

# Coroutine Manager Pro

## 1.0

Mon Jan 4 2016 15:58:08



# Contents

<b>1</b>	<b>Hierarchical Index</b>	<b>1</b>
1.1	Class Hierarchy	1
<b>2</b>	<b>Class Index</b>	<b>3</b>
2.1	Class List	3
<b>3</b>	<b>Class Documentation</b>	<b>5</b>
3.1	CM_Dispatcher Class Reference	5
3.1.1	Detailed Description	5
3.1.2	Member Function Documentation	5
3.1.2.1	AddToExecuteQueue(IEumerator job)	5
3.2	CM_GlobalCoroutineManager< T > Class Template Reference	6
3.2.1	Detailed Description	6
3.2.2	Property Documentation	6
3.2.2.1	Global	6
3.3	CM_Job Class Reference	6
3.3.1	Detailed Description	9
3.3.2	Member Function Documentation	9
3.3.2.1	AddChild(CM_Job childJob)	9
3.3.2.2	AddChild(IEumerator childJob)	9
3.3.2.3	Builder(params IEumerator[] coroutines)	9
3.3.2.4	Clone()	9
3.3.2.5	Clone(int numOfCopies)	9
3.3.2.6	Kill()	9
3.3.2.7	Kill(float delayInSeconds)	10
3.3.2.8	Make(IEumerator coroutine)	11
3.3.2.9	Make(IEumerator coroutine, string id)	11
3.3.2.10	NotifyOnChildJobComplete(EventHandler< CM_JobEventArgs > e)	11
3.3.2.11	NotifyOnChildJobStarted(EventHandler< CM_JobEventArgs > e)	11
3.3.2.12	NotifyOnJobComplete(EventHandler< CM_JobEventArgs > e)	11
3.3.2.13	NotifyOnJobFinishedRunning(EventHandler< CM_JobEventArgs > e)	11
3.3.2.14	NotifyOnJobPaused(EventHandler< CM_JobEventArgs > e)	12

3.3.2.15	NotifyOnJobResumed(EventHandler< CM_JobEventArgs > e)	12
3.3.2.16	NotifyOnJobStarted(EventHandler< CM_JobEventArgs > e)	12
3.3.2.17	OnChildJobsComplete(CM_JobEventArgs e)	12
3.3.2.18	OnChildJobsStarted(CM_JobEventArgs e)	12
3.3.2.19	OnJobComplete(CM_JobEventArgs e)	12
3.3.2.20	OnJobFinishedRunning(CM_JobEventArgs e)	13
3.3.2.21	OnJobPaused(CM_JobEventArgs e)	13
3.3.2.22	OnJobResumed(CM_JobEventArgs e)	13
3.3.2.23	OnJobStarted(CM_JobEventArgs e)	13
3.3.2.24	Pause()	13
3.3.2.25	Pause(float delayInSeconds)	13
3.3.2.26	RemoveChildJob(CM_Job childJob)	13
3.3.2.27	RemoveNotifyOnChildJobComplete(EventHandler< CM_JobEventArgs > e)	14
3.3.2.28	RemoveNotifyOnChildJobStarted(EventHandler< CM_JobEventArgs > e)	14
3.3.2.29	RemoveNotifyOnJobComplete(EventHandler< CM_JobEventArgs > e)	14
3.3.2.30	RemoveNotifyOnJobFinishedRunning(EventHandler< CM_JobEventArgs > e)	14
3.3.2.31	RemoveNotifyOnJobPaused(EventHandler< CM_JobEventArgs > e)	14
3.3.2.32	RemoveNotifyOnJobResumed(EventHandler< CM_JobEventArgs > e)	14
3.3.2.33	RemoveNotifyOnJobStarted(EventHandler< CM_JobEventArgs > e)	15
3.3.2.34	Repeat()	15
3.3.2.35	Repeat(int numOfTimes)	15
3.3.2.36	Resume()	15
3.3.2.37	Resume(float delayInSeconds)	15
3.3.2.38	Start()	15
3.3.2.39	Start(float delayInSeconds)	15
3.3.2.40	StopRepeat(float delayInSeconds)	15
3.3.3	Property Documentation	16
3.3.3.1	coroutine	16
3.3.3.2	id	16
3.3.3.3	jobKilled	16
3.3.3.4	numOfTimesExecuted	16
3.3.3.5	paused	16
3.3.3.6	repeating	16
3.3.3.7	running	16
3.4	CM_JobEventArgs Class Reference	16
3.4.1	Detailed Description	17
3.4.2	Constructor & Destructor Documentation	17
3.4.2.1	CM_JobEventArgs(CM_Job job, CM_Job[] childJobs)	17
3.4.3	Property Documentation	17
3.4.3.1	childJobs	17

3.4.3.2	hasChildJobs	17
3.4.3.3	job	18
3.5	CM_JobManager Class Reference	18
3.5.1	Detailed Description	20
3.5.2	Member Function Documentation	20
3.5.2.1	AddJob(CM_Job job)	20
3.5.2.2	AddJob(string id, IEnumerator routine)	21
3.5.2.3	AddJob(IList< CM_Job > jobs)	21
3.5.2.4	AddJob(params CM_Job[] jobs)	21
3.5.2.5	ClearJobList()	21
3.5.2.6	HasJob(string id)	22
3.5.2.7	IsRunning(CM_Job job)	22
3.5.2.8	IsRunning(string id)	22
3.5.2.9	KillAll()	22
3.5.2.10	KillAll(float delayInSeconds)	22
3.5.2.11	Make()	23
3.5.2.12	NotifyOnAllJobsCleared(EventHandler< CM_JobManagerEventArgs > e)	23
3.5.2.13	NotifyOnAllJobsKilled(EventHandler< CM_JobManagerEventArgs > e)	23
3.5.2.14	NotifyOnAllJobsPaused(EventHandler< CM_JobManagerEventArgs > e)	23
3.5.2.15	NotifyOnAllJobsResumed(EventHandler< CM_JobManagerEventArgs > e)	23
3.5.2.16	NotifyOnJobAdded(EventHandler< CM_JobManagerJobEditedEventArgs > e)	23
3.5.2.17	NotifyOnJobRemoved(EventHandler< CM_JobManagerJobEditedEventArgs > e)	24
3.5.2.18	OnAllJobsCleared(CM_JobManagerEventArgs e)	25
3.5.2.19	OnAllJobsKilled(CM_JobManagerEventArgs e)	25
3.5.2.20	OnAllJobsPaused(CM_JobManagerEventArgs e)	25
3.5.2.21	OnAllJobsResumed(CM_JobManagerEventArgs e)	25
3.5.2.22	OnJobAdded(CM_JobManagerJobEditedEventArgs e)	25
3.5.2.23	OnJobRemoved(CM_JobManagerJobEditedEventArgs e)	25
3.5.2.24	PauseAll()	26
3.5.2.25	PauseAll(float delayInSeconds)	26
3.5.2.26	PauseCoroutine(CM_Job job)	26
3.5.2.27	PauseCoroutine(string id)	26
3.5.2.28	RemoveJob(CM_Job job)	26
3.5.2.29	RemoveJob(string id)	27
3.5.2.30	RemoveNotifyOnAllJobsCleared(EventHandler< CM_JobManagerEventArgs > e)	27
3.5.2.31	RemoveNotifyOnAllJobsKilled(EventHandler< CM_JobManagerEventArgs > e)	27
3.5.2.32	RemoveNotifyOnAllJobsPaused(EventHandler< CM_JobManagerEventArgs > e)	27
3.5.2.33	RemoveNotifyOnAllJobsResumed(EventHandler< CM_JobManagerEventArgs > e)	27

3.5.2.34	RemoveNotifyOnJobAdded(EventHandler< CM_JobManagerJobEditedEvent↔ Args > e)	28
3.5.2.35	RemoveNotifyOnJobRemoved(EventHandler< CM_JobManagerJobEdited↔ EventArgs > e)	29
3.5.2.36	ResumeAll()	29
3.5.2.37	ResumeAll(float delayInSeconds)	29
3.5.2.38	ResumeCoroutine(CM_Job job)	29
3.5.2.39	ResumeCoroutine(string id)	29
3.5.2.40	StartAll()	30
3.5.2.41	StartAll(float delayInSeconds)	30
3.5.2.42	StartCoroutine(CM_Job job)	30
3.5.2.43	StartCoroutine(string id)	30
3.5.2.44	StopCoroutine(CM_Job job)	30
3.5.2.45	StopCoroutine(string id)	31
3.5.3	Event Documentation	31
3.5.3.1	allJobsCleared	31
3.5.3.2	allJobsKilled	31
3.5.3.3	allJobsPaused	31
3.5.3.4	allJobsResumed	31
3.5.3.5	jobAdded	31
3.5.3.6	jobRemoved	31
3.6	CM_JobManagerEventArgs Class Reference	31
3.6.1	Detailed Description	32
3.6.2	Constructor & Destructor Documentation	32
3.6.2.1	CM_JobManagerEventArgs(Dictionary< string, CM_Job > ownedJobs)	32
3.6.3	Property Documentation	32
3.6.3.1	ownedJobs	32
3.6.3.2	pausedJobs	32
3.6.3.3	runningJobs	33
3.7	CM_JobManagerJobEditedEventArgs Class Reference	33
3.7.1	Detailed Description	33
3.7.2	Constructor & Destructor Documentation	33
3.7.2.1	CM_JobManagerJobEditedEventArgs(Dictionary< string, CM_Job > owned↔ Jobs, CM_Job jobEdited)	33
3.7.3	Property Documentation	34
3.7.3.1	jobEdited	34
3.7.3.2	otherArgs	34
3.8	CM_JobQueue Class Reference	34
3.8.1	Detailed Description	36
3.8.2	Member Function Documentation	36
3.8.2.1	Clone()	36

3.8.2.2	Clone(int numOfCopies)	36
3.8.2.3	ContinuousRunning()	37
3.8.2.4	Enqueue(CM_JobQueue other)	37
3.8.2.5	Enqueue(params CM_Job[] jobs)	37
3.8.2.6	Enqueue(ICollection< CM_Job > jobs)	37
3.8.2.7	Enqueue(string id, IEnumerator routine)	37
3.8.2.8	Enqueue(IEnumerator routine)	37
3.8.2.9	Enqueue(CM_Job job)	37
3.8.2.10	HandleJobComplete(object sender, CM_JobEventArgs e)	38
3.8.2.11	KillAll()	38
3.8.2.12	KillAll(float delayInSeconds)	38
3.8.2.13	KillCurrent()	38
3.8.2.14	KillCurrent(float delayInSeconds)	38
3.8.2.15	Make()	38
3.8.2.16	NotifyOnJobProcessed(EventHandler< CM_QueueEventArgs > e)	39
3.8.2.17	NotifyOnQueueComplete(EventHandler< CM_QueueEventArgs > e)	40
3.8.2.18	NotifyOnQueueStarted(EventHandler< CM_QueueEventArgs > e)	40
3.8.2.19	OnJobProcessed(CM_QueueEventArgs e)	40
3.8.2.20	OnQueueComplete(CM_QueueEventArgs e)	40
3.8.2.21	OnQueueStarted(CM_QueueEventArgs e)	40
3.8.2.22	Pause()	40
3.8.2.23	Pause(float delayInSeconds)	40
3.8.2.24	RemoveNotifyOnJobProcessed(EventHandler< CM_QueueEventArgs > e)	41
3.8.2.25	RemoveNotifyOnQueueComplete(EventHandler< CM_QueueEventArgs > e)	41
3.8.2.26	RemoveNotifyOnQueueStarted(EventHandler< CM_QueueEventArgs > e)	41
3.8.2.27	Repeat()	41
3.8.2.28	Repeat(int numOfTimes)	41
3.8.2.29	Resume()	41
3.8.2.30	Resume(float delayInSeconds)	41
3.8.2.31	Start()	41
3.8.2.32	Start(float delayInSeconds)	42
3.8.2.33	StopContinuousRunning()	43
3.8.2.34	StopRepeat()	43
3.8.2.35	StopRepeat(float delayInSeconds)	43
3.8.3	Property Documentation	43
3.8.3.1	continuousRunning	43
3.8.3.2	numOfTimesExecuted	43
3.8.3.3	repeating	43
3.8.3.4	running	43
3.8.4	Event Documentation	44

3.8.4.1	jobProcessed	44
3.8.4.2	queueComplete	44
3.8.4.3	queueStarted	44
3.9	CM_Logger Class Reference	44
3.9.1	Detailed Description	45
3.9.2	Member Enumeration Documentation	45
3.9.2.1	Status	45
3.9.3	Member Function Documentation	45
3.9.3.1	Log(object message)	45
3.9.3.2	Log(object context, object message)	45
3.9.3.3	Log(object message, Status status)	45
3.10	CM_QueueEventArgs Class Reference	45
3.10.1	Detailed Description	46
3.10.2	Constructor & Destructor Documentation	46
3.10.2.1	CM_QueueEventArgs(CM_Job[] queuedJobs, CM_Job[] completedJobs, CM_Queue jobQueue)	46
3.10.3	Property Documentation	46
3.10.3.1	completedJobs	46
3.10.3.2	hasCompletedJobs	46
3.10.3.3	hasJobsInQueue	47
3.10.3.4	queuedJobs	47
3.11	CM_Singleton< T > Class Template Reference	47
3.11.1	Detailed Description	47
3.11.2	Property Documentation	47
3.11.2.1	instance	47
3.11.2.2	IsDestroyed	48
3.12	ExampleCharacterDamage Class Reference	48
3.12.1	Detailed Description	48
3.12.2	Member Function Documentation	48
3.12.2.1	AddActionToQueue(CM_Job action)	48
3.12.2.2	ApplyDamage(string damageType, float time)	48
3.12.2.3	RestoreHealth(float time)	49
3.13	ExampleCharacterMovement Class Reference	49
3.13.1	Detailed Description	49
3.13.2	Member Data Documentation	49
3.13.2.1	moveSpeed	49
3.14	ExampleContinuousSpawner Class Reference	50
3.14.1	Detailed Description	50
3.14.2	Member Data Documentation	50
3.14.2.1	numToSpawn	50



3.14.2.2	prefab	50
3.14.2.3	timeBetweenSpawns	50
3.15	ExampleDamageApplier Class Reference	50
3.15.1	Detailed Description	51
3.15.2	Member Data Documentation	51
3.15.2.1	character	51
3.16	ExampleGUI Class Reference	51
3.16.1	Detailed Description	51
3.16.2	Member Data Documentation	51
3.16.2.1	titleText	52
3.17	ExampleJobManagerTest Class Reference	52
3.17.1	Detailed Description	53
3.17.2	Member Function Documentation	53
3.17.2.1	DelayedKillAllTest()	53
3.17.2.2	DelayedPauseAllTest()	53
3.17.2.3	DelayedResumeAllTest()	53
3.17.2.4	GlobalJobManagerEventTest()	53
3.17.2.5	GlobalJobPauseTest()	53
3.17.2.6	GlobalJobResumeTest()	53
3.17.2.7	GlobalJobStartTest()	53
3.17.2.8	GlobalJobStopTest()	53
3.17.2.9	GlobalStartAllTest()	53
3.17.2.10	LocalJobManagerTest()	54
3.18	ExampleJobQueueTest Class Reference	54
3.18.1	Detailed Description	55
3.18.2	Member Function Documentation	55
3.18.2.1	AddLocalQueueToGlobalQueueTest()	55
3.18.2.2	AddRepeatingJobToQueueTest()	55
3.18.2.3	ClonedRepeatingQueueTest()	55
3.18.2.4	DelayedKillAllTest()	55
3.18.2.5	DelayedKillCurrentTest()	55
3.18.2.6	LocalQueueDelayedPauseAndResumeTest()	55
3.18.2.7	LocalQueueDelayedStartTest()	55
3.18.2.8	MultipleClonedQueueTest()	55
3.18.2.9	QueueEventTest()	56
3.18.2.10	SetNumberRepeatingQueueTest()	56
3.18.2.11	SimpleGlobalQueueTest()	56
3.18.2.12	SimpleLocalQueueTest()	56
3.18.2.13	TimedRepeatingQueueTest()	56
3.19	ExampleJobTest Class Reference	56

