</td>
</tr>
</tbody>
</table>

Copyright © 2018 Jordan Duerksen
Tungsten

$^7\text{W}$
ReaderWriterLocker Constructor

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReaderWriterLocker</td>
<td>Constructs a new ReaderWriterLocker with a LockRecursionPolicy of NoRecursion.</td>
</tr>
<tr>
<td>ReaderWriterLocker(LockRecursionPolicy)</td>
<td>Constructs a new ReaderWriterLocker using the specified LockRecursionPolicy.</td>
</tr>
</tbody>
</table>

## See Also

### Reference
- ReaderWriterLocker Class
- W.Threading.Lockers Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
# ReaderWriterLocker Constructor

Constructs a new ReaderWriterLocker with a LockRecursionPolicy of NoRecursion

**Namespace:**  W.Threading.Lockers  
**Assembly:**  Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

## Syntax

```csharp
public ReaderWriterLocker()
```

## See Also

**Reference**
- ReaderWriterLocker Class
- ReaderWriterLocker Overload
- W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
ReaderWriterLocker Constructor (LockRecursionPolicy)

Constructs a new ReaderWriterLocker using the specified LockRecursionPolicy

Namespace: W.Threading.Lockers

Syntax

```csharp
public ReaderWriterLocker(
    LockRecursionPolicy lockPolicy
)
```

Parameters

`lockPolicy`  
Type: System.Threading.LockRecursionPolicy  
The lock recursion policy to use

See Also

Reference
- ReaderWriterLocker Class
- ReaderWriterLocker Overload
- W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
| Tungsten | W |
The `ReaderWriterLocker` type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="" /> Locker</td>
<td>The <code>ReaderWriterLockSlim</code> used to perform locks</td>
</tr>
</tbody>
</table>

### See Also

Reference

- `ReaderWriterLocker Class`
- `W.Threading.Lockers Namespace`

Copyright © 2018 Jordan Duerksen
Tungsten

\( W \)
ReaderWriterLocker

Property

The ReaderWriterLockSlim used to perform locks

**Namespace:** W.Threading.Lockers

**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```csharp
public ReaderWriterLockSlim Locker { get; }
```

### Property Value

**Type:** ReaderWriterLockSlim

**Implements**

ILocker TLocker

### See Also

**Reference**

ReaderWriterLocker Class

W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
# ReaderWriterLocker Methods

The `ReaderWriterLocker` type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dispose</strong></td>
<td>Disposes the instance and releases resources.</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified object is equal to the current object. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as the default hash function. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the <code>Type</code> of the current instance.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Performs the action in a read lock</td>
</tr>
<tr>
<td>InLock(LockTypeEnum, Action)</td>
<td>Executes an action from within a ReaderWriterLockSlim</td>
</tr>
<tr>
<td>InLockTResult(FuncTResult)</td>
<td>Performs the function in a read lock</td>
</tr>
<tr>
<td>InLockTValue(LockTypeEnum, FuncTValue)</td>
<td>Executes a function from within a ReaderWriterLockSlim</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Asynchronously performs the action in a read lock</td>
</tr>
<tr>
<td>InLockAsync(LockTypeEnum, Action)</td>
<td>Executes an action from within a ReaderWriterLockSlim</td>
</tr>
<tr>
<td>InLockAsyncTResult(FuncTResult)</td>
<td>Asynchronously performs the function in a read lock</td>
</tr>
<tr>
<td>InLockAsyncTValue(LockTypeEnum, FuncTValue)</td>
<td>Executes a function from within a ReaderWriterLockSlim</td>
</tr>
<tr>
<td>Lock</td>
<td>Enters a read or write lock on the ReaderWriterLockSlim</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow</td>
</tr>
</tbody>
</table>
copy of the current 
Object.
(Inherited from 
Object.)

ToString

Returns a string that represents the current object.
(Inherited from Object.)

Unlock

Exits a read or write lock on the ReaderWriterLockSlim.

Top

## Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| AsTTType              | Use Generic syntax for the as operator. 
(Defined by AsExtensions.) |
| InitializeProperties  | Scans the fields and properties of "owner" and sets the member's Owner property to "owner"  
This method should be called in the constructor of any class which has IOwnedProperty members 
(Defined by PropertyHostExtensions.) |
<p>| InLock(Action)         | Overloaded. |</p>
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLockTType(Func&lt;TType&gt;)</td>
<td>Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTType(Func&lt;TType&gt;)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false</td>
</tr>
</tbody>
</table>
Unlock

Performs a Monitor unlock

See Also

Reference
ReaderWriterLocker Class
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>$^\text{W}$</td>
</tr>
</tbody>
</table>
ReaderWriterLocker Dispose Method

Disposes the instance and releases resources

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

**Syntax**

```csharp
public void Dispose()
```

Implements  
**IDisposableDispose**

**See Also**

Reference  
ReaderWriterLocker Class  
W.Threading.Lockers Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
# ReaderWriterLockerInLock Method

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLock(Action)</td>
<td>Performs the action in a read lock</td>
</tr>
<tr>
<td>InLockTResult(FuncTResult)</td>
<td>Performs the function in a read lock</td>
</tr>
<tr>
<td>InLock(LockTypeEnum, Action)</td>
<td>Executes an action from within a ReaderWriterLockSlim</td>
</tr>
<tr>
<td>InLockTValue(LockTypeEnum, FuncTValue)</td>
<td>Executes a function from within a ReaderWriterLockSlim</td>
</tr>
</tbody>
</table>

## See Also

### Reference
- ReaderWriterLocker Class
- W.Threading.Lockers Namespace
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>$^\text{W}$</td>
</tr>
</tbody>
</table>
ReaderWriterLocker.InLock Method (Action)

Performs the action in a read lock

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```
public void InLock(
    Action action
)
```

### Parameters

**action**
- Type: System.Action
- The action to perform

### Implements

ILockerInLock(Action)

### See Also

Reference
- ReaderWriterLocker Class
- InLock Overload
- W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
**ReaderWriterLocker.InLock<TResult> Method (Func<TResult>)**

Performs the function in a read lock

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```csharp
public TResult InLock<TResult>(
    Func<TResult> func
)
```

**Parameters**

*func*

Type: `System.Func<TResult>`  
The function to perform

**Type Parameters**

*TResult*  
[Missing `<typeparam name="TResult"` documentation for "M:W.Threading.Lockers.ReaderWriterLocker.InLock`1(System.Func`0)""]

**Return Value**

Type: `TResult`  

**Implements**
See Also

Reference
- ReaderWriterLocker Class
- InLock Overload
- W.Threading.Lockers Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
ReaderWriterLocker.InLock Method (LockTypeEnum, Action)

Executes an action from within a ReaderWriterLockSlim

**Namespace:** W.Threading.Lockers
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```csharp
public void InLock(
    LockTypeEnum lockType,
    Action action
)
```

### Parameters

**lockType**
- Type: W.Threading.LockersLockTypeEnum

**action**
- Type: SystemAction
- The action to run

### See Also

**Reference**
- ReaderWriterLocker Class
- InLock Overload
- W.Threading.Lockers Namespace
Tungsten

W
ReaderWriterLocker.InLock\<TValue\>(LockTypeEnum, Func\<TValue\>)

Executes a function from within a ReaderWriterLockSlim

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```csharp
public TValue InLock<TValue>(
    LockTypeEnum lockType,
    Func<TValue> func
)
```

### Parameters

- **lockType**
  - Type: W.Threading.LockersLockTypeEnum  

- **func**
  - Type: System.Func\<TValue\>
  - The function to run

### Type Parameters

- **TValue**
  - The type of return value
Return Value
Type: **TValue**
The result of the function call (a value of type TValue)

**See Also**

Reference
- ReaderWriterLocker Class
- InLock Overload
- W.Threading.Lockers Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

W
## ReaderWriterLocker InLockAsync Method

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLockAsync(Action)</td>
<td>Asynchronously performs the action in a read lock</td>
</tr>
<tr>
<td>InLockAsyncTResult(FuncTResult)</td>
<td>Asynchronously performs the function in a read lock</td>
</tr>
<tr>
<td>InLockAsync(LockTypeEnum, Action)</td>
<td>Executes an action from within a ReaderWriterLockSlim</td>
</tr>
<tr>
<td>InLockAsyncTValue(LockTypeEnum, FuncTValue)</td>
<td>Executes a function from within a ReaderWriterLockSlim</td>
</tr>
</tbody>
</table>

**Top**

### See Also

**Reference**
- [ReaderWriterLocker Class](#)
- [W.Threading.Lockers Namespace](#)

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
ReaderWriterLocker.InLockAsync Method (Action)

Asynchronously performs the action in a read lock

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

## Syntax

```csharp
public Task InLockAsync(
    Action action
)
```

### Parameters

- **action**
  - Type: `SystemAction`
  - The action to perform

### Return Value

- Type: `Task`

### Implements

- `ILockerInLockAsync(Action)`

## See Also

Reference
- `ReaderWriterLocker Class`
Tungsten

W
ReaderWriterLocker.InLockAsync<TResult> Method (Func<TResult>)

Asynchronously performs the function in a read lock

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```csharp
public Task<TResult> InLockAsync<TResult>(
    Func<TResult> func
)
```

### Parameters

**func**
- Type: `System.Func<TResult>`
- The action to perform

### Type Parameters

**TResult**
- [Missing <typeparam name="TResult"/> documentation for "M:W.Threading.Lockers.ReaderWriterLocker.InLockAsync`1(System.Func`0)""]

### Return Value

Type: `Task<TResult>`

### Implements
ILockerInLockAsyncTResult(FuncTResult)

See Also

Reference
ReaderWriterLocker Class
InLockAsync Overload
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
</tr>
</tbody>
</table>
# ReaderWriterLocker.InLockAsync Method (LockTypeEnum, Action)

Executes an action from within a ReaderWriterLockSlim

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

## Syntax

```csharp
public Task InLockAsync(
    LockTypeEnum lockType,
    Action action
)
```

## Parameters

- **lockType**
  - Type: W.Threading.Lockers.LockTypeEnum  
  
- **action**
  - Type: System.Action  
  - The action to run

## Return Value

- Type: Task

## See Also
Reference
ReaderWriterLocker Class
InLockAsync Overload
W Threading Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
UploaderWriterLockerInLockAsync<TValue> Method (LockTypeEnum, Func<TValue> Executes a function from within a ReaderWriterLockSlim

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```csharp
public Task<TValue> InLockAsync<TValue>(
    LockTypeEnum lockType,
    Func<TValue> func
)
```

### Parameters

**lockType**  
Type: W.Threading.LockersLockTypeEnum  
[Missing <param name="lockType"/> documentation for "M:W.Threading.Lockers.ReaderWriterLocker.InLockAsync``1(W.Threading.Lockers.LockTypeEnum,System.Func{``0})"]

**func**  
Type: System.Func<TValue>  
The function to run

### Type Parameters

**TValue**  
The type of return value

### Return Value
Type: **Task**\(T\text{Value}\)

The result of the function call (a value of type TValue)

**See Also**

**Reference**
- ReaderWriterLocker Class
- InLockAsync Overload
- W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
**ReaderWriterLocker Lock Method**

Enters a read or write lock on the ReaderWriterLockSlim

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

**Syntax**

```csharp
public void Lock(
    LockTypeEnum lockType
);
```

**Parameters**

*lockType*
- **Type:** W.Threading.Lockers.LockTypeEnum
- The type of lock to enter

**See Also**

**Reference**
- ReaderWriterLocker Class
- W.Threading.Lockers Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

W
ReaderWriterLockerUnlock Method

Exits a read or write lock on the ReaderWriterLockSlim

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

**Syntax**

```csharp
public void Unlock(
    LockTypeEnum lockType
)
```

**Parameters**

*lockType*  
Type: W.Threading.Lockers.LockTypeEnum  
The type of lock to exit

**See Also**

Reference  
ReaderWriterLocker Class  
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
ReaderWriterLocker<TState> Class

Extends ReaderWriterLocker with an internal state variable

Inheritance Hierarchy

- System
  - Object
  - W.Threading.Lockers.ReaderWriterLocker<TState>
    - WLockableSlim<TValue>

Namespace: W.Threading.Lockers

Syntax

