

# LinksPlatform's Platform.Timestamps Class Library

## 1.1 ./csharp/Platform.Timestamps/Timestamp.cs

```
1 using System;
2 using System.Runtime.CompilerServices;
3
4 namespace Platform.Timestamps
5 {
6     /// <summary>
7     /// <para>Represents a timestamp.</para>
8     /// <para>Представляет метку времени.</para>
9     /// </summary>
10    /// <remarks>
11    /// <para>To make this timestamp truly unique, it is recommended to use <see
12    ↪ cref="UniqueTimestampFactory"/>.</para>
13    /// <para>Чтобы эта метка времени была действительно уникальна рекомендуется использовать
14    ↪ <see cref="UniqueTimestampFactory"/>.</para>
15    /// </remarks>
16    public struct Timestamp : IEquatable<Timestamp>
17    {
18        /// <summary>
19        /// <para>Returns a string containg the default DateTime format for Timestamp.</para>
20        /// <para>Возвращает строку, содержащую формат даты и времени по умолчанию для метки
21        ↪ времени.</para>
22        /// </summary>
23        public static readonly string DefaultFormat = "yyyy.MM.dd hh:mm:ss.ffffff";
24
25        /// <summary>
26        /// <para>Gets or sets the number of ticks that represent the date and time in
27        ↪ UTC.</para>
28        /// <para>Возвращает или устанавливает количество тиков, которые представляют дату и
29        ↪ время в UTC.</para>
30        /// </summary>
31        public readonly ulong Ticks;
32
33        /// <summary>
34        /// <para>Creates a timestamp.</para>
35        /// <para>Создаёт метку времени.</para>
36        /// </summary>
37        /// <param name="ticks"><para>A number representing the number of
38        ↪ ticks.</para><para>Число представляющие количество тиков.</para></param>
39        [MethodImpl(MethodImplOptions.AggressiveInlining)]
40        public Timestamp(ulong ticks) => Ticks = ticks;
41
42        /// <summary>
43        /// <para>Defines an implicit conversion of a DateTime to a Timestamp.</para>
44        /// <para>Определяет неявное преобразование DateTime в метку времени.</para>
45        /// </summary>
46        /// <param name="dateTime"><para>The DateTime struct.</para><para>Структура
47        ↪ DateTime.</para></param>
48        [MethodImpl(MethodImplOptions.AggressiveInlining)]
49        public static implicit operator Timestamp(DateTime dateTime) => new
50        ↪ Timestamp((ulong)dateTime.ToUniversalTime().Ticks);
51
52        /// <summary>
53        /// <para>Defines an implicit conversion of a Timestamp to a DateTime.</para>
54        /// <para>Определяет неявное преобразование метки времени в DateTime.</para>
55        /// </summary>
56        /// <param name="timestamp"><para>The Timestamp.</para><para>Отметка
57        ↪ времени.</para></param>
58        [MethodImpl(MethodImplOptions.AggressiveInlining)]
59        public static implicit operator DateTime(Timestamp timestamp) => new
60        ↪ DateTime((long)timestamp.Ticks, DateTimeKind.Utc);
61
62        /// <summary>
63        /// <para>Defines an implicit conversion of a 64-bit unsigned integer to a
64        ↪ Timestamp.</para>
65        /// <para>Определяет неявное преобразование 64-разрядного целого числа без знака в метку
66        ↪ времени.</para>
67        /// </summary>
68        /// <param name="ticks"><para>The number of ticks represented as a 64-bit
69        ↪ integer.</para><para>Количество тиков представленное в виде 64-разрядного целого
70        ↪ числа.</para></param>
71        [MethodImpl(MethodImplOptions.AggressiveInlining)]
72        public static implicit operator Timestamp(ulong ticks) => new Timestamp(ticks);
73
74        /// <summary>
```

```

61    /// <para>Defines an implicit conversion of a Timestamp to a 64-bit unsigned
    ↪ integer.</para>
62    /// <para>Определяет неявное преобразование метки времени в 64-разрядное целое число без
    ↪ знака.</para>
63    /// </summary>
64    /// <param name="timestamp"><para>The Timestamp.</para><para>Отметка
    ↪ времени.</para></param>
65    [MethodImpl(MethodImplOptions.AggressiveInlining)]
66    public static implicit operator ulong(Timestamp timestamp) => timestamp.Ticks;
67
68    /// <summary>
69    /// <para>Returns a string that represents the current Timestamp.</para>
70    /// <para>Возвращает строку, которая представляет текущую метку времени.</para>
71    /// </summary>
72    /// <returns><para>A string that represents the current Timestamp.</para><para>Строка,
    ↪ представляющая текущую метку времени.</para></returns>
73    [MethodImpl(MethodImplOptions.AggressiveInlining)]
74    public override string ToString() => ((DateTime)this).ToString(DefaultFormat);
75
76    /// <summary>
77    /// <para>Определяет, равна ли текущая отметка времени другой отметке времени.</para>
78    /// <para>Indicates whether the current Timestamp is equal to another Timestamp.</para>
79    /// </summary>
80    /// <param name="other"><para>Other Timestamp.</para><para>Другая отметка
    ↪ времени.</para></param>
81    /// <returns><para>True if the current Timestamp is equal to the other Timestamp;
    ↪ otherwise, false.</para><para>Истину, если текущая отметка времени равна другой
    ↪ отметке времени; иначе ложь.</para></returns>
82    [MethodImpl(MethodImplOptions.AggressiveInlining)]
83    public bool Equals(Timestamp other) => Ticks == other.Ticks;
84
85    /// <summary>
86    /// <para>Determines whether the specified object is equal to the current object.</para>
87    /// <para>Определяет, равен ли указанный объект текущему объекту.</para>
88    /// </summary>
89    /// <param name="obj"><para>The object to compare with the current
    ↪ object.</para><para>Объект для сравнения с текущим объектом.</para></param>
90    /// <returns><para>True if the specified object is equal to the current object;
    ↪ otherwise, false.</para><para>Истину, если указанный объект равен текущему объекту;
    ↪ иначе ложь.</para></returns>
91    [MethodImpl(MethodImplOptions.AggressiveInlining)]
92    public override bool Equals(object obj) => obj is Timestamp timestamp ?
    ↪ Equals(timestamp) : false;
93
94    /// <summary>
95    /// <para>Serves as the default hash function.</para>
96    /// <para>Служит в качестве хэш-функции по умолчанию.</para>
97    /// </summary>
98    /// <returns><para>A hash code for the current object.</para><para>Хеш-код для текущего
    ↪ объекта.</para></returns>
99    [MethodImpl(MethodImplOptions.AggressiveInlining)]
100    public override int GetHashCode() => Ticks.GetHashCode();
101
102    /// <summary>
103    /// <para>Determines if the specified timestamp is equal to the current timestamp.</para>
104    /// <para>Определяет, равна ли указанная метка времени текущей метке времени.</para>
105    /// </summary>
106    /// <param name="left"><para>The current timestamp.</para><para>Текущая метка
    ↪ времени.</para></param>
107    /// <param name="right"><para>A timestamp to compare with this
    ↪ timestamp.</para><para>Метка времени для сравнения с этой меткой
    ↪ времени.</para></param>
108    /// <returns><para>True if the current timestamp is equal to the other timestamp;
    ↪ otherwise, false.</para><para>True, если текущий метка времени равна другой метке
    ↪ времени; иначе false.</para></returns>
109    [MethodImpl(MethodImplOptions.AggressiveInlining)]
110    public static bool operator ==(Timestamp left, Timestamp right) => left.Equals(right);
111
112    /// <summary>
113    /// <para>Determines if the specified timestamp is not equal to the current
    ↪ timestamp.</para>
114    /// <para>Определяет, не равна ли указанная метка времени текущей метке времени.</para>
115    /// </summary>
116    /// <param name="left"><para>The current timestamp.</para><para>Текущая метка
    ↪ времени.</para></param>