3.19.1 Detailed Description . . . . .	57
3.19.2 Member Function Documentation . . . . .	57
3.19.2.1 ChildJobTest() . . . . .	57
3.19.2.2 InfinitelyRepeatableJobTest() . . . . .	57
3.19.2.3 JobTestWithDelayedKill() . . . . .	57
3.19.2.4 JobTestWithDelayedPause() . . . . .	57
3.19.2.5 JobTestWithDelayedResume() . . . . .	57
3.19.2.6 JobTestWithDelayedStart() . . . . .	58
3.19.2.7 JobTestWithStartAndEndEvents() . . . . .	58
3.19.2.8 MultipleCloneJobTest() . . . . .	58
3.19.2.9 MultipleRepeatableJobTest() . . . . .	58
3.19.2.10 MultipleRepeatableJobTestWithChild() . . . . .	58
3.19.2.11 SimpleJobTest() . . . . .	58
3.19.2.12 SingleCloneJobTest() . . . . .	58
3.20 ExampleTimer Class Reference . . . . .	58
3.20.1 Detailed Description . . . . .	59
3.21 ExampleTimerPauseResume Class Reference . . . . .	59
3.21.1 Detailed Description . . . . .	59
3.22 ICM_Cloneable< T > Interface Template Reference . . . . .	59
3.22.1 Detailed Description . . . . .	59
3.22.2 Member Function Documentation . . . . .	59
3.22.2.1 Clone() . . . . .	59
3.22.2.2 Clone(int numOfCopies) . . . . .	60
3.23 CM_Logger.Message Class Reference . . . . .	61
3.23.1 Detailed Description . . . . .	61
3.23.2 Constructor & Destructor Documentation . . . . .	61
3.23.2.1 Message(object invoker, object content) . . . . .	61
3.23.3 Member Function Documentation . . . . .	61
3.23.3.1 ToString() . . . . .	61
3.23.4 Property Documentation . . . . .	62
3.23.4.1 content . . . . .	62
3.23.4.2 invoker . . . . .	62
<b>Index</b>	<b>63</b>

# Chapter 1

## Hierarchical Index

### 1.1 Class Hierarchy

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

CM_GlobalCoroutineManager< T > . . . . .	6
CM_GlobalCoroutineManager< CM_JobManager > . . . . .	6
CM_JobManager . . . . .	18
CM_GlobalCoroutineManager< CM_JobQueue > . . . . .	6
CM_JobQueue . . . . .	34
CM_Singleton< CM_Dispatcher > . . . . .	47
CM_Dispatcher . . . . .	5
CM_Singleton< CM_Logger > . . . . .	47
CM_Logger . . . . .	44
EventArgs	
CM_JobEventArgs . . . . .	16
CM_JobManagerEventArgs . . . . .	31
CM_JobManagerJobEditedEventArgs . . . . .	33
CM_QueueEventArgs . . . . .	45
ICM_Cloneable< T > . . . . .	59
ICM_Cloneable< CM_Job > . . . . .	59
CM_Job . . . . .	6
ICM_Cloneable< CM_JobQueue > . . . . .	59
CM_JobQueue . . . . .	34
CM_Logger.Message . . . . .	61
MonoBehaviour	
CM_Singleton< T > . . . . .	47
ExampleCharacterDamage . . . . .	48
ExampleCharacterMovement . . . . .	49
ExampleContinousSpawner . . . . .	50
ExampleDamageApplier . . . . .	50
ExampleGUI . . . . .	51
ExampleJobManagerTest . . . . .	52
ExampleJobQueueTest . . . . .	54
ExampleJobTest . . . . .	56
ExampleTimer . . . . .	58
ExampleTImerPauseResume . . . . .	59



## Chapter 2

# Class Index

### 2.1 Class List

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

<a href="#">CM_Dispatcher</a>	Used to start coroutines in the main thread from separate threads. . . . .	5
<a href="#">CM_GlobalCoroutineManager&lt; T &gt;</a>	The base class of Global Coroutine Managers. Provides access to a singular global copy of the class. . . . .	6
<a href="#">CM_Job</a>	The main coroutine job class. Encapsulates the behaviour for a single coroutine job. Provides access to status (i.e. running, paused, killed etc), . . . . .	6
<a href="#">CM_JobEventArgs</a>	Arguments used in events raised by <a href="#">CM_Job</a> . . . . .	16
<a href="#">CM_JobManager</a>	The main job manager class. Encapsulates the behaviour for global and local job managers. Provides access to events and public access to stored jobs. . . . .	18
<a href="#">CM_JobManagerEventArgs</a>	Arguments used in events raised by <a href="#">CM_JobManager</a> . . . . .	31
<a href="#">CM_JobManagerJobEditedEventArgs</a>	Arguments used in events raised by <a href="#">CM_JobManager</a> . . . . .	33
<a href="#">CM_JobQueue</a>	The main job queue class. Encapsulates all behaviour related to queueing a job. Provides access to events, and status (i.e. running, repeating). . . . .	34
<a href="#">CM_Logger</a>	Simple logging class used by the Coroutine Manager. . . . .	44
<a href="#">CM_QueueEventArgs</a>	Arguments used by events raised by <a href="#">CM_JobQueue</a> . . . . .	45
<a href="#">CM_Singleton&lt; T &gt;</a>	A base class for any Singleton. Provides global singular access to a MonoBehaviour. . . . .	47
<a href="#">ExampleCharacterDamage</a>	Example character with action queue. . . . .	48
<a href="#">ExampleCharacterMovement</a>	Example Script. A simple script showing how you can use <a href="#">CM_JobQueue</a> to create easily repeatable character movement. . . . .	49
<a href="#">ExampleContinuousSpawner</a>	Example Script. Used to spawn a number of objects using <a href="#">CM_Job</a> . . . . .	50
<a href="#">ExampleDamageApplier</a>	Applies damage actions to the example character. . . . .	50
<a href="#">ExampleGUI</a>	Simple GUI controller. Uses a <a href="#">CM_JobQueue</a> to enqueue a number of gui actions. . . . .	51

<a href="#">ExampleJobManagerTest</a>	Includes a number of methods to test the functionality of the <a href="#">CM_JobManager</a> class. Each method showcases a particular functionality of the <a href="#">CM_JobManager</a> that you can implement in your own projects. . . . .	52
<a href="#">ExampleJobQueueTest</a>	Includes a number of methods to test the functionality of the <a href="#">CM_JobQueue</a> class. Each method showcases a particular functionality of the <a href="#">CM_JobQueue</a> that you can implement in your own projects. Each method returns an ienumerator so that it can added to a seperate job queue to be run in sequence for test purposes. . . . .	54
<a href="#">ExampleJobTest</a>	Includes a number of methods to test the functionality of the <a href="#">CM_Job</a> class. Each method showcases a particular functionality of the <a href="#">CM_Job</a> class that you can implement in your own projects. Each method returns an ienumerator so that it can added to a job queue to be run in sequence for test purposes. . . . .	56
<a href="#">ExampleTimer</a>	Creates new coroutine job to update a text object with time since startup. Adds job to global job manager so that it can be paused and resumed as required. . . . .	58
<a href="#">ExampleTimerPauseResume</a>	Pauses all jobs associated with the JobManager when user clicks left mouse button and resumes all jobs with user clicks right mouse button. . . . .	59
<a href="#">ICM_Cloneable&lt; T &gt;</a>	An interface for all classes used by the coroutine manager that can be cloned. . . . .	59
<a href="#">CM_Logger.Message</a>	Encapsulates a message used by <a href="#">CM_Logger</a> . . . . .	61

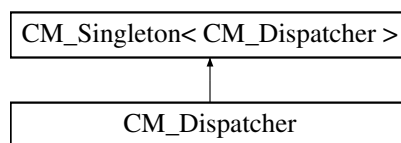
## Chapter 3

# Class Documentation

### 3.1 CM\_Dispatcher Class Reference

Used to start coroutines in the main thread from seperate threads.

Inheritance diagram for CM\_Dispatcher:



#### Public Member Functions

- void [AddToExecuteQueue](#) (IEnumerator job)

*Adds to execute queue. Coroutines in the queue are executed during the next timestep in the update method.*

#### Additional Inherited Members

##### 3.1.1 Detailed Description

Used to start coroutines in the main thread from seperate threads.

##### 3.1.2 Member Function Documentation

###### 3.1.2.1 void CM\_Dispatcher.AddToExecuteQueue ( IEnumerator job )

Adds to execute queue. Coroutines in the queue are executed during the next timestep in the update method.

Parameters

<i>job</i>	Job.
------------	------

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

- CM\_Dispatcher.cs

## 3.2 CM\_GlobalCoroutineManager< T > Class Template Reference

The base class of Global Coroutine Managers. Provides access to a singular global copy of the class.

### Protected Member Functions

- virtual [CM\\_Job](#) **MakeJob** (string id, IEnumerator routine, bool addListenerToOnComplete=true)
- virtual [CM\\_Job](#) **MakeJob** ([CM\\_Job](#) job)
- void **AutoGenerateJobId** ([CM\\_Job](#) job)
- abstract void **HandleJobComplete** (object sender, [CM\\_JobEventArgs](#) e)

### Properties

- static T [Global](#) [get]  
*Access to a global instance.*

#### 3.2.1 Detailed Description

The base class of Global Coroutine Managers. Provides access to a singular global copy of the class.

#### Type Constraints

**T** : *new()*

#### 3.2.2 Property Documentation

##### 3.2.2.1 T CM\_GlobalCoroutineManager< T >.Global [static], [get]

Access to a global instance.

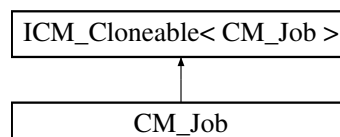
The documentation for this class was generated from the following file:

- CM\_GlobalCoroutineManager.cs

## 3.3 CM\_Job Class Reference

The main coroutine job class. Encapsulates the behaviour for a single coroutine job. Provides access to status (i.e. running, paused, killed etc),

Inheritance diagram for CM\_Job:



### Public Member Functions

- [CM\\_Job](#) **Clone** ()  
*Clone this instance.*



- [CM\\_Job\[\] Clone](#) (int numOfCopies)  
*Clone this instance.*
- [CM\\_Job Start](#) ()  
*Start this instance. Runs the coroutine immediately.*
- [CM\\_Job Start](#) (float delayInSeconds)  
*Start the specified instance after delayInSeconds. The coroutine is added to [CM\\_Dispatcher](#) job queue to be executed in the next timestep as a coroutine cannot be started in a seperate thread.*
- [CM\\_Job Repeat](#) ()  
*Sets this instance to repeat. The job is repeated when it has finished processing.*
- [CM\\_Job Repeat](#) (int numOfTimes)  
*Sets this instance to repeat. The job is repeated a set number of times.*
- [CM\\_Job StopRepeat](#) ()
- [CM\\_Job StopRepeat](#) (float delayInSeconds)  
*Stops the repeat after a specified delay in seconds.*
- [CM\\_Job Pause](#) ()  
*Pause this instance.*
- [CM\\_Job Pause](#) (float delayInSeconds)  
*Pause the specified instance after delayInSeconds.*
- [CM\\_Job Resume](#) ()  
*Resume this instance.*
- [CM\\_Job Resume](#) (float delayInSeconds)  
*Resume the specified instance after delayInSeconds.*
- void [Kill](#) ()  
*Kill this instance. Stops the running coroutine.*
- void [Kill](#) (float delayInSeconds)  
*Kill this instance. Stops the running coroutine after delayInSeconds.*
- [CM\\_Job AddChild](#) ([CM\\_Job](#) childJob)  
*Adds a child job.*
- [CM\\_Job AddChild](#) (IEnumerator childJob)  
*Create a new job using the provided Enumerator and adds as a child job.*
- [CM\\_Job RemoveChildJob](#) ([CM\\_Job](#) childJob)  
*Removes a child job if present.*
- [CM\\_Job NotifyOnJobFinishedRunning](#) (EventHandler< [CM\\_JobEventArgs](#) > e)  
*Subscribes to the jobFinishedRunning event*
- [CM\\_Job RemoveNotifyOnJobFinishedRunning](#) (EventHandler< [CM\\_JobEventArgs](#) > e)  
*Unsubscribes to the jobFinishedRunning event*
- [CM\\_Job NotifyOnJobStarted](#) (EventHandler< [CM\\_JobEventArgs](#) > e)  
*Subscribes to the jobStarted event.*
- [CM\\_Job RemoveNotifyOnJobStarted](#) (EventHandler< [CM\\_JobEventArgs](#) > e)  
*Unsubscribes to the jobStarted event.*
- [CM\\_Job NotifyOnJobPaused](#) (EventHandler< [CM\\_JobEventArgs](#) > e)  
*Subscribes to the job paused event.*
- [CM\\_Job RemoveNotifyOnJobPaused](#) (EventHandler< [CM\\_JobEventArgs](#) > e)  
*Unsubscribes to the job paused event.*
- [CM\\_Job NotifyOnJobResumed](#) (EventHandler< [CM\\_JobEventArgs](#) > e)  
*Subscribes to the job resumed event.*
- [CM\\_Job RemoveNotifyOnJobResumed](#) (EventHandler< [CM\\_JobEventArgs](#) > e)  
*Unsubscribes to the job resumed event.*
- [CM\\_Job NotifyOnJobComplete](#) (EventHandler< [CM\\_JobEventArgs](#) > e)  
*Subscribes to the the jobComplete event.*
- [CM\\_Job RemoveNotifyOnJobComplete](#) (EventHandler< [CM\\_JobEventArgs](#) > e)

- Unsubscribes to the the jobComplete event.*
  - [CM\\_Job NotifyOnChildJobStarted](#) (EventHandler< [CM\\_JobEventArgs](#) > e)
 *Subscribes to the the childJobsStarted event.*
  - [CM\\_Job RemoveNotifyOnChildJobStarted](#) (EventHandler< [CM\\_JobEventArgs](#) > e)
 *Unsubscribes to the the childJobsStarted event.*
  - [CM\\_Job NotifyOnChildJobComplete](#) (EventHandler< [CM\\_JobEventArgs](#) > e)
 *Subscribes to the the childJobsComplete event.*
  - [CM\\_Job RemoveNotifyOnChildJobComplete](#) (EventHandler< [CM\\_JobEventArgs](#) > e)
 *Unsubscribes to the the childJobsComplete event.*

## Static Public Member Functions

- static [CM\\_Job Make](#) (IEnumerator [coroutine](#))
 *Returns an initialised [CM\\_Job](#) instance. Provides static access to class.*
- static [CM\\_Job Make](#) (IEnumerator [coroutine](#), string [id](#))
 *Returns an initialised [CM\\_Job](#) instance with the specified id. Provides static access to class.*
- static [CM\\_Job\[\] Builder](#) (params IEnumerator[] [coroutines](#))
 *Builds the specified coroutines into [CM\\_Job](#) instances.*

## Protected Member Functions

- void [OnJobFinishedRunning](#) ([CM\\_JobEventArgs](#) e)
 *Raises the job finished running event.*
- void [OnJobStarted](#) ([CM\\_JobEventArgs](#) e)
 *Raises the job started event.*
- void [OnJobComplete](#) ([CM\\_JobEventArgs](#) e)
 *Raises the job complete event.*
- void [OnJobPaused](#) ([CM\\_JobEventArgs](#) e)
 *Raises the job paused event.*
- void [OnJobResumed](#) ([CM\\_JobEventArgs](#) e)
 *Raises the job resumed event.*
- void [OnChildJobsStarted](#) ([CM\\_JobEventArgs](#) e)
 *Raises the child jobs started event.*
- void [OnChildJobsComplete](#) ([CM\\_JobEventArgs](#) e)
 *Raises the child jobs complete event.*

## Properties

- string [id](#) [get, set]
 *Gets or sets the identifier. The identifier is a unique key used by [CM\\_JobManager](#) to reference individual jobs.*
- bool [running](#) [get]
 *Gets a value indicating whether this [CM\\_Job](#) is running.*
- bool [paused](#) [get]
 *Gets a value indicating whether this [CM\\_Job](#) is paused.*
- bool [jobKilled](#) [get]
 *Gets a value indicating whether this [CM\\_Job](#) job was killed or was allowed to complete.*
- bool [repeating](#) [get]
 *Gets a value indicating whether this [CM\\_Job](#) is repeating.*
- int [numOfTimesExecuted](#) [get]
 *Gets the number of times this job has been executed.*
- IEnumerator [coroutine](#) [get]
 *Gets the coroutine of this job.*

### 3.3.1 Detailed Description

The main coroutine job class. Encapsulates the behaviour for a single coroutine job. Provides access to status (i.e. running, paused, killed etc),

### 3.3.2 Member Function Documentation

#### 3.3.2.1 **CM\_Job** CM\_Job.AddChild ( **CM\_Job** *childJob* )

Adds a child job.

##### Returns

The child.

##### Parameters

<i>childJob</i>	Child job.
-----------------	------------

#### 3.3.2.2 **CM\_Job** CM\_Job.AddChild ( **IEnumerator** *childJob* )

Create a new job using the provided Enumerator and adds as a child job.

##### Returns

The child.

##### Parameters

<i>childJob</i>	Child job.
-----------------	------------

#### 3.3.2.3 **static CM\_Job [ ]** CM\_Job.Builder ( **params** **IEnumerator [ ]** *coroutines* ) **[static]**

Builds the specified coroutines into [CM\\_Job](#) instances.

##### Parameters

<i>coroutines</i>	The built jobs.
-------------------	-----------------

#### 3.3.2.4 **CM\_Job** CM\_Job.Clone ( )

Clone this instance.

#### 3.3.2.5 **CM\_Job [ ]** CM\_Job.Clone ( **int** *numOfCopies* )

Clone this instance.

