```
LinksPlatform's Platform. Threading Class Library
./ConcurrentQueueExtensions.cs
   using System;
   using System. Collections. Concurrent;
2
   using System. Threading. Tasks;
   using Platform.Collections.Concurrent;
4
   #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
6
   namespace Platform. Threading
8
       public static class ConcurrentQueueExtensions
10
11
            public static async Task AwaitAll(this ConcurrentQueue<Task> queue)
12
13
                foreach (var item in queue.DequeueAll())
14
15
                    await item;
16
                }
            }
19
            public static async Task AwaitOne(this ConcurrentQueue<Task> queue)
20
21
                   (queue.TryDequeue(out Task item))
22
                    await item;
24
                }
25
            }
26
27
           public static void EnqueueAsRunnedTask(this ConcurrentQueue<Task> queue, Action action)
               => queue.Enqueue(Task.Run(action));
       }
29
30
./Synchronization/ISynchronization.cs
   using System;
2
   namespace Platform. Threading. Synchronization
3
4
        /// <summary>
5
       /// <para>Represents a synchronization object that supports read and write operations.</para>
       /// <para>Представляет объект синхронизации с поддержкой операций чтения и записи.</para>
        /// </summary>
       public interface ISynchronization
10
            /// <summary>
1.1
            /// <para>Executes action in read access mode.</para>
12
            /// <para>Выполняет действие в режиме доступа для чтения.</para>
            /// </summary>
14
            /// <param name="action"><para>The action.</para><para>Действие.</para></param>
15
            void ExecuteReadOperation(Action action);
16
17
            /// <summary>
18
            /// <para>Executes a function in read access mode and returns the function's
               result.</para>
            /// <para>Выполняет функцию в режиме доступа для чтения и возвращает полученный из неё
20
            → результат.</para>
            /// </summary>
21
            /// <typeparam name="TResult"><para>Туре of function's result.</para><para>Тип
22
               результата функции.</para></typeparam>
            /// <param name="function"><para>The function.</para><para>Функция.</para></para>
            /// <returns><para>The function's result.</para><para>Результат функции.</para></returns>
2.4
            TResult ExecuteReadOperation<TResult>(Func<TResult> function);
25
            /// <summary>
27
            /// <para>Executes action in write access mode.</para>
28
            /// <para>Выполняет действие в режиме доступа для записи.</para>
29
            /// </summary>
30
            /// <param name="action"><para>The action.</para><para>Действие.</para></param>
3.1
            void ExecuteWriteOperation(Action action);
33
            /// <summary>
34
            /// <para>Executes a function in write access mode and returns the function's
35
                result.</para>
            /// <para>Выполняет функцию в режиме доступа для записи и возвращает полученный из неё
36
                результат.</para>
            /// </summary>
            /// <typeparam name="TResult"><para>Type of function's result.</para><para>Тип
            → результата функции.</para></typeparam>
```

```
/// <param name="function"><para>The function.</para><para>Функция.</para></param>
            /// <returns><para>The function's result.</para><para>Результат функции.</para></returns>
           TResult ExecuteWriteOperation<TResult>(Func<TResult> function);
41
42
./Synchronization/ISynchronizationExtensions.cs
   using System;
   namespace Platform. Threading. Synchronization
       /// <summary>
5
       /// <para>Contains extension methods for the <see cref="ISynchronization"/> interface.</para>
6
       /// <para>Содержит методы расширения для интерфейса <see cref="ISynchronization"/>.</para>
       /// </summary>
       public static class ISynchronizationExtensions
10
            /// <summary>
11
            /// <para>Executes a function in read access mode and returns the function's
12
               result.</para>
            /// <para>Выполняет функцию в режиме доступа для чтения и возвращает полученный из неё
13
           → результат.</para>
/// </summary>
