

## Threads

Generated by Doxygen 1.9.1

<b>1 Threads</b>	<b>1</b>
<b>1 Threads</b>	<b>1</b>
1.1 API	2
1.1.1 Synchronize to the main-thread	2
<b>2 CHANGELOG</b>	<b>2</b>
<b>3 LICENSE</b>	<b>2</b>
<b>4 Namespace Index</b>	<b>3</b>
4.1 Namespace List	3
<b>5 Hierarchical Index</b>	<b>3</b>
5.1 Class Hierarchy	3
<b>6 Class Index</b>	<b>3</b>
6.1 Class List	3
<b>7 Namespace Documentation</b>	<b>4</b>
7.1 HamerSoft Namespace Reference	4
7.2 HamerSoft.Threads Namespace Reference	4
7.3 HamerSoft.Threads.Editor Namespace Reference	4
7.4 HamerSoft.Threads.Tests Namespace Reference	4
7.5 HamerSoft.Threads.Tests.Editor Namespace Reference	4
7.6 HamerSoft.Threads.Tests.Runtime Namespace Reference	4
<b>8 Class Documentation</b>	<b>4</b>
8.1 HamerSoft.Threads.Dispatcher Class Reference	4
8.1.1 Detailed Description	5
8.1.2 Member Function Documentation	5
8.2 HamerSoft.Threads.IDispatchResult< out out TResult > Interface Template Reference	6
8.2.1 Detailed Description	6
8.2.2 Property Documentation	7
8.3 HamerSoft.Threads.MainThreadAwaiter Class Reference	7
8.3.1 Detailed Description	8
8.3.2 Member Function Documentation	8
8.3.3 Property Documentation	8
8.4 HamerSoft.Threads.MainThreadSync Class Reference	9
8.4.1 Detailed Description	9
8.4.2 Member Function Documentation	9
<b>Index</b>	<b>11</b>

## 1 Threads

A light weight library to dispatch actions to the Unity3D main-thread. Threads can be used at both Editor-time and Run-time straight out of the box. No initialization is required. **When playmode is toggled in the editor, the**

**actions in the queue will be purged, and thus will never complete!** The dispatcher synchronizes with Unity during the `Update` loop or `EditorApplication.Update` loop.

## 1.1 API

The dispatcher API can be accessed through the static `Dispatcher` class.

Method	Description
<code>void Dispatcher.Post</code>	Post an action on the main-thread
<code>Task Dispatcher.PostAsync</code>	Post an awaitable action on the main-thread
<code>IDispatchResult Dispatcher.PostAsync&lt;T&gt;</code>	Post an awaitable <code>Func&lt;T&gt;</code> on the main-thread. Results can be retrieved through the <code>IDispatchResult</code>
<code>Dispatcher.ToMainThread()</code>	An awaiter to synchronize to the main-tread

See the [Docs](Documentation~) for more information or [Tests](Tests) for how to use the Dispatcher and we have provides some [Samples](#).

### 1.1.1 Synchronize to the main-thread

A very useful utility in this Dispatcher is the fact that code is easily synchronized to the main-thread. See the example below:

```
Task.Run(async () => {
    // We are on some other thread depending on scheduling.
    await Dispatcher.ToMainThread();
    // We are back on the main-thread now
    // so we can access UnityEngine code
    // without getting an exception!
    var myGameObject = Object.Instantiate(myPrefab);
    // F*cking Awesome XD!!!

    // We can also await a background thread
    await Dispatcher.ToBackgroundThread();
    // We are on a background thread again
    // so we can do expensive processing
    // without blocking the main-thread
});
```

## 2 CHANGELOG

## 3 LICENSE

Copyright (c) 2023 [HamerSoft](#)

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

## 4 Namespace Index

### 4.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

<a href="#">HamerSoft</a>	4
<a href="#">HamerSoft.Threads</a>	4
<a href="#">HamerSoft.Threads.Editor</a>	4
<a href="#">HamerSoft.Threads.Tests</a>	4
<a href="#">HamerSoft.Threads.Tests.Editor</a>	4
<a href="#">HamerSoft.Threads.Tests.Runtime</a>	4

## 5 Hierarchical Index

### 5.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

<a href="#">HamerSoft.Threads.Dispatcher</a>	4
<a href="#">HamerSoft.Threads.IDispatchResult&lt; out out TResult &gt;</a>	6
<a href="#">HamerSoft.Threads.IDispatchResult&lt; TResult &gt;</a>	6
<a href="#">HamerSoft.Threads.MainThreadAwaiter</a>	7
<a href="#">HamerSoft.Threads.MainThreadSync</a>	9