##### Parameters

<i>numOfCopies</i>	Number of copies to create.
--------------------	-----------------------------

#### 3.3.2.6 **void** CM\_Job.Kill ( )

Kill this instance. Stops the running coroutine.

### 3.3.2.7 void CM\_Job.Kill ( float *delayInSeconds* )

Kill this instance. Stops the running coroutine after *delayInSeconds*.

## Parameters

<i>delayInSeconds</i>	Delay in seconds until instance killed.
-----------------------	---

3.3.2.8 static CM\_Job CM\_Job.Make ( IEnumerator *coroutine* ) [static]

Returns an initialised [CM\\_Job](#) instance. Provides static access to class.

## Parameters

<i>coroutine</i>	Coroutine.
------------------	------------

3.3.2.9 static CM\_Job CM\_Job.Make ( IEnumerator *coroutine*, string *id* ) [static]

Returns an initialised [CM\\_Job](#) instance with the specified id. Provides static access to class.

## Parameters

<i>coroutine</i>	Coroutine.
<i>id</i>	Identifier.

3.3.2.10 CM\_Job CM\_Job.NotifyOnChildJobComplete ( EventHandler< CM\_JobEventArgs > *e* )

Subscribes to the the childJobsComplete event.

## Parameters

<i>e</i>	The eventhandler to be invoked on event.
----------	--

3.3.2.11 CM\_Job CM\_Job.NotifyOnChildJobStarted ( EventHandler< CM\_JobEventArgs > *e* )

Subscribes to the the childJobsStarted event.

## Parameters

<i>e</i>	The eventhandler to be invoked on event.
----------	--

3.3.2.12 CM\_Job CM\_Job.NotifyOnJobComplete ( EventHandler< CM\_JobEventArgs > *e* )

Subscribes to the the jobComplete event.

## Parameters

<i>e</i>	The eventhandler to be invoked on event.
----------	--

3.3.2.13 CM\_Job CM\_Job.NotifyOnJobFinishedRunning ( EventHandler< CM\_JobEventArgs > *e* )

Subscribes to the jobFinishedRunning event

## Parameters

---

<i>e</i>	The eventhandler to be invoked on event.
----------	--

#### 3.3.2.14 **CM\_Job** **CM\_Job.NotifyOnJobPaused ( EventHandler< CM\_JobEventArgs > *e* )**

Subscribes to the job paused event.

##### Returns

The on job paused.

##### Parameters

<i>e</i>	E.
----------	----

#### 3.3.2.15 **CM\_Job** **CM\_Job.NotifyOnJobResumed ( EventHandler< CM\_JobEventArgs > *e* )**

Subscribes to the job resumed event.

##### Returns

The on job paused.

##### Parameters

<i>e</i>	E.
----------	----

#### 3.3.2.16 **CM\_Job** **CM\_Job.NotifyOnJobStarted ( EventHandler< CM\_JobEventArgs > *e* )**

Subscribes to the jobStarted event.

##### Parameters

<i>e</i>	The eventhandler to be invoked on event.
----------	--

#### 3.3.2.17 **void CM\_Job.OnChildJobsComplete ( CM\_JobEventArgs *e* )** [protected]

Raises the child jobs complete event.

##### Parameters

<i>e</i>	E.
----------	----

#### 3.3.2.18 **void CM\_Job.OnChildJobsStarted ( CM\_JobEventArgs *e* )** [protected]

Raises the child jobs started event.

##### Parameters

<i>e</i>	E.
----------	----

#### 3.3.2.19 **void CM\_Job.OnJobComplete ( CM\_JobEventArgs *e* )** [protected]

Raises the job complete event.

## Parameters

<i>e</i>	E.
----------	----

**3.3.2.20** void CM\_Job.OnJobFinishedRunning ( CM\_JobEventArgs *e* ) [protected]

Raises the job finished running event.

## Parameters

<i>e</i>	E.
----------	----

**3.3.2.21** void CM\_Job.OnJobPaused ( CM\_JobEventArgs *e* ) [protected]

Raises the job paused event.

## Parameters

<i>e</i>	E.
----------	----

**3.3.2.22** void CM\_Job.OnJobResumed ( CM\_JobEventArgs *e* ) [protected]

Raises the job resumed event.

## Parameters

<i>e</i>	E.
----------	----

**3.3.2.23** void CM\_Job.OnJobStarted ( CM\_JobEventArgs *e* ) [protected]

Raises the job started event.

## Parameters

<i>e</i>	E.
----------	----

**3.3.2.24** CM\_Job CM\_Job.Pause ( )

Pause this instance.

**3.3.2.25** CM\_Job CM\_Job.Pause ( float *delayInSeconds* )

Pause the specified instance after delayInSeconds.

## Parameters

<i>delayInSeconds</i>	Delay in seconds until instance is paused.
-----------------------	--

**3.3.2.26** CM\_Job CM\_Job.RemoveChildJob ( CM\_Job *childJob* )

Removes a child job if present.

**Returns**

The child job.

**Parameters**

<i>childJob</i>	Child job.
-----------------	------------

**3.3.2.27 CM\_Job CM\_Job.RemoveNotifyOnChildJobComplete ( EventHandler< CM\_JobEventArgs > e )**

Unsubscribes to the the childJobsComplete event.

**Parameters**

<i>e</i>	The eventhandler to be unsubscribed.
----------	--------------------------------------

**3.3.2.28 CM\_Job CM\_Job.RemoveNotifyOnChildJobStarted ( EventHandler< CM\_JobEventArgs > e )**

Unsubscribes to the the childJobsStarted event.

**Parameters**

<i>e</i>	The eventhandler to be unsubscribed.
----------	--------------------------------------

**3.3.2.29 CM\_Job CM\_Job.RemoveNotifyOnJobComplete ( EventHandler< CM\_JobEventArgs > e )**

Unsubscribes to the the jobComplete event.

**Parameters**

<i>e</i>	The eventhandler to be unsubscribed.
----------	--------------------------------------

**3.3.2.30 CM\_Job CM\_Job.RemoveNotifyOnJobFinishedRunning ( EventHandler< CM\_JobEventArgs > e )**

Unsubscribes to the jobFinishedRunning event

**Parameters**

<i>e</i>	The eventhandler to be invoked on event.
----------	--

**3.3.2.31 CM\_Job CM\_Job.RemoveNotifyOnJobPaused ( EventHandler< CM\_JobEventArgs > e )**

Unsubscribes to the job paused event.

**Parameters**

<i>e</i>	The eventhandler to be unsubscribed.
----------	--------------------------------------

**3.3.2.32 CM\_Job CM\_Job.RemoveNotifyOnJobResumed ( EventHandler< CM\_JobEventArgs > e )**

Unsubscribes to the job resumed event.



## Parameters

<i>e</i>	The eventhandler to be unsubscribed.
----------	--------------------------------------

**3.3.2.33 CM\_Job CM\_Job.RemoveNotifyOnJobStarted ( EventHandler< CM\_JobEventArgs > e )**

Unsubscribes to the jobStarted event.

## Parameters

<i>e</i>	The eventhandler to be unsubscribed.
----------	--------------------------------------

**3.3.2.34 CM\_Job CM\_Job.Repeat ( )**

Sets this instance to repeat. The job is repeated when it has finished processing.

**3.3.2.35 CM\_Job CM\_Job.Repeat ( int numOfTimes )**

Sets this instance to repeat. The job is repeated a set number of times.

**3.3.2.36 CM\_Job CM\_Job.Resume ( )**

Resume this instance.

**3.3.2.37 CM\_Job CM\_Job.Resume ( float delayInSeconds )**

Resume the specified instance after delayInSeconds.

## Parameters

<i>delayInSeconds</i>	Delay in seconds until instance is resumed.
-----------------------	---

**3.3.2.38 CM\_Job CM\_Job.Start ( )**

Start this instance. Runs the coroutine immediately.

**3.3.2.39 CM\_Job CM\_Job.Start ( float delayInSeconds )**

Start the specified instance after delayInSeconds. The coroutine is added to [CM\\_Dispatcher](#) job queue to be executed in the next timestep as a coroutine cannot be started in a separate thread.

## Parameters

<i>delayInSeconds</i>	Delay in seconds until instance is processed.
-----------------------	---

**3.3.2.40 CM\_Job CM\_Job.StopRepeat ( float delayInSeconds )**

Stops the repeat after a specified delay in seconds.

## Returns

The repeat.

## Parameters

<i>delayInSeconds</i>	Delay in seconds.
-----------------------	-------------------

### 3.3.3 Property Documentation

#### 3.3.3.1 IEnumerator `CM_Job.coroutine` [get]

Gets the coroutine of this job.

The coroutine.

#### 3.3.3.2 string `CM_Job.id` [get], [set]

Gets or sets the identifier. The identifier is a unique key used by [CM\\_JobManager](#) to reference individual jobs.

The identifier.

#### 3.3.3.3 bool `CM_Job.jobKilled` [get]

Gets a value indicating whether this [CM\\_Job](#) job was killed or was allowed to complete.

true if job killed; otherwise, false.

#### 3.3.3.4 int `CM_Job.numOfTimesExecuted` [get]

Gets the number of times this job has been executed.

The number of times executed.

#### 3.3.3.5 bool `CM_Job.paused` [get]

Gets a value indicating whether this [CM\\_Job](#) is paused.

true if paused; otherwise, false.

#### 3.3.3.6 bool `CM_Job.repeating` [get]

Gets a value indicating whether this [CM\\_Job](#) is repeating.

true if repeating; otherwise, false.

#### 3.3.3.7 bool `CM_Job.running` [get]

Gets a value indicating whether this [CM\\_Job](#) is running.

true if running; otherwise, false.

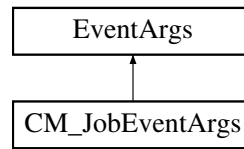
The documentation for this class was generated from the following file:

- `CM_Job.cs`

## 3.4 CM\_JobEventArgs Class Reference

Arguments used in events raised by [CM\\_Job](#).

Inheritance diagram for CM\_JobEventArgs:



## Public Member Functions

- [CM\\_JobEventArgs](#) ([CM\\_Job](#) job, [CM\\_Job\[\]](#) childJobs)  
*Initializes a new instance of the [CM\\_JobEventArgs](#) class.*

## Properties

- [CM\\_Job](#) job [get]  
*Gets the current job.*
- [CM\\_Job\[\]](#) childJobs [get]  
*Gets the child jobs (if any).*
- bool [hasChildJobs](#) [get]  
*Gets a value indicating whether this [CM\\_JobEventArgs](#) has child jobs.*

### 3.4.1 Detailed Description

Arguments used in events raised by [CM\\_Job](#).

### 3.4.2 Constructor & Destructor Documentation

#### 3.4.2.1 CM\_JobEventArgs.CM\_JobEventArgs ( CM\_Job job, CM\_Job[] childJobs )

Initializes a new instance of the [CM\\_JobEventArgs](#) class.

Parameters

<i>job</i>	Job.
<i>childJobs</i>	Child jobs.

### 3.4.3 Property Documentation

#### 3.4.3.1 CM\_Job[] CM\_JobEventArgs.childJobs [get]

Gets the child jobs (if any).

The child jobs.

#### 3.4.3.2 bool CM\_JobEventArgs.hasChildJobs [get]

Gets a value indicating whether this [CM\\_JobEventArgs](#) has child jobs.

true if has child jobs; otherwise, false.

### 3.4.3.3 CM\_Job CM\_JobEventArgs.job [get]

Gets the current job.

The job.

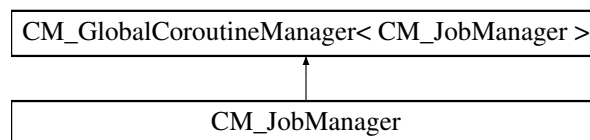
The documentation for this class was generated from the following file:

- CM\_JobEventArgs.cs

## 3.5 CM\_JobManager Class Reference

The main job manager class. Encapsulates the behaviour for global and local job managers. Provides access to events and public access to stored jobs.

Inheritance diagram for CM\_JobManager:



### Public Member Functions

- [CM\\_JobManager AddJob](#) ([CM\\_Job](#) job)  
*Adds a job to the job manager.*
- [CM\\_JobManager AddJob](#) (string id, IEnumerator routine)  
*Creates a job with the specified id and routine and adds to job manager.*
- [CM\\_JobManager AddJob](#) (IList< [CM\\_Job](#) > jobs)  
*Adds the provided jobs to the job manager.*
- [CM\\_JobManager AddJob](#) (params [CM\\_Job](#)[] jobs)  
*Adds the provided jobs to the job manager.*
- [CM\\_JobManager RemoveJob](#) ([CM\\_Job](#) job)  
*Removes the job if owned by this instance of job manager.*
- [CM\\_JobManager RemoveJob](#) (string id)  
*Removes the job if owned by this instance of job manager.*
- [CM\\_JobManager StartCoroutine](#) ([CM\\_Job](#) job)  
*Starts the specified coroutine if owned by this instance of job manager. Job is searched using job id.*
- [CM\\_JobManager StartCoroutine](#) (string id)  
*Starts the specified coroutine if owned by this instance of job manager.*
- [CM\\_JobManager StartAll](#) ()  
*Starts all jobs owned by this instance.*
- [CM\\_JobManager StartAll](#) (float delayInSeconds)  
*Starts all jobs owned by this instance after delay in seconds.*
- [CM\\_JobManager StopCoroutine](#) ([CM\\_Job](#) job)  
*Stops the specified coroutine if owned by this instance of job manager. Job is searched using job id.*
- [CM\\_JobManager StopCoroutine](#) (string id)  
*Stops the specified coroutine if owned by this instance of job manager.*
- [CM\\_JobManager PauseCoroutine](#) ([CM\\_Job](#) job)  
*Pauses the specified coroutine if owned by this instance of job manager. Job is searched using job id.*
- [CM\\_JobManager PauseCoroutine](#) (string id)

- Stops the specified coroutine if owned by this instance of job manager.*
- [CM\\_JobManager ResumeCoroutine](#) ([CM\\_Job](#) job)
  - Resumes the specified coroutine if owned by this instance of job manager. Job is searched using job id.*
- [CM\\_JobManager ResumeCoroutine](#) (string id)
  - Stops the specified coroutine if owned by this instance of job manager.*
- [CM\\_JobManager PauseAll](#) ()
  - Pauses all jobs owned by this instance. Raises allJobsPaused event.*
- [CM\\_JobManager PauseAll](#) (float delayInSeconds)
  - Pauses all jobs owned by this instance after delay in seconds. Raises allJobsPaused event.*
- [CM\\_JobManager ResumeAll](#) ()
  - Resumes all jobs owned by this instance. Raises allJobsResumedEvent.*
- [CM\\_JobManager ResumeAll](#) (float delayInSeconds)
  - Resumes all jobs owned by this instance after delay in seconds. Raises allJobsResumedEvent.*
- [CM\\_JobManager KillAll](#) ()
  - Kills all jobs owned by this instance. Raises allJobsKilled event.*
- [CM\\_JobManager KillAll](#) (float delayInSeconds)
  - Kills all jobs owned by this instance after delay in seconds. Raises allJobsKilled event.*
- [CM\\_JobManager ClearJobList](#) ()
  - Clears the job list owned by this instance. It does not kill the jobs so they will continue to run. Raises allJobsCleared event.*
- bool [HasJob](#) (string id)
  - Determines whether this instance has the job with the specified id.*
- bool [IsRunning](#) ([CM\\_Job](#) job)
  - Determines whether the specified job is executing.*
- bool [IsRunning](#) (string id)
  - Determines whether the specified job is executing.*
- [CM\\_JobManager NotifyOnJobAdded](#) (EventHandler< [CM\\_JobManagerJobEditedEventArgs](#) > e)
  - Subscribes to the the jobAdded event.*
- [CM\\_JobManager RemoveNotifyOnJobAdded](#) (EventHandler< [CM\\_JobManagerJobEditedEventArgs](#) > e)
  - Unsubscribes to the the jobAdded event.*
- [CM\\_JobManager NotifyOnJobRemoved](#) (EventHandler< [CM\\_JobManagerJobEditedEventArgs](#) > e)
  - Subscribes to the the jobRemoved event.*
- [CM\\_JobManager RemoveNotifyOnJobRemoved](#) (EventHandler< [CM\\_JobManagerJobEditedEventArgs](#) > e)
  - Unsubscribes to the the jobRemoved event.*
- [CM\\_JobManager NotifyOnAllJobsPaused](#) (EventHandler< [CM\\_JobManagerEventArgs](#) > e)
  - Subscribes to the the jobPaused event.*
- [CM\\_JobManager RemoveNotifyOnAllJobsPaused](#) (EventHandler< [CM\\_JobManagerEventArgs](#) > e)
  - Unsubscribes to the the jobPaused event.*
- [CM\\_JobManager NotifyOnAllJobsResumed](#) (EventHandler< [CM\\_JobManagerEventArgs](#) > e)
  - Subscribes to the the jobResumed event.*
- [CM\\_JobManager RemoveNotifyOnAllJobsResumed](#) (EventHandler< [CM\\_JobManagerEventArgs](#) > e)
  - Unsubscribes to the the jobResumed event.*
- [CM\\_JobManager NotifyOnAllJobsKilled](#) (EventHandler< [CM\\_JobManagerEventArgs](#) > e)
  - Subscribes to the the allJobsKilled event.*
- [CM\\_JobManager RemoveNotifyOnAllJobsKilled](#) (EventHandler< [CM\\_JobManagerEventArgs](#) > e)
  - Unsubscribes to the the allJobsKilled event.*
- [CM\\_JobManager NotifyOnAllJobsCleared](#) (EventHandler< [CM\\_JobManagerEventArgs](#) > e)
  - Subscribes to the the allJobsCleared event.*
- [CM\\_JobManager RemoveNotifyOnAllJobsCleared](#) (EventHandler< [CM\\_JobManagerEventArgs](#) > e)
  - Unsubscribes to the the allJobsCleared event.*