            /// <typeparam name="TResult"><para>Type of function's result.</para><para>Тип
               результата функции.</para></typeparam>
            /// <typeparam name="TParam"><para>The parameter type.</para><para>Тип
16
               параметра.</para></typeparam>
            /// <param name="synchronization"><para>Synchronization
               object.</para><para>Синхронизация объекта.</para></param>
            /// <param name="parameter"><para>The parameter</para><para>Параметр.</para></para>
            /// <param name="function"><para>The function.</para><para>Функция.</para></param>
19
            /// <returns><para>The function's result.</para><para>Результат функции.</para></returns>
20
           public static TResult ExecuteReadOperation<TResult, TParam>(this ISynchronization
               synchronization, TParam parameter, Func<TParam, TResult> function) =>
               synchronization.ExecuteReadOperation(() => function(parameter));
            /// <summary>
            /// <para>Executes action in read access mode.</para>
24
            /// <para>Выполняет действие в режиме доступа для чтения.</para>
25
            /// </summary>
26
            /// <typeparam name="TParam"><para>The parameter type.</para><para>Тип
               параметра.</para></typeparam>
            /// <param name="synchronization"><para>Synchronization
               object.</para><para>Синхронизация объекта.</para></param>
            /// <param name="parameter"><para>The parameter</para><para>Параметр.</para></para>
29
            /// <param name="action"><para>The action.</para><para>Действие.</para></param>
30
           public static void ExecuteReadOperation<TParam>(this ISynchronization synchronization,
            TParam parameter, Action<TParam> action) => synchronization.ExecuteReadOperation(()
               => action(parameter));
32
            /// <summary>
            /// <para>Executes a function in write access mode and returns the function's
34
               result.</para>
            /// <para>Выполняет функцию в режиме доступа для записи и возвращает полученный из неё
35
               результат. </para>
            /// </summary>
36
            /// <typeparam name="TResult"><para>Туре of function's result.</para><para>Тип
               результата функции.</para></typeparam>
            /// <typeparam name="TParam"><para>The parameter type.</para><para>Тип
               параметра.</para></typeparam>
            /// <param name="synchronization"><para>Synchronization
3.9
               object.</para><para>Синхронизация объекта.</para></param>
            /// <param name="parameter"><para>The parameter</para><para>Параметр.</para></param>
40
            /// <param name="function"><para>The function.</para><para>Функция.</para></param>
41
            /// <returns><para>The function's result.</para><para>Результат функции.</para></returns>
           public static TResult ExecuteWriteOperation<TResult, TParam>(this ISynchronization
43
               synchronization, TParam parameter, Func<TParam, TResult> function) =>
               synchronization.ExecuteWriteOperation(() => function(parameter));
            /// <summary>
45
            /// <para>Executes action in write access mode.</para>
46
            /// <para>Выполняет действие в режиме доступа для записи.</para>
            /// </summary>
            /// <typeparam name="TParam"><para>The parameter type.</para><para>Тип
49
               параметра.</para></typeparam>
            /// <param name="synchronization"><para>Synchronization
               object.</para><para>Синхронизация объекта.</para></param>
```

```
/// <param name="parameter"><para>The parameter</para><para>Параметр.</para></param>
            /// <param name="action"><para>The action.</para><para>Действие.</para></param>
           public static void ExecuteWriteOperation<TParam>(this ISynchronization synchronization,
53
               TParam parameter, Action<TParam> action) => synchronization.ExecuteWriteOperation(()
               => action(parameter));
            /// <summary>
            /// <para>Executes a function in read access mode and returns the function's
56
               result.</para>
            /// <para>Выполняет функцию в режиме доступа для чтения и возвращает полученный из неё
57

    peзультат.
