

CommandPattern

Generated by Doxygen 1.9.4

1 Namespace Index	1
1.1 Package List	1
2 Hierarchical Index	3
2.1 Class Hierarchy	3
3 Class Index	5
3.1 Class List	5
4 Namespace Documentation	7
4.1 SadSapphicGames Namespace Reference	7
4.2 SadSapphicGames.CommandPattern Namespace Reference	7
4.2.1 Enumeration Type Documentation	8
4.2.1.1 ExecuteCode	8
5 Class Documentation	9
5.1 SadSapphicGames.CommandPattern.AsyncCommand Class Reference	9
5.1.1 Detailed Description	10
5.1.2 Member Function Documentation	10
5.1.2.1 Execute()	10
5.1.2.2 ExecuteAsync()	11
5.1.3 Property Documentation	11
5.1.3.1 CancellationTokenSource	11
5.1.3.2 CommandTask	11
5.2 SadSapphicGames.CommandPattern.Command Class Reference	12
5.2.1 Detailed Description	12
5.2.2 Member Function Documentation	12
5.2.2.1 Execute()	12
5.3 SadSapphicGames.CommandPattern.CommandStream Class Reference	12
5.3.1 Detailed Description	14
5.3.2 Constructor & Destructor Documentation	14
5.3.2.1 CommandStream()	14
5.3.3 Member Function Documentation	14
5.3.3.1 CancelRunningCommandTask() [1/2]	14
5.3.3.2 CancelRunningCommandTask() [2/2]	15
5.3.3.3 DropHistory()	15
5.3.3.4 DropQueue()	15
5.3.3.5 ExecuteFullQueue()	15
5.3.3.6 ForceQueueUndoCommand()	16
5.3.3.7 GetCommandHistory()	16
5.3.3.8 GetCommandQueue()	16
5.3.3.9 GetFaultedCommandTasks()	17
5.3.3.10 GetRunningCommandTasks()	17
5.3.3.11 QueueCommand()	17

5.3.3.12 QueueCommands()	17
5.3.3.13 TryExecuteNext() [1/2]	18
5.3.3.14 TryExecuteNext() [2/2]	18
5.3.3.15 TryQueueUndoCommand()	18
5.4 SadSapphicGames.CommandPattern.CompositeCommand Class Reference	19
5.4.1 Detailed Description	20
5.4.2 Member Function Documentation	20
5.4.2.1 AddChild()	20
5.4.2.2 Execute()	20
5.5 SadSapphicGames.CommandPattern.IAsyncCommand Interface Reference	21
5.5.1 Detailed Description	21
5.5.2 Member Function Documentation	22
5.5.2.1 ExecuteAsync()	22
5.5.3 Property Documentation	22
5.5.3.1 CancellationTokenSource	22
5.5.3.2 CommandTask	22
5.6 SadSapphicGames.CommandPattern.ICommand Interface Reference	23
5.6.1 Detailed Description	23
5.6.2 Member Function Documentation	23
5.6.2.1 Execute()	23
5.7 SadSapphicGames.CommandPattern.IFailable Interface Reference	23
5.7.1 Detailed Description	24
5.7.2 Member Function Documentation	24
5.7.2.1 WouldFail()	24
5.8 SadSapphicGames.CommandPattern.IreversibleCompositeFailureException Class Reference	24
5.8.1 Detailed Description	25
5.9 SadSapphicGames.CommandPattern.IUndoable Interface Reference	25
5.9.1 Detailed Description	25
5.9.2 Member Function Documentation	25
5.9.2.1 GetUndoCommand()	26
5.10 SadSapphicGames.CommandPattern.NullCommand Class Reference	26
5.10.1 Detailed Description	26
5.10.2 Member Function Documentation	26
5.10.2.1 Execute()	27
5.10.2.2 GetUndoCommand()	27
5.11 SadSapphicGames.CommandPattern.NullCompositeCommand Class Reference	27
5.11.1 Detailed Description	28
5.11.2 Constructor & Destructor Documentation	28
5.11.2.1 NullCompositeCommand()	28
5.11.3 Member Function Documentation	28
5.11.3.1 GetUndoCommand()	28
5.12 SadSapphicGames.CommandPattern.ReversibleCompositeFailureException Class Reference	29

5.12.1 Detailed Description	29
5.13 SadSapphicGames.CommandPattern.SimpleComposite Class Reference	29
5.13.1 Detailed Description	30
5.13.2 Constructor & Destructor Documentation	30
5.13.2.1 SimpleComposite()	30
5.14 SadSapphicGames.CommandPattern.SingletonCommandManager Class Reference	30
5.14.1 Detailed Description	31
5.14.2 Member Function Documentation	32
5.14.2.1 DropCommandHistory()	32
5.14.2.2 ForceQueueUndoCommand()	32
5.14.2.3 GetCommandHistory()	32
5.14.2.4 GetRunningCommandTasks()	32
5.14.2.5 QueueCommand()	33
5.14.2.6 QueueCommands()	33
5.14.2.7 ToggleCommandExecution()	33
5.14.2.8 TryQueueUndoCommand()	33
Index	35