```csharp
public class ReaderWriterLocker<TState> : IReader
IDisposable
```

Type Parameters

- **TState**
  - The state Type

The `ReaderWriterLocker<TState>` type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReaderWriterLocker&lt;TState&gt;</td>
<td>Constructs a new ReaderWriterLocker</td>
</tr>
</tbody>
</table>

Top
## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dispose</strong></td>
<td>Disposes the ReaderWriterLocker and releases resources</td>
</tr>
<tr>
<td><strong>Equals</strong></td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td><strong>Finalize</strong></td>
<td>Allows an object to try to free resources and perform other cleanup</td>
</tr>
<tr>
<td></td>
<td>operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td><strong>GetHashCode</strong></td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td><strong>GetState</strong></td>
<td>Retrieves the internal state from within a ReaderWriterLockSlim.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the Type of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from</td>
</tr>
</tbody>
</table>
### Fields

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLock(LockTypeEnum, ActionTState)</td>
<td>Executes an action from within a ReaderWriterLockSlim</td>
</tr>
<tr>
<td>InLockTValue(LockTypeEnum, FuncTState, TValue)</td>
<td>Executes a function from within a ReaderWriterLockSlim</td>
</tr>
<tr>
<td>InLockAsync(LockTypeEnum, ActionTState)</td>
<td>Executes an action from within a ReaderWriterLockSlim</td>
</tr>
<tr>
<td>InLockAsyncTValue(LockTypeEnum, FuncTState, TValue)</td>
<td>Executes a function from within a ReaderWriterLockSlim</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>SetState</td>
<td>Sets the internal state from within a ReaderWriterLockSlim</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object. (Inherited from Object.)</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Locker</strong></td>
<td>The ReaderWriterLocker used to access the State</td>
</tr>
<tr>
<td><strong>State</strong></td>
<td>The internal state</td>
</tr>
</tbody>
</table>

**Extension Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AsTType</strong></td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td><strong>InitializeProperties</strong></td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;! This method should be called in the constructor of any class which has IOwnedProperty members (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td><strong>InLock(Action)</strong></td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>InLockTType(FuncTType)</strong></td>
<td>Overloaded. Performs the function in a Monitor lock</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>InLockAsync(Action)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>InLockAsyncTTyped(FuncTTyped)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>IsDirty</code></td>
<td>Scans the IsDirty value of each field and property of type <code>IProperty</code> (Defined by <code>PropertyHostExtensions</code>.)</td>
</tr>
<tr>
<td><code>Lock</code></td>
<td>Performs a Monitor lock (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>MarkAsClean</code></td>
<td>Scans each field and property of type <code>IProperty</code> and sets its IsDirty flag to false (Defined by <code>PropertyHostExtensions</code>.)</td>
</tr>
<tr>
<td><code>Unlock</code></td>
<td>Performs a Monitor unlock (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
</tbody>
</table>
Tungsten

W
ReaderWriterLocker<TState> Constructor

Constructs a new ReaderWriterLocker

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

## Syntax

```
public ReaderWriterLocker(
    LockRecursionPolicy lockRecursionPolicy
)
```

### Parameters

- **lockRecursionPolicy**  
  Type: System.Threading.LockRecursionPolicy  
  The lock recursion policy to use

## See Also

- Reference:  
  ReaderWriterLockerTState Class  
  W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
## Methods

The **ReaderWriterLockerTState** generic type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispose</td>
<td>Disposes the ReaderWriterLocker and releases resources</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup</td>
</tr>
<tr>
<td></td>
<td>operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td><strong>GetState</strong></td>
<td>Retrieves the internal state from within a ReaderWriterLockSlim.</td>
</tr>
<tr>
<td><strong>GetType</strong></td>
<td>Gets the Type of the current instance. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>InLock(LockTypeEnum, ActionTState)</strong></td>
<td>Executes an action from within a ReaderWriterLockSlim.</td>
</tr>
<tr>
<td><strong>InLockTValue(LockTypeEnum, FuncTState, TValue)</strong></td>
<td>Executes a function from within a ReaderWriterLockSlim.</td>
</tr>
<tr>
<td><strong>InLockAsync(LockTypeEnum, ActionTState)</strong></td>
<td>Executes an action from within a ReaderWriterLockSlim.</td>
</tr>
<tr>
<td><strong>InLockAsyncTValue(LockTypeEnum, FuncTState, TValue)</strong></td>
<td>Executes a function from within a ReaderWriterLockSlim.</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current <strong>Object</strong>. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>SetState</strong></td>
<td>Sets the internal state from within a ReaderWriterLockSlim.</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Returns a string that</td>
</tr>
</tbody>
</table>
represents the current object.
(Inherited from Object.)

## Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot; This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTType(FuncTType)</td>
<td>Overloaded. Performs the function in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>InLockAsync(Action)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by <code>MonitorExtensions</code>).</td>
</tr>
<tr>
<td><code>InLockAsync&lt;TType&gt;(Func&lt;TType&gt;)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by <code>MonitorExtensions</code>).</td>
</tr>
<tr>
<td><code>IsDirty</code></td>
<td>Scans the IsDirty value of each field and property of type <code>IProperty</code> (Defined by <code>PropertyHostExtensions</code>).</td>
</tr>
<tr>
<td><code>Lock</code></td>
<td>Performs a Monitor lock (Defined by <code>MonitorExtensions</code>).</td>
</tr>
<tr>
<td><code>MarkAsClean</code></td>
<td>Scans each field and property of type <code>IProperty</code> and sets its IsDirty flag to false (Defined by <code>PropertyHostExtensions</code>).</td>
</tr>
<tr>
<td><code>Unlock</code></td>
<td>Performs a Monitor unlock (Defined by <code>MonitorExtensions</code>).</td>
</tr>
</tbody>
</table>
See Also

Reference
ReaderWriterLockerTState Class
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
ReaderWriterLocker<TState> Dispose Method

Disposes the ReaderWriterLocker and releases resources

Namespace: W.Threading.Lockers

Syntax

```csharp
public void Dispose()
```

Implements

IDisposable.Dispose

See Also

Reference

ReaderWriterLocker<TState> Class
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
ReaderWriterLocker $TState$ GetState Method

Retrieves the internal state from within a ReaderWriterLockSlim

**Namespace:**  W.Threading.Lockers  
**Assembly:**  Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```
public TState GetState()
```

**Return Value**

Type: $TState$

The current state

Implements

$IReaderWriterStateLocker$ $TState$ GetState

### See Also

Reference

ReaderWriterLocker $TState$ Class  
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
ReaderWriterLocker TState InLock Method

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLock(LockTypeEnum, Action&lt;TState&gt;)</td>
<td>Executes an action from within a ReaderWriterLockSlim</td>
</tr>
<tr>
<td>InLockTValue(LockTypeEnum, Func&lt;TState, TValue&gt;)</td>
<td>Executes a function from within a ReaderWriterLockSlim</td>
</tr>
</tbody>
</table>

**See Also**

Reference
- ReaderWriterLockerTState Class
- W.Threading.Lockers Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$^63\text{W}$
**ReaderWriterLocker**

**TState InLock Method (LockTypeEnum, Action<TState>)**

Executes an action from within a ReaderWriterLockSlim

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```c#
public void InLock(
    LockTypeEnum lockType,
    Action<TState> action
)
```

### Parameters

- **lockType**
  - Type: W.Threading.Lockers.LockTypeEnum
  - Specifies whether to use a Read or Write lock

- **action**
  - Type: System.Action<TState>
  - The action to run

### Implements

IRenderWriterStateLocker

IReaderWriterStateLockerTStateInLock(LockTypeEnum, Action<TState>)

### See Also
| Tungsten | $^\text{W}$ |
ReaderWriterLocker $TState$ InLock$TV$ Method (LockTypeEnum, Func $TStat \ TValue$)

Executes a function from within a ReaderWriterLockSlim

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```c#
public TValue InLock<TValue>(
    LockTypeEnum lockType,
    Func<TState, TValue> func
)
```

### Parameters

**lockType**
- Type: W.Threading.LockersLockTypeEnum
- Specifies whether to use a Read or Write lock

**func**
- Type: System(Func<TState, TValue>
- The function to run

### Type Parameters

**TValue**
- The type of return value

### Return Value
Type: \textit{TValue}

The result of the function call (a value of type \textit{TValue})

Implements
\texttt{IReaderWriterStateLocker}\texttt{TState}\texttt{InLock}\texttt{TValue}(\texttt{LockTypeEnum}, \texttt{FuncTState}, \texttt{TValue})

See Also

Reference
\texttt{ReaderWriter Locker TState Class}
\texttt{InLock Overload}
\texttt{W.Threading.Lockers Namespace}

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>$W$</td>
</tr>
</tbody>
</table>
ReaderWriterLocker<TState> InLockAsync Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLockAsync(LockTypeEnum, Action&lt;TState&gt;)</td>
<td>Executes an action from within a ReaderWriterLockSlim</td>
</tr>
<tr>
<td>InLockAsyncTValue(LockTypeEnum, Func&lt;TState, TValue&gt;)</td>
<td>Executes a function from within a ReaderWriterLockSlim</td>
</tr>
</tbody>
</table>

See Also

Reference
ReaderWriterLocker<TState> Class
W.Threading.Lockers Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$^\text{W}$
ReaderWriterLocker<TState>InLockAsync Method (LockTypeEnum, Action<TState>)

Executes an action from within a ReaderWriterLockSlim

**Namespace:**  W.Threading.Lockers  
**Assembly:**  Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

**Syntax**

```csharp
public Task InLockAsync(
    LockTypeEnum lockType,
    Action<TState> action
)
```

**Parameters**

- **lockType**
  
  Type:  W.Threading.Lockers.LockTypeEnum
  
  Specifies whether to use a Read or Write lock

- **action**
  
  Type:  System.Action<TState>
  
  The action to run

**Return Value**

Type:  Task


**Implements**

IReaderWriterStateLocker<TState>InLockAsync(LockTypeEnum, Action<TState>)
See Also

Reference
ReaderWriterLockerTState Class
InLockAsync Overload
W.Threading.Lockers Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

W
ReaderWriterLocker\textit{TState}\text{InLockAsync} Method (\text{LockTypeEnum}, \text{Func}\text{\textit{TStat}})

Executes a function from within a ReaderWriterLockSlim

\textbf{Namespace: } W.Threading.Lockers

\textbf{Syntax}

\begin{verbatim}
public Task<TValue> InLockAsync<TValue>(
    LockTypeEnum lockType,
    Func<TState, TValue> func
)
\end{verbatim}

\textbf{Parameters}

\textit{lockType}
\begin{itemize}
\item Type: W.Threading.Lockers\text{LockTypeEnum}
\item Specifies whether to use a Read or Write lock
\end{itemize}

\textit{func}
\begin{itemize}
\item Type: System\text{Func\textit{TState}, \textit{TValue}}
\item The function to run
\end{itemize}

\textbf{Type Parameters}

\textit{TValue}
\begin{itemize}
\item The type of return value
\end{itemize}

\textbf{Return Value}
\begin{itemize}
\item Type: Task\text{\textit{TValue}}
\item The result of the function call (a value of type \textit{TValue})
\end{itemize}
Implements
IReadWriterStateLocker<TState> InLockAsync<TValue>(LockTypeEnum, Func<TState, TValue>)

See Also

Reference

ReaderWriterLocker<TState> Class
InLockAsync Overload
W.Threading.Lockers Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

W
**ReaderWriterLocker**

Sets the internal state from within a ReaderWriterLockSlim

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

**Syntax**

```csharp
public void SetState(
    TState newState
)
```

**Parameters**

*newState*

Type: `TState`  
The new value

**Implements**

`IReaderWriterStateLocker<TState>`

**See Also**

Reference

ReaderWriterLocker TState Class  
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
# ReaderWriterLocker<sup>T</sup>State

The `ReaderWriterLocker<sup>T</sup>State` generic type exposes the following members.

## Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locker</td>
<td>The ReaderWriterLocker used to access the State</td>
</tr>
<tr>
<td>State</td>
<td>The internal state</td>
</tr>
</tbody>
</table>

## See Also

- **Reference**
  - `ReaderWriterLocker<sup>T</sup>State Class`
  - `W.Threading.Lockers Namespace`

Copyright @ 2018 Jordan Duerksen
Tungsten

W
ReaderWriterLocker

Field

The ReaderWriterLocker used to access the State

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```csharp
protected ReaderWriterLocker Locker
```

### Field Value

**Type:** ReaderWriterLocker

### See Also

**Reference**  
ReaderWriterLockerTState Class  
W.Threading.Lockers Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
ReaderWriterLocker

Field

The internal state

**Namespace:** W.Threading.Lockers

**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```
protected TState State
```

### Field Value

Type: *TState*

### See Also

Reference

ReaderWriterLockerTState Class

W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
ReaderWriterLockSlimExtensions Class

Extensions to simplify locking with ReaderWriterLockSlim

Inheritance Hierarchy

```
System
    Object
    W.Threading.Lockers
        ReaderWriterLockSlimExtensions
```

Namespace: W.Threading.Lockers

Syntax

```
public static class ReaderWriterLockSlimExtensions
```

The ReaderWriterLockSlimExtensions type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="InLock" /> ReaderWriterLockSlim, LockTypeEnum, Action)</td>
<td>Performs the action in a lock</td>
</tr>
<tr>
<td><img src="image" alt="InLockT" /> ReaderWriterLockSlim, LockTypeEnum, FuncTType)</td>
<td>Performs the function in a lock</td>
</tr>
<tr>
<td><img src="image" alt="InLockAsync" /> ReaderWriterLockSlim, LockTypeEnum, Action)</td>
<td>Asynchronous performs the action</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| **InLockAsync**<br>
InLockAsync(ReaderWriterLockSlim, LockTypeEnum, Func<TType>) | Asynchronous performs the function in a lock.                               |
| **Lock**                                    | Enters a read or write lock on the ReaderWriterLockSlim.                    |
| **Unlock**                                  | Exits a read or write lock on the ReaderWriterLockSlim.                     |

**See Also**

Reference

W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
</tr>
</tbody>
</table>
ReaderWriterLockSlimExtensions Methods

The ReaderWriterLockSlimExtensions type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLock(ReaderWriterLockSlim, LockTypeEnum, Action)</td>
<td>Performs the action in a lock</td>
</tr>
<tr>
<td>InLockTType(ReaderWriterLockSlim, LockTypeEnum, Func&lt;TType&gt;)</td>
<td>Performs the function in a lock</td>
</tr>
<tr>
<td>InLockAsync(ReaderWriterLockSlim, LockTypeEnum, Action)</td>
<td>Asynchronous performs the action in a lock</td>
</tr>
<tr>
<td>InLockAsyncTType(ReaderWriterLockSlim, LockTypeEnum, Func&lt;TType&gt;)</td>
<td>Asynchronous performs the function in a lock</td>
</tr>
<tr>
<td>Lock</td>
<td>Enters a read or write lock on the ReaderWriterLockSlim</td>
</tr>
<tr>
<td>Unlock</td>
<td>Exits a read or write lock on the ReaderWriterLockSlim</td>
</tr>
</tbody>
</table>
See Also

Reference
ReaderWriterLockSlimExtensions Class
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
# ReaderWriterLockSlimExtensions.InLock Method

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLock(ReaderWriterLockSlim, LockTypeEnum, Action)</td>
<td>Performs the action in a lock</td>
</tr>
<tr>
<td>InLockT&lt;T&gt;(ReaderWriterLockSlim, LockTypeEnum, Func&lt;T&gt;)</td>
<td>Performs the function in a lock</td>
</tr>
</tbody>
</table>

## See Also

Reference
- ReaderWriterLockSlimExtensions Class
- W.Threading.Lockers Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
**ReaderWriterLockSlimExtensionsInLock Method (ReaderWriterLockSlim, LockTypeEnum, Action)**

Performs the action in a lock

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```csharp
public static void InLock(
    this ReaderWriterLockSlim this,
    LockTypeEnum lockType,
    Action action
)
```

### Parameters

- **this**
  - Type: `System.Threading.ReaderWriterLockSlim`
  - The ReaderWriterLockSlim to provide resource locking

- **lockType**
  - Type: `W.Threading.Lockers.LockTypeEnum`
  - The type of lock to obtain

- **action**
  - Type: `SystemAction`
  - The action to perform

### Usage Note
In Visual Basic and C#, you can call this method as an instance method on any object of type ReaderWriterLockSlim. When you use instance method syntax to call this method, omit the first parameter. For more information, see Extension Methods (Visual Basic) or Extension Methods (C# Programming Guide).

See Also

Reference
ReaderWriterLockSlimExtensions Class
InLock Overload
W.Threading.Lockers Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
ReaderWriterLockSlimExtensionsInLock Method (ReaderWriterLockSlim, LockTypeEnum, Func<TType>)

Performs the function in a lock

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```csharp
public static TType InLock<TType>(
    this ReaderWriterLockSlim this,
    LockTypeEnum lockType,
    Func<TType> func
)
```

**Parameters**

- **this**
  Type: System.Threading.ReaderWriterLockSlim  
The ReaderWriterLockSlim to provide resource locking

- **lockType**
  Type: W.Threading.Lockers.LockTypeEnum  
The type of lock to obtain

- **func**
  Type: System.Func<TType>  
The function to perform

**Type Parameters**
**TType**


**Return Value**

Type: **TType**


**Usage Note**

In Visual Basic and C#, you can call this method as an instance method on any object of type `ReaderWriterLockSlim`. When you use instance method syntax to call this method, omit the first parameter. For more information, see [Extension Methods (Visual Basic)](Extension Methods (Visual Basic)) or [Extension Methods (C# Programming Guide)](Extension Methods (C# Programming Guide)).

**See Also**

- Reference
  - `ReaderWriterLockSlimExtensions` Class
  - `InLock` Overload
  - `W.Threading.Lockers` Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
## ReaderWriterLockSlimExtensionsInLock Method

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![S] InLockAsync(ReaderWriterLockSlim, LockTypeEnum, Action)</td>
<td>Asynchronous performs the action in a lock</td>
</tr>
<tr>
<td>![S] InLockAsyncTType(ReaderWriterLockSlim, LockTypeEnum, FuncTType)</td>
<td>Asynchronous performs the function in a lock</td>
</tr>
</tbody>
</table>

### See Also

**Reference**
- ReaderWriterLockSlimExtensions Class
- W.Threading.Lockers Namespace

---

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
ReaderWriterLockSlimExtensionsInLockAsync Method (ReaderWriterLockSlim, LockTypeEnum, Action)

Asynchronously performs the action in a lock

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```csharp
public static Task InLockAsync(
    this ReaderWriterLockSlim this,
    LockTypeEnum lockType,
    Action action
)
```

### Parameters

**this**
- Type: `System.Threading.ReaderWriterLockSlim`
- The ReaderWriterLockSlim to provide resource locking

**lockType**
- Type: `W.Threading.Lockers.LockTypeEnum`
- The type of lock to obtain

**action**
- Type: `System.Action`
- The action to perform

### Return Value
Type: Task

Usage Note
In Visual Basic and C#, you can call this method as an instance method on any object of type ReaderWriterLockSlim. When you use instance method syntax to call this method, omit the first parameter. For more information, see Extension Methods (Visual Basic) or Extension Methods (C# Programming Guide).

See Also

Reference
ReaderWriterLockSlimExtensions Class
InLockAsync Overload
W.Threading.Lockers Namespace

Copyright © 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
</tr>
</tbody>
</table>
ReaderWriterLockSlimExtensionsInLock Method (ReaderWriterLockSlim, LockType, Func<TType>)

Asynchronously performs the function in a lock

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```csharp
public static Task<TType> InLockAsync<TType>(
    this ReaderWriterLockSlim this,
    LockTypeEnum lockType,
    Func<TType> func
)
```

### Parameters

- **this**
  - Type: System.Threading.ReaderWriterLockSlim  
  - The ReaderWriterLockSlim to provide resource locking

- **lockType**
  - Type: W.Threading.Lockers.LockTypeEnum  
  - The type of lock to obtain

- **func**
  - Type: System.Func<TType>  
  - The function to perform

### Type Parameters
**Type**


**Return Value**

Type: Task<TType>


**Usage Note**

In Visual Basic and C#, you can call this method as an instance method on any object of type ReaderWriterLockSlim. When you use instance method syntax to call this method, omit the first parameter. For more information, see Extension Methods (Visual Basic) or Extension Methods (C# Programming Guide).

**See Also**

Reference

ReaderWriterLockSlimExtensions Class
InLockAsync Overload
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
ReaderWriterLockSlimExtensionsLock Method

Enters a read or write lock on the ReaderWriterLockSlim

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```csharp
public static void Lock(
    this ReaderWriterLockSlim this,
    LockTypeEnum lockType
)
```

### Parameters

- **this**
  - Type: System.Threading.ReaderWriterLockSlim
  - The object to provide resource locking

- **lockType**
  - Type: W.Threading.Lockers.LockTypeEnum
  - The type of lock to enter

### Usage Note

In Visual Basic and C#, you can call this method as an instance method on any object of type ReaderWriterLockSlim. When you use instance method syntax to call this method, omit the first parameter. For more information, see Extension Methods (Visual Basic) or Extension Methods (C# Programming Guide).
See Also

Reference
ReaderWriterLockSlimExtensions Class
W.Threading.Lockers Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
ReaderWriterLockSlimExtensionsUnlock Method

Exits a read or write lock on the ReaderWriterLockSlim

**Namespace:**  W.Threading.Lockers  
**Assembly:**  Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll)  Version: 2.0.1

### Syntax

```csharp
public static void Unlock(
    this ReaderWriterLockSlim this,
    LockTypeEnum lockType
)
```

### Parameters

- **this**
  
  Type:  System.Threading.ReaderWriterLockSlim  
  The object to provide resource locking

- **lockType**
  
  Type:  W.Threading.Lockers.LockTypeEnum  
  The type of lock to exit

### Usage Note

In Visual Basic and C#, you can call this method as an instance method on any object of type ReaderWriterLockSlim. When you use instance method syntax to call this method, omit the first parameter. For more information, see Extension Methods (Visual Basic) or Extension Methods (C# Programming Guide).
See Also

Reference

ReaderWriterLockSlimExtensions Class
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( \text{W} \)
SemaphoreSlimExtensions Class

Extensions to simplify locking with SemaphoreSlim

Inheritance Hierarchy

- System
- Object
- W.Threading.Lockers
- SemaphoreSlimExtensions

Namespace: W.Threading.Lockers

Syntax

```csharp
public static class SemaphoreSlimExtensions
```

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLock(SemaphoreSlim, Action)</td>
<td>Performs the action in a lock</td>
</tr>
<tr>
<td>InLockT&lt;TType&gt;(SemaphoreSlim, Func&lt;TType&gt;)</td>
<td>Performs the action in a lock</td>
</tr>
<tr>
<td>InLockAsync(SemaphoreSlim, Action)</td>
<td>Asynchronously performs the action in a lock</td>
</tr>
<tr>
<td>InLockAsync&lt;TType&gt;(SemaphoreSlim, Func&lt;TType&gt;)</td>
<td>Asynchronously performs the action in a lock</td>
</tr>
<tr>
<td>FuncTType()</td>
<td>performs the function in a lock</td>
</tr>
<tr>
<td>------------</td>
<td>--------------------------------</td>
</tr>
</tbody>
</table>

See Also

Reference

W.Threading.Lockers Namespace
Tungsten

$W$
SemaphoreSlimExtensions

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLock(SemaphoreSlim, Action)</td>
<td>Performs the action in a lock</td>
</tr>
<tr>
<td>InLockTType(SemaphoreSlim, FuncTType)</td>
<td>Performs the action in a lock</td>
</tr>
<tr>
<td>InLockAsync(SemaphoreSlim, Action)</td>
<td>Asynchronously performs the action in a lock</td>
</tr>
<tr>
<td>InLockAsyncTType(SemaphoreSlim, FuncTType)</td>
<td>Asynchronously performs the function in a lock</td>
</tr>
</tbody>
</table>

See Also

Reference
SemaphoreSlimExtensions Class
W.Threading.Lockers Namespace

Copyright © 2018 Jordan Duerksen
| Tungsten | $W$ |
SemaphoreSlimExtensions\n
InLock Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLock(SemaphoreSlim, Action)</td>
<td>Performs the action in a lock</td>
</tr>
<tr>
<td>InLockTType(SemaphoreSlim, FuncTType)</td>
<td>Performs the action in a lock</td>
</tr>
</tbody>
</table>

See Also

Reference
SemaphoreSlimExtensions Class
W.Threading.Lockers Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

W
SemaphoreSlimExtensionsInLock Method (SemaphoreSlim, Action)

Performs the action in a lock

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

## Syntax