/// </summary>
            /// <typeparam name="TResult"><para>Type of function's result.</para><para>Тип
           → результата функции.</para></typeparam>
/// <typeparam name="TParam1"><para>The first parameter type.</para>Tип первого
60
               параметра.</para></typeparam>
            /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
               параметра.</para></typeparam>
            /// <param name="synchronization"><para>Synchronization
               object.</para><para>Синхронизация объекта.</para></param>
            /// <param name="parameter1"><para>The first parameter</para><para>Первый
               параметр.</para></param>
            /// <param name="parameter2"><para>The second parameter</para><para>Второй
               параметр.</para></param>
            /// <param name="function"><para>The function.</para><para>Функция.</para></param>
65
            /// <returns><para>The function's result.</para><para>Результат функции.</para></returns>
66
           public static TResult ExecuteReadOperation<TResult, TParam1, TParam2>(this
               ISynchronization synchronization, TParam1 parameter1, TParam2 parameter2,
               Func<TParam1, TParam2, TResult> function) => synchronization.ExecuteReadOperation(()
               => function(parameter1, parameter2));
            /// <summary>
69
            /// <para>Executes action in read access mode.</para>
70
            /// <para>Выполняет действие в режиме доступа для чтения.</para>
            /// </summary>
72
            /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
               параметра.</para></typeparam>
            /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
               параметра.</para></typeparam>
            /// <param name="synchronization"><para>Synchronization
               object.</para><para>Синхронизация объекта.</para></param>
            /// <param name="parameter1"><para>The first parameter</para><para>Первый
               параметр.</para></param>
            /// <param name="parameter2"><para>The second parameter</para><para>Второй
               параметр.</para></param>
            /// <param name="action"><para>The action.</para><para>Действие.</para></param>
           public static void ExecuteReadOperation<TParam1, TParam2>(this ISynchronization
               synchronization, TParam1 parameter1, TParam2 parameter2, Action<TParam1, TParam2>
                action) => synchronization.ExecuteReadOperation(() => action(parameter1,
               parameter2));
            /// <summary>
            /// <para>Executes a function in write access mode and returns the function's
82
               result.</para>
            /// <para>Выполняет функцию в режиме доступа для записи и возвращает полученный из неё
83
            → результат.</para>
/// </summary>
            /// <typeparam name="TResult"><para>Type of function's result.</para><para>Тип
               результата функции.</para></typeparam>
            /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
               параметра.</para></typeparam>
            /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
               параметра.</para></typeparam>
            /// <param name="synchronization"><para>Synchronization
               object.</para><para>Синхронизация объекта.</para></param>
            /// <param name="parameter1"><para>The first parameter</para><para>Первый
               параметр.</para></param>
            /// <param name="parameter2"><para>The second parameter</para><para>Второй
90
               параметр.</para></param>
            /// <param name="function"><para>The function.</para><para>Функция.</para></param>
91
            /// <returns><para>The function's result.</para><para>Результат функции.</para></returns>
           public static TResult ExecuteWriteOperation<TResult, TParam1, TParam2>(this
               ISynchronization synchronization, TParam1 parameter1, TParam2 parameter2,
               Func<TParam1, TParam2, TResult> function) =>
               synchronization.ExecuteWriteOperation(() => function(parameter1, parameter2));
```

```
/// <summary>
            /// <para>Executes action in write access mode.</para>
            /// <para>Выполняет действие в режиме доступа для записи.</para>
97
            /// </summary>
98
            /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
               параметра.</para></typeparam>
            /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
                параметра.</para></typeparam>
            /// <param name="synchronization"><para>Synchronization
                object.</para><para>Синхронизация объекта.</para></param>
            /// <param name="parameter1"><para>The first parameter</para><para>Первый
102
                параметр.</para></param>
            /// <param name="parameter2"><para>The second parameter</para><para>Второй
103
                параметр.</para></param>
            /// <param name="action"><para>The action.</para><para>Действие.</para></param>
            public static void ExecuteWriteOperation<TParam1, TParam2>(this ISynchronization