## 6 Class Index

### 6.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">HamerSoft.Threads.Dispatcher</a>	
Main-thread dispatcher which can be used to post actions on the mainthread both Editor-time and Run-time	4
<a href="#">HamerSoft.Threads.IDispatchResult&lt; out out TResult &gt;</a>	
Result object of a function executed on the main-thread	6
<a href="#">HamerSoft.Threads.MainThreadAwaiter</a>	
An awaitable to synchronize to the main thread	7
<a href="#">HamerSoft.Threads.MainThreadSync</a>	
Object that facilitates synchronization to the main-thread	9

## 7 Namespace Documentation

### 7.1 HamerSoft Namespace Reference

### 7.2 HamerSoft.Threads Namespace Reference

#### Classes

- class [Dispatcher](#)  
*Main-thread dispatcher which can be used to post actions on the mainthread both Editor-time and Run-time*
- interface [IDispatchResult](#)  
*Result object of a function executed on the main-thread*
- class [MainThreadAwaiter](#)  
*An awaitable to synchronize to the main thread*
- class [MainThreadSync](#)  
*Object that facilitates synchronization to the main-thread*

### 7.3 HamerSoft.Threads.Editor Namespace Reference

### 7.4 HamerSoft.Threads.Tests Namespace Reference

### 7.5 HamerSoft.Threads.Tests.Editor Namespace Reference

### 7.6 HamerSoft.Threads.Tests.Runtime Namespace Reference

## 8 Class Documentation

### 8.1 HamerSoft.Threads.Dispatcher Class Reference

Main-thread dispatcher which can be used to post actions on the mainthread both Editor-time and Run-time

#### Static Public Member Functions

- static void [Post](#) (Action action)  
*Post an action on the main-thread*
- static Task [PostAsync](#) (Action action)  
*Post an action on the main-thread, and allow for waiting for its completion*
- static Task< [IDispatchResult](#)< TResult > > [PostAsync](#)< TResult > (Func< TResult > function)  
*Post a function on the main-thread, and allow for waiting for its completion*
- static [MainThreadSync ToMainThread](#) ()  
*Synchronize a thread to the main-thread*
- static BackgroundThreadSync [ToBackgroundThread](#) ()  
*Synchronize a thread to some background-thread*

### 8.1.1 Detailed Description

Main-thread dispatcher which can be used to post actions on the mainthread both Editor-time and Run-time

### 8.1.2 Member Function Documentation

**8.1.2.1 Post()** `static void HamerSoft.Threads.Dispatcher.Post ( Action action ) [inline], [static]`

Post an action on the main-thread

#### Parameters

<i>action</i>	The function to be executed on the main-thread
---------------	--

**8.1.2.2 PostAsync()** `static Task HamerSoft.Threads.Dispatcher.PostAsync ( Action action ) [inline], [static]`

Post an action on the main-thread, and allow for waiting for its completion

#### Parameters

<i>action</i>	The function to be executed on the main-thread
---------------	--

#### Returns

An awaitable task

**8.1.2.3 PostAsync<TResult>()** `static Task<IDispatchResult<TResult>> HamerSoft.Threads.Dispatcher.PostAsync ( TResult > ( Func< TResult > function ) [inline], [static]`

Post a function on the main-thread, and allow for waiting for its completion

#### Parameters

<i>function</i>	The function to be executed on the main-thread
-----------------	--

#### Template Parameters

<i>TResult</i>	Result type of the function
----------------	-----------------------------

### Returns

`IDispatchResult<TResult>` that contains the result of the function, or the exception thrown when it was not successful

**8.1.2.4 ToBackgroundThread()** `static BackgroundThreadSync HamerSoft.Threads.Dispatcher.ToBackgroundThread ( ) [inline], [static]`

Synchronize a thread to some background-thread

This will mostly be used to sync from the main-thread to some background thread for further processing that doesn't need UnityEngine access.

### Returns

an awaiter to synchronize to some background thread

**8.1.2.5 ToMainThread()** `static MainThreadSync HamerSoft.Threads.Dispatcher.ToMainThread ( ) [inline], [static]`

Synchronize a thread to the main-thread

### Returns

an awaiter to synchronize to the main thread

The documentation for this class was generated from the following file:

- Runtime/Dispatcher.cs

## 8.2 HamerSoft.Threads.IDispatchResult< out TResult > Interface Template Reference

Result object of a function executed on the main-thread

### Properties

- TResult `Result` [get]  
*The return value of the function*
- bool `Succeeded` [get]  
*Completion flag of the function*
- Exception `Exception` [get]  
*Optional exception throw, when unsuccessful*

#### 8.2.1 Detailed Description

Result object of a function executed on the main-thread

## Template Parameters

<i>TResult</i>	result of the Func<T>
----------------	-----------------------

## 8.2.2 Property Documentation

**8.2.2.1 Exception** `Exception HamerSoft.Threads.IDispatchResult< out out TResult >.Exception [get]`

Optional exception throw, when unsuccessful

**8.2.2.2 Result** `TResult HamerSoft.Threads.IDispatchResult< out out TResult >.Result [get]`

The return value of the function

**8.2.2.3 Succeeded** `bool HamerSoft.Threads.IDispatchResult< out out TResult >.Succeeded [get]`

Completion flag of the function

The documentation for this interface was generated from the following file:

- Runtime/IDispatchResult.cs

## 8.3 HamerSoft.Threads.MainThreadAwaiter Class Reference

An awaitable to synchronize to the main thread

Inherits INotifyCompletion.