- void [OnJobRemoved](#) ([CM\\_JobManagerJobEditedEventArgs](#) e)  
*Raises the job removed event.*
- void [OnAllJobsResumed](#) ([CM\\_JobManagerEventArgs](#) e)  
*Raises the all jobs resumed event.*
- void [OnAllJobsPaused](#) ([CM\\_JobManagerEventArgs](#) e)  
*Raises the all jobs paused event.*
- void [OnAllJobsKilled](#) ([CM\\_JobManagerEventArgs](#) e)  
*Raises the all jobs killed event.*
- void [OnAllJobsCleared](#) ([CM\\_JobManagerEventArgs](#) e)  
*Raises the all jobs cleared event.*

## Static Public Member Functions

- static [CM\\_JobManager Make](#) ()  
*Returns an initialised [CM\\_JobManager](#) instance. Provides static access to class.*

## Protected Member Functions

- void [OnJobAdded](#) ([CM\\_JobManagerJobEditedEventArgs](#) e)  
*Raises the job added event.*
- override void **HandleJobComplete** (object sender, [CM\\_JobEventArgs](#) e)

## Events

- EventHandler< [CM\\_JobManagerJobEditedEventArgs](#) > [jobAdded](#)  
*Raised when job added.*
- EventHandler< [CM\\_JobManagerJobEditedEventArgs](#) > [jobRemoved](#)  
*Raised when job removed.*
- EventHandler< [CM\\_JobManagerEventArgs](#) > [allJobsKilled](#)  
*Raised when all jobs killed.*
- EventHandler< [CM\\_JobManagerEventArgs](#) > [allJobsResumed](#)  
*Raised when all jobs resumed.*
- EventHandler< [CM\\_JobManagerEventArgs](#) > [allJobsPaused](#)  
*Raised when all jobs paused.*
- EventHandler< [CM\\_JobManagerEventArgs](#) > [allJobsCleared](#)  
*Raised when all jobs cleared.*

## Additional Inherited Members

### 3.5.1 Detailed Description

The main job manager class. Encapsulates the behaviour for global and local job managers. Provides access to events and public access to stored jobs.

### 3.5.2 Member Function Documentation

#### 3.5.2.1 [CM\\_JobManager](#) [CM\\_JobManager.AddJob](#) ( [CM\\_Job](#) job )

Adds a job to the job manager.

**Returns**

The job.

**Parameters**

<i>job</i>	Job.
------------	------

**3.5.2.2 CM\_JobManager CM\_JobManager.AddJob ( string *id*, IEnumerator *routine* )**

Creates a job with the specified id and routine and adds to job manager.

**Returns**

The job manager.

**Parameters**

<i>id</i>	Identifier of job.
<i>routine</i>	Routine.

**3.5.2.3 CM\_JobManager CM\_JobManager.AddJob ( IList< CM\_Job > *jobs* )**

Adds the provided jobs to the job manager.

**Returns**

The job manager.

**Parameters**

<i>jobs</i>	Jobs to add to this instance.
-------------	-------------------------------

**3.5.2.4 CM\_JobManager CM\_JobManager.AddJob ( params CM\_Job[] *jobs* )**

Adds the provided jobs to the job manager.

**Returns**

The job manager.

**Parameters**

<i>jobs</i>	Jobs to add to this instance.
-------------	-------------------------------

**3.5.2.5 CM\_JobManager CM\_JobManager.ClearJobList ( )**

Clears the job list owned by this instance. It does not kill the jobs so they will continue to run. Raises allJobsCleared event.

**Returns**

The job list.

### 3.5.2.6 bool CM\_JobManager.HasJob ( string *id* )

Determines whether this instance has the job with the specified id.

#### Returns

true if this instance has a job with the specified id; otherwise, false.

#### Parameters

<i>id</i>	Identifier.
-----------	-------------

### 3.5.2.7 bool CM\_JobManager.IsRunning ( CM\_Job *job* )

Determines whether the specified job is executing.

#### Returns

true if the job is currently running; otherwise, false.

#### Parameters

<i>job</i>	Job.
------------	------

### 3.5.2.8 bool CM\_JobManager.IsRunning ( string *id* )

Determines whether the specified job is executing.

#### Returns

true if the job is currently running; otherwise, false.

#### Parameters

<i>job</i>	Job ID.
------------	---------

### 3.5.2.9 CM\_JobManager CM\_JobManager.KillAll ( )

Kills all jobs owned by this instance. Raises allJobsKilled event.

#### Returns

The all.

### 3.5.2.10 CM\_JobManager CM\_JobManager.KillAll ( float *delayInSeconds* )

Kills all jobs owned by this instance after delay in seconds. Raises allJobsKilled event.

#### Returns

The all.



## Parameters

<i>delayInSeconds</i>	Delay in seconds.
-----------------------	-------------------

## 3.5.2.11 static CM\_JobManager CM\_JobManager.Make ( ) [static]

Returns an initialised [CM\\_JobManager](#) instance. Provides static access to class.

## 3.5.2.12 CM\_JobManager CM\_JobManager.NotifyOnAllJobsCleared ( EventHandler&lt; CM\_JobManagerEventArgs &gt; e )

Subscribes to the the allJobsCleared event.

## Parameters

<i>e</i>	The eventhandler to be invoked on event.
----------	--

## 3.5.2.13 CM\_JobManager CM\_JobManager.NotifyOnAllJobsKilled ( EventHandler&lt; CM\_JobManagerEventArgs &gt; e )

Subscribes to the the allJobsKilled event.

## Parameters

<i>e</i>	The eventhandler to be invoked on event.
----------	--

## 3.5.2.14 CM\_JobManager CM\_JobManager.NotifyOnAllJobsPaused ( EventHandler&lt; CM\_JobManagerEventArgs &gt; e )

Subscribes to the the jobPaused event.

## Parameters

<i>e</i>	The eventhandler to be invoked on event.
----------	--

## 3.5.2.15 CM\_JobManager CM\_JobManager.NotifyOnAllJobsResumed ( EventHandler&lt; CM\_JobManagerEventArgs &gt; e )

Subscribes to the the jobResumed event.

## Parameters

<i>e</i>	The eventhandler to be invoked on event.
----------	--

## 3.5.2.16 CM\_JobManager CM\_JobManager.NotifyOnJobAdded ( EventHandler&lt; CM\_JobManagerJobEditedEventArgs &gt; e )

Subscribes to the the jobAdded event.

## Parameters

<i>e</i>	The eventhandler to be invoked on event.
----------	--

**3.5.2.17 CM\_JobManager** **CM\_JobManager.NotifyOnJobRemoved** ( **EventHandler** < **CM\_JobManagerJobEdited**↔  
**EventArgs** > *e* )

Subscribes to the the jobRemoved event.

## Parameters

<i>e</i>	The eventhandler to be invoked on event.
----------	--

**3.5.2.18 void CM\_JobManager.OnAllJobsCleared ( CM\_JobManagerEventArgs *e* )**

Raises the all jobs cleared event.

## Parameters

<i>e</i>	E.
----------	----

**3.5.2.19 void CM\_JobManager.OnAllJobsKilled ( CM\_JobManagerEventArgs *e* )**

Raises the all jobs killed event.

## Parameters

<i>e</i>	E.
----------	----

**3.5.2.20 void CM\_JobManager.OnAllJobsPaused ( CM\_JobManagerEventArgs *e* )**

Raises the all jobs paused event.

## Parameters

<i>e</i>	E.
----------	----

**3.5.2.21 void CM\_JobManager.OnAllJobsResumed ( CM\_JobManagerEventArgs *e* )**

Raises the all jobs resumed event.

## Parameters

<i>e</i>	E.
----------	----

**3.5.2.22 void CM\_JobManager.OnJobAdded ( CM\_JobManagerJobEditedEventArgs *e* ) [protected]**

Raises the job added event.

## Parameters

<i>e</i>	E.
----------	----

**3.5.2.23 void CM\_JobManager.OnJobRemoved ( CM\_JobManagerJobEditedEventArgs *e* )**

Raises the job removed event.

## Parameters

<i>e</i>	E.
----------	----

**3.5.2.24 CM\_JobManager CM\_JobManager.PauseAll ( )**

Pauses all jobs owned by this instance. Raises allJobsPaused event.

**Returns**

The all.

**3.5.2.25 CM\_JobManager CM\_JobManager.PauseAll ( float *delayInSeconds* )**

Pauses all jobs owned by this instance after delay in seconds. Raises allJobsPaused event.

**Returns**

The all.

**Parameters**

<i>delayInSeconds</i>	Delay in seconds.
-----------------------	-------------------

**3.5.2.26 CM\_JobManager CM\_JobManager.PauseCoroutine ( CM\_Job *job* )**

Pauses the specified coroutine if owned by this instance of job manager. Job is searched using job id.

**Returns**

The job manager.

**Parameters**

<i>job</i>	Job.
------------	------

**3.5.2.27 CM\_JobManager CM\_JobManager.PauseCoroutine ( string *id* )**

Stops the specified coroutine if owned by this instance of job manager.

**Returns**

The job manager.

**Parameters**

<i>job</i>	Job.
------------	------

**3.5.2.28 CM\_JobManager CM\_JobManager.RemoveJob ( CM\_Job *job* )**

Removes the job if owned by this instance of job manager.

**Returns**

The job manager.

## Parameters

<i>job</i>	Job.
------------	------

3.5.2.29 CM\_JobManager CM\_JobManager.RemoveJob ( string *id* )

Removes the job if owned by this instance of job manager.

## Returns

The job manager.

## Parameters

<i>job</i>	Job.
------------	------

3.5.2.30 CM\_JobManager CM\_JobManager.RemoveNotifyOnAllJobsCleared ( EventHandler< CM\_JobManagerEventArgs > *e* )

Unsubscribes to the the allJobsCleared event.

## Parameters

<i>e</i>	The eventhandler to be invoked on event.
----------	--

3.5.2.31 CM\_JobManager CM\_JobManager.RemoveNotifyOnAllJobsKilled ( EventHandler< CM\_JobManagerEventArgs > *e* )

Unsubscribes to the the allJobsKilled event.

## Parameters

<i>e</i>	The eventhandler to be invoked on event.
----------	--

3.5.2.32 CM\_JobManager CM\_JobManager.RemoveNotifyOnAllJobsPaused ( EventHandler< CM\_JobManagerEventArgs > *e* )

Unsubscribes to the the jobPaused event.

## Parameters

<i>e</i>	The eventhandler to be invoked on event.
----------	--

3.5.2.33 CM\_JobManager CM\_JobManager.RemoveNotifyOnAllJobsResumed ( EventHandler< CM\_JobManagerEventArgs > *e* )

Unsubscribes to the the jobResumed event.

## Parameters

<i>e</i>	The eventhandler to be invoked on event.
----------	--

3.5.2.34 **CM\_JobManager** **CM\_JobManager.RemoveNotifyOnJobAdded** ( **EventHandler**<  
**CM\_JobManagerJobEditedEventArgs** > *e* )

Unsubscribes to the the jobAdded event.

## Parameters

<i>e</i>	The eventhandler to be invoked on event.
----------	--

### 3.5.2.35 CM\_JobManager CM\_JobManager.RemoveNotifyOnJobRemoved ( EventHandler< CM\_JobManagerJobEditedEventArgs > *e* )

Unsubscribes to the the jobRemoved event.

## Parameters

<i>e</i>	The eventhandler to be invoked on event.
----------	--

### 3.5.2.36 CM\_JobManager CM\_JobManager.ResumeAll ( )

Resumes all jobs owned by this instance. Raises allJobsResumedEvent.

## Returns

The all.

### 3.5.2.37 CM\_JobManager CM\_JobManager.ResumeAll ( float *delayInSeconds* )

Resumes all jobs owned by this instance after delay in seconds. Raises allJobsResumedEvent.

## Returns

The all.

## Parameters

<i>delayInSeconds</i>	Delay in seconds.
-----------------------	-------------------

### 3.5.2.38 CM\_JobManager CM\_JobManager.ResumeCoroutine ( CM\_Job *job* )

Resumes the specified coroutine if owned by this instance of job manager. Job is searched using job id.

## Returns

The job manager.

## Parameters

<i>job</i>	Job.
------------	------

### 3.5.2.39 CM\_JobManager CM\_JobManager.ResumeCoroutine ( string *id* )

Stops the specified coroutine if owned by this instance of job manager.

## Returns

The job manager.

## Parameters

<i>job</i>	Job.
------------	------

**3.5.2.40 CM\_JobManager CM\_JobManager.StartAll ( )**

Starts all jobs owned by this instance.

## Returns

The all.

**3.5.2.41 CM\_JobManager CM\_JobManager.StartAll ( float *delayInSeconds* )**

Starts all jobs owned by this instance after delay in seconds.

## Returns

The all.

## Parameters

<i>delayInSeconds</i>	Delay in seconds.
-----------------------	-------------------

**3.5.2.42 CM\_JobManager CM\_JobManager.StartCoroutine ( CM\_Job *job* )**

Starts the specified coroutine if owned by this instance of job manager. Job is searched using job id.

## Returns

The job manager.

## Parameters

<i>job</i>	Job.
------------	------

**3.5.2.43 CM\_JobManager CM\_JobManager.StartCoroutine ( string *id* )**

Starts the specified coroutine if owned by this instance of job manager.

## Returns

The job manager.

## Parameters

<i>job</i>	Job.
------------	------

**3.5.2.44 CM\_JobManager CM\_JobManager.StopCoroutine ( CM\_Job *job* )**

Stops the specified coroutine if owned by this instance of job manager. Job is searched using job id.

## Returns

The job manager.



## Parameters

<i>job</i>	Job.
------------	------

## 3.5.2.45 CM\_JobManager CM\_JobManager.StopCoroutine ( string id )

Stops the specified coroutine if owned by this instance of job manager.

## Returns

The job manager.

## Parameters

<i>job</i>	Job.
------------	------

## 3.5.3 Event Documentation

## 3.5.3.1 EventHandler&lt;CM\_JobManagerEventArgs&gt; CM\_JobManager.allJobsCleared [protected]

Raised when all jobs cleared.

## 3.5.3.2 EventHandler&lt;CM\_JobManagerEventArgs&gt; CM\_JobManager.allJobsKilled [protected]

Raised when all jobs killed.

## 3.5.3.3 EventHandler&lt;CM\_JobManagerEventArgs&gt; CM\_JobManager.allJobsPaused [protected]

Raised when all jobs paused.

## 3.5.3.4 EventHandler&lt;CM\_JobManagerEventArgs&gt; CM\_JobManager.allJobsResumed [protected]

Raised when all jobs resumed.

## 3.5.3.5 EventHandler&lt;CM\_JobManagerJobEditedEventArgs&gt; CM\_JobManager.jobAdded [protected]

Raised when job added.

## 3.5.3.6 EventHandler&lt;CM\_JobManagerJobEditedEventArgs&gt; CM\_JobManager.jobRemoved [protected]

Raised when job removed.

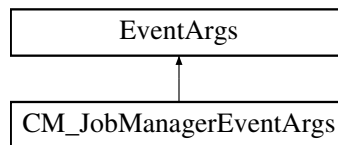
The documentation for this class was generated from the following file:

- CM\_JobManager.cs

## 3.6 CM\_JobManagerEventArgs Class Reference

Arguments used in events raised by [CM\\_JobManager](#).

Inheritance diagram for CM\_JobManagerEventArgs:



## Public Member Functions

- [CM\\_JobManagerEventArgs](#) (Dictionary< string, [CM\\_Job](#) > [ownedJobs](#))

*Initializes a new instance of the [CM\\_JobManagerEventArgs](#) class.*

## Properties

- [CM\\_Job\[\]](#) [ownedJobs](#) [get]  
*Gets the owned jobs of the job manager at the time of the event being raised.*
- [CM\\_Job\[\]](#) [runningJobs](#) [get]  
*Gets the currently running jobs of the job manager at the time of the event being raised.*
- [CM\\_Job\[\]](#) [pausedJobs](#) [get]  
*Gets the paused jobs of the job manager at the time of the event being raised.*

### 3.6.1 Detailed Description

Arguements used in events raised by [CM\\_JobManager](#).