```csharp
public static void InLock(
    this SemaphoreSlim this,
    Action action
)
```

### Parameters

**this**  
Type: `System.ThreadingSemaphoreSlim`  
The SemaphoreSlim to provide resource locking  

**action**  
Type: `SystemAction`  
The action to perform

### Usage Note
In Visual Basic and C#, you can call this method as an instance method on any object of type `SemaphoreSlim`. When you use instance method syntax to call this method, omit the first parameter. For more information, see Extension Methods (Visual Basic) or Extension Methods (C# Programming Guide).
See Also

Reference
SemaphoreSlimExtensions Class
InLock Overload
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
SemaphoreSlimExtensions.InLock<TType>

Method (SemaphoreSlim, Func<TType>)

Performs the action in a lock

**Namespace:** W.Threading.Lockers

**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

**Syntax**

```csharp
public static TType InLock<TType>(
    this SemaphoreSlim this,
    Func<TType> func
)
```

**Parameters**

**this**

Type: `System.Threading.SemaphoreSlim`

The SemaphoreSlim to provide resource locking

**func**

Type: `System.Func<TType>`

The function to perform

**Type Parameters**

**TType**

[Missing <typeparam name="TType"/> documentation for "M:W.Threading.Lockers.SemaphoreSlimExtensions.InLock`1(System.Threading...

**Return Value**
Type: \textit{TTy}\text{pe}

Usage Note

In Visual Basic and C#, you can call this method as an instance method on any object of type \textit{SemaphoreSlim}. When you use instance method syntax to call this method, omit the first parameter. For more information, see \textit{Extension Methods (Visual Basic)} or \textit{Extension Methods (C# Programming Guide)}.

\textbf{See Also}

Reference

\begin{itemize}
  \item \textit{SemaphoreSlimExtensions Class}
  \item \textit{InLock Overload}
  \item \textit{W.Threading.Lockers Namespace}
\end{itemize}
### Tungsten

$W$
SemaphoreSlimExtensions.InLockAsync Method

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLockAsync(SemaphoreSlim, Action)</td>
<td>Asynchronously performs the action in a lock</td>
</tr>
<tr>
<td>InLockAsyncTTType(SemaphoreSlim, FuncTTType)</td>
<td>Asynchronously performs the function in a lock</td>
</tr>
</tbody>
</table>

## See Also

**Reference**
- SemaphoreSlimExtensions Class
- W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\(W\)
SemaphoreSlimExtensions

Method (SemaphoreSlim, Action)

Asynchronously performs the action in a lock

**Namespace:**  W.Threading.Lockers

**Assembly:**  Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

**Syntax**

```csharp
public static Task InLockAsync(
    this SemaphoreSlim this, 
    Action action
)
```

**Parameters**

- **this**
  - Type: `System.Threading.SemaphoreSlim`
  - The SemaphoreSlim to provide resource locking

- **action**
  - Type: `System.Action`
  - The action to perform

**Return Value**

Type: `Task`

**Usage Note**

In Visual Basic and C#, you can call this method as an instance method on any object of type `SemaphoreSlim`. When you use
instance method syntax to call this method, omit the first parameter. For more information, see Extension Methods (Visual Basic) or Extension Methods (C# Programming Guide).

See Also

Reference
SemaphoreSlimExtensions Class
InLockAsync Overload
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
| Tungsten |
|----------|---|
| W        |
SemaphoreSlimExtensions.InLockAsync Method (SemaphoreSlim, Func<TType>)

Asynchronously performs the function in a lock

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

**Syntax**

```csharp
public static Task<TType> InLockAsync<TType>(
    SemaphoreSlim this,
    Func<TType> func
)
```

**Parameters**

- **this**
  
  Type: System.Threading.SemaphoreSlim  
  The SemaphoreSlim to provide resource locking

- **func**
  
  Type: System.Func<TType>  
  The function to perform

**Type Parameters**

- **TType**


**Return Value**
Type: `Task<TType>`


Usage Note
In Visual Basic and C#, you can call this method as an instance method on any object of type `SemaphoreSlim`. When you use instance method syntax to call this method, omit the first parameter. For more information, see Extension Methods (Visual Basic) or Extension Methods (C# Programming Guide).

See Also

Reference
SemaphoreSlimExtensions Class
InLockAsync Overload
W.Threading.Lockers Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
SemaphoreSlimLocker Class

Uses SemaphoreSlim to provide resource locking

Inheritance Hierarchy

- System
  - Object
    - W.Threading.Lockers
      - SemaphoreSlimLocker

Namespace: W.Threading.Lockers

Syntax

```csharp
public class SemaphoreSlimLocker : ILocker<SemaphoreSlimLocker, IDisposable
```

The SemaphoreSlimLocker type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SemaphoreSlimLocker</td>
<td>Constructs a new SemaphoreSlimLocker with an initial request count of 1 and maximum request count of 1</td>
</tr>
<tr>
<td>SemaphoreSlimLocker(Int32)</td>
<td>Constructs a new SemaphoreSlimLocker</td>
</tr>
</tbody>
</table>
SemaphoreSlimLocker(Int32, Int32)

Constructs a new SemaphoreSlimLocker

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locker</td>
<td>The SemaphoreSlim used to perform locks</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispose</td>
<td>Disposes the SemaphoreSlimLocker and releases resources</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup</td>
</tr>
<tr>
<td></td>
<td>operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from Object.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Executes an action from within a SemaphoreSlim</td>
</tr>
<tr>
<td>InLockTValue(FuncTValue)</td>
<td>Executes a function from within a SemaphoreSlim</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Executes an action from within a SemaphoreSlim</td>
</tr>
<tr>
<td>InLockAsyncTValue(FuncTValue)</td>
<td>Executes a function from within a SemaphoreSlim</td>
</tr>
<tr>
<td>Lock</td>
<td>Locks the resource</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object.</td>
</tr>
</tbody>
</table>
Unlock

Unlocks the resource

## Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTType(FuncTType)</td>
<td>Overloaded. Performs the function in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTT(TTTT)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>
See Also

Reference

W Threading Lockers Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

W
# SemaphoreSlimLocker Constructor

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SemaphoreSlimLocker</td>
<td>Constructs a new SemaphoreSlimLocker with an initial request count of 1 and maximum request count of 1</td>
</tr>
<tr>
<td>SemaphoreSlimLocker(Int32)</td>
<td>Constructs a new SemaphoreSlimLocker</td>
</tr>
<tr>
<td>SemaphoreSlimLocker(Int32, Int32)</td>
<td>Constructs a new SemaphoreSlimLocker</td>
</tr>
</tbody>
</table>

## See Also

**Reference**

- SemaphoreSlimLocker Class
- W.Threading.Lockers Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

W
SemaphoreSlimLocker Constructor

Constructs a new SemaphoreSlimLocker with an initial request count of 1 and maximum request count of 1

Namespace: W.Threading.Lockers

Syntax

```
public SemaphoreSlimLocker()
```

See Also

Reference

SemaphoreSlimLocker Class
SemaphoreSlimLocker Overload
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
SemaphoreSlimLocker Constructor (Int32)

Constructs a new SemaphoreSlimLocker

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```csharp
public SemaphoreSlimLocker(
    int initialCount
)
```

### Parameters

*`initialCount`*

Type: `System.Int32`  
The initial number of requests that the semaphore can grant concurrently

### See Also

- Reference
  - SemaphoreSlimLocker Class
  - SemaphoreSlimLocker Overload
  - W.Threading.Lockers Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$^W$
SemaphoreSlimLocker Constructor (Int32, Int32)

Constructs a new SemaphoreSlimLocker

**Namespace:**  W.Threading.Lockers  
**Assembly:**  Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```csharp
public SemaphoreSlimLocker(
    int initialCount,
    int maxCount
)
```

### Parameters

- **initialCount**
  - Type: `SystemInt32`
  - The initial number of requests that the semaphore can grant concurrently

- **maxCount**
  - Type: `SystemInt32`
  - The maximum number of requests that can be granted concurrently

### See Also

- Reference
  - SemaphoreSlimLocker Class
  - SemaphoreSlimLocker Overload
Tungsten

W
SemaphoreSlimLocker Properties

The SemaphoreSlimLocker type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locker</td>
<td>The SemaphoreSlim used to perform locks</td>
</tr>
</tbody>
</table>

See Also

Reference

SemaphoreSlimLocker Class

W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
SemaphoreSlimLocker

Property

The SemaphoreSlim used to perform locks

**Namespace:** W.Threading.Lockers
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

Syntax

```csharp
public SemaphoreSlim Locker { get; }
```

Property Value

Type: SemaphoreSlim

Implements

ILocker<TLockerLocker>

See Also

Reference

SemaphoreSlimLocker Class
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$^\text{W}$
SemaphoreSlimLocker Methods

The `SemaphoreSlimLocker` type exposes the following members.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dispose</td>
<td>Disposes the <code>SemaphoreSlimLocker</code> and releases resources</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup</td>
</tr>
<tr>
<td></td>
<td>operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <code>Type</code> of the</td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLock(Action)</td>
<td>Executes an action from within a SemaphoreSlim</td>
</tr>
<tr>
<td>InLockTValue(FuncTValue)</td>
<td>Executes a function from within a SemaphoreSlim</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Executes an action from within a SemaphoreSlim</td>
</tr>
<tr>
<td>InLockAsyncTValue(FuncTValue)</td>
<td>Executes a function from within a SemaphoreSlim</td>
</tr>
<tr>
<td>Lock</td>
<td>Locks the resource</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object. (Inherited from Object.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Unlocks the resource</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTType(FuncTType)</td>
<td>Overloaded. Performs the function in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock. (Defined by</td>
</tr>
</tbody>
</table>
### MonitorExtensions.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLockAsync&lt;TType&gt;(Func&lt;TType&gt;)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>

### See Also

**Reference**
- SemaphoreSlimLocker Class
- W.Threading.Lockers Namespace
**Tungsten**

$W$
SemaphoreSlimLockerDispose Method

Disposes the SemaphoreSlimLocker and releases resources

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```csharp
public void Dispose()
```

Implements  
`IDisposable.Dispose`

### See Also

Reference  
SemaphoreSlimLocker Class  
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
SemaphoreSlimLocker\n
**InLock**

**Overload List**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLock(Action)</td>
<td>Executes an action from within a SemaphoreSlim</td>
</tr>
<tr>
<td>InLockTValue(FuncTValue)</td>
<td>Executes a function from within a SemaphoreSlim</td>
</tr>
</tbody>
</table>

**See Also**

Reference
SemaphoreSlimLocker Class
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
SemaphoreSlimLocker.InLock Method (Action)

Executes an action from within a SemaphoreSlim

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```csharp
public void InLock(
    Action action
)
```

### Parameters

**action**  
Type: System.Action  
The action to run

### Implements

ILockerInLock(Action)

### See Also

Reference  
SemaphoreSlimLocker Class  
InLock Overload  
W.Threading.Lockers Namespace

---

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
SemaphoreSlimLocker

InLock

Method (Func<TValue>)

Executes a function from within a SemaphoreSlim

**Namespace:**  W.Threading.Lockers
**Assembly:**  Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

⚠️ Syntax