## Public Member Functions

- void [GetResult](#) ()  
*Get the result of the awaitable*
- void [Complete](#) (Exception e)  
*Complete the awaitable*
- void [OnCompleted](#) (Action continuation)  
*Function invoked when completed*



## Properties

- bool `IsCompleted` [get]  
*Flag if the awaitable is complete*

### 8.3.1 Detailed Description

An awaitable to synchronize to the main thread

### 8.3.2 Member Function Documentation

**8.3.2.1 Complete()** `void HamerSoft.Threads.MainThreadAwaiter.Complete ( Exception e ) [inline]`

Complete the awaitable

#### Parameters

<i>e</i>	optional exception
----------	--------------------

**8.3.2.2 GetResult()** `void HamerSoft.Threads.MainThreadAwaiter.GetResult ( ) [inline]`

Get the result of the awaitable

**8.3.2.3 OnCompleted()** `void HamerSoft.Threads.MainThreadAwaiter.OnCompleted ( Action continuation ) [inline]`

Function invoked when completed

#### Parameters

<i>continuation</i>	continuation action
---------------------	---------------------

### 8.3.3 Property Documentation

**8.3.3.1 IsCompleted** `bool HamerSoft.Threads.MainThreadAwaiter.IsCompleted [get]`

Flag if the awaitable is complete

The documentation for this class was generated from the following file:

- Runtime/MainThreadAwaiter.cs

## 8.4 HamerSoft.Threads.MainThreadSync Class Reference

Object that facilitates synchronization to the main-thread

### Public Member Functions

- [MainThreadAwaiter](#) [GetAwaiter](#) ()  
*Get the awaiter*

#### 8.4.1 Detailed Description

Object that facilitates synchronization to the main-thread

#### 8.4.2 Member Function Documentation

##### 8.4.2.1 [GetAwaiter\(\)](#) [MainThreadAwaiter](#) HamerSoft.Threads.MainThreadSync.GetAwaiter ( ) [inline]

Get the awaiter

#### Returns

Main-thread Awaiter

The documentation for this class was generated from the following file:

- Runtime/MainThreadSync.cs



## Index

- Complete
  - HamerSoft.Threads.MainThreadAwaiter, [8](#)
- Exception
  - HamerSoft.Threads.IDispatchResult< out out TResult >, [7](#)
- GetAwaiter
  - HamerSoft.Threads.MainThreadSync, [9](#)
- GetResult
  - HamerSoft.Threads.MainThreadAwaiter, [8](#)
- HamerSoft, [4](#)
- HamerSoft.Threads, [4](#)
- HamerSoft.Threads.Dispatcher, [4](#)
  - Post, [5](#)
  - PostAsync, [5](#)
  - PostAsync< TResult >, [5](#)
  - ToBackgroundThread, [6](#)
  - ToMainThread, [6](#)
- HamerSoft.Threads.Editor, [4](#)
- HamerSoft.Threads.IDispatchResult< out out TResult >, [6](#)
  - Exception, [7](#)
  - Result, [7](#)
  - Succeeded, [7](#)
- HamerSoft.Threads.MainThreadAwaiter, [7](#)
  - Complete, [8](#)
  - GetResult, [8](#)
  - IsCompleted, [8](#)
  - OnCompleted, [8](#)
- HamerSoft.Threads.MainThreadSync, [9](#)
  - GetAwaiter, [9](#)
- HamerSoft.Threads.Tests, [4](#)
- HamerSoft.Threads.Tests.Editor, [4](#)
- HamerSoft.Threads.Tests.Runtime, [4](#)
- IsCompleted
  - HamerSoft.Threads.MainThreadAwaiter, [8](#)
- OnCompleted
  - HamerSoft.Threads.MainThreadAwaiter, [8](#)
- Post
  - HamerSoft.Threads.Dispatcher, [5](#)
- PostAsync
  - HamerSoft.Threads.Dispatcher, [5](#)
- PostAsync< TResult >
  - HamerSoft.Threads.Dispatcher, [5](#)
- Result
  - HamerSoft.Threads.IDispatchResult< out out TResult >, [7](#)
- Succeeded
  - HamerSoft.Threads.IDispatchResult< out out TResult >, [7](#)
- ToBackgroundThread
  - HamerSoft.Threads.Dispatcher, [6](#)
- ToMainThread
  - HamerSoft.Threads.Dispatcher, [6](#)