105
                synchronization, TParam1 parameter1, TParam2 parameter2, Action<TParam1, TParam2>
                action) => synchronization.ExecuteWriteOperation(() => action(parameter1,
                parameter2));
106
            /// <summary>
107
            /// <para>Executes a function in read access mode and returns the function's
108
                result.</para>
            /// <para>Выполняет функцию в режиме доступа для чтения и возвращает полученный из неё
                результат.</para>
            /// </summary>
110
            /// <typeparam name="TResult"><para>Type of function's result.</para><para>Тип
111
                результата функции.</para></typeparam>
            /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
112
               параметра.</para></typeparam>
            /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
                параметра.</para></typeparam>
            /// <typeparam name="TParam3"><para>The third parameter type.</para><para>Тип третьего
                параметра.</para></typeparam>
            /// <param name="synchronization"><para>Synchronization
115
                object.</para><para>Синхронизация объекта.</para></param>
            /// <param name="parameter1"><para>The first parameter</para><para>Первый
116
               параметр.</para></param>
            /// <param name="parameter2"><para>The second parameter</para><para>Второй
117
                параметр.</para></param>
            /// <param name="parameter3"><para>The third parameter</para><para>Третий
               параметр.</para></param>
            /// <param name="function"><para>The function.</para><para>Функция.</para></param>
119
            /// <returns><para>The function's result.</para><para>Результат функции.</para></returns>
120
            public static TResult ExecuteReadOperation<TResult, TParam1, TParam2, TParam3>(this
                ISynchronization synchronization, TParam1 parameter1, TParam2 parameter2, TParam3
                parameter3, Func<TParam1, TParam2, TParam3, TResult> function) =>
                synchronization.ExecuteReadOperation(() => function(parameter1, parameter2,
                parameter3));
            /// <summary>
123
            /// <para>Executes action in read access mode.</para>
124
            /// <para>Выполняет действие в режиме доступа для чтения.</para>
            /// <\bar{\gammary>}
126
            /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
127
               параметра.</para></typeparam>
            /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
128
                параметра.</para></typeparam>
            /// <typeparam name="TParam3"><para>The third parameter type.</para><para>Тип третьего
                параметра.</para></typeparam>
            /// <param name="synchronization"><para>Synchronization
                object.</para><para>Синхронизация объекта.</para></param>
            /// <param name="parameter1"><para>The first parameter</para><para>Первый
                параметр.</para></param>
            /// <param name="parameter2"><para>The second parameter</para><para>Второй
132
                параметр.</para></param>
            /// <param name="parameter3"><para>The third parameter</para><para>Третий
133
               параметр.</para></param>
            /// <param name="action"><para>The action.</para><para>Действие.</para></param>
            public static void ExecuteReadOperation<TParam1, TParam2, TParam3>(this ISynchronization
135
               synchronization, TParam1 parameter1, TParam2 parameter2, TParam3 parameter3,
                Action<TParam1, TParam2, TParam3> action) => synchronization.ExecuteReadOperation(()
                => action(parameter1, parameter2, parameter3));
136
```

/// <summary>

137

```
/// <para>Executes a function in write access mode and returns the function's
138
                result.</para>
            /// <para>Выполняет функцию в режиме доступа для записи и возвращает полученный из неё
139
                результат.</para>
            /// </summary>
            /// <typeparam name="TResult"><para>Туре of function's result.</para><para>Тип
141
                результата функции.</para></typeparam>
            /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
142
                параметра.</para></typeparam>
            /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
                параметра.</para></typeparam>
            /// <typeparam name="TParam3"><para>The third parameter type.</para><para>Тип третьего
                параметра.</para></typeparam>
            /// <param name="synchronization"><para>Synchronization
                object.</para><para>Синхронизация объекта.</para></param>
            /// <param name="parameter1"><para>The first parameter</para><para>Первый
146
                параметр.</para></param>
            /// <param name="parameter2"><para>The second parameter</para><para>Второй
147
                параметр.</para></param>
            /// <param name="parameter3"><para>The third parameter</para><para>Третий
               параметр.</para></param>