### 3.6.2 Constructor & Destructor Documentation

#### 3.6.2.1 [CM\\_JobManagerEventArgs.CM\\_JobManagerEventArgs](#) ( Dictionary< string, [CM\\_Job](#) > [ownedJobs](#) )

Initializes a new instance of the [CM\\_JobManagerEventArgs](#) class.

#### Parameters

<a href="#">ownedJobs</a>	Owned jobs.
---------------------------	-------------

### 3.6.3 Property Documentation

#### 3.6.3.1 [CM\\_Job \[\]](#) [CM\\_JobManagerEventArgs.ownedJobs](#) [get]

Gets the owned jobs of the job manager at the time of the event being raised.

The owned jobs.

#### 3.6.3.2 [CM\\_Job \[\]](#) [CM\\_JobManagerEventArgs.pausedJobs](#) [get]

Gets the paused jobs of the job manager at the time of the event being raised.

The paused jobs.

### 3.6.3.3 CM\_Job [] CM\_JobManagerEventArgs.runningJobs [get]

Gets the currently running jobs of the job manager at the time of the event being raised.

The running jobs.

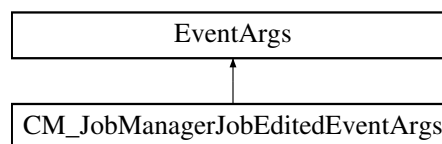
The documentation for this class was generated from the following file:

- CM\_JobManagerEventArgs.cs

## 3.7 CM\_JobManagerJobEditedEventArgs Class Reference

Arguments used in events raised by [CM\\_JobManager](#).

Inheritance diagram for CM\_JobManagerJobEditedEventArgs:



### Public Member Functions

- [CM\\_JobManagerJobEditedEventArgs](#) (Dictionary< string, [CM\\_Job](#) > ownedJobs, [CM\\_Job](#) jobEdited)  
*Initializes a new instance of the [CM\\_JobManagerJobEditedEventArgs](#) class.*

### Properties

- [CM\\_JobManagerEventArgs](#) otherArgs [get]  
*Gets the other arguments.*
- [CM\\_Job](#) jobEdited [get]  
*Gets the job currently changed that caused the event to be raised.*

### 3.7.1 Detailed Description

Arguments used in events raised by [CM\\_JobManager](#).

### 3.7.2 Constructor & Destructor Documentation

#### 3.7.2.1 CM\_JobManagerJobEditedEventArgs.CM\_JobManagerJobEditedEventArgs ( Dictionary< string, CM\_Job > ownedJobs, CM\_Job jobEdited )

Initializes a new instance of the [CM\\_JobManagerJobEditedEventArgs](#) class.

Parameters

<i>ownedJobs</i>	Owned jobs.
<i>jobEdited</i>	Job edited.

### 3.7.3 Property Documentation

#### 3.7.3.1 **CM\_Job** **CM\_JobManagerJobEditedEventArgs.jobEdited** [get]

Gets the job currently changed that caused the event to be raised.

The job edited.

#### 3.7.3.2 **CM\_JobManagerEventArgs** **CM\_JobManagerJobEditedEventArgs.otherArgs** [get]

Gets the other arguments.

The other arguments.

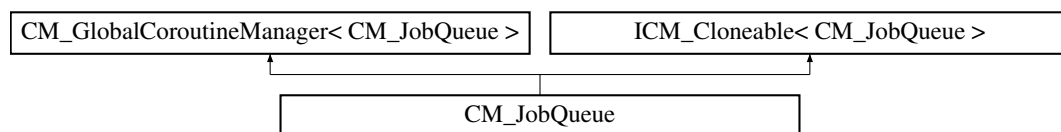
The documentation for this class was generated from the following file:

- CM\_JobManagerJobEditedEventArgs.cs

## 3.8 CM\_JobQueue Class Reference

The main job queue class. Encapsulates all behaviour related to queueing a job. Provides access to events, and status (i.e. running, repeating).

Inheritance diagram for CM\_JobQueue:



### Public Member Functions

- **CM\_JobQueue Clone** ()  
*Clone this instance.*
- **CM\_JobQueue[] Clone** (int numOfCopies)  
*Clone this instance the specified numOfCopies.*
- **CM\_JobQueue Enqueue** (CM\_JobQueue other)  
*Enqueues the specified other queue. Adds the jobs from one queue to this queue and also adds the other queues event subscriptions.*
- **CM\_JobQueue Enqueue** (params CM\_Job[] jobs)  
*Enqueues the specified jobs.*
- **CM\_JobQueue Enqueue** (IEnumerable< CM\_Job > jobs)  
*Enqueues the specified jobs.*
- **CM\_JobQueue Enqueue** (string id, IEnumerator routine)  
*Creates a new job with specified id and coroutine and adds job to queue.*
- **CM\_JobQueue Enqueue** (IEnumerator routine)  
*Creates a new job with specified id and coroutine and adds job to queue.*
- **CM\_JobQueue Enqueue** (CM\_Job job)  
*Enqueues the specified job.*
- **CM\_JobQueue Start** ()  
*Start this instance of the queue immediately.*
- **CM\_JobQueue Start** (float delayInSeconds)

- Start the specified instance after delayInSeconds.*

  - [CM\\_JobQueue Repeat](#) ()
 

*Sets this instance to repeat. The job is repeated when it has finished processing.*
  - [CM\\_JobQueue Repeat](#) (int numOfTimes)
 

*Sets this instance to repeat a number of times. The job is repeated when it has finished processing.*
  - [CM\\_JobQueue StopRepeat](#) ()
 

*Stops the repeat.*
  - [CM\\_JobQueue StopRepeat](#) (float delayInSeconds)
 

*Stops the repeat after a specified delay in seconds.*
  - [CM\\_JobQueue Pause](#) ()
 

*Pauses this instance.*
  - [CM\\_JobQueue Pause](#) (float delayInSeconds)
 

*Pause this instance after the specified delayInSeconds.*
  - [CM\\_JobQueue Resume](#) ()
 

*Resume this instance immediately.*
  - [CM\\_JobQueue Resume](#) (float delayInSeconds)
 

*Resume the instance after the specified delayInSeconds.*
  - [CM\\_JobQueue ContinuousRunning](#) ()
 

*Set the queue to run continuously.*
  - [CM\\_JobQueue StopContinuousRunning](#) ()
 

*Stops the continuous running of this queue.*
  - [CM\\_JobQueue KillAll](#) ()
 

*Kill all currently queued jobs immediately. Clears queue list.*
  - [CM\\_JobQueue KillAll](#) (float delayInSeconds)
 

*Kill all currently queued jobs after the specified delayInSeconds.*
  - [CM\\_JobQueue KillCurrent](#) ()
 

*Kills the current running job immediately.*
  - [CM\\_JobQueue KillCurrent](#) (float delayInSeconds)
 

*Kills the current running job after the specified delayInSeconds.*
  - [CM\\_JobQueue NotifyOnQueueStarted](#) (EventHandler< [CM\\_QueueEventArgs](#) > e)
 

*Subscribes to the queue started event.*
  - [CM\\_JobQueue RemoveNotifyOnQueueStarted](#) (EventHandler< [CM\\_QueueEventArgs](#) > e)
 

*Unsubscribes to the the queue started event.*
  - [CM\\_JobQueue NotifyOnQueueComplete](#) (EventHandler< [CM\\_QueueEventArgs](#) > e)
 

*Subscribes to the queue completed event.*
  - [CM\\_JobQueue RemoveNotifyOnQueueComplete](#) (EventHandler< [CM\\_QueueEventArgs](#) > e)
 

*Unsubscribes to the queue completed event.*
  - [CM\\_JobQueue NotifyOnJobProcessed](#) (EventHandler< [CM\\_QueueEventArgs](#) > e)
 

*Subscribes to the the job processed event.*
  - [CM\\_JobQueue RemoveNotifyOnJobProcessed](#) (EventHandler< [CM\\_QueueEventArgs](#) > e)
 

*Unsubscribes to the the job processed event. This event is invoked every time a job in the queue has finished running.*

### Static Public Member Functions

- static [CM\\_JobQueue Make](#) ()
 

*Returns an initialised [CM\\_JobQueue](#) instance. Provides static access to class.*

## Protected Member Functions

- void [OnQueueStarted](#) ([CM\\_QueueEventArgs](#) e)  
*Raises the queue started event.*
- void [OnQueueComplete](#) ([CM\\_QueueEventArgs](#) e)  
*Raises the queue complete event.*
- void [OnJobProcessed](#) ([CM\\_QueueEventArgs](#) e)  
*Raises the job processed event.*
- override void [HandleJobComplete](#) (object sender, [CM\\_JobEventArgs](#) e)  
*Invoked whenever a queued job has finished processing. Handles maintenance of queue and raising [OnJobProcessed](#) and [OnQueueComplete](#) events.*

## Properties

- bool [repeating](#) [get]  
*Gets a value indicating whether this [CM\\_JobQueue](#) is repeating.*
- int [numOfTimesExecuted](#) [get]  
*Gets the number of times this queue executed (used if repeating).*
- bool [running](#) [get]  
*Gets a value indicating whether this [CM\\_JobQueue](#) is running.*
- bool [continousRunning](#) [get]  
*Gets a value indicating whether this [CM\\_JobQueue](#) is running continuously i.e. will not stop running until [StopContinousRunning](#) is called.*

## Events

- EventHandler< [CM\\_QueueEventArgs](#) > [queueStarted](#)  
*Raised when queue started.*
- EventHandler< [CM\\_QueueEventArgs](#) > [queueComplete](#)  
*Raised when queue complete.*
- EventHandler< [CM\\_QueueEventArgs](#) > [jobProcessed](#)  
*Raised when a job in the queue has finished.*

### 3.8.1 Detailed Description

The main job queue class. Encapsulates all behaviour related to queueing a job. Provides access to events, and status (i.e. running, repeating).

### 3.8.2 Member Function Documentation

#### 3.8.2.1 [CM\\_JobQueue](#) [CM\\_JobQueue.Clone](#) ( )

Clone this instance.

#### 3.8.2.2 [CM\\_JobQueue](#) [ ] [CM\\_JobQueue.Clone](#) ( int *numOfCopies* )

Clone this instance the specified *numOfCopies*.

## Parameters

<i>numOfCopies</i>	Number of copies.
--------------------	-------------------

## 3.8.2.3 CM\_JobQueue CM\_JobQueue.ContinuousRunning ( )

Set the queue to run continuously.

3.8.2.4 CM\_JobQueue CM\_JobQueue.Enqueue ( CM\_JobQueue *other* )

Enqueues the specified other queue. Adds the jobs from one queue to this queue and also adds the other queues event subscriptions.

## Parameters

<i>other</i>	Other.
--------------	--------

3.8.2.5 CM\_JobQueue CM\_JobQueue.Enqueue ( params CM\_Job[] *jobs* )

Enqueues the specified jobs.

## Parameters

<i>jobs</i>	Jobs.
-------------	-------

3.8.2.6 CM\_JobQueue CM\_JobQueue.Enqueue ( IList< CM\_Job > *jobs* )

Enqueues the specified jobs.

## Parameters

<i>jobs</i>	Jobs.
-------------	-------

3.8.2.7 CM\_JobQueue CM\_JobQueue.Enqueue ( string *id*, IEnumerator *routine* )

Creates a new job with specified id and coroutine and adds job to queue.

## Parameters

<i>id</i>	Job Identifier.
<i>routine</i>	Routine.

3.8.2.8 CM\_JobQueue CM\_JobQueue.Enqueue ( IEnumerator *routine* )

Creates a new job with specified id and coroutine and adds job to queue.

## Parameters

<i>id</i>	Job Identifier.
<i>routine</i>	Routine.

3.8.2.9 CM\_JobQueue CM\_JobQueue.Enqueue ( CM\_Job *job* )

Enqueues the specified job.

## Parameters

<i>job</i>	Job.
------------	------

**3.8.2.10** `override void CM_JobQueue.HandleJobComplete ( object sender, CM_JobEventArgs e )` [protected], [virtual]

Invoked whenever a queued job has finished processing. Handles maintenance of queue and raising OnJobProcessed and OnQueueComplete events.

## Parameters

<i>sender</i>	Sender.
<i>e</i>	E.

Implements [CM\\_GlobalCoroutineManager](#)< [CM\\_JobQueue](#) >.

**3.8.2.11** `CM_JobQueue CM_JobQueue.KillAll ( )`

Kill all currently queued jobs immediately. Clears queue list.

**3.8.2.12** `CM_JobQueue CM_JobQueue.KillAll ( float delayInSeconds )`

Kill all currently queued jobs after the specified delayInSeconds.

## Parameters

<i>delayInSeconds</i>	Delay in seconds.
-----------------------	-------------------

**3.8.2.13** `CM_JobQueue CM_JobQueue.KillCurrent ( )`

Kills the current running job immediately.

## Returns

The current.

**3.8.2.14** `CM_JobQueue CM_JobQueue.KillCurrent ( float delayInSeconds )`

Kills the current running job after the specified delayInSeconds.

## Returns

The current.

## Parameters

<i>delayInSeconds</i>	Delay in seconds.
-----------------------	-------------------

**3.8.2.15** `static CM_JobQueue CM_JobQueue.Make ( )` [static]

Returns an initialised [CM\\_JobQueue](#) instance. Provides static access to class.



#### 3.8.2.16 CM\_JobQueue CM\_JobQueue.NotifyOnJobProcessed ( EventHandler< CM\_QueueEventArgs > e )

Subscribes to the the job processed event.

## Parameters

<i>e</i>	The event handler to be invoked on event.
----------	---

**3.8.2.17 CM\_JobQueue CM\_JobQueue.NotifyOnQueueComplete ( EventHandler< CM\_QueueEventArgs > e )**

Subscribes to the queue completed event.

## Parameters

<i>e</i>	The event handler to be invoked on event.
----------	---

**3.8.2.18 CM\_JobQueue CM\_JobQueue.NotifyOnQueueStarted ( EventHandler< CM\_QueueEventArgs > e )**

Subscribes to the queue started event.

## Parameters

<i>e</i>	The event handler to be invoked on event.
----------	---

**3.8.2.19 void CM\_JobQueue.OnJobProcessed ( CM\_QueueEventArgs e ) [protected]**

Raises the job processed event.

## Parameters

<i>e</i>	E.
----------	----

**3.8.2.20 void CM\_JobQueue.OnQueueComplete ( CM\_QueueEventArgs e ) [protected]**

Raises the queue complete event.

## Parameters

<i>e</i>	E.
----------	----

**3.8.2.21 void CM\_JobQueue.OnQueueStarted ( CM\_QueueEventArgs e ) [protected]**

Raises the queue started event.

## Parameters

<i>e</i>	E.
----------	----

**3.8.2.22 CM\_JobQueue CM\_JobQueue.Pause ( )**

Pauses this instance.

**3.8.2.23 CM\_JobQueue CM\_JobQueue.Pause ( float delayInSeconds )**

Pause this instance after the specified delayInSeconds.

## Parameters

<i>delayInSeconds</i>	Delay in seconds.
-----------------------	-------------------

## 3.8.2.24 CM\_JobQueue CM\_JobQueue.RemoveNotifyOnJobProcessed ( EventHandler&lt; CM\_QueueEventArgs &gt; e )

Unsubscribes to the the job processed event. This event is invoked every time a job in the queue has finished running.

## Parameters

<i>e</i>	The event handler to be invoked on event.
----------	---

## 3.8.2.25 CM\_JobQueue CM\_JobQueue.RemoveNotifyOnQueueComplete ( EventHandler&lt; CM\_QueueEventArgs &gt; e )

Unsubscribes to the queue completed event.

## Parameters

<i>e</i>	The event handler to be invoked on event.
----------	---

## 3.8.2.26 CM\_JobQueue CM\_JobQueue.RemoveNotifyOnQueueStarted ( EventHandler&lt; CM\_QueueEventArgs &gt; e )

Unsubscribes to the the queue started event.

## Parameters

<i>e</i>	The event handler to be invoked on event.
----------	---

## 3.8.2.27 CM\_JobQueue CM\_JobQueue.Repeat ( )

Sets this instance to repeat. The job is repeated when it has finished processing.

## 3.8.2.28 CM\_JobQueue CM\_JobQueue.Repeat ( int numOfTimes )

Sets this instance to repeat a number of times. The job is repeated when it has finished processing.

## 3.8.2.29 CM\_JobQueue CM\_JobQueue.Resume ( )

Resume this instance immediately.

## 3.8.2.30 CM\_JobQueue CM\_JobQueue.Resume ( float delayInSeconds )

Resume the instance after the specified delayInSeconds.

## Parameters

<i>delayInSeconds</i>	Delay in seconds.
-----------------------	-------------------

## 3.8.2.31 CM\_JobQueue CM\_JobQueue.Start ( )

Start this instance of the queue immediately.