```

```

117     /// <param name="right"><para>A timestamp to compare with this
    ↪ timestamp.</para><para>Метка времени для сравнения с этой меткой
    ↪ времени.</para></param>
118     /// <returns><para>True if the current timestamp is not equal to the other timestamp;
    ↪ otherwise, false.</para><para>True, если текущий метка времени не равна другой метке
    ↪ времени; иначе false.</para></returns>
119     [MethodImpl(MethodImplOptions.AggressiveInlining)]
120     public static bool operator !=(Timestamp left, Timestamp right) => !(left == right);
121 }
122 }

```

## 1.2 ./csharp/Platform.Timestamps/UniqueTimestampFactory.cs

```

1  using System;
2  using System.Runtime.CompilerServices;
3  using Platform.Interfaces;
4
5  namespace Platform.Timestamps
6  {
7      /// <summary>
8      /// <para>Represents a factory for creating unique timestamps.</para>
9      /// <para>Представляет фабрику по созданию уникальных отметок времени.</para>
10     /// </summary>
11     public class UniqueTimestampFactory : IFactory<Timestamp>
12     {
13         private ulong _lastTicks;
14
15         /// <summary>
16         /// <para>Creates a timestamp corresponding to the current UTC date and time or next
    ↪ unique timestamp.</para>
17         /// <para>Создаёт отметку времени соответствующую текущей дате и времени по UTC или
    ↪ следующую уникальную отметку времени.</para>
18         /// </summary>
19         [MethodImpl(MethodImplOptions.AggressiveInlining)]
20         public Timestamp Create()
21         {
22             var utcTicks = (ulong)DateTime.UtcNow.Ticks;
23             _lastTicks = utcTicks > _lastTicks ? utcTicks : _lastTicks + 1;
24             return new Timestamp(_lastTicks);
25         }
26     }
27 }

```

## 1.3 ./csharp/Platform.Timestamps.Tests/UniqueTimestampFactoryTests.cs

```

1  using Xunit;
2
3  namespace Platform.Timestamps.Tests
4  {
5      public class UniqueTimestampFactoryTests
6      {
7          [Fact]
8          public void UniqueTimestampTest()
9          {
10             var factory = new UniqueTimestampFactory();
11             var timestamp1 = factory.Create();
12             var timestamp2 = factory.Create();
13             Assert.NotEqual(timestamp1, timestamp2);
14         }
15     }
16 }

```

## Index

./csharp/Platform.Timestamps.Tests/UniqueTimestampFactoryTests.cs, 3

./csharp/Platform.Timestamps/Timestamp.cs, 1

./csharp/Platform.Timestamps/UniqueTimestampFactory.cs, 3