Chapter 1

Namespace Index

1.1 Package List

Here are the packages with brief descriptions (if available):

SadSapphicGames	7
SadSapphicGames.CommandPattern	7

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

SadSapphicGames.CommandPattern.CommandStream	12
System.Exception	
SadSapphicGames.CommandPattern.IrreversibleCompositeFailureException	24
SadSapphicGames.CommandPattern.ReversibleCompositeFailureException	29
SadSapphicGames.CommandPattern.ICommand	23
SadSapphicGames.CommandPattern.Command	12
SadSapphicGames.CommandPattern.AsyncCommand	9
SadSapphicGames.CommandPattern.CompositeCommand	19
SadSapphicGames.CommandPattern.NullCompositeCommand	27
SadSapphicGames.CommandPattern.SimpleComposite	29
SadSapphicGames.CommandPattern.NullCommand	26
SadSapphicGames.CommandPattern.IAsyncCommand	21
SadSapphicGames.CommandPattern.AsyncCommand	9
SadSapphicGames.CommandPattern.IFailable	23
SadSapphicGames.CommandPattern.IUndoable	25
SadSapphicGames.CommandPattern.NullCommand	26
SadSapphicGames.CommandPattern.NullCompositeCommand	27
MonoBehaviour	
SadSapphicGames.CommandPattern.SingletonCommandManager	30

Chapter 3

Class Index

3.1 Class List

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

SadSapphicGames.CommandPattern.AsyncCommand	
The base class of all commands who's execute method involves asynchronous tasks	9
SadSapphicGames.CommandPattern.Command	
The base class of all commands	12
SadSapphicGames.CommandPattern.CommandStream	
This is the object that stores commands to be invoked and executes them when told to by the client. It has no knowledge of the implementation of commands beyond their interfaces.	12
SadSapphicGames.CommandPattern.CompositeCommand	
A Command that is composed of multiple child commands, all of which are executed together and leave one record in the CommandStream 's history. <remark> For more information on this type of object seek external documentation on the composite design pattern </remark> . . .	19
SadSapphicGames.CommandPattern.IAsyncCommand	
This Interface of the AsyncCommand abstract class, It is strongly recommended you use that class rather than implement this yourself unless you are very familiar with asynchronous programming	
21	
SadSapphicGames.CommandPattern.ICommand	
This is the Interface of the Command abstract class, unless you are defining your own base type for commands you should probably inherit from Command over this	23
SadSapphicGames.CommandPattern.IFailable	
Interface implemented by commands that could fail to execute. Commands that would fail do not have their execute method invoked and are not recorded in the CommandStream 's history . .	23
SadSapphicGames.CommandPattern.IrreversibleCompositeFailureException	
An exception that indicates a CompositeCommand is executed but one of its children failed and the composite cannot undo its executed commands	24
SadSapphicGames.CommandPattern.IUndoable	
Indicates a command can be undone	25
SadSapphicGames.CommandPattern.NullCommand	
A Command that does nothing	26
SadSapphicGames.CommandPattern.NullCompositeCommand	
Like the NullCommand this is a composite command that does nothing, multiple times.	27
SadSapphicGames.CommandPattern.ReversibleCompositeFailureException	
An exception that indicates a CompositeCommand is executed but one of its children failed, however the composite was able to undo the commands it had executed	29
SadSapphicGames.CommandPattern.SimpleComposite	
A CompositeCommand created from a collection of Command 's that cannot fail	29

[SadSapphicGames.CommandPattern.SingletonCommandManager](#)

A singleton manager for a single-stream, out of the box implementation of the [Command](#) pattern. For more complicated implementations create your own [CommandStream](#) wrappers. Does not drop Commands from its CommandHistory. Executes the next command in the [CommandStream](#) every frame.