            /// <param name="function"><para>The function.</para><para>Функция.</para></param>
149
            /// <returns><para>The function's result.</para><para>Результат функции.</para></returns>
150
            public static TResult ExecuteWriteOperation<TResult, TParam1, TParam2, TParam3>(this
151
                ISynchronization synchronization, TParam1 parameter1, TParam2 parameter2, TParam3
                parameter3, Func<TParam1, TParam2, TParam3, TResult> function) =>
                synchronization.ExecuteWriteOperation(() => function(parameter1, parameter2,
                parameter3));
            /// <summary>
153
            /// <para>Executes action in write access mode.</para>
154
            /// <para>Выполняет действие в режиме доступа для записи.</para>
155
            /// </summary>
            /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
157
            → параметра.</para></typeparam>
            /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
158
               параметра.</para></typeparam>
            /// <typeparam name="TParam3"><para>The third parameter type.</para><para>Тип третьего
159
               параметра.</para></typeparam>
            /// <param name="synchronization"><para>Synchronization
                object.</para><para>Синхронизация объекта.</para></param>
            /// <param name="parameter1"><para>The first parameter</para><para>Первый
               параметр.</para></param>
            /// <param name="parameter2"><para>The second parameter</para><para>Второй
162
               параметр.</para></param>
            /// <param name="parameter3"><para>The third parameter</para><para>Третий
163
                параметр.</para></param>
            /// <param name="action"><para>The action.</para><para>Действие.</para></param>
            public static void ExecuteWriteOperation<TParam1, TParam2, TParam3>(this
                ISynchronization synchronization, TParam1 parameter1, TParam2 parameter2, TParam3
                parameter3, Action<TParam1, TParam2, TParam3> action) =>
                synchronization.ExecuteWriteOperation(() => action(parameter1, parameter2,
                parameter3));
166
            /// <summary>
167
            /// <para>Executes a function in read access mode and returns the function's
168
                result.</para>
            /// <para>Выполняет функцию в режиме доступа для чтения и возвращает полученный из неё
               результат.</para>
            /// </summary>
170
            /// <typeparam name="TResult"><para>Туре of function's result.</para><para>Тип
171
            → результата функции.</para></typeparam>
/// <typeparam name="TParam1"><para>The first parameter type.</para>Tип первого
               параметра.</para></typeparam>
            /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
               параметра.</para></typeparam>
            /// <typeparam name="TParam3"><para>The third parameter type.</para><para>Тип третьего
               параметра.</para></typeparam>
            /// <typeparam name="TParam4"><para>The forth parameter type.</para><para>Тип четвёртого
                параметра.</para></typeparam>
            /// <param name="synchronization"><para>Synchronization
176
                object.</para><para>Синхронизация объекта.</para></param>
            /// <param name="parameter1"><para>The first parameter</para><para>Первый
               параметр.</para></param>
```

```
/// <param name="parameter2"><para>The second parameter</para><para>Второй
178
                параметр. </para></param>
             /// <param name="parameter3"><para>The third parameter</para><para>Третий
                параметр.</para></param>
             /// <param name="parameter4"><para>The forth parameter</para><para>Чертвёртый
                параметр.</para></param>
             /// <param name="function"><para>The function.</para><para>Функция.</para></para>
181
             /// <returns><para>The function's result.</para><para>Результат функции.</para></returns>
182
            public static TResult ExecuteReadOperation<TResult, TParam1, TParam2, TParam3,
183
                TParam4>(this ISynchronization synchronization, TParam1 parameter1, TParam2 parameter2, TParam3 parameter3, TParam4 parameter4, Func<TParam1, TParam2, TParam3, TParam4, TResult> function) => synchronization.ExecuteReadOperation(() =>
                function(parameter1, parameter2, parameter3, parameter4));
184
             /// <summary>
185
             /// <para>Executes action in read access mode.</para>
186
             /// <para>Выполняет действие в режиме доступа для чтения.</para>
187
             /// </summary>
188
             /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
                параметра.</para></typeparam>
             /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
190
                параметра.</para></typeparam>
             /// <typeparam name="TParam3"><para>The third parameter type.</para><para>Тип третьего
191
                параметра.</para></typeparam>
             /// <typeparam name="TParam4"><para>The forth parameter type.</para><para>Тип четвёртого