```csharp
public TValue InLock<TValue>(
    Func<TValue> func
)
```

**Parameters**

*func*

Type:  `System.Func<TValue>`
The function to run

**Type Parameters**

*TValue*

The type of return value

**Return Value**

Type:  `TValue`
The result of the function call (a value of type TValue)

⚠️ See Also

Reference
Tungsten

W
SemaphoreSlimLocker

InLockAsync Method

![Overload List](image)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>InLockAsync(Action)</code></td>
<td>Executes an action from within a SemaphoreSlim</td>
</tr>
<tr>
<td><code>InLockAsyncTValue(FuncTValue)</code></td>
<td>Executes a function from within a SemaphoreSlim</td>
</tr>
</tbody>
</table>

See Also

Reference

SemaphoreSlimLocker Class

W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( W \)
SemaphoreSlimLocker

SemaphoreSlimLocker.InLockAsync Method (Action)

Executes an action from within a SemaphoreSlim

**Namespace:**  W.Threading.Lockers  
**Assembly:**  Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```csharp
public Task InLockAsync(
    Action action
)
```

### Parameters

**action**
Type:  SystemAction  
The action to run

### Return Value

Type:  Task  

### Implements

ILockerInLockAsync(Action)

### See Also

Reference
SemaphoreSlimLocker Class
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
</tr>
</tbody>
</table>
SemaphoreSlimLockerInLockAsync Method (Func<TValue>)

Executes a function from within a SemaphoreSlim

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```csharp
public Task<TValue> InLockAsync<TValue>(
    Func<TValue> func
)
```

### Parameters

**func**

Type: System.Func<TValue>

The function to run

### Type Parameters

**TValue**

The type of return value

### Return Value

Type: Task<TValue>

The result of the function call (a value of type TValue)

### See Also

Reference
SemaphoreSlimLocker Class
InLockAsync Overload
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
SemaphoreSlimLocker

Method

Locks the resource

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```csharp
public void Lock()
```

### See Also

**Reference**  
SemaphoreSlimLocker Class  
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
SemaphoreSlimLockerUnlock Method

Unlocks the resource

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```csharp
public void Unlock()
```

### See Also

Reference
- SemaphoreSlimLocker Class
- W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
| Tungsten | $W$ |
SemaphoreSlimLocker<TState>

Class

Extends SemaphoreSlimLocker with an internal state variable

Inheritance Hierarchy

- `System.Object`
- `W.Threading.Lockers.StateLockerSemaphoreSlimLocker<TState>`
- `W.Threading.LockersSemaphoreSlimLocker<TState>`

Namespace: `W.Threading.Lockers`  
Version: 2.0.1

Syntax

```csharp
public class SemaphoreSlimLocker<TState> : StateLocker
```

Type Parameters

- `TState`
  - The state Type

The `SemaphoreSlimLocker<TState>` type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>SemaphoreSlimLocker&lt;TState&gt;</code></td>
<td>Initializes a new instance of</td>
</tr>
</tbody>
</table>
the SemaphoreSlimLockerTState class

## Top

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locker</td>
<td>The locking mechanism (SpinLock, Monitor, SemaphoreSlim, ReaderWriterLock)</td>
</tr>
<tr>
<td></td>
<td>(Inherited from StateLockerTLocker, TState.)</td>
</tr>
</tbody>
</table>

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup</td>
</tr>
<tr>
<td></td>
<td>operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from Object.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>GetHashCode</code></td>
<td>Serves as the default hash function. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><code>GetType</code></td>
<td>Gets the <code>Type</code> of the current instance. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><code>InLock(Action)</code></td>
<td>Performs an action from within a lock (Inherited from <code>StateLocker&lt;TLocker, TState&gt;</code>.)</td>
</tr>
<tr>
<td><code>InLock(Action&lt;TState&gt;)</code></td>
<td>Performs an action from within a lock, passing in the current state (Inherited from <code>StateLocker&lt;TLocker, TState&gt;</code>.)</td>
</tr>
<tr>
<td><code>InLock(Func&lt;TState, TState&gt;)</code></td>
<td>Performs a function from within a lock, passing in the current state and assigning the state to the function result (Inherited from <code>StateLocker&lt;TLocker, TState&gt;</code>.)</td>
</tr>
<tr>
<td><code>InLockTResult(Func&lt;TResult&gt;)</code></td>
<td>Performs a function</td>
</tr>
</tbody>
</table>
InLockAsync(Action)

Asynchronously performs an action from within a lock (Inherited from StateLocker<TLocker, TState>.)

InLockAsync(Action<TState>)

Asynchronously performs an action from within a lock, passing in the current state (Inherited from StateLocker<TLocker, TState>.)

InLockAsync(Func<TState, TState>)

Asynchronously performs a function from within a lock, passing in the current state and assigning the state to the function result (Inherited from StateLocker<TLocker, TState>.)

InLockAsync<TResult(Func<TResult>)>

Asynchronously performs a function from within a lock (Inherited from StateLocker<TLocker, TState>.)
MemberwiseClone Creates a shallow copy of the current Object. (Inherited from Object.)

ToString Returns a string that represents the current object. (Inherited from Object.)

Top

Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
<td>The internal state (Inherited from StateLockerTLocker, TState.)</td>
</tr>
</tbody>
</table>

Top

Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method</td>
</tr>
</tbody>
</table>
should be called in the constructor of any class which has I OwnedProperty members
(Defined by PropertyHostExtensions.)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
</table>
| InLock(Action)                | Overloaded. Performs the action in a Monitor lock
(Defined by MonitorExtensions.) |
| InLockTTType(FuncTTType)      | Overloaded. Performs the function in a Monitor lock
(Defined by MonitorExtensions.) |
| InLockAsync(Action)           | Overloaded. Asynchronously performs the action in a Monitor lock
(Defined by MonitorExtensions.) |
| InLockAsyncTTType(FuncTTType) | Overloaded. Asynchronously performs the action in a Monitor lock
(Defined by MonitorExtensions.) |
| IsDirty                       | Scans the IsDirty value of each field and property of type IProperty
(Defined by PropertyHostExtensions.) |
<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Lock</strong></td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><strong>MarkAsClean</strong></td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td><strong>Unlock</strong></td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>

### Remarks

Same as StateLocker<SemaphoreSlimLocker, TState>

### See Also

Reference

W.Threading.Lockers Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

\( W \)
SemaphoreSlimLockerTState Constructor

Initializes a new instance of the SemaphoreSlimLockerTState class

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```csharp
public SemaphoreSlimLocker()
```

### See Also

**Reference**  
SemaphoreSlimLockerTState Class  
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
SemaphoreSlimLocker<TState>

Properties

The `SemaphoreSlimLocker<TState>` generic type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locker</td>
<td>The locking mechanism (SpinLock, Monitor, SemaphoreSlim, ReaderWriterLock)</td>
</tr>
<tr>
<td></td>
<td>(Inherited from <code>StateLocker&lt;TLocker, TState&gt;</code>)</td>
</tr>
</tbody>
</table>

See Also

Reference

- [SemaphoreSlimLocker<TState> Class](#)
- [W.Threading.Lockers Namespace](#)

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
SemaphoreSlimLocker<TState> Methods

The `SemaphoreSlimLocker<TState>` generic type exposes the following members.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the <code>Type</code> of the</td>
</tr>
<tr>
<td><strong>InLock</strong>&lt;br&gt;(Action)</td>
<td>Performs an action from within a lock (Inherited from StateLocker&lt;TLock, TState&gt;.)</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>InLock</strong>&lt;br&gt;(Action&lt;TState&gt;)</td>
<td>Performs an action from within a lock, passing in the current state (Inherited from StateLocker&lt;TLock, TState&gt;.)</td>
</tr>
<tr>
<td><strong>InLock</strong>&lt;br&gt;(Func&lt;TState, TState&gt;)</td>
<td>Performs a function from within a lock, passing in the current state and assigning the state to the function result (Inherited from StateLocker&lt;TLock, TState&gt;.)</td>
</tr>
<tr>
<td><strong>InLock&lt;TResult&gt;</strong>&lt;br&gt;(Func&lt;TResult&gt;)</td>
<td>Performs a function from within a lock (Inherited from StateLocker&lt;TLock, TState&gt;.)</td>
</tr>
<tr>
<td><strong>InLockAsync</strong>&lt;br&gt;(Action)</td>
<td>Asynchronously performs an action from within a lock (Inherited from</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>InLockAsync(Action&lt;TState&gt;)</strong></td>
<td>Asynchronously performs an action from within a lock, passing in the current state (Inherited from StateLocker&lt;TLocker, TState&gt;.)</td>
</tr>
<tr>
<td><strong>InLockAsync(Func&lt;TState, TState&gt;)</strong></td>
<td>Asynchronously performs a function from within a lock, passing in the current state and assigning the state to the function result (Inherited from StateLocker&lt;TLocker, TState&gt;.)</td>
</tr>
<tr>
<td><strong>InLockAsync&lt;TResult(Func&lt;TResult&gt;)&gt;</strong></td>
<td>Asynchronously performs a function from within a lock (Inherited from StateLocker&lt;TLocker, TState&gt;.)</td>
</tr>
<tr>
<td><strong>MemberwiseClone</strong></td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td><strong>ToString</strong></td>
<td>Returns a string that represents the</td>
</tr>
</tbody>
</table>
current object. (Inherited from Object.)

### Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AsTType</code></td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td><code>InitializeProperties</code></td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td><code>InLock(Action)</code></td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><code>InLockTType(FuncTType)</code></td>
<td>Overloaded. Performs the function in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTTType(FuncTTType)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>
See Also

Reference
SemaphoreSlimLocker
TState Class
W.Threading.Lockers Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

W
SemaphoreSlimLocker<TState>

Fields

The SemaphoreSlimLocker<TState> generic type exposes the following members.

Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
<td>The internal state (Inherited from StateLocker&lt;TLock, TState&gt;.)</td>
</tr>
</tbody>
</table>

See Also

Reference
SemaphoreSlimLocker<TState> Class
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
SpinLocker Class

Uses SpinLock to provide resource locking

▲ Inheritance Hierarchy

SystemObject W.Threading.LockersSpinLocker

Namespace: W.Threading.Lockers

▲ Syntax

```csharp
public class SpinLocker : ILocker<SpinLock>, ILocker
```

The SpinLocker type exposes the following members.

▲ Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SpinLocker</td>
<td>Initializes a new instance of the SpinLocker class</td>
</tr>
</tbody>
</table>

▲ Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locker</td>
<td>The SpinLock used to perform locks</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td><em>(Inherited from Object.)</em></td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. <em>(Inherited from Object.)</em></td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. <em>(Inherited from Object.)</em></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Performs an action from within a SpinLock</td>
</tr>
<tr>
<td>InLockAsyncResult(Func&lt;TResult&gt;)</td>
<td>Performs a function from within a SpinLock</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Performs an action from within a SpinLock</td>
</tr>
<tr>
<td>InLockAsyncAsyncResult(Func&lt;TResult&gt;)</td>
<td>Performs a function from within a SpinLock</td>
</tr>
<tr>
<td>Lock</td>
<td>Locks the resource</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a</td>
</tr>
</tbody>
</table>
string that represents the current object.  
(Inherited from Object.)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unlock</td>
<td>Unlocks the resource</td>
</tr>
</tbody>
</table>

### Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| AsTType                   | Use Generic syntax for the as operator.  
(Defined by AsExtensions.)|
| InitializeProperties      | Scans the fields and properties of "owner" and sets the member's Owner property to  
"owner"  
This method should be called in the constructor of any class which has  
IOwnedProperty members  
(Defined by PropertyHostExtensions.)|
| InLock(Action)            | Overloaded.  
Performs the action in a Monitor lock  
(Defined by MonitorExtensions.)|
<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLockTTType(FuncTTType)</td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTTType(FuncTTType)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock</td>
</tr>
</tbody>
</table>
Remarks

Can be overridden to provide additional functionality

See Also

Reference

W.Threading.Lockers Namespace
Tungsten

$W$
SpinLocker Constructor

Initializes a new instance of the SpinLocker class

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

## Syntax

```c#
public SpinLocker()
```

## See Also

**Reference**  
SpinLocker Class  
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
SpinLocker Properties

The SpinLocker type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locker</td>
<td>The SpinLock used to perform locks</td>
</tr>
</tbody>
</table>

See Also

Reference

SpinLocker Class
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
SpinLockerLocker Property