30

Chapter 4

Namespace Documentation

4.1 SadSapphicGames Namespace Reference

4.2 SadSapphicGames.CommandPattern Namespace Reference

Classes

- class [AsyncCommand](#)
The base class of all commands who's execute method involves asynchronous tasks
- class [Command](#)
The base class of all commands
- class [CommandStream](#)
This is the object that stores commands to be invoked and executes them when told to by the client. It has no knowledge of the implementation of commands beyond their interfaces.
- class [CompositeCommand](#)
A [Command](#) that is composed of multiple child commands, all of which are executed together and leave one record in the [CommandStream](#)'s history. <remark> For more information on this type of object seek external documentation on the composite design pattern </remark>
- interface [IAsyncCommand](#)
This Interface of the [AsyncCommand](#) abstract class, It is strongly recommended you use that class rather than implement this yourself unless you are very familiar with asynchronous programming
- interface [ICommand](#)
This is the Interface of the [Command](#) abstract class, unless you are defining your own base type for commands you should probably inherit from [Command](#) over this
- interface [IFailable](#)
Interface implemented by commands that could fail to execute. Commands that would fail do not have their execute method invoked and are not recorded in the [CommandStream](#)'s history
- class [IrreversibleCompositeFailureException](#)
An exception that indicates a [CompositeCommand](#) is executed but one of its children failed and the composite cannot undo its executed commands
- interface [IUndoable](#)
Indicates a command can be undone
- class [NullCommand](#)
A [Command](#) that does nothing
- class [NullCompositeCommand](#)

Like the [NullCommand](#) this is a composite command that does nothing, multiple times.

- class [ReversibleCompositeFailureException](#)

An exception that indicates a [CompositeCommand](#) is executed but one of its children failed, however the composite was able to undo the commands it had executed

- class [SimpleComposite](#)

A [CompositeCommand](#) created from a collection of [Command](#)'s that cannot fail

- class [SingletonCommandManager](#)

A singleton manager for a single-stream, out of the box implementation of the [Command](#) pattern. For more complicated implementations create your own [CommandStream](#) wrappers. Does not drop Commands from its [CommandStream](#) History. Executes the next command in the [CommandStream](#) every frame.

Enumerations

- enum [ExecuteCode](#) {
[Success](#) , [Failure](#) , [QueueEmpty](#) , [CompositeFailure](#) ,
[AwaitingCompletion](#) }

The possible return values of [CommandStream.TryExecuteNext\(\)](#)

4.2.1 Enumeration Type Documentation

4.2.1.1 ExecuteCode

```
enum SadSapphicGames.CommandPattern.ExecuteCode
```

The possible return values of [CommandStream.TryExecuteNext\(\)](#)

Enumerator

Success	Top Command executed successfully
Failure	Top Command would fail
QueueEmpty	Command queue was empty
CompositeFailure	Top Command was a CompositeCommand that did not indicate it would fail but failed partway through execution - was reversible
AwaitingCompletion	Top command was an AsyncCommand that is awaiting completion

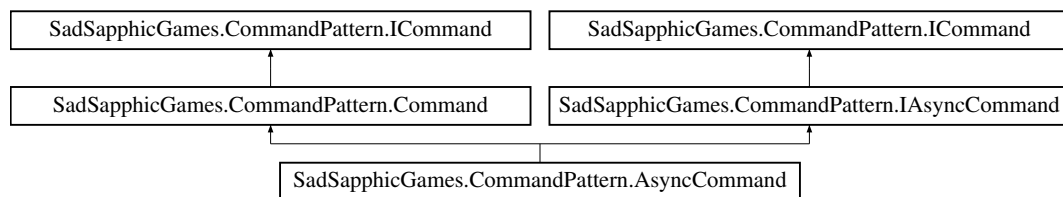
Chapter 5

Class Documentation

5.1 SadSapphicGames.CommandPattern.AsyncCommand Class Reference

The base class of all commands who's execute method involves asynchronous tasks

Inheritance diagram for SadSapphicGames.CommandPattern.AsyncCommand:



Public Member Functions

- sealed override void [Execute](#) ()
This method may not be overridden in asynchronous commands. Use `ExecuteAsync` for your command's logic instead.
- abstract Task [ExecuteAsync](#) ()
Executes the command asynchronously. Remember to add the `async` keyword as that is not considered part of the method signature and cannot be added to abstract methods.
IMPORTANT: if you want to be able to cancel the `AsyncCommand`'s task you must invoke `CancellationToken.ThrowIfCancellationRequested()` within this method somewhere after the first `await`. If you do not do so attempting to cancel it will do nothing.

Protected Member Functions

- AsyncCommand** ()
Created the cancellation token of the async command and sets up its disposal once the task completes (wether that is from success, cancellation, or faulting)

Properties

- Task [CommandTask](#) [get]
The task for the completion of the ExecuteAsync method after it reaches its first await and returns control back to the calling method ([CommandStream.TryExecuteNext\(\)](#))
- CancellationTokenSource [CancellationTokenSource](#) [get]
This can be used to signal to CommandTask that it should be canceled, provided that you respond to Cancellation←Token indicating it has been canceled from ExecuteAsync
- CancellationToken **CancellationToken** [get]
This can be used in ExecuteAsync to determine if the CommandTask has been canceled.