192
                параметра.</para></typeparam>
             /// <param name="synchronization"><para>Synchronization
                object.</para><para>Синхронизация объекта.</para></param>
             /// <param name="parameter1"><para>The first parameter</para><para>Первый
                параметр.</para></param>
             /// <param name="parameter2"><para>The second parameter</para><para>Второй
195
                параметр.</para></param>
             /// <param name="parameter3"><para>The third parameter</para><para>Третий
196
                параметр.</para></param>
             /// <param name="parameter4"><para>The forth parameter</para><para>Чертвёртый
197
                параметр.</para></param>
             /// <param name="action"><para>The action.</para><para>Действие.</para></param>
            public static void ExecuteReadOperation<TParam1, TParam2, TParam3, TParam4>(this
199
                ISynchronization synchronization, TParam1 parameter1, TParam2 parameter2, TParam3
                parameter3, TParam4 parameter4, Action<TParam1, TParam2, TParam3, TParam4> action)
                 => synchronization.ExecuteReadOperation(() => action(parameter1, parameter2,
                parameter3, parameter4));
200
             /// <summary>
201
             /// <para>Executes a function in write access mode and returns the function's
                result.</para>
             /// <para>Выполняет функцию в режиме доступа для записи и возвращает полученный из неё
203
                результат.</para>
             /// </summary>
204
             /// <typeparam name="TResult"><para>Туре of function's result.</para><para>Тип
205
                результата функции.</para></typeparam>
             /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
                параметра.</para></typeparam>
             /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
207
                параметра.</para></typeparam>
             /// <typeparam name="TParam3"><para>The third parameter type.</para><para>Тип третьего
208
                параметра.</para></typeparam>
             /// <typeparam name="TParam4"><para>The forth parameter type.</para><para>Тип четвёртого
209
                параметра.</para></typeparam>
             /// <param name="synchronization"><para>Synchronization
                object.</para><para>Синхронизация объекта.</para></param>
             /// <param name="parameter1"><para>The first parameter</para><para>Первый
                параметр.</para></param>
             /// <param name="parameter2"><para>The second parameter</para><para>Второй
                параметр.</para></param>
             /// <param name="parameter3"><para>The third parameter</para><para>Третий
213
                параметр.</para></param>
             /// <param name="parameter4"><para>The forth parameter</para><para>Чертвёртый
214
                параметр.</para></param>
             /// <param name="function"><para>The function.</para><para>Функция.</para></param>
             /// <returns><para>The function's result.</para><para>Результат функции.</para></returns>
216
```

```
public static TResult ExecuteWriteOperation<TResult, TParam1, TParam2, TParam3,</pre>
217
                TParam4>(this ISynchronization synchronization, TParam1 parameter1, TParam2 parameter2, TParam3 parameter3, TParam4 parameter4, Func<TParam1, TParam2, TParam3,
                 TParam4, TResult> function) => synchronization.ExecuteWriteOperation(() =>
                 function(parameter1, parameter2, parameter3, parameter4));
             /// <summary>
219
             /// <para>Executes action in write access mode.</para>
220
             /// <para>Выполняет действие в режиме доступа для записи.</para>
             /// </summary>
222
             /// <typeparam name="TParam1"><para>The first parameter type.</para><para>Тип первого
223
                параметра.</para></typeparam>
             /// <typeparam name="TParam2"><para>The second parameter type.</para><para>Тип второго
224
                параметра.</para></typeparam>
             /// <typeparam name="TParam3"><para>The third parameter type.</para><para>Тип третьего
             → параметра.</para></typeparam>
             /// <typeparam name="TParam4"><para>The forth parameter type.</para><para>Тип четвёртого
                параметра.</para></typeparam>
             /// <param name="synchronization"><para>Synchronization
227
                object.</para><para>Синхронизация объекта.</para></param>
             /// <param name="parameter1"><para>The first parameter</para><para>Первый
228
                параметр.</para></param>
             /// <param name="parameter2"><para>The second parameter</para><para>Второй
229
                параметр.</para></param>
             /// <param name="parameter3"><para>The third parameter</para><para>Третий
                 параметр.</para></param>
             /// <param name="parameter4"><para>The forth parameter</para><para>Чертвёртый
231
                параметр.</para></param>
             /// <param name="action"><para>The action.</para><para>Действие.</para></param>
