

# LinksPlatform's Platform.Converters Class Library

./To.cs

```
1  using System;
2  using System.Runtime.CompilerServices;
3
4  #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
5
6  namespace Platform.Converters
7  {
8      public static class To
9      {
10         public static readonly char UnknownCharacter = '\0';
11
12         [MethodImpl(MethodImplOptions.AggressiveInlining)]
13         public static ulong UInt64(ulong value) => value;
14
15         [MethodImpl(MethodImplOptions.AggressiveInlining)]
16         public static long Int64(ulong value) => value > long.MaxValue ? long.MaxValue :
17             ↳ (long)value;
18
19         [MethodImpl(MethodImplOptions.AggressiveInlining)]
20         public static uint UInt32(ulong value) => value > uint.MaxValue ? uint.MaxValue :
21             ↳ (uint)value;
22
23         [MethodImpl(MethodImplOptions.AggressiveInlining)]
24         public static int Int32(ulong value) => value > int.MaxValue ? int.MaxValue : (int)value;
25
26         [MethodImpl(MethodImplOptions.AggressiveInlining)]
27         public static ushort UInt16(ulong value) => value > ushort.MaxValue ? ushort.MaxValue :
28             ↳ (ushort)value;
29
30         [MethodImpl(MethodImplOptions.AggressiveInlining)]
31         public static short Int16(ulong value) => value > (ulong)short.MaxValue ? short.MaxValue
32             ↳ : (short)value;
33
34         [MethodImpl(MethodImplOptions.AggressiveInlining)]
35         public static byte Byte(ulong value) => value > byte.MaxValue ? byte.MaxValue :
36             ↳ (byte)value;
37
38         [MethodImpl(MethodImplOptions.AggressiveInlining)]
39         public static sbyte SByte(ulong value) => value > (ulong)sbyte.MaxValue ? sbyte.MaxValue
40             ↳ : (sbyte)value;
41
42         [MethodImpl(MethodImplOptions.AggressiveInlining)]
43         public static bool Boolean(ulong value) => value > 0;
44
45         [MethodImpl(MethodImplOptions.AggressiveInlining)]
46         public static char Char(ulong value) => value > char.MaxValue ? UnknownCharacter :
47             ↳ (char)value;
48
49         [MethodImpl(MethodImplOptions.AggressiveInlining)]
50         public static DateTime DateTime(ulong value) => value > long.MaxValue ?
51             ↳ System.DateTime.MaxValue : new DateTime((long)value);
52
53         [MethodImpl(MethodImplOptions.AggressiveInlining)]
54         public static TimeSpan TimeSpan(ulong value) => value > long.MaxValue ?
55             ↳ System.TimeSpan.MaxValue : new TimeSpan((long)value);
56
57         [MethodImpl(MethodImplOptions.AggressiveInlining)]
58         public static ulong UInt64(long value) => value < (long)ulong.MinValue ? ulong.MinValue
59             ↳ : (ulong)value;
60
61         [MethodImpl(MethodImplOptions.AggressiveInlining)]
62         public static ulong UInt64(int value) => value < (int)ulong.MinValue ? ulong.MinValue :
63             ↳ (ulong)value;
64
65         [MethodImpl(MethodImplOptions.AggressiveInlining)]
66         public static ulong UInt64(short value) => value < (short)ulong.MinValue ?
67             ↳ ulong.MinValue : (ulong)value;
68
69         [MethodImpl(MethodImplOptions.AggressiveInlining)]
70         public static ulong UInt64(sbyte value) => value < (sbyte)ulong.MinValue ?
71             ↳ ulong.MinValue : (ulong)value;
72
73         [MethodImpl(MethodImplOptions.AggressiveInlining)]
74         public static ulong UInt64(bool value) => value ? 1UL : 0UL;
75
76         [MethodImpl(MethodImplOptions.AggressiveInlining)]
```

```

64     public static ulong UInt64(char value) => value;
65
66     [MethodImpl(MethodImplOptions.AggressiveInlining)]
67     public static long Signed(ulong value) => (long)value;
68
69     [MethodImpl(MethodImplOptions.AggressiveInlining)]
70     public static int Signed(uint value) => (int)value;
71
72     [MethodImpl(MethodImplOptions.AggressiveInlining)]
73     public static short Signed(ushort value) => (short)value;
74
75     [MethodImpl(MethodImplOptions.AggressiveInlining)]
76     public static sbyte Signed(byte value) => (sbyte)value;
77
78     [MethodImpl(MethodImplOptions.AggressiveInlining)]
79     public static object Signed<T>(T value) => To<T>.Signed(value);
80
81     [MethodImpl(MethodImplOptions.AggressiveInlining)]
82     public static ulong Unsigned(long value) => (ulong)value;
83
84     [MethodImpl(MethodImplOptions.AggressiveInlining)]
85     public static uint Unsigned(int value) => (uint)value;
86
87     [MethodImpl(MethodImplOptions.AggressiveInlining)]
88     public static ushort Unsigned(short value) => (ushort)value;
89
90     [MethodImpl(MethodImplOptions.AggressiveInlining)]
91     public static byte Unsigned(sbyte value) => (byte)value;
92
93     [MethodImpl(MethodImplOptions.AggressiveInlining)]
94     public static object Unsigned<T>(T value) => To<T>.Unsigned(value);
95
96     [MethodImpl(MethodImplOptions.AggressiveInlining)]
97     public static T UnsignedAs<T>(object value) => To<T>.UnsignedAs(value);
98 }
99 }

```

./To[T].cs

```

1  using System;
2  using System.Reflection;
3  using Platform.Exceptions;
4  using Platform.Reflection;
5  using Platform.Reflection.Sigil;
6
7  #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
8
9  namespace Platform.Converters
10 {
11     public sealed class To<T>
12     {
13         public static readonly Func<T, object> Signed;
14         public static readonly Func<T, object> Unsigned;
15         public static readonly Func<object, T> UnsignedAs;
16
17         static To()
18         {
19             Signed = DelegateHelpers.Compile<Func<T, object>>(emitter =>
20             {
21                 Ensure.Always.IsUnsignedInteger<T>();
22                 emitter.LoadArgument(0);
23                 var method = typeof(To).GetTypeInfo().GetMethod("Signed", Types<T>.Array);
24                 emitter.Call(method);
25                 emitter.Box(method.ReturnType);
26                 emitter.Return();
27             });
28             Unsigned = DelegateHelpers.Compile<Func<T, object>>(emitter =>
29             {
30                 Ensure.Always.IsSignedInteger<T>();
31                 emitter.LoadArgument(0);
32                 var method = typeof(To).GetTypeInfo().GetMethod("Unsigned", Types<T>.Array);
33                 emitter.Call(method);
34                 emitter.Box(method.ReturnType);
35                 emitter.Return();
36             });
37             UnsignedAs = DelegateHelpers.Compile<Func<object, T>>(emitter =>
38             {
39                 Ensure.Always.IsUnsignedInteger<T>();
40                 emitter.LoadArgument(0);
41                 var signedVersion = CachedTypeInfo<T>.SignedVersion;
42                 emitter.UnboxAny(signedVersion);

```

```
43         var method = typeof(To).GetTypeInfo().GetMethod("Unsigned", new[] {  
44             ↪ signedVersion });  
45         emitter.Call(method);  
46         emitter.Return();  
47     });  
48 }  
49 private To()  
50 {  
51 }  
52 }  
53 }
```

## Index

./To.cs, 1  
./To[T].cs, 2