### 3.8.2.32 **CM\_JobQueue** **CM\_JobQueue.Start** ( float *delayInSeconds* )

Start the specified instance after *delayInSeconds*.

## Parameters

<i>delayInSeconds</i>	Delay in seconds.
-----------------------	-------------------

**3.8.2.33 CM\_JobQueue CM\_JobQueue.StopContinuousRunning ( )**

Stops the continuous running of this queue.

**3.8.2.34 CM\_JobQueue CM\_JobQueue.StopRepeat ( )**

Stops the repeat.

## Returns

The repeat.

**3.8.2.35 CM\_JobQueue CM\_JobQueue.StopRepeat ( float *delayInSeconds* )**

Stops the repeat after a specified delay in seconds.

## Returns

The repeat.

## Parameters

<i>delayInSeconds</i>	Delay in seconds.
-----------------------	-------------------

**3.8.3 Property Documentation****3.8.3.1 bool CM\_JobQueue.continuousRunning [get]**

Gets a value indicating whether this [CM\\_JobQueue](#) is running continuously i.e. will not stop running until [StopContinuousRunning](#) is called.

true if continuous running; otherwise, false.

**3.8.3.2 int CM\_JobQueue.numOfTimesExecuted [get]**

Gets the number of times this queue executed (used if repeating).

The number of times executed.

**3.8.3.3 bool CM\_JobQueue.repeating [get]**

Gets a value indicating whether this [CM\\_JobQueue](#) is repeating.

true if repeating; otherwise, false.

**3.8.3.4 bool CM\_JobQueue.running [get]**

Gets a value indicating whether this [CM\\_JobQueue](#) is running.

true if running; otherwise, false.

### 3.8.4 Event Documentation

#### 3.8.4.1 EventHandler<CM\_QueueEventArgs> CM\_JobQueue.jobProcessed [protected]

Raised when a job in the queue has finished.

#### 3.8.4.2 EventHandler<CM\_QueueEventArgs> CM\_JobQueue.queueComplete [protected]

Raised when queue complete.

#### 3.8.4.3 EventHandler<CM\_QueueEventArgs> CM\_JobQueue.queueStarted [protected]

Raised when queue started.

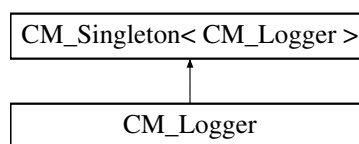
The documentation for this class was generated from the following file:

- CM\_JobQueue.cs

## 3.9 CM\_Logger Class Reference

Simple logging class used by the Coroutine Manager.

Inheritance diagram for CM\_Logger:



### Classes

- class [Message](#)  
*Encapsulates a message used by [CM\\_Logger](#).*

### Public Types

- enum [Status](#) { **Log**, **Warning**, **Error** }  
*The status level of the message.*

### Public Member Functions

- void [Log](#) (object message)  
*Log the specified message with default log status.*
- void [Log](#) (object context, object message)  
*Log the message with the specified context.*
- void [Log](#) (object message, [Status](#) status)  
*Uses a lookup to either write a log, warning, or error based on the status.*

## Additional Inherited Members

### 3.9.1 Detailed Description

Simple logging class used by the Coroutine Manager.

### 3.9.2 Member Enumeration Documentation

#### 3.9.2.1 enum CM\_Logger.Status [strong]

The status level of the message.

### 3.9.3 Member Function Documentation

#### 3.9.3.1 void CM\_Logger.Log ( object *message* )

Log the specified message with default log status.

Parameters

<i>message</i>	<a href="#">Message.</a>
----------------	--------------------------

#### 3.9.3.2 void CM\_Logger.Log ( object *context*, object *message* )

Log the message with the specified context.

Parameters

<i>context</i>	Context i.e. the calling class.
<i>message</i>	<a href="#">Message.</a>

#### 3.9.3.3 void CM\_Logger.Log ( object *message*, Status *status* )

Uses a lookup to either write a log, warning, or error based on the status.

Parameters

<i>message</i>	<a href="#">Message.</a>
<i>status</i>	Status.

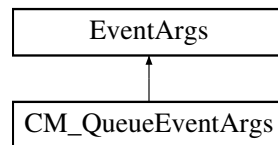
The documentation for this class was generated from the following file:

- CM\_Logger.cs

## 3.10 CM\_QueueEventArgs Class Reference

Arguments used by events raised by [CM\\_JobQueue](#).

Inheritance diagram for CM\_QueueEventArgs:



## Public Member Functions

- [CM\\_QueueEventArgs](#) ([CM\\_Job\[\]](#) [queuedJobs](#), [CM\\_Job\[\]](#) [completedJobs](#), [CM\\_JobQueue](#) [jobQueue](#))  
*Initializes a new instance of the [CM\\_QueueEventArgs](#) class.*

## Properties

- [bool](#) [hasJobsInQueue](#) [get]  
*Gets a value indicating whether this [CM\\_QueueEventArgs](#) has jobs in queue.*
- [CM\\_Job\[\]](#) [queuedJobs](#) [get]  
*Gets the queued jobs.*
- [bool](#) [hasCompletedJobs](#) [get]  
*Gets a value indicating whether this [CM\\_QueueEventArgs](#) has completed jobs.*
- [CM\\_Job\[\]](#) [completedJobs](#) [get]  
*Gets the completed jobs.*
- [CM\\_JobQueue](#) [jobQueue](#) [get]

### 3.10.1 Detailed Description

Arguments used by events raised by [CM\\_JobQueue](#).

### 3.10.2 Constructor & Destructor Documentation

- 3.10.2.1 [CM\\_QueueEventArgs.CM\\_QueueEventArgs](#) ( [CM\\_Job\[\]](#) [queuedJobs](#), [CM\\_Job\[\]](#) [completedJobs](#), [CM\\_JobQueue](#) [jobQueue](#) )

Initializes a new instance of the [CM\\_QueueEventArgs](#) class.

Parameters

<i>queuedJobs</i>	Queued jobs.
<i>completedJobs</i>	Completed jobs.

### 3.10.3 Property Documentation

- 3.10.3.1 [CM\\_Job\[\]](#) [CM\\_QueueEventArgs.completedJobs](#) [get]

Gets the completed jobs.

The completed jobs.

- 3.10.3.2 [bool](#) [CM\\_QueueEventArgs.hasCompletedJobs](#) [get]

Gets a value indicating whether this [CM\\_QueueEventArgs](#) has completed jobs.

true if has completed jobs; otherwise, false.



## 3.10.3.3 bool CM\_QueueEventArgs.hasJobsInQueue [get]

Gets a value indicating whether this [CM\\_QueueEventArgs](#) has jobs in queue.

true if has jobs in queue; otherwise, false.

## 3.10.3.4 CM\_Job [] CM\_QueueEventArgs.queuedJobs [get]

Gets the queued jobs.

The queued jobs.

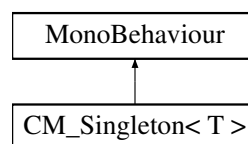
The documentation for this class was generated from the following file:

- CM\_QueueEventArgs.cs

## 3.11 CM\_Singleton&lt; T &gt; Class Template Reference

A base class for any Singleton. Provides global singular access to a MonoBehaviour.

Inheritance diagram for CM\_Singleton< T >:



## Protected Member Functions

- virtual void **OnDestroy** ()
- virtual void **OnApplicationQuit** ()

## Properties

- static bool [IsDestroyed](#) [get]  
*Gets a value indicating whether this instance is destroyed.*
- static T [instance](#) [get]  
*Gets the instance. The instance is created if not currently past of the scene.*

## 3.11.1 Detailed Description

A base class for any Singleton. Provides global singular access to a MonoBehaviour.

## Type Constraints

**T** : *MonoBehaviour*

## 3.11.2 Property Documentation

## 3.11.2.1 T CM\_Singleton&lt; T &gt;.instance [static],[get]

Gets the instance. The instance is created if not currently past of the scene.

The instance.

### 3.11.2.2 bool CM\_Singleton< T >.IsDestroyed [static],[get]

Gets a value indicating whether this instance is destroyed.

true if is destroyed; otherwise, false.

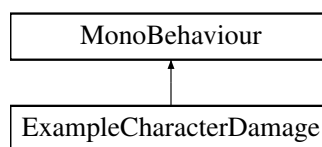
The documentation for this class was generated from the following file:

- CM\_Singleton.cs

## 3.12 ExampleCharacterDamage Class Reference

Example character with action queue.

Inheritance diagram for ExampleCharacterDamage:



### Public Member Functions

- void [AddActionToQueue](#) (CM\_Job action)  
*Adds the action to the current queue. An action can be any coroutine you want.*
- IEnumerator [ApplyDamage](#) (string damageType, float time)  
*Simulates the application of damage over time.*
- IEnumerator [RestoreHealth](#) (float time)  
*Simulates restoring of health over time.*

### Public Attributes

- float **health** = 20f

### 3.12.1 Detailed Description

Example character with action queue.

### 3.12.2 Member Function Documentation

#### 3.12.2.1 void ExampleCharacterDamage.AddActionToQueue ( CM\_Job action )

Adds the action to the current queue. An action can be any coroutine you want.

Parameters

<i>action</i>	Action.
---------------	---------

#### 3.12.2.2 IEnumerator ExampleCharacterDamage.ApplyDamage ( string damageType, float time )

Simulates the application of damage over time.

**Returns**

The damage.

**Parameters**

<i>damageType</i>	Damage type.
<i>time</i>	Time.

**3.12.2.3 IEnumerator ExampleCharacterDamage.RestoreHealth ( float *time* )**

Simulates restoring of health over time.

**Returns**

The health.

**Parameters**

<i>time</i>	Time.
-------------	-------

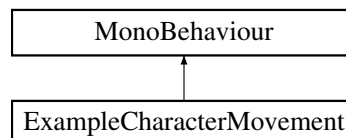
The documentation for this class was generated from the following file:

- ExampleCharacterDamage.cs

**3.13 ExampleCharacterMovement Class Reference**

Example Script. A simple script showing how you can use [CM\\_JobQueue](#) to create easily repeatable character movement.

Inheritance diagram for ExampleCharacterMovement:

**Public Attributes**

- float [moveSpeed](#) = 10f  
*The movement speed of the character.*

**3.13.1 Detailed Description**

Example Script. A simple script showing how you can use [CM\\_JobQueue](#) to create easily repeatable character movement.

**3.13.2 Member Data Documentation****3.13.2.1 float ExampleCharacterMovement.moveSpeed = 10f**

The movement speed of the character.

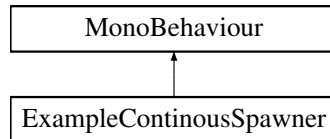
The documentation for this class was generated from the following file:

- ExampleCharacterMovement.cs

### 3.14 ExampleContinuousSpawner Class Reference

Example Script. Used to spawn a number of objects using [CM\\_Job](#).

Inheritance diagram for ExampleContinuousSpawner:



#### Public Attributes

- GameObject [prefab](#)  
*The prefab to spawn.*
- int [numToSpawn](#) = 200  
*The number of objects to spawn.*
- float [timeBetweenSpawns](#) = 0.1f  
*The time between spawns.*

#### 3.14.1 Detailed Description

Example Script. Used to spawn a number of objects using [CM\\_Job](#).

#### 3.14.2 Member Data Documentation

##### 3.14.2.1 int ExampleContinuousSpawner.numToSpawn = 200

The number of objects to spawn.

##### 3.14.2.2 GameObject ExampleContinuousSpawner.prefab

The prefab to spawn.

##### 3.14.2.3 float ExampleContinuousSpawner.timeBetweenSpawns = 0.1f

The time between spawns.

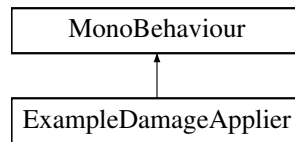
The documentation for this class was generated from the following file:

- ExampleContinuousSpawner.cs

### 3.15 ExampleDamageApplier Class Reference

Applies damage actions to the example character.

Inheritance diagram for ExampleDamageApplier:



## Public Attributes

- [ExampleCharacterDamage character](#)  
*The character to apply damage actions to.*

### 3.15.1 Detailed Description

Applies damage actions to the example character.

### 3.15.2 Member Data Documentation

#### 3.15.2.1 [ExampleCharacterDamage](#) `ExampleDamageApplier.character`

The character to apply damage actions to.

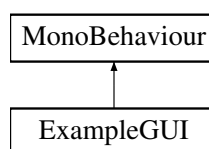
The documentation for this class was generated from the following file:

- `ExampleDamageApplier.cs`

## 3.16 ExampleGUI Class Reference

Simple GUI controller. Uses a [CM\\_JobQueue](#) to enqueue a number of gui actions.

Inheritance diagram for ExampleGUI:



## Public Attributes

- Text [titleText](#)  
*The GUI text*

### 3.16.1 Detailed Description

Simple GUI controller. Uses a [CM\\_JobQueue](#) to enqueue a number of gui actions.

### 3.16.2 Member Data Documentation

### 3.16.2.1 Text ExampleGUI.titleText

The GUI text

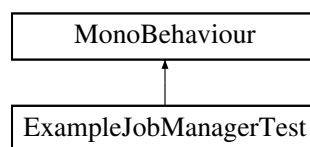
The documentation for this class was generated from the following file:

- ExampleGUI.cs

## 3.17 ExampleJobManagerTest Class Reference

Includes a number of methods to test the functionality of the [CM\\_JobManager](#) class. Each method showcases a particular functionality of the [CM\\_JobManager](#) that you can implement in your own projects.

Inheritance diagram for ExampleJobManagerTest:



### Public Member Functions

- void [GlobalJobManagerEventTest](#) ()  
*Subscribes to each of the global managers events. NotifyOnJobAdded: called everytime a job is added to the global manager, NotifyOnJobRemoved: called everytime a job is removed, NotifyOnAllJobsKilled: called when [CM\\_JobManager.KillAll](#) is invoked, NotifyOnAllJobsResumed: called when [CM\\_JobManager.ResumeAll](#) is called, NotifyOnAllJobsPaused: called when [CM\\_JobManager.PauseAll](#) is invoked, and NotifyOnAllJobsCleared: called when [CM\\_JobManager.ClearJobList](#) is invoked.*
- void [GlobalJobStartTest](#) ()  
*Starts the test job.*
- void [GlobalJobPauseTest](#) ()  
*Pauses the test job.*
- void [GlobalJobResumeTest](#) ()  
*Resumes the test job.*
- void [GlobalJobStopTest](#) ()  
*Stops the test job. This also removes the reference from the JobManager.*
- void [GlobalStartAllTest](#) ()  
*Starts all jobs owned by the global JobManager.*
- void [DelayedPauseAllTest](#) ()  
*Pauses all jobs owned by the global JobManager after 1 second has passed.*
- void [DelayedResumeAllTest](#) ()  
*Resumes all jobs owned by the global JobManager after 1.5 seconds have passed.*
- void [DelayedKillAllTest](#) ()  
*Kills all jobs owned by the global JobManager after 2 seconds have passed. This also removes all references to those jobs from the JobManager.*
- void [LocalJobManagerTest](#) ()  
*Local job manager test. Creates a new local JobManager, subscribes to [CM\\_JobManager.jobAdded](#) and [CM\\_JobManager.jobRemoved](#) events, adds a test job to the local JobManager, and finally starts, pauses, resumes, and stops this test job. This is used to show that anything you can do with the global JobManager you can also do with a local JobManager. This is useful if you want to create separate JobManagers for separate parts of your codebase.*

### 3.17.1 Detailed Description

Includes a number of methods to test the functionality of the [CM\\_JobManager](#) class. Each method showcases a particular functionality of the [CM\\_JobManager](#) that you can implement in your own projects.

### 3.17.2 Member Function Documentation

#### 3.17.2.1 void ExampleJobManagerTest.DelayedKillAllTest ( )

Kills all jobs owned by the global JobManager after 2 seconds have passed. This also removes all references to those jobs from the JobManager.

#### 3.17.2.2 void ExampleJobManagerTest.DelayedPauseAllTest ( )

Pauses all jobs owned by the global JobManager after 1 second has passed.

#### 3.17.2.3 void ExampleJobManagerTest.DelayedResumeAllTest ( )

Resumes all jobs owned by the global JobManager after 1.5 seconds have passed.

#### 3.17.2.4 void ExampleJobManagerTest.GlobalJobManagerEventTest ( )

Subscribes to each of the global managers events. NotifyOnJobAdded: called everytime a job is added to the global manager, NotifyOnJobRemoved: called everytime a job is removed, NotifyOnAllJobsKilled: called when [CM\\_JobManager.KillAll](#) is invoked, NotifyOnAllJobsResumed: called when [CM\\_JobManager.ResumeAll](#) is called, NotifyOnAllJobsPaused: called when [CM\\_JobManager.PauseAll](#) is invoked, and NotifyOnAllJobsCleared: called when [CM\\_JobManager.ClearJobList](#) is invoked.