232
            public static void ExecuteWriteOperation<TParam1, TParam2, TParam3, TParam4>(this
233
                 ISynchronization synchronization, TParam1 parameter1, TParam2 parameter2, TParam3
                parameter3, TParam4 parameter4, Action<TParam1, TParam2, TParam3, TParam4> action)
                => synchronization.ExecuteWriteOperation(() => action(parameter1, parameter2,
                parameter3, parameter4));
        }
234
./Synchronization/ISynchronized.cs
    namespace Platform. Threading. Synchronization
 1
 2
        /// <summary>
 3
        /// <para>Represents extendable synchronized interface access gate.</para>
 4
        /// <para>Представляет расширяемый интерфейс шлюза синхронизированного доступа.</para>
        /// </summary>
        /// <typeparam name="TInterface"><para>Synchronized interface.</para><para>Синхронизируемый
            интерфейс.</para></typeparam>
        public interface ISynchronized<out TInterface>
             /// <summary>
10
             /// <para>Gets sychronization method.</para>
11
             /// <para>Возвращает способ синхронизации.</para>
             /// </summary>
13
             ISynchronization SyncRoot { get; }
14
15
             /// <summary>
16
             /// <para>Get source version of <typeparamref name="TInterface"/>, that does not
17
                garantee thread safe access synchronization.</para>
             /// <para>Возвращает исходную версию <typeparamref name="TInterface"/>, которая не
             → гарантирует потокобезопасную синхронизацию доступа.</para>
             /// </summary>
19
             /// <remarks>
20
             /// <para>It is unsafe to use it directly, unless compound context using SyncRoot is
21
                created.</para>
             /// <para>Использовать напрямую небезопасно, за исключением ситуации когда создаётся
                 составной контекст с использованием SyncRoot.</para>
             /// </remarks>
23
            TInterface Unsync { get; }
2.4
             /// <summary>
26
             /// <para>Get wrapped/decorated version of <typeparamref name="TInterface"/>, that does
27
             _{\hookrightarrow} garantee thread safe access synchronization.</para> /// <para>Возвращает обернутую/декорированную версию <typeparamref name="TInterface"/>,
                которая гарантирует потокобезопасную синхронизацию доступа.</para>
             /// </summary>
29
             /// <remarks>
30
```

```
/// <para>It is safe to use it directly, because it must be thread safe
31
                implementation.</para>
            /// <para>Безопасно использовать напрямую, так как реализация должна быть
32
                потокобезопасной.</para>
            /// </remarks>
33
            TInterface Sync { get; }
34
        }
35
   }
./Synchronization/ReaderWriterLockSynchronization.cs
   using System;
   using System. Threading;
2
   namespace Platform. Threading. Synchronization
4
        /// <summary>
        /// <para>Implementation of <see cref="ISynchronization"/> based on <see
           cref="ReaderWriterLockSlim"/>.</para>
        /// <para>Peaлизация <see cref="ISynchronization"/> на основе <see
           cref="ReaderWriterLockSlim"/>.</para>
        /// </summary>
       public class ReaderWriterLockSynchronization : ISynchronization
10
11
            private readonly ReaderWriterLockSlim _rwLock = new
12
            ReaderWriterLockSlim(LockRecursionPolicy.SupportsRecursion);
13
            /// <include file='bin\Release\netstandard2.0\Platform.Threading.xml'
               path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio_
               n.ExecuteReadOperation(System.Action)"]/*'/>
            /// <inheritdoc/>
1.5
            public void ExecuteReadOperation(Action action)
16
                _rwLock.EnterReadLock();
                try
19
                {
20
                    action();
21
22
                finally
23
24
                    _rwLock.ExitReadLock();
25
                }
26
            }
27
            /// <include file='bin\Release\netstandard2.0\Platform.Threading.xml'
29
            path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio
               n.ExecuteReadOperation``1(System.Func{``0})"]/*'/>
            /// <inheritdoc/>
30
            public TResult ExecuteReadOperation<TResult>(Func<TResult> function)
31
32
                 _rwLock.EnterReadLock();
33
                try
35
                    return function();
36
                finally
38
                {
39
                    _rwLock.ExitReadLock();
                }
41
            }
42
43
            /// <include file='bin\Release\netstandard2.0\Platform.Threading.xml'
44
               path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronizatio-
                n.ExecuteWriteOperation(System.Action)"]/*'/>
            /// <inheritdoc/>
45
            public void ExecuteWriteOperation(Action action)
46
47
                 _rwLock.EnterWriteLock();
48
                try
49
50
                    action();
5.1
                finally
53
                    _rwLock.ExitWriteLock();
55
56
            }