The SpinLock used to perform locks

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```csharp
public SpinLock Locker { get; }
```

### Property Value

Type: SpinLock

Implements

ILocker TLocker

### See Also

Reference

SpinLocker Class

W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
SpinLocker Methods

The SpinLocker type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td>finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td>gethashcode</td>
<td>Serves as the default</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from Object.)</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Performs an action from within a SpinLock</td>
</tr>
<tr>
<td>InLockAsyncResult(FuncTResult)</td>
<td>Performs a function from within a SpinLock</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Performs an action from within a SpinLock</td>
</tr>
<tr>
<td>InLockAsyncResult(FuncTResult)</td>
<td>Performs a function from within a SpinLock</td>
</tr>
<tr>
<td>Lock</td>
<td>Locks the resource</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current hash function. (Inherited from Object.)</td>
</tr>
</tbody>
</table>
**ToString**

Returns a string that represents the current object. (Inherited from `Object`.)

**Unlock**

Unlocks the resource

---

**Extension Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="" /> AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by <code>AsExtensions</code>.)</td>
</tr>
<tr>
<td><img src="image2.png" alt="" /> InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by <code>PropertyHostExtensions</code>.)</td>
</tr>
<tr>
<td><img src="image3.png" alt="" /> InLock(Action)</td>
<td>Overloaded.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>InLockTTtype(FuncTTtype)</code></td>
<td>Performs the action in a Monitor lock (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>InLockAsync(TAction)</code></td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>InLockAsyncTTtype(FuncTTtype)</code></td>
<td>Asynchronously performs the action in a Monitor lock (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>IsDirty</code></td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by <code>PropertyHostExtensions</code>.)</td>
</tr>
<tr>
<td><code>Lock</code></td>
<td>Performs a Monitor lock (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>MarkAsClean</code></td>
<td>Scans each field and property of type IProperty and sets it’s IsDirty flag to false</td>
</tr>
</tbody>
</table>
Unlock

Performs a Monitor unlock
(Defined by MonitorExtensions.)

See Also

Reference
SpinLocker Class
W.Threading.Lockers Namespace
Tungsten

$W$
SpinLocker InLock Method

**Overload List**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLock&lt;Action&gt;</td>
<td>Performs an action from within a SpinLock</td>
</tr>
<tr>
<td>InLock&lt;TResult&lt;Func&lt;TResult)&gt;</td>
<td>Performs a function from within a SpinLock</td>
</tr>
</tbody>
</table>

**See Also**

Reference
SpinLocker Class
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
SpinLockerInLock Method (Action)

Performs an action from within a SpinLock

**Namespace:**  W.Threading.Lockers  
**Assembly:**  Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```csharp
public void InLock(
    Action action
)
```

### Parameters

- **action**
  - Type: SystemAction
  - The action to run

### See Also

- Reference
  - SpinLocker Class
  - InLock Overload
  - W.Threading.Lockers Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
SpinLockerInLock\textit{TResult} Method (Func\textit{TResult})

Performs a function from within a SpinLock

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```c#
public TResult InLock\textless T\text{Result}\textgreater (Func\textless T\text{Result}\textgreater func)
```

### Parameters

- **func**
  - Type: System.Func\textless T\text{Result}\textgreater
  - The function to run

### Type Parameters

- **TResult**
  - The type of return value

### Return Value

- Type: TResult
  - The result of the function call (a value of type TValue)

### Implements

ILockerInLock\textit{TResult}(Func\textit{TResult})
See Also

Reference
SpinLocker Class
InLock Overload
W.Threading.Lockers Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
SpinLocker

InLockAsync Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLockAsync(Action)</td>
<td>Performs an action from within a SpinLock</td>
</tr>
<tr>
<td>InLockAsyncTResult(FuncTResult)</td>
<td>Performs a function from within a SpinLock</td>
</tr>
</tbody>
</table>

See Also

Reference
SpinLocker Class
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
SpinLockerInLockAsync Method (Action)

Performs an action from within a SpinLock

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```csharp
public Task InLockAsync(
    Action action
)
```

### Parameters

**action**
- Type: SystemAction  
  The action to run

### Return Value

Type: Task

**Missing <returns> documentation for**


### Implements

ILockerInLockAsync(Action)

### See Also

Reference  
SpinLocker Class
Tungsten

W
SpinLockerInLockAsync<TResult> Method (Func<TResult>)

Performs a function from within a SpinLock

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```csharp
public Task<TResult> InLockAsync<TResult>(
    Func<TResult> func
)
```

### Parameters

**func**  
Type: `System.Func<TResult>`  
The function to run

### Type Parameters

**TResult**  
The type of return value

### Return Value

Type: `Task<TResult>`  
The result of the function call (a value of type TValue)

### Implements

`ILockerInLockAsync<TResult>(Func<TResult>)`
See Also

Reference
SpinLocker Class
InLockAsync Overload
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
SpinLockerLock Method

Locks the resource

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```csharp
public void Lock(
    ref bool lockTaken
)
```

**Parameters**

- **lockTaken**
  - Type: `SystemBoolean`
  - [Missing <param name="lockTaken"/> documentation for "M:W.Threading.Lockers.SpinLocker.Lock(System.Boolean@)"]

### See Also

**Reference**
- SpinLocker Class
- W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
SpinLockerUnlock Method

Unlocks the resource

**Namespace:**  W.Threading.Lockers  
**Assembly:**  Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll)  
**Version:**  2.0.1

Syntax

```csharp
public void Unlock(
    bool useMemberBarrier = false
)
```

Parameters

**useMemberBarrier (Optional)**

Type:  System.Boolean

[Missing <param name="useMemberBarrier"/> documentation for "M:W.Threading.Lockers.SpinLocker.Unlock(System.Boolean)""]

See Also

Reference

SpinLocker Class  
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
SpinLocker\textit{\textit{TState}} Class

Extends SpinLocker with an internal state variable

\section*{Inheritance Hierarchy}

\begin{itemize}
  \item System\object ~ W.Threading.Lockers\textit{StateLockerSpinLocker}, \textit{TState}
  \item W.Threading.Lockers\textit{SpinLockerTState}
\end{itemize}

\textbf{Namespace:} W.Threading.Lockers


\section*{Syntax}

\texttt{
\begin{Verbatim}
public class SpinLocker\textless TState\textgreater : StateLocker\textless SpinLocker\textgreater \\
\end{Verbatim}
}

\textbf{Type Parameters}

\textit{TState}

The state Type

The \texttt{SpinLocker\textit{TState}} type exposes the following members.

\section*{Constructors}

\begin{tabular}{|l|l|}
\hline
\textbf{Name} & \textbf{Description} \\
\hline
- & \texttt{SpinLocker\textit{TState}} \quad \textit{Initializes a new instance of the \texttt{SpinLocker\textit{TState}} class} \\
\hline
\end{tabular}
## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Image" /></td>
<td><strong>Locker</strong></td>
</tr>
</tbody>
</table>

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Image" /></td>
<td><strong>Equals</strong></td>
</tr>
<tr>
<td><img src="image" alt="Image" /></td>
<td><strong>Finalize</strong></td>
</tr>
<tr>
<td><img src="image" alt="Image" /></td>
<td><strong>GetHashCode</strong></td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Get Type</strong></td>
<td>Gets the <strong>Type</strong> of the current instance. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td><strong>InLock(Action)</strong></td>
<td>Performs an action from within a lock (Inherited from <strong>StateLocker</strong>,<strong>TLocker,</strong>,<strong>TState</strong>.)</td>
</tr>
<tr>
<td><strong>InLock(Action,</strong> <strong>TState</strong>)</td>
<td>Performs an action from within a lock, passing in the current state (Inherited from <strong>StateLocker</strong>,<strong>TLocker,</strong>,<strong>TState</strong>.)</td>
</tr>
<tr>
<td><strong>InLock(Func,</strong> <strong>TState,</strong> <strong>TState</strong>)</td>
<td>Performs a function from within a lock, passing in the current state and assigning the state to the function result (Inherited from <strong>StateLocker</strong>,<strong>TLocker,</strong>,<strong>TState</strong>.)</td>
</tr>
<tr>
<td><strong>InLockTResult(Func,</strong> <strong>TResult</strong>)</td>
<td>Performs a function from within a lock (Inherited from <strong>StateLocker</strong>,<strong>TLocker,</strong>,<strong>TState</strong>.)</td>
</tr>
<tr>
<td><strong>InLockAsync(Action)</strong></td>
<td>Asynchronously</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>InLockAsync(Action&lt;TState&gt;)</td>
<td>Asynchronously performs an action from within a lock, passing in the current state (Inherited from StateLocker&lt;TLocker, TState&gt;).</td>
</tr>
<tr>
<td>InLockAsync(Func&lt;TState, TState&gt;)</td>
<td>Asynchronously performs a function from within a lock, passing in the current state and assigning the state to the function result (Inherited from StateLocker&lt;TLocker, TState&gt;).</td>
</tr>
<tr>
<td>InLockAsync&lt;TResult&gt;(Func&lt;TResult&gt;)</td>
<td>Asynchronously performs a function from within a lock (Inherited from StateLocker&lt;TLocker, TState.).</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
</tbody>
</table>
**ToString**

Returns a string that represents the current object. (Inherited from **Object**.)

---

### Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>State</strong></td>
<td>The internal state (Inherited from <strong>StateLocker</strong>&lt;sup&gt;TLocker&lt;/sup&gt;, <strong>TState</strong>.)</td>
</tr>
</tbody>
</table>

---

### Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AsTType</strong></td>
<td>Use Generic syntax for the as operator. (Defined by <strong>AsExtensions</strong>.)</td>
</tr>
<tr>
<td><strong>InitializeProperties</strong></td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by <strong>PropertyHostExtensions</strong>.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTType(FuncTType)</td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsyncTType(FuncTType)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty</td>
</tr>
</tbody>
</table>
and sets its IsDirty flag to false
(Defined by PropertyHostExtensions.)

**Unlock**

Performs a Monitor unlock
(Defined by MonitorExtensions.)

Remarks

Same as StateLocker<SpinLocker;, TState>

See Also

Reference

W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

W
SpinLocker<TState> Constructor

Initializes a new instance of the SpinLocker<TState> class

**Namespace:**  W.Threading.Lockers  
**Assembly:**  Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

**Syntax**

```csharp
public SpinLocker()
```

**See Also**

Reference  
SpinLockerTState Class  
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
SpinLocker<TState> Properties

The SpinLocker<TState> generic type exposes the following members.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locker</td>
<td>The locking mechanism (SpinLock, Monitor, SemaphoreSlim, ReaderWriterLock) (Inherited from StateLockerTLocker, TState.)</td>
</tr>
</tbody>
</table>

See Also

Reference
SpinLockerTState Class
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\( \text{W} \)
SpinLocker\textit{TState} Methods

The \textit{SpinLockerTState} generic type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from \textit{Object}.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from \textit{Object}.)</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from \textit{Object}.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the \textbf{Type} of the current instance.</td>
</tr>
<tr>
<td></td>
<td>(Inherited from \textit{Object}.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>InLock(Action)</strong></td>
<td>Performs an action from within a lock (Inherited from StateLocker TLocker, TState.)</td>
</tr>
<tr>
<td><strong>InLock(ActionTState)</strong></td>
<td>Performs an action from within a lock, passing in the current state (Inherited from StateLocker TLocker, TState.)</td>
</tr>
<tr>
<td><strong>InLock(FuncTState, TState)</strong></td>
<td>Performs a function from within a lock, passing in the current state and assigning the state to the function result (Inherited from StateLocker TLocker, TState.)</td>
</tr>
<tr>
<td><strong>InLockTResult(FuncTResult)</strong></td>
<td>Performs a function from within a lock (Inherited from StateLocker TLocker, TState.)</td>
</tr>
<tr>
<td><strong>InLockAsync(Action)</strong></td>
<td>Asynchronously performs an action from within a lock (Inherited from StateLocker TLocker, TState.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>InLockAsync(Action&lt;TState&gt;)</td>
<td>Asynchronously performs an action from within a lock, passing in the current state (Inherited from StateLocker&lt;TLocker, TState&gt;.)</td>
</tr>
<tr>
<td>InLockAsync(Func&lt;TState, TState&gt;)</td>
<td>Asynchronously performs a function from within a lock, passing in the current state and assigning the state to the function result (Inherited from StateLocker&lt;TLocker, TState&gt;.)</td>
</tr>
<tr>
<td>InLockAsync&lt;TResult&gt;(Func&lt;TResult&gt;)</td>
<td>Asynchronously performs a function from within a lock (Inherited from StateLocker&lt;TLocker, TState&gt;.)</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
<tr>
<td>ToString</td>
<td>Returns a string that represents the current object. (Inherited from Object.)</td>
</tr>
</tbody>
</table>
## Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>as TType</td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>InLock(Action)</td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockTType(FuncTType)</td>
<td>Overloaded. Performs the function in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>InLockAsyncTTType(TType)</td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>

**See Also**

Reference

SpinLockerTState Class
Tungsten

$W$
# SpinLocker\textit{TState} Fields

The \textit{SpinLockerTState} generic type exposes the following members.

## Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
<td>The internal state (Inherited from \textit{StateLockerTLocker}, \textit{TState}.)</td>
</tr>
</tbody>
</table>

## See Also

Reference
- \textit{SpinLockerTState Class}
- \textit{W.Threading.Lockers Namespace}

Copyright © 2018 Jordan Duerksen
Tungsten

W
SpinLockExtensions Class

Extensions to simplify locking with SpinLock

Inheritance Hierarchy

- System
- Object
- W.Threading.Lockers
- SpinLockExtensions

Namespace: W.Threading.Lockers

Syntax

```csharp
public static class SpinLockExtensions
```

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLock(SpinLock, Action)</td>
<td>Performs the action in a lock</td>
</tr>
<tr>
<td>InLockTType(SpinLock, FuncTType)</td>
<td>Performs the function in a lock</td>
</tr>
<tr>
<td>InLockAsync(SpinLock, Action)</td>
<td>Asynchronously performs the action in a lock</td>
</tr>
<tr>
<td>InLockAsyncTType(SpinLock, FuncTType)</td>
<td>Asynchronously performs the function in a lock</td>
</tr>
</tbody>
</table>
See Also

Reference

W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$\text{W}$
SpinLockExtensions Methods

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLock(SpinLock, Action)</td>
<td>Performs the action in a lock</td>
</tr>
<tr>
<td>InLockTType(SpinLock, FuncTType)</td>
<td>Performs the function in a lock</td>
</tr>
<tr>
<td>InLockAsync(SpinLock, Action)</td>
<td>Asynchronously performs the action in a lock</td>
</tr>
<tr>
<td>InLockAsyncTType(SpinLock, FuncTType)</td>
<td>Asynchronously performs the function in a lock</td>
</tr>
</tbody>
</table>

See Also

Reference
SpinLockExtensions Class
W.Threading.Lockers Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
SpinLockExtensions InLock Method

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLock(SpinLock, Action)</td>
<td>Performs the action in a lock</td>
</tr>
<tr>
<td>InLockTType(SpinLock, FuncTType)</td>
<td>Performs the function in a lock</td>
</tr>
</tbody>
</table>

See Also

Reference
SpinLockExtensions Class
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
SpinLockExtensionsInLock
Method (SpinLock, Action)

Performs the action in a lock

**Namespace:**  W.Threading.Lockers  
**Assembly:**  Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```
public static void InLock(  
    this SpinLock this,  
    Action action
)
```

### Parameters

**this**
- Type: `System.ThreadingSpinLock`
  - The SpinLock to provide resource locking

**action**
- Type: `SystemAction`
  - The action to perform

### Usage Note

In Visual Basic and C#, you can call this method as an instance method on any object of type `SpinLock`. When you use instance method syntax to call this method, omit the first parameter. For more information, see Extension Methods (Visual Basic) or Extension Methods (C# Programming Guide).
See Also

Reference
- SpinLockExtensions Class
- InLock Overload
- W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$^W$
SpinLockExtensions.InLock<TType>
Method (SpinLock, Func<TType>)

Performs the function in a lock

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```csharp
public static TType InLock<TType>(
    this SpinLock this,
    Func<TType> func
)
```

### Parameters

- **this**
  - Type: `System.Threading.SpinLock`
  - The SpinLock to provide resource locking

- **func**
  - Type: `System.Func<TType>`
  - The function to perform

### Type Parameters

- **TType**

### Return Value
Type: `TType`

Usage Note

See Also

Reference
- `SpinLockExtensions Class`
- `InLock Overload`
- `W.Threading.Lockers Namespace`
Tungsten

$W$
SpinLockExtensions.InLockAsync Method

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLockAsync(SpinLock, Action)</td>
<td>Asynchronously performs the action in a lock</td>
</tr>
<tr>
<td>InLockAsyncTType(SpinLock, FuncTType)</td>
<td>Asynchronously performs the function in a lock</td>
</tr>
</tbody>
</table>

## See Also

Reference
SpinLockExtensions Class
W.Threading.Lockers Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
SpinLockExtensions.InLockAsync Method (SpinLock, Action)

Asynchronously performs the action in a lock

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

**Syntax**

```csharp
public static Task InLockAsync(
    this SpinLock this,
    Action action
)
```

**Parameters**

*this*
- Type: `System.Threading.SpinLock`  
The SpinLock to provide resource locking

*action*
- Type: `SystemAction`  
The action to perform

**Return Value**

Type: `Task`

**Usage Note**

In Visual Basic and C#, you can call this method as an instance method on any object of type `SpinLock`. When you use instance
method syntax to call this method, omit the first parameter. For more information, see Extension Methods (Visual Basic) or Extension Methods (C# Programming Guide).

**See Also**

Reference
- SpinLockExtensions Class
- InLockAsync Overload
- W.Threading.Lockers Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
SpinLockExtensionsInLockAsync<TType> Method (SpinLock, Func<TType>)

Asynchronously performs the function in a lock

**Namespace:** W.Threading.Lockers
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```csharp
public static Task<TType> InLockAsync<TType>(
    this SpinLock this,
    Func<TType> func
)
```

### Parameters

**this**
*Type:* System.Threading.SpinLock
The SpinLock to provide resource locking

**func**
*Type:* System.Func<TType>
The function to perform

### Type Parameters

**TType**


### Return Value
Type: **Task**\texttt{TType}\n
Usage Note

In Visual Basic and C#, you can call this method as an instance method on any object of type **SpinLock**. When you use instance method syntax to call this method, omit the first parameter. For more information, see [Extension Methods (Visual Basic)](ExtensionMethods) or [Extension Methods (C# Programming Guide)](ExtensionMethods).

See Also

Reference

[SpinLockExtensions Class](SpinLockExtensions)
[InLockAsync Overload](InLockAsync)
[W.Threading.Lockers Namespace](W.Threading.Lockers)

Copyright © 2018 Jordan Duerksen
Tungsten

W
StateAssignmentDelegate<TState>
Delegate

Delegate which can be used to assign a new value to the internal state

**Namespace:**  W.Threading.Lockers
**Assembly:**  Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

**Syntax**

```c#
public delegate TState StateAssignmentDelegate<TState>(TState state)
```

**Parameters**

*state*
- Type: *TState*
- The current state

**Type Parameters**

*TState*

[Missing <typeparam name="TState"/> documentation for "T:W.Threading.Lockers.StateAssignmentDelegate`1"]

**Return Value**

- Type: *TState*
- The new value for the internal state
See Also

Reference

W.Threading.Lockers Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

W
StateLocker<TLocker, TState> Class

Extends a locker (SpinLocker, MonitorLocker, ReaderWriterLocker, SemaphoreSlimLocker) with an internal state value

**Inheritance Hierarchy**

```
System
    Object
        W.Threading.Lockers
            StateLocker<TLocker, TState>
                MonitorLocker<TState>
                SemaphoreSlimLocker<TState>
                SpinLocker<TState>
```

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll)  
**Version:** 2.0.1

**Syntax**

```c#
public abstract class StateLocker<TLocker, TState>
    ILocker<TLocker>, ILocker
where TLocker : new(), ILocker
```

**Type Parameters**

- **TLocker**  
  The Type of Locker to extend

- **TState**  
  The Type of the internal state value

The StateLocker<TLocker, TState> type exposes the following members.
### Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StateLockerTLocker, TState</td>
<td>Initializes a new instance of the StateLockerTLocker, TState class</td>
</tr>
</tbody>
</table>

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locker</td>
<td>The locking mechanism (SpinLock, Monitor, SemaphoreSlim, ReaderWriterLock)</td>
</tr>
</tbody>
</table>

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals</td>
<td>Determines whether the specified object is equal to the current object. (Inherited from Object.)</td>
</tr>
<tr>
<td>Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup</td>
</tr>
</tbody>
</table>
operations before it is reclaimed by garbage collection. (Inherited from **Object**.)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetHashCode</td>
<td>Serves as the default hash function. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>GetType</td>
<td>Gets the Type of the current instance. (Inherited from <strong>Object</strong>.)</td>
</tr>
<tr>
<td>InLock( Action )</td>
<td>Performs an action from within a lock</td>
</tr>
<tr>
<td>InLock( Action, TState )</td>
<td>Performs an action from within a lock, passing in the current state</td>
</tr>
<tr>
<td>InLock( Func, TState, TState )</td>
<td>Performs a function from within a lock, passing in the current state and assigning the state to the function result</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>InLockTResult(Func&lt;TResult&gt;)</td>
<td>Performs a function from within a lock</td>
</tr>
<tr>
<td>InLockAsync(Action)</td>
<td>Asynchronously performs an action from within a lock</td>
</tr>
<tr>
<td>InLockAsync(Action&lt;TState&gt;)</td>
<td>Asynchronously performs an action from within a lock, passing in the current state</td>
</tr>
<tr>
<td>InLockAsync(Func&lt;TState, TState&gt;)</td>
<td>Asynchronously performs a function from within a lock, passing in the current state and assigning the state to the function result</td>
</tr>
<tr>
<td>InLockAsyncTResult(Func&lt;TResult&gt;)</td>
<td>Asynchronously performs a function from within a lock</td>
</tr>
<tr>
<td>MemberwiseClone</td>
<td>Creates a shallow copy of the current Object. (Inherited from Object.)</td>
</tr>
</tbody>
</table>
**ToString**

Returns a string that represents the current object. (Inherited from `Object`.)

### Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
<td>The internal state</td>
</tr>
</tbody>
</table>

### Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AsTType</td>
<td>Use Generic syntax for the as operator. (Defined by <code>AsExtensions</code>.)</td>
</tr>
<tr>
<td>InitializeProperties</td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by <code>PropertyHostExtensions</code>.)</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td><code>InLock(Action)</code></td>
<td>Overloaded. Performs the action in a Monitor lock (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>InLock&lt;TType&gt;(Func&lt;TType&gt;)</code></td>
<td>Overloaded. Performs the function in a Monitor lock (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>InLockAsync(Action)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>InLockAsync&lt;TType&gt;(Func&lt;TType&gt;)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>IsDirty</code></td>
<td>Scans the IsDirty value of each field and property of type <code>IProperty</code> (Defined by <code>PropertyHostExtensions</code>.)</td>
</tr>
<tr>
<td><code>Lock</code></td>
<td>Performs a Monitor lock (Defined by <code>MonitorExtensions</code>.)</td>
</tr>
<tr>
<td><code>MarkAsClean</code></td>
<td>Scans each field and property of type <code>IProperty</code> and sets its IsDirty flag to...</td>
</tr>
</tbody>
</table>
false
(Defined by
PropertyHostExtensions.)

Unlock
Performs a Monitor unlock
(Defined by
MonitorExtensions.)

Remarks
This class adds the state functionality by wrapping the TLocker and re-implementing the ILocker interface

See Also

Reference
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
</tr>
</tbody>
</table>
StateLocker<TLocker, TState> Constructor

Initializes a new instance of the StateLocker<TLocker, TState> class

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

**Syntax**