#### 3.17.2.5 void ExampleJobManagerTest.GlobalJobPauseTest ( )

Pauses the test job.

#### 3.17.2.6 void ExampleJobManagerTest.GlobalJobResumeTest ( )

Resumes the test job.

#### 3.17.2.7 void ExampleJobManagerTest.GlobalJobStartTest ( )

Starts the test job.

#### 3.17.2.8 void ExampleJobManagerTest.GlobalJobStopTest ( )

Stops the test job. This also removes the reference from the JobManager.

#### 3.17.2.9 void ExampleJobManagerTest.GlobalStartAllTest ( )

Starts all jobs owned by the global JobManager.

### 3.17.2.10 void ExampleJobManagerTest.LocalJobManagerTest ( )

Local job manager test. Creates a new local JobManager, subscribes to [CM\\_JobManager.jobAdded](#) and [CM\\_JobManager.jobRemoved](#) events, adds a test job to the local JobManager, and finally starts, pauses, resumes, and stops this test job. This is used to show that anything you can do with the global JobManager you can also do with a local JobManager. This is useful if you want to create separate JobManagers for separate parts of your codebase.

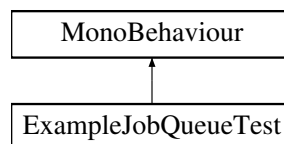
The documentation for this class was generated from the following file:

- ExampleJobManagerTest.cs

## 3.18 ExampleJobQueueTest Class Reference

Includes a number of methods to test the functionality of the [CM\\_JobQueue](#) class. Each method showcases a particular functionality of the [CM\\_JobQueue](#) that you can implement in your own projects. Each method returns an [IEnumerator](#) so that it can be added to a separate job queue to be run in sequence for test purposes.

Inheritance diagram for ExampleJobQueueTest:



### Public Member Functions

- [IEnumerator SimpleGlobalQueueTest \( \)](#)  
*Creates a list of jobs, adds them to the global queue and starts global queue. The global queue can be accessed from any script.*
- [IEnumerator SimpleLocalQueueTest \( \)](#)  
*Creates a list of jobs, adds them to a newly created local queue and starts the queue.*
- [IEnumerator LocalQueueDelayedStartTest \( \)](#)  
*Creates and starts a queue after a delay of two seconds.*
- [IEnumerator LocalQueueDelayedPauseAndResumeTest \( \)](#)  
*Creates a local queue, starts queue, pauses queue after 1 seconds, and lastly resumes queue after 3 seconds.*
- [IEnumerator QueueEventTest \( \)](#)  
*Subscribes to the global queue "queueStarted", "jobProcessed", and "queueComplete" events.*
- [IEnumerator DelayedKillCurrentTest \( \)](#)  
*Creates a local queue and sets the current job in the queue to be killed after three seconds. This kills the currently running job but the queue still executes.*
- [IEnumerator DelayedKillAllTest \( \)](#)  
*Creates a local queue and sets all jobs in the queue to be killed after one second. This kills all jobs and clears the queue.*
- [IEnumerator AddLocalQueueToGlobalQueueTest \( \)](#)  
*Creates a local queue and then adds it to the global queue. The event subscriptions are also added to the global queue.*
- [IEnumerator AddRepeatingJobToQueueTest \( \)](#)  
*Adds a repeating job to a queue. Queue will not progress if a repeating job is added until that job is manually killed or has reached its set number of times to repeat.*
- [IEnumerator SetNumberRepeatingQueueTest \( \)](#)  
*Creates a new local queue, adds a number of test jobs and sets the queue to repeat two times.*
- [IEnumerator TimedRepeatingQueueTest \( \)](#)



*Creates a new local queue, sets it to repeat and then stop repeating after 1 seconds.*

- IEnumerator [ClonedRepeatingQueueTest](#) ()

*Creates a local queue and sets it to repeat twice. The queue is then cloned and started. The new cloned queue will contain the original queues repeat status and event subscriptions.*

- IEnumerator [MultipleClonedQueueTest](#) ()

*Creates a local queue and clones the queue twice. Both of the cloned queues are then started.*

### 3.18.1 Detailed Description

Includes a number of methods to test the functionality of the [CM\\_JobQueue](#) class. Each method showcases a particular functionality of the [CM\\_JobQueue](#) that you can implement in your own projects. Each method returns an enumerator so that it can added to a seperate job queue to be run in sequence for test purposes.

### 3.18.2 Member Function Documentation

#### 3.18.2.1 IEnumerator ExampleJobQueueTest.AddLocalQueueToGlobalQueueTest ( )

Creates a local queue and then adds it to the glocal queue. The event subscriptions are also added to the global queue.

#### 3.18.2.2 IEnumerator ExampleJobQueueTest.AddRepeatingJobToQueueTest ( )

Adds a repeating job to a queue. Queue will not progress if a repeating job is added until that job is manually killed or has reached its set number of times to repeat.

#### 3.18.2.3 IEnumerator ExampleJobQueueTest.ClonedRepeatingQueueTest ( )

Creates a local queue and sets it to repeat twice. The queue is then cloned and started. The new cloned queue will contain the original queues repeat status and event subscriptions.

#### 3.18.2.4 IEnumerator ExampleJobQueueTest.DelayedKillAllTest ( )

Creates a local queue and sets all jobs in the queue to be killed after one second. This kills all jobs and clears the queue.

#### 3.18.2.5 IEnumerator ExampleJobQueueTest.DelayedKillCurrentTest ( )

Creates a local queue and sets the current job in the queue to be killed after three seconds. This kills the currently running job but the queue still executes.

#### 3.18.2.6 IEnumerator ExampleJobQueueTest.LocalQueueDelayedPauseAndResumeTest ( )

Creates a local queue, starts queue, pauses queue after 1 seconds, and lastly resumes queue after 3 seconds.

#### 3.18.2.7 IEnumerator ExampleJobQueueTest.LocalQueueDelayedStartTest ( )

Creates and starts a queue after a delay of two seconds.

#### 3.18.2.8 IEnumerator ExampleJobQueueTest.MultipleClonedQueueTest ( )

Creates a local queue and clones the queue twice. Both of the cloned queues are then started.

### 3.18.2.9 IEnumerator ExampleJobQueueTest.QueueEventTest ( )

Subscribes to the global queue "queueStarted", "jobProcessed", and "queueComplete" events.

### 3.18.2.10 IEnumerator ExampleJobQueueTest.SetNumberRepeatingQueueTest ( )

Creates a new local queue, adds a number of test jobs and sets the queue to repeat two times.

### 3.18.2.11 IEnumerator ExampleJobQueueTest.SimpleGlobalQueueTest ( )

Creates a list of jobs, adds them to the global queue and starts global queue. The global queue can be accessed from any script.

### 3.18.2.12 IEnumerator ExampleJobQueueTest.SimpleLocalQueueTest ( )

Creates a list of jobs, adds them to a newly created local queue and starts the queue.

### 3.18.2.13 IEnumerator ExampleJobQueueTest.TimedRepeatingQueueTest ( )

Creates a new local queue, sets it to repeat and then stop repeating after 1 seconds.

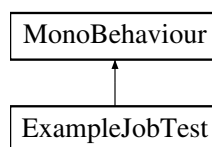
The documentation for this class was generated from the following file:

- ExampleJobQueueTest.cs

## 3.19 ExampleJobTest Class Reference

Includes a number of methods to test the functionality of the [CM\\_Job](#) class. Each method showcases a particular functionality of the [CM\\_Job](#) class that you can implement in your own projects. Each method returns an enumerator so that it can be added to a job queue to be run in sequence for test purposes.

Inheritance diagram for ExampleJobTest:



### Public Member Functions

- IEnumerator [SimpleJobTest](#) ()  
*Creates and starts a job.*
- IEnumerator [JobTestWithDelayedStart](#) ()  
*Creates and starts a job after a delay.*
- IEnumerator [JobTestWithDelayedPause](#) ()  
*Creates coroutine job, starts job immediately and then pauses coroutine after 4 seconds. This paused job is then stored and can be resumed/killed from any class that has a reference to that job.*
- IEnumerator [JobTestWithDelayedResume](#) ()  
*Creates new job, pauses after 1 second, and resumes after 3 seconds.*
- IEnumerator [JobTestWithDelayedKill](#) ()

- Creates coroutine job, starts job immediately and then sets the job to be killed after 4 seconds.*
- IEnumerator [JobTestWithStartAndEndEvents](#) ()
  - Creates new job, subscribes to the job start and end events and then starts job to be run immediately.*
- IEnumerator [ChildJobTest](#) ()
  - Child job test. Creates new job, adds two children (parent will not complete until children have finished processing) and starts the job.*
- IEnumerator [SingleCloneJobTest](#) ()
  - Creates a new job, clones job, and then runs clone.*
- IEnumerator [MultipleCloneJobTest](#) ()
  - Creates a new job, clones five copies of the original job, and then starts a random clone.*
- IEnumerator [InfinitelyRepeatableJobTest](#) ()
  - Creates and starts a job that will repeat for 3 seconds. The job complete event is subscribed to, this is used to display how many times the job will be repeated.*
- IEnumerator [MultipleRepeatableJobTest](#) ()
  - Creates and starts a job that will repeat three times. The job complete event is subscribed to, this is used to display how many times the job will be repeated.*
- IEnumerator [MultipleRepeatableJobTestWithChild](#) ()
  - Creates and starts a job that will repeat three times. Adds a child job that will also repeat three times. The job complete event and child job complete events are subscribed to, this is used to display how many times the job will be repeated.*

### 3.19.1 Detailed Description

Includes a number of methods to test the functionality of the [CM\\_Job](#) class. Each method showcases a particular functionality of the [CM\\_Job](#) class that you can implement in your own projects. Each method returns an enumerator so that it can be added to a job queue to be run in sequence for test purposes.

### 3.19.2 Member Function Documentation

#### 3.19.2.1 IEnumerator ExampleJobTest.ChildJobTest ( )

Child job test. Creates new job, adds two children (parent will not complete until children have finished processing) and starts the job.

#### 3.19.2.2 IEnumerator ExampleJobTest.InfinitelyRepeatableJobTest ( )

Creates and starts a job that will repeat for 3 seconds. The job complete event is subscribed to, this is used to display how many times the job will be repeated.

#### 3.19.2.3 IEnumerator ExampleJobTest.JobTestWithDelayedKill ( )

Creates coroutine job, starts job immediately and then sets the job to be killed after 4 seconds.

#### 3.19.2.4 IEnumerator ExampleJobTest.JobTestWithDelayedPause ( )

Creates coroutine job, starts job immediately and then pauses coroutine after 4 seconds. This paused job is then stored and can be resumed/killed from any class that has a reference to that job.

#### 3.19.2.5 IEnumerator ExampleJobTest.JobTestWithDelayedResume ( )

Creates new job, pauses after 1 second, and resumes after 3 seconds.

**3.19.2.6 IEnumerator ExampleJobTest.JobTestWithDelayedStart ( )**

Creates and starts a job after a delay.

**3.19.2.7 IEnumerator ExampleJobTest.JobTestWithStartAndEndEvents ( )**

Creates new job, subscribes to the job start and end events and then starts job to be run immediately.

**3.19.2.8 IEnumerator ExampleJobTest.MultipleCloneJobTest ( )**

Creates a new job, clones five copies of the original job, and then starts a random clone.

**3.19.2.9 IEnumerator ExampleJobTest.MultipleRepeatableJobTest ( )**

Creates and starts a job that will repeat three times. The job complete event is subscribed to, this is used to display how many times the job will be repeated.

**3.19.2.10 IEnumerator ExampleJobTest.MultipleRepeatableJobTestWithChild ( )**

Creates and starts a job that will repeat three times. Adds a child job that will also repeat three times. The job complete event and child job complete events are subscribed to, this is used to display how many times the job will be repeated.

**3.19.2.11 IEnumerator ExampleJobTest.SimpleJobTest ( )**

Creates and starts a job.

**3.19.2.12 IEnumerator ExampleJobTest.SingleCloneJobTest ( )**

Creates a new job, clones job, and then runs clone.

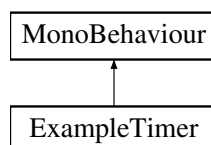
The documentation for this class was generated from the following file:

- ExampleJobTest.cs

**3.20 ExampleTimer Class Reference**

Creates new coroutine job to update a text object with time since startup. Adds job to global job manager so that it can be paused and resumed as required.

Inheritance diagram for ExampleTimer:



### 3.20.1 Detailed Description

Creates new coroutine job to update a text object with time since startup. Adds job to global job manager so that it can be paused and resumed as required.

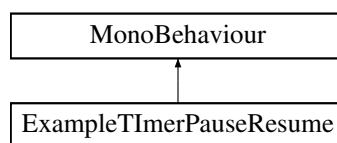
The documentation for this class was generated from the following file:

- ExampleTimer.cs

## 3.21 ExampleTimerPauseResume Class Reference

Pauses all jobs associated with the JobManager when user clicks left mouse button and resumes all jobs with user clicks right mouse button.

Inheritance diagram for ExampleTimerPauseResume:



### 3.21.1 Detailed Description

Pauses all jobs associated with the JobManager when user clicks left mouse button and resumes all jobs with user clicks right mouse button.

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

- ExampleTimerPauseResume.cs

## 3.22 ICM\_Cloneable< T > Interface Template Reference

An interface for all classes used by the coroutine manager that can be cloned.

### Public Member Functions

- `T Clone ()`  
*Clone this instance.*
- `T[] Clone (int numOfCopies)`  
*Clone the specified numOfCopies.*

### 3.22.1 Detailed Description

An interface for all classes used by the coroutine manager that can be cloned.

### 3.22.2 Member Function Documentation

#### 3.22.2.1 T ICM\_Cloneable< T >.Clone ( )

Clone this instance.

3.22.2.2 `T[] ICM_Cloneable<T>.Clone ( int numOfCopies )`

Clone the specified `numOfCopies`.

## Parameters

<i>numOfCopies</i>	Number of copies.
--------------------	-------------------

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

- ICM\_Cloneable.cs

## 3.23 CM\_Logger.Message Class Reference

Encapsulates a message used by [CM\\_Logger](#).

### Public Member Functions

- [Message](#) (object [invoker](#), object [content](#))  
*Initializes a new instance of the CM\_Logger+Message class.*
- override string [ToString](#) ()  
*Returns a System.String that represents the current CM\_Logger+Message.*

### Properties

- object [content](#) [get]  
*Gets the content of the message.*
- object [invoker](#) [get]  
*Gets the invoker of the message.*

#### 3.23.1 Detailed Description

Encapsulates a message used by [CM\\_Logger](#).

#### 3.23.2 Constructor & Destructor Documentation

##### 3.23.2.1 CM\_Logger.Message.Message ( object *invoker*, object *content* )

Initializes a new instance of the CM\_Logger+Message class.

## Parameters

<i>invoker</i>	Invoker.
<i>content</i>	Content.

#### 3.23.3 Member Function Documentation

##### 3.23.3.1 override string CM\_Logger.Message.ToString ( )

Returns a System.String that represents the current CM\_Logger+Message.

## Returns

A System.String that represents the current CM\_Logger+Message.

### 3.23.4 Property Documentation

#### 3.23.4.1 object `CM_Logger.Message.content` [get]

Gets the content of the message.

The content.

#### 3.23.4.2 object `CM_Logger.Message.invoker` [get]

Gets the invoker of the message.

The invoker.