57
58
```

```
/// <include file='bin\Release\netstandard2.0\Platform.Threading.xml'
5.9
            \rightarrow path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronization]
                n.ExecuteWriteOperation``1(System.Func{``0})"]/*'/>
            /// <inheritdoc/>
60
            public TResult ExecuteWriteOperation<TResult>(Func<TResult> function)
62
                 _rwLock.EnterWriteLock();
63
                try
64
65
                    return function();
66
67
                finally
68
                {
                     _rwLock.ExitWriteLock();
70
                }
71
            }
72
        }
73
   }
74
./Synchronization/Unsynchronization.cs
   using System;
   namespace Platform. Threading. Synchronization
3
4
        /// <summary>
5
        /// <para>Implementation of <see cref="ISynchronization"/> that makes no actual
6
           synchronization.</para>
        /// /// cref="ISynchronization"/>, которая не выполняет фактическую
           синхронизацию.</para>
        /// </summary>
        public class Unsynchronization : ISynchronization
10
11
            /// <include file='bin\Release\netstandard2.0\Platform.Threading.xml'
            _{\rightarrow} \quad \texttt{path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronization]} \\
                n.ExecuteReadOperation(System.Action)"]/*'/>
            /// <inheritdoc/>
            public void ExecuteReadOperation(Action action) => action();
13
            /// <include file='bin\Release\netstandard2.0\Platform.Threading.xml'
15
            \rightarrow path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronization]
               n.ExecuteReadOperation``1(System.Func{``0})"]/*'/>
            /// <inheritdoc/>
16
            public TResult ExecuteReadOperation<TResult>(Func<TResult> function) => function();
17
18
            /// <include file='bin\Release\netstandard2.0\Platform.Threading.xml'
19
            _{\rightarrow} \quad \texttt{path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronization]} \\
                n.ExecuteWriteOperation(System.Action)"]/*'/>
            /// <inheritdoc/>
20
            public void ExecuteWriteOperation(Action action) => action();
22
23
            /// <include file='bin\Release\netstandard2.0\Platform.Threading.xml'
            _{\rightarrow} \quad \texttt{path='doc/members/member[@name="M:Platform.Threading.Synchronization.ISynchronization]} \\
                n.ExecuteWriteOperation``1(System.Func{``0})"]/*'/>
            /// <inheritdoc/>
            public TResult ExecuteWriteOperation<TResult>(Func<TResult> function) => function();
25
        }
26
   }
./TaskExtensions.cs
   using System.Threading.Tasks;
2
   #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
3
   namespace Platform. Threading
5
        public static class TaskExtensions
            public static T AwaitResult<T>(this Task<T> task) => task.GetAwaiter().GetResult();
9
10
   }
11
./ThreadHelpers.cs
   using System;
   using System. Threading;
2
   #pragma warning disable CS1591 // Missing XML comment for publicly visible type or member
```

```
namespace Platform. Threading
6
        public static class ThreadHelpers
9
           public static readonly int DefaultMaxStackSize;
public static readonly int ExtendedMaxStackSize = 200 * 1024 * 1024;
public static readonly int DefaultSleepTimeout = 1;
10
11
12
13
            public static void SyncInvokeWithExtendedStack<T>(T param, Action<object> action) =>
14
            SyncInvokeWithExtendedStack(param, action, ExtendedMaxStackSize);
            public static void SyncInvokeWithExtendedStack<T>(T param, Action<object> action, int
16
            maxStackSize) => StartNew(param, action, maxStackSize).Join();
            public static void SyncInvokeWithExtendedStack(Action action) =>

→ SyncInvokeWithExtendedStack(action, ExtendedMaxStackSize);

            public static void SyncInvokeWithExtendedStack(Action action, int maxStackSize) =>
20
            public static Thread StartNew<T>(T param, Action<object> action) => StartNew(param,
            → action, DefaultMaxStackSize);
23
            public static Thread StartNew<T>(T param, Action<object> action, int maxStackSize)
24
25
                var thread = new Thread(new ParameterizedThreadStart(action), maxStackSize);
26
                thread.Start(param);
27
28
                return thread;
29
30
            public static Thread StartNew(Action action) => StartNew(action, DefaultMaxStackSize);
31
32
            public static Thread StartNew(Action action, int maxStackSize)
33
34
                var thread = new Thread(new ThreadStart(action), maxStackSize);
35
                thread.Start();
36
                return thread;
38
39
            public static void Sleep() => Thread.Sleep(DefaultSleepTimeout);
       }
41
   }
42
```

Index

- ./ConcurrentQueueExtensions.cs, 1 ./Synchronization/ISynchronization.cs, 1 ./Synchronization/ISynchronizationExtensions.cs, 2 ./Synchronization/ISynchronized.cs, 7 ./Synchronization/ReaderWriterLockSynchronization.cs, 8 ./Synchronization/Unsynchronization.cs, 9 ./TaskExtensions.cs, 9 ./ThreadHelpers.cs, 9