```
protected StateLocker()
```

**See Also**

Reference

StateLocker<TLocker, TState> Class  
W.Threading.Lockers Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$^W$
StateLocker\(TLocker, TState\) Properties

The StateLocker\(TLocker, TState\) generic type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locker</td>
<td>The locking mechanism (SpinLock, Monitor, SemaphoreSlim, ReaderWriterLock)</td>
</tr>
</tbody>
</table>

See Also

Reference

StateLocker\(TLocker, TState\) Class

W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

\[ \text{W} \]
StateLocker

StateLocker

TLocker, TStateLocker Property

The locking mechanism (SpinLock, Monitor, SemaphoreSlim, ReaderWriterLock)

Namespace: W.Threading.Lockers

Syntax

```c#
public TLocker Locker { get; }
```

Property Value

Type: `TLocker`

Implements

`ILocker` TLocker

See Also

Reference

StateLocker TLocker, TState Class
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
</tr>
</tbody>
</table>
StateLocker\textit{TLocker, TState} Methods

The \textit{StateLockerTLocker, TState} generic type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\textbullet Equals</td>
<td>Determines whether the specified object is equal to the current object. (Inherited from \textit{Object}.)</td>
</tr>
<tr>
<td>\textbullet Finalize</td>
<td>Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. (Inherited from \textit{Object}.)</td>
</tr>
<tr>
<td>\textbullet GetHashCode</td>
<td>Serves as the default hash function.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>GetTypeInfo()</code></td>
<td>Gets the Type of the current instance. (Inherited from <code>Object</code>.)</td>
</tr>
<tr>
<td><code>InLock(Action)</code></td>
<td>Performs an action from within a lock</td>
</tr>
<tr>
<td><code>InLock(ActionTState)</code></td>
<td>Performs an action from within a lock, passing in the current state</td>
</tr>
<tr>
<td><code>InLock(FuncTState, TState)</code></td>
<td>Performs a function from within a lock, passing in the current state and assigning the state to the function result</td>
</tr>
<tr>
<td><code>InLockTResult(FuncTResult)</code></td>
<td>Performs a function from within a lock</td>
</tr>
<tr>
<td><code>InLockAsync(Action)</code></td>
<td>Asynchronously performs an action from within a lock</td>
</tr>
<tr>
<td><code>InLockAsync(ActionTState)</code></td>
<td>Asynchronously...</td>
</tr>
</tbody>
</table>
InLockAsync(Func<TState, TState>)  
Asynchronously performs a function from within a lock, passing in the current state and assigning the state to the function result.

InLockAsyncTResult(FuncTResult)  
Asynchronously performs a function from within a lock.

MemberwiseClone  
Creates a shallow copy of the current `Object`. (Inherited from `Object`.)

ToString  
Returns a string that represents the current object. (Inherited from `Object`.)
## Extension Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AsTType</code></td>
<td>Use Generic syntax for the as operator. (Defined by AsExtensions.)</td>
</tr>
<tr>
<td><code>InitializeProperties</code></td>
<td>Scans the fields and properties of &quot;owner&quot; and sets the member's Owner property to &quot;owner&quot;. This method should be called in the constructor of any class which has IOwnedProperty members. (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td><code>InLock(Action)</code></td>
<td>Overloaded. Performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><code>InLockTType(FuncTType)</code></td>
<td>Overloaded. Performs the function in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td><code>InLockAsync(Action)</code></td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock. (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>
### MonitorExtensions.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLockAsync&lt;TType(Func&lt;TType&gt;)</td>
<td>Overloaded. Asynchronously performs the action in a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>IsDirty</td>
<td>Scans the IsDirty value of each field and property of type IProperty (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Lock</td>
<td>Performs a Monitor lock (Defined by MonitorExtensions.)</td>
</tr>
<tr>
<td>MarkAsClean</td>
<td>Scans each field and property of type IProperty and sets its IsDirty flag to false (Defined by PropertyHostExtensions.)</td>
</tr>
<tr>
<td>Unlock</td>
<td>Performs a Monitor unlock (Defined by MonitorExtensions.)</td>
</tr>
</tbody>
</table>

### See Also

Reference
- StateLockerTLocker, TState Class
- W.Threading.Lockers Namespace
## StateLocker

**TLocker, TState**

### InLock Method

#### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InLock(Action)</td>
<td>Performs an action from within a lock</td>
</tr>
<tr>
<td>InLock(Action&lt;TState&gt;)</td>
<td>Performs an action from within a lock, passing in the current state</td>
</tr>
<tr>
<td>InLock&lt;TResult(Func&lt;TResult)&gt;)</td>
<td>Performs a function from within a lock</td>
</tr>
<tr>
<td>InLock(Func&lt;TState, TState&gt;)</td>
<td>Performs a function from within a lock, passing in the current state and assigning the state to the function result</td>
</tr>
</tbody>
</table>

### See Also

**Reference**
- StateLocker<TLocker, TState> Class
- W.Threading.Lockers Namespace
Tungsten

$W$
StateLocker

**TLocker, TState** InLock Method (Action)

Performs an action from within a lock

**Namespace:** W.Threading.Lockers
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```csharp
public void InLock(
    Action action
)
```

### Parameters

**action**
- Type: `System.Action`
- The action to run

### Implements

`ILockerInLock(Action)`

### See Also

- Reference
  - StateLocker
  - TLocker, TState Class
  - InLock Overload
  - W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>$W$</td>
</tr>
</tbody>
</table>
StateLocker\textit{TLocker}, \textit{TState}InLock Method (Action\textit{TState})

Performs an action from within a lock, passing in the current state

\textbf{Namespace:} W.Threading.Lockers  

\section*{Syntax}

\begin{verbatim}
public void InLock(
    Action<TState> action
)
\end{verbatim}

\section*{Parameters}

\textit{action}  
Type: \textit{SystemAction<TState>}

The action to run

\section*{Implements}

ISTateLockerTLocker, TStateInLock(ActionTState)

\section*{See Also}

Reference

StateLockerTLocker, TState Class  
InLock Overload  
W.Threading.Lockers Namespace
Tungsten

$W$
StateLocker<TLocker, TState>InLock<TResult> Method (Func<TResult>)

Performs a function from within a lock

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```csharp
public TResult InLock<TResult>(
    Func<TResult> func
)
```

### Parameters

**func**
- **Type:** System.Func<TResult>
- The function to run

### Type Parameters

**TResult**

[Missing <typeparam name="TResult"/> documentation for "M:W.Threading.Lockers.StateLocker`2.InLock`1(System.Func{`0})"]

### Return Value

**Type:** TResult

[Missing <returns> documentation for "M:W.Threading.Lockers.StateLocker`2.InLock`1(System.Func{`0})"]
Implements
ILocker
InLock
TResult(FuncTResult)

See Also

Reference
StateLocker TLocker, TState Class
InLock Overload
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
</tr>
</tbody>
</table>
StateLocker<TLocker, TState>InLock Method (Func<TState, TState>)

Performs a function from within a lock, passing in the current state and assigning the state to the function result

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

**Syntax**

```csharp
public TState InLock(
    Func<TState, TState> func
)
```

**Parameters**

*func*
Type: System.Func<TState, TState>
The function to run

**Return Value**

Type: TState

[Missing <returns> documentation for "M:W.Threading.Lockers.StateLocker`2.InLock(System.Func`1,`1)""]

**Implements**

IStateLocker<TLocker, TStateInLock(Func<TState, TState)>
See Also

Reference
StateLocker TLocker, TState Class
InLock Overload
W.Threading.Lockers Namespace

Copyright © 2018 Jordan Duerksen
### StateLocker TLocker, TState InLockAsync Method

#### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>InLockAsync(Action)</code></td>
<td>Asynchronously performs an action from within a lock</td>
</tr>
<tr>
<td><code>InLockAsync(Action&lt;TState&gt;)</code></td>
<td>Asynchronously performs an action from within a lock, passing in the current state</td>
</tr>
<tr>
<td><code>InLockAsync&lt;TResult(Func&lt;TResult&gt;)</code></td>
<td>Asynchronously performs a function from within a lock</td>
</tr>
<tr>
<td><code>InLockAsync(Func&lt;TState, TState&gt;)</code></td>
<td>Asynchronously performs a function from within a lock, passing in the current state and assigning the state to the function result</td>
</tr>
</tbody>
</table>
See Also

Reference
StateLocker
TLocker, TState Class
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$^w$
StateLocker

InLockAsync Method

Asynchronously performs an action from within a lock

**Namespace:** W.Threading.Lockers

**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

### Syntax

```csharp
public Task InLockAsync(
    Action action
)
```

### Parameters

*action*

Type: `SystemAction`

The action to run

### Return Value

Type: `Task`


### Implements

ILockerInLockAsync(Action)

### See Also
Reference
StateLocker TLocker, TState Class
InLockAsync Overload
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
StateLocker\text{TLocker}, \text{TState}\text{InLockAsync} Method (Action\text{TState})

Asynchronously performs an action from within a lock, passing in the current state

\textbf{Namespace: } W.Threading.Lockers

\section*{Syntax}

\begin{verbatim}
    public Task InLockAsync(
        Action<TState> action
    )
\end{verbatim}

\section*{Parameters}

\textit{action}
\begin{itemize}
    \item Type: System\text{Action}\text{TState} \\
    \text{The action to run}
\end{itemize}

\section*{Return Value}

\begin{itemize}
    \item Type: Task
\end{itemize}

[Missing \texttt{<returns>} documentation for \texttt{M:W.Threading.Lockers.StateLocker`2.InLockAsync(System.Action`1)}]

\section*{Implements}

IStateLocker\text{TLocker}, TState\text{InLockAsync}(Action\text{TState})
See Also

Reference

StateLocker TLocker, TState Class
InLockAsync Overload
W.Threading.Lockers Namespace

Copyright © 2018 Jordan Duerksen
Tungsten

$W$
StateLocker

InLockAsync

Method (Func<TResult>)

Asynchronously performs a function from within a lock

Namespace:  W.Threading.Lockers

Syntax

```csharp
public Task<TResult> InLockAsync<TResult>(
    Func<TResult> func
)
```

Parameters

func
Type: System.Func<TResult>
The function to run

Type Parameters

TResult

Return Value
Type: Task<TResult>

[Missing <typeparam name="TResult"/> documentation for "M:W.Threading.Lockers.StateLocker`2.InLockAsync`1(System.Func{`0})""]
Implements
ILocker
InLockAsync<TResult>(Func<TResult)

See Also

Reference
StateLocker
TLocker, TState Class
InLockAsync Overload
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
StateLocker<TLocker, TState> InLockAsync Method (Func<TState, TState>)

Asynchronously performs a function from within a lock, passing in the current state and assigning the state to the function result

**Namespace**: W.Threading.Lockers
**Assembly**: Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

**Syntax**

```csharp
public Task<TState> InLockAsync(
    Func<TState, TState> func
)
```

**Parameters**

`func`
- Type: System.Func<TState, TState>
- The function to run

**Return Value**
- Type: Task<TState>

[Missing <returns> documentation for "M:W.Threading.Lockers.StateLocker\`2.InLockAsync(System.Func\`1,\`1)"]

**Implements**
- IStateLocker<TLocker, TState> InLockAsync(Func<TState, TState>)
See Also

Reference
StateLocker, TState Class
InLockAsync Overload
W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen
Tungsten

$W$
## StateLocker\textit{TLocker}, \textit{TState}

Fields

The \textit{StateLockerTLocker, TState} generic type exposes the following members.

### Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
<td>The internal state</td>
</tr>
</tbody>
</table>

See Also

Reference

- \textit{StateLockerTLocker, TState Class}
- \textit{W.Threading.Lockers Namespace}

Copyright @ 2018 Jordan Duerksen
<table>
<thead>
<tr>
<th>Tungsten</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
</tr>
</tbody>
</table>
StateLocker<TLocker, TState> State Field

The internal state

**Namespace:** W.Threading.Lockers  
**Assembly:** Tungsten.Threading.Lockers (in Tungsten.Threading.Lockers.dll) Version: 2.0.1

## Syntax

```csharp
protected TState State
```

## Field Value

Type: `TState`

## See Also

Reference

- StateLocker<TLocker, TState> Class
- W.Threading.Lockers Namespace

Copyright @ 2018 Jordan Duerksen