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

- `CM_Logger.cs`



# Index

- AddActionToQueue
  - ExampleCharacterDamage, [48](#)
- AddChild
  - CM\_Job, [9](#)
- AddJob
  - CM\_JobManager, [20, 21](#)
- AddLocalQueueToGlobalQueueTest
  - ExampleJobQueueTest, [55](#)
- AddRepeatingJobToQueueTest
  - ExampleJobQueueTest, [55](#)
- AddToExecuteQueue
  - CM\_Dispatcher, [5](#)
- allJobsCleared
  - CM\_JobManager, [31](#)
- allJobsKilled
  - CM\_JobManager, [31](#)
- allJobsPaused
  - CM\_JobManager, [31](#)
- allJobsResumed
  - CM\_JobManager, [31](#)
- ApplyDamage
  - ExampleCharacterDamage, [48](#)
- Builder
  - CM\_Job, [9](#)
- CM\_Dispatcher, [5](#)
  - AddToExecuteQueue, [5](#)
- CM\_GlobalCoroutineManager
  - Global, [6](#)
- CM\_GlobalCoroutineManager< T >, [6](#)
- CM\_Job, [6](#)
  - AddChild, [9](#)
  - Builder, [9](#)
  - Clone, [9](#)
  - coroutine, [16](#)
  - id, [16](#)
  - jobKilled, [16](#)
  - Kill, [9](#)
  - Make, [11](#)
  - NotifyOnChildJobComplete, [11](#)
  - NotifyOnChildJobStarted, [11](#)
  - NotifyOnJobComplete, [11](#)
  - NotifyOnJobFinishedRunning, [11](#)
  - NotifyOnJobPaused, [12](#)
  - NotifyOnJobResumed, [12](#)
  - NotifyOnJobStarted, [12](#)
  - numOfTimesExecuted, [16](#)
  - OnChildJobsComplete, [12](#)
  - OnChildJobsStarted, [12](#)
  - OnJobComplete, [12](#)
  - OnJobFinishedRunning, [13](#)
  - OnJobPaused, [13](#)
  - OnJobResumed, [13](#)
  - OnJobStarted, [13](#)
  - Pause, [13](#)
  - paused, [16](#)
  - RemoveChildJob, [13](#)
  - RemoveNotifyOnChildJobComplete, [14](#)
  - RemoveNotifyOnChildJobStarted, [14](#)
  - RemoveNotifyOnJobComplete, [14](#)
  - RemoveNotifyOnJobFinishedRunning, [14](#)
  - RemoveNotifyOnJobPaused, [14](#)
  - RemoveNotifyOnJobResumed, [14](#)
  - RemoveNotifyOnJobStarted, [15](#)
  - Repeat, [15](#)
  - repeating, [16](#)
  - Resume, [15](#)
  - running, [16](#)
  - Start, [15](#)
  - StopRepeat, [15](#)
- CM\_JobEventArgs, [16](#)
  - CM\_JobEventArgs, [17](#)
  - childJobs, [17](#)
  - hasChildJobs, [17](#)
  - job, [17](#)
- CM\_JobManager, [18](#)
  - AddJob, [20, 21](#)
  - allJobsCleared, [31](#)
  - allJobsKilled, [31](#)
  - allJobsPaused, [31](#)
  - allJobsResumed, [31](#)
  - ClearJobList, [21](#)
  - HasJob, [21](#)
  - IsRunning, [22](#)
  - jobAdded, [31](#)
  - jobRemoved, [31](#)
  - KillAll, [22](#)
  - Make, [23](#)
  - NotifyOnAllJobsCleared, [23](#)
  - NotifyOnAllJobsKilled, [23](#)
  - NotifyOnAllJobsPaused, [23](#)
  - NotifyOnAllJobsResumed, [23](#)
  - NotifyOnJobAdded, [23](#)
  - NotifyOnJobRemoved, [23](#)
  - OnAllJobsCleared, [25](#)
  - OnAllJobsKilled, [25](#)
  - OnAllJobsPaused, [25](#)
  - OnAllJobsResumed, [25](#)

- OnJobAdded, 25
- OnJobRemoved, 25
- PauseAll, 25, 26
- PauseCoroutine, 26
- RemoveJob, 26, 27
- RemoveNotifyOnAllJobsCleared, 27
- RemoveNotifyOnAllJobsKilled, 27
- RemoveNotifyOnAllJobsPaused, 27
- RemoveNotifyOnAllJobsResumed, 27
- RemoveNotifyOnJobAdded, 27
- RemoveNotifyOnJobRemoved, 29
- ResumeAll, 29
- ResumeCoroutine, 29
- StartAll, 30
- StartCoroutine, 30
- StopCoroutine, 30, 31
- CM\_JobManagerEventArgs, 31
  - CM\_JobManagerEventArgs, 32
  - ownedJobs, 32
  - pausedJobs, 32
  - runningJobs, 32
- CM\_JobManagerJobEditedEventArgs, 33
  - CM\_JobManagerJobEditedEventArgs, 33
  - jobEdited, 34
  - otherArgs, 34
- CM\_JobQueue, 34
  - Clone, 36
  - ContinuousRunning, 37
  - continuousRunning, 43
  - Enqueue, 37
  - HandlejobComplete, 38
  - jobProcessed, 44
  - KillAll, 38
  - KillCurrent, 38
  - Make, 38
  - NotifyOnJobProcessed, 38
  - NotifyOnQueueComplete, 40
  - NotifyOnQueueStarted, 40
  - numOfTimesExecuted, 43
  - OnJobProcessed, 40
  - OnQueueComplete, 40
  - OnQueueStarted, 40
  - Pause, 40
  - queueComplete, 44
  - queueStarted, 44
  - RemoveNotifyOnJobProcessed, 41
  - RemoveNotifyOnQueueComplete, 41
  - RemoveNotifyOnQueueStarted, 41
  - Repeat, 41
  - repeating, 43
  - Resume, 41
  - running, 43
  - Start, 41
  - StopContinuousRunning, 43
  - StopRepeat, 43
- CM\_Logger, 44
  - Log, 45
  - Status, 45
- CM\_Logger.Message, 61
- CM\_Logger::Message
  - content, 62
  - invoker, 62
  - Message, 61
  - ToString, 61
- CM\_QueueEventArgs, 45
  - CM\_QueueEventArgs, 46
  - completedJobs, 46
  - hasCompletedJobs, 46
  - hasJobsInQueue, 46
  - queuedJobs, 47
- CM\_Singleton
  - instance, 47
  - IsDestroyed, 47
- CM\_Singleton< T >, 47
- character
  - ExampleDamageApplier, 51
- ChildJobTest
  - ExampleJobTest, 57
- childJobs
  - CM\_JobEventArgs, 17
- ClearJobList
  - CM\_JobManager, 21
- Clone
  - CM\_Job, 9
  - CM\_JobQueue, 36
  - ICM\_Cloneable, 59
- ClonedRepeatingQueueTest
  - ExampleJobQueueTest, 55
- completedJobs
  - CM\_QueueEventArgs, 46
- content
  - CM\_Logger::Message, 62
- ContinuousRunning
  - CM\_JobQueue, 37
- continuousRunning
  - CM\_JobQueue, 43
- coroutine
  - CM\_Job, 16
- DelayedKillAllTest
  - ExampleJobManagerTest, 53
  - ExampleJobQueueTest, 55
- DelayedKillCurrentTest
  - ExampleJobQueueTest, 55
- DelayedPauseAllTest
  - ExampleJobManagerTest, 53
- DelayedResumeAllTest
  - ExampleJobManagerTest, 53
- Enqueue
  - CM\_JobQueue, 37
- ExampleCharacterDamage, 48
  - AddActionToQueue, 48
  - ApplyDamage, 48
  - RestoreHealth, 49
- ExampleCharacterMovement, 49
  - moveSpeed, 49

- ExampleContinuousSpawner, 50
  - numToSpawn, 50
  - prefab, 50
  - timeBetweenSpawns, 50
- ExampleDamageApplier, 50
  - character, 51
- ExampleGUI, 51
  - titleText, 51
- ExampleJobManagerTest, 52
  - DelayedKillAllTest, 53
  - DelayedPauseAllTest, 53
  - DelayedResumeAllTest, 53
  - GlobalJobManagerEventTest, 53
  - GlobalJobPauseTest, 53
  - GlobalJobResumeTest, 53
  - GlobalJobStartTest, 53
  - GlobalJobStopTest, 53
  - GlobalStartAllTest, 53
  - LocalJobManagerTest, 53
- ExampleJobQueueTest, 54
  - AddLocalQueueToGlobalQueueTest, 55
  - AddRepeatingJobToQueueTest, 55
  - ClonedRepeatingQueueTest, 55
  - DelayedKillAllTest, 55
  - DelayedKillCurrentTest, 55
  - LocalQueueDelayedPauseAndResumeTest, 55
  - LocalQueueDelayedStartTest, 55
  - MultipleClonedQueueTest, 55
  - QueueEventTest, 55
  - SetNumberRepeatingQueueTest, 56
  - SimpleGlobalQueueTest, 56
  - SimpleLocalQueueTest, 56
  - TimedRepeatingQueueTest, 56
- ExampleJobTest, 56
  - ChildJobTest, 57
  - InfinitelyRepeatableJobTest, 57
  - JobTestWithDelayedKill, 57
  - JobTestWithDelayedPause, 57
  - JobTestWithDelayedResume, 57
  - JobTestWithDelayedStart, 57
  - JobTestWithStartAndEndEvents, 58
  - MultipleCloneJobTest, 58
  - MultipleRepeatableJobTest, 58
  - MultipleRepeatableJobTestWithChild, 58
  - SimpleJobTest, 58
  - SingleCloneJobTest, 58
- ExampleTimerPauseResume, 59
- ExampleTimer, 58
- Global
  - CM\_GlobalCoroutineManager, 6
- GlobalJobManagerEventTest
  - ExampleJobManagerTest, 53
- GlobalJobPauseTest
  - ExampleJobManagerTest, 53
- GlobalJobResumeTest
  - ExampleJobManagerTest, 53
- GlobalJobStartTest
  - ExampleJobManagerTest, 53
- GlobalJobStopTest
  - ExampleJobManagerTest, 53
- GlobalStartAllTest
  - ExampleJobManagerTest, 53
- HandleJobComplete
  - CM\_JobQueue, 38
- hasChildJobs
  - CM\_JobEventArgs, 17
- hasCompletedJobs
  - CM\_QueueEventArgs, 46
- HasJob
  - CM\_JobManager, 21
- hasJobsInQueue
  - CM\_QueueEventArgs, 46
- ICM\_Cloneable
  - Clone, 59
- ICM\_Cloneable< T >, 59
- id
  - CM\_Job, 16
- InfinitelyRepeatableJobTest
  - ExampleJobTest, 57
- instance
  - CM\_Singleton, 47
- invoker
  - CM\_Logger::Message, 62
- IsDestroyed
  - CM\_Singleton, 47
- IsRunning
  - CM\_JobManager, 22
- job
  - CM\_JobEventArgs, 17
- jobAdded
  - CM\_JobManager, 31
- jobEdited
  - CM\_JobManagerJobEditedEventArgs, 34
- jobKilled
  - CM\_Job, 16
- jobProcessed
  - CM\_JobQueue, 44
- jobRemoved
  - CM\_JobManager, 31
- JobTestWithDelayedKill
  - ExampleJobTest, 57
- JobTestWithDelayedPause
  - ExampleJobTest, 57
- JobTestWithDelayedResume
  - ExampleJobTest, 57
- JobTestWithDelayedStart
  - ExampleJobTest, 57
- JobTestWithStartAndEndEvents
  - ExampleJobTest, 58
- Kill
  - CM\_Job, 9
- KillAll
  - CM\_JobManager, 22

- CM\_JobQueue, 38
- KillCurrent
  - CM\_JobQueue, 38
- LocalJobManagerTest
  - ExampleJobManagerTest, 53
- LocalQueueDelayedPauseAndResumeTest
  - ExampleJobQueueTest, 55
- LocalQueueDelayedStartTest
  - ExampleJobQueueTest, 55
- Log
  - CM\_Logger, 45
- Make
  - CM\_Job, 11
  - CM\_JobManager, 23
  - CM\_JobQueue, 38
- Message
  - CM\_Logger::Message, 61
- moveSpeed
  - ExampleCharacterMovement, 49
- MultipleCloneJobTest
  - ExampleJobTest, 58
- MultipleClonedQueueTest
  - ExampleJobQueueTest, 55
- MultipleRepeatableJobTest
  - ExampleJobTest, 58
- MutltipleRepeatableJobTestWithChild
  - ExampleJobTest, 58
- NotifyOnAllJobsCleared
  - CM\_JobManager, 23
- NotifyOnAllJobsKilled
  - CM\_JobManager, 23
- NotifyOnAllJobsPaused
  - CM\_JobManager, 23
- NotifyOnAllJobsResumed
  - CM\_JobManager, 23
- NotifyOnChildJobComplete
  - CM\_Job, 11
- NotifyOnChildJobStarted
  - CM\_Job, 11
- NotifyOnJobAdded
  - CM\_JobManager, 23
- NotifyOnJobComplete
  - CM\_Job, 11
- NotifyOnJobFinishedRunning
  - CM\_Job, 11
- NotifyOnJobPaused
  - CM\_Job, 12
- NotifyOnJobProcessed
  - CM\_JobQueue, 38
- NotifyOnJobRemoved
  - CM\_JobManager, 23
- NotifyOnJobResumed
  - CM\_Job, 12
- NotifyOnJobStarted
  - CM\_Job, 12
- NotifyOnQueueComplete
  - CM\_JobQueue, 40
- NotifyOnQueueStarted
  - CM\_JobQueue, 40
- numOfTimesExecuted
  - CM\_Job, 16
  - CM\_JobQueue, 43
- numToSpawn
  - ExampleContinousSpawner, 50
- OnAllJobsCleared
  - CM\_JobManager, 25
- OnAllJobsKilled
  - CM\_JobManager, 25
- OnAllJobsPaused
  - CM\_JobManager, 25
- OnAllJobsResumed
  - CM\_JobManager, 25
- OnChildJobsComplete
  - CM\_Job, 12
- OnChildJobsStarted
  - CM\_Job, 12
- OnJobAdded
  - CM\_JobManager, 25
- OnJobComplete
  - CM\_Job, 12
- OnJobFinishedRunning
  - CM\_Job, 13
- OnJobPaused
  - CM\_Job, 13
- OnJobProcessed
  - CM\_JobQueue, 40
- OnJobRemoved
  - CM\_JobManager, 25
- OnJobResumed
  - CM\_Job, 13
- OnJobStarted
  - CM\_Job, 13
- OnQueueComplete
  - CM\_JobQueue, 40
- OnQueueStarted
  - CM\_JobQueue, 40
- otherArgs
  - CM\_JobManager.JobEditedEventArgs, 34
- ownedJobs
  - CM\_JobManagerEventArgs, 32
- Pause
  - CM\_Job, 13
  - CM\_JobQueue, 40
- PauseAll
  - CM\_JobManager, 25, 26
- PauseCoroutine
  - CM\_JobManager, 26
- paused
  - CM\_Job, 16
- pausedJobs
  - CM\_JobManagerEventArgs, 32
- prefab
  - ExampleContinousSpawner, 50

- queueComplete
  - CM\_JobQueue, 44
- QueueEventTest
  - ExampleJobQueueTest, 55
- queueStarted
  - CM\_JobQueue, 44
- queuedJobs
  - CM\_QueueEventArgs, 47
- RemoveChildJob
  - CM\_Job, 13
- RemoveJob
  - CM\_JobManager, 26, 27
- RemoveNotifyOnAllJobsCleared
  - CM\_JobManager, 27
- RemoveNotifyOnAllJobsKilled
  - CM\_JobManager, 27
- RemoveNotifyOnAllJobsPaused
  - CM\_JobManager, 27
- RemoveNotifyOnAllJobsResumed
  - CM\_JobManager, 27
- RemoveNotifyOnChildJobComplete
  - CM\_Job, 14
- RemoveNotifyOnChildJobStarted
  - CM\_Job, 14
- RemoveNotifyOnJobAdded
  - CM\_JobManager, 27
- RemoveNotifyOnJobComplete
  - CM\_Job, 14
- RemoveNotifyOnJobFinishedRunning
  - CM\_Job, 14
- RemoveNotifyOnJobPaused
  - CM\_Job, 14
- RemoveNotifyOnJobProcessed
  - CM\_JobQueue, 41
- RemoveNotifyOnJobRemoved
  - CM\_JobManager, 29
- RemoveNotifyOnJobResumed
  - CM\_Job, 14
- RemoveNotifyOnJobStarted
  - CM\_Job, 15
- RemoveNotifyOnQueueComplete
  - CM\_JobQueue, 41
- RemoveNotifyOnQueueStarted
  - CM\_JobQueue, 41
- Repeat
  - CM\_Job, 15
  - CM\_JobQueue, 41
- repeating
  - CM\_Job, 16
  - CM\_JobQueue, 43
- RestoreHealth
  - ExampleCharacterDamage, 49
- Resume
  - CM\_Job, 15
  - CM\_JobQueue, 41
- ResumeAll
  - CM\_JobManager, 29
- ResumeCoroutine
  - CM\_JobManager, 29
- running
  - CM\_Job, 16
  - CM\_JobQueue, 43
- runningJobs
  - CM\_JobManagerEventArgs, 32
- SetNumberRepeatingQueueTest
  - ExampleJobQueueTest, 56
- SimpleGlobalQueueTest
  - ExampleJobQueueTest, 56
- SimpleJobTest
  - ExampleJobTest, 58
- SimpleLocalQueueTest
  - ExampleJobQueueTest, 56
- SingleCloneJobTest
  - ExampleJobTest, 58
- Start
  - CM\_Job, 15
  - CM\_JobQueue, 41
- StartAll
  - CM\_JobManager, 30
- StartCoroutine
  - CM\_JobManager, 30
- Status
  - CM\_Logger, 45
- StopContinuousRunning
  - CM\_JobQueue, 43
- StopCoroutine
  - CM\_JobManager, 30, 31
- StopRepeat
  - CM\_Job, 15
  - CM\_JobQueue, 43
- timeBetweenSpawns
  - ExampleContinuousSpawner, 50
- TimedRepeatingQueueTest
  - ExampleJobQueueTest, 56
- titleText
  - ExampleGUI, 51
- ToString
  - CM\_Logger::Message, 61