

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) => unchecked(value > long.MaxValue ? long.MaxValue
17             ↳ : (long)value);
18
19         [MethodImpl(MethodImplOptions.AggressiveInlining)]
20         public static uint UInt32(ulong value) => unchecked(value > uint.MaxValue ?
21             ↳ uint.MaxValue : (uint)value);
22
23         [MethodImpl(MethodImplOptions.AggressiveInlining)]
24         public static int Int32(ulong value) => unchecked(value > int.MaxValue ? int.MaxValue :
25             ↳ (int)value);
26
27         [MethodImpl(MethodImplOptions.AggressiveInlining)]
28         public static ushort UInt16(ulong value) => unchecked(value > ushort.MaxValue ?
29             ↳ ushort.MaxValue : (ushort)value);
30
31         [MethodImpl(MethodImplOptions.AggressiveInlining)]
32         public static short Int16(ulong value) => unchecked(value > (ulong)short.MaxValue ?
33             ↳ short.MaxValue : (short)value);
34
35         [MethodImpl(MethodImplOptions.AggressiveInlining)]
36         public static byte Byte(ulong value) => unchecked(value > byte.MaxValue ? byte.MaxValue
37             ↳ : (byte)value);
38
39         [MethodImpl(MethodImplOptions.AggressiveInlining)]
40         public static sbyte SByte(ulong value) => unchecked(value > (ulong)sbyte.MaxValue ?
41             ↳ sbyte.MaxValue : (sbyte)value);
42
43         [MethodImpl(MethodImplOptions.AggressiveInlining)]
44         public static bool Boolean(ulong value) => value > 0UL;
45
46         [MethodImpl(MethodImplOptions.AggressiveInlining)]
47         public static char Char(ulong value) => unchecked(value > char.MaxValue ?
48             ↳ UnknownCharacter : (char)value);
49
50         [MethodImpl(MethodImplOptions.AggressiveInlining)]
51         public static DateTime DateTime(ulong value) => unchecked(value > long.MaxValue ?
52             ↳ System.DateTime.MaxValue : new DateTime((long)value));
53
54         [MethodImpl(MethodImplOptions.AggressiveInlining)]
55         public static TimeSpan TimeSpan(ulong value) => unchecked(value > long.MaxValue ?
56             ↳ System.TimeSpan.MaxValue : new TimeSpan((long)value));
57
58         [MethodImpl(MethodImplOptions.AggressiveInlining)]
59         public static ulong UInt64(long value) => unchecked(value < (long)ulong.MinValue ?
60             ↳ ulong.MinValue : (ulong)value);
61
62         [MethodImpl(MethodImplOptions.AggressiveInlining)]
63         public static ulong UInt64(int value) => unchecked(value < (int)ulong.MinValue ?
64             ↳ ulong.MinValue : (ulong)value);
65
66         [MethodImpl(MethodImplOptions.AggressiveInlining)]
67         public static ulong UInt64(short value) => unchecked(value < (short)ulong.MinValue ?
68             ↳ ulong.MinValue : (ulong)value);
69
70         [MethodImpl(MethodImplOptions.AggressiveInlining)]
71         public static ulong UInt64(sbyte value) => unchecked(value < (sbyte)ulong.MinValue ?
72             ↳ ulong.MinValue : (ulong)value);
73
74         [MethodImpl(MethodImplOptions.AggressiveInlining)]
75         public static ulong UInt64(bool value) => value ? 1UL : 0UL;
```

```

63     [MethodImpl(MethodImplOptions.AggressiveInlining)]
64     public static ulong UInt64(char value) => value;
65
66     [MethodImpl(MethodImplOptions.AggressiveInlining)]
67     public static long Signed(ulong value) => unchecked((long)value);
68
69     [MethodImpl(MethodImplOptions.AggressiveInlining)]
70     public static int Signed(uint value) => unchecked((int)value);
71
72     [MethodImpl(MethodImplOptions.AggressiveInlining)]
73     public static short Signed(ushort value) => unchecked((short)value);
74
75     [MethodImpl(MethodImplOptions.AggressiveInlining)]
76     public static sbyte Signed(byte value) => unchecked((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) => unchecked((ulong)value);
83
84     [MethodImpl(MethodImplOptions.AggressiveInlining)]
85     public static uint Unsigned(int value) => unchecked((uint)value);
86
87     [MethodImpl(MethodImplOptions.AggressiveInlining)]
88     public static ushort Unsigned(short value) => unchecked((ushort)value);
89
90     [MethodImpl(MethodImplOptions.AggressiveInlining)]
91     public static byte Unsigned(sbyte value) => unchecked((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
6  #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
7
8  namespace Platform.Converters
9  {
10     public static class To<T>
11     {
12         public static readonly Func<T, object> Signed;
13         public static readonly Func<T, object> Unsigned;
14         public static readonly Func<object, T> UnsignedAs;
15
16         static To()
17         {
18             Signed = CompileSignedDelegate();
19             Unsigned = CompileUnsignedDelegate();
20             UnsignedAs = CompileUnsignedAsDelegate();
21         }
22
23         static private Func<T, object> CompileSignedDelegate()
24         {
25             return DelegateHelpers.Compile<Func<T, object>>(emitter =>
26             {
27                 Ensure.Always.IsUnsignedInteger<T>();
28                 emitter.LoadArgument(0);
29                 var method = typeof(To).GetTypeInfo().GetMethod("Signed", Types<T>.Array);
30                 emitter.Call(method);
31                 emitter.Box(method.ReturnType);
32                 emitter.Return();
33             });
34         }
35
36         static private Func<T, object> CompileUnsignedDelegate()
37         {
38             return DelegateHelpers.Compile<Func<T, object>>(emitter =>
39             {
40                 Ensure.Always.IsSignedInteger<T>();
41                 emitter.LoadArgument(0);

```

```

42         var method = typeof(To).GetTypeInfo().GetMethod("Unsigned", Types<T>.Array);
43         emitter.Call(method);
44         emitter.Box(method.ReturnType);
45         emitter.Return();
46     });
47 }
48
49 static private Func<object, T> CompileUnsignedAsDelegate()
50 {
51     return DelegateHelpers.Compile<Func<object, T>>(emitter =>
52     {
53         Ensure.Always.IsUnsignedInteger<T>();
54         emitter.LoadArgument(0);
55         var signedVersion = NumericType<T>.SignedVersion;
56         emitter.UnboxValue(signedVersion);
57         var method = typeof(To).GetTypeInfo().GetMethod("Unsigned", new[] {
58             ↪ signedVersion });
59         emitter.Call(method);
60         emitter.Return();
61     });
62 }
63 }

```

Index

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