Events

- Action **OnTaskCompleted**
This event is invoked when CommandTask is completed. I.E. - when we reach the end of [ExecuteAsync\(\)](#). Subscribe to it to preform an action at after the command has fully completed.
- Action **OnTaskCanceled**
This event is invoked if the command task is canceled. Most bookkeeping that occurs in this situation is handled by the package but you can subscribe to this as well if needed.
- Action< Exception > **OnTaskFaulted**
This event is invoked if the command task throws an exception.

5.1.1 Detailed Description

The base class of all commands who's execute method involves asynchronous tasks

5.1.2 Member Function Documentation

5.1.2.1 Execute()

```
sealed override void SadSapphicGames.CommandPattern.AsyncCommand.Execute ( ) [virtual]
```

This method may not be overridden in asynchronous commands. Use ExecuteAsync for your command's logic instead.

Implements [SadSapphicGames.CommandPattern.Command](#).

5.1.2.2 ExecuteAsync()

```
abstract Task SadSapphicGames.CommandPattern.AsyncCommand.ExecuteAsync ( ) [pure virtual]
```

Executes the command asynchronously. Remember to add the async keyword as that is not considered part of the method signature and cannot be added to abstract methods.

IMPORTANT: if you want to be able to cancel the [AsyncCommand](#)'s task you must invoke `CancellationToken.ThrowIfCancellationRequested()` within this method somewhere after the first await. If you do not do so attempting to cancel it will do nothing.

Returns

The task for the completion of this method after it reaches its first await and returns control to the calling method.

Implements [SadSapphicGames.CommandPattern.IAsyncCommand](#).

5.1.3 Property Documentation

5.1.3.1 CancellationTokenSource

```
CancellationTokenSource SadSapphicGames.CommandPattern.AsyncCommand.CancellationTokenSource  
[get]
```

This can be used to signal to `CommandTask` that it should be canceled, provided that you respond to `CancellationToken` indicating it has been canceled from `ExecuteAsync`

Implements [SadSapphicGames.CommandPattern.IAsyncCommand](#).

5.1.3.2 CommandTask

```
Task SadSapphicGames.CommandPattern.AsyncCommand.CommandTask [get]
```

The task for the completion of the `ExecuteAsync` method after it reaches its first await and returns control back to the calling method ([CommandStream.TryExecuteNext\(\)](#))

Implements [SadSapphicGames.CommandPattern.IAsyncCommand](#).

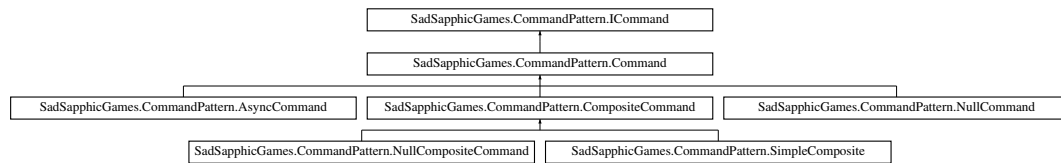
The documentation for this class was generated from the following file:

- C:/Users/Layla/Documents/My Docs/GameDev/CommandPatternDevelopment/Assets/Packages/CommandPattern/Runtime/Commands/AsyncCommand.cs

5.2 SadSapphicGames.CommandPattern.Command Class Reference

The base class of all commands

Inheritance diagram for SadSapphicGames.CommandPattern.Command:



Public Member Functions

- abstract void [Execute](#) ()
Executes the command

5.2.1 Detailed Description

The base class of all commands

5.2.2 Member Function Documentation

5.2.2.1 Execute()

```
abstract void SadSapphicGames.CommandPattern.Command.Execute ( ) [pure virtual]
```

Executes the command

Implements [SadSapphicGames.CommandPattern.ICommand](#).

Implemented in [SadSapphicGames.CommandPattern.AsyncCommand](#), [SadSapphicGames.CommandPattern.CompositeCommand](#), and [SadSapphicGames.CommandPattern.NullCommand](#).

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

- C:/Users/Layla/Documents/My Docs/GameDev/CommandPatternDevelopment/Assets/Packages/CommandPattern/Runtime/Commands/Command.cs

5.3 SadSapphicGames.CommandPattern.CommandStream Class Reference

This is the object that stores commands to be invoked and executes them when told to by the client. It has no knowledge of the implementation of commands beyond their interfaces.

Public Member Functions

- `ReadOnlyCollection< ICommand > GetCommandHistory ()`
Get the [CommandStream](#)'s history of executed Commands.
- `ReadOnlyCollection< ICommand > GetCommandQueue ()`
Get the queue of commands to be executed by the command stream.
- `ReadOnlyCollection< Task > GetRunningCommandTasks ()`
Get the Tasks being being run by AsyncCommands executed by this [CommandStream](#)
- `void CancelRunningCommandTask (Task task)`
Cancels a task if that task is currently running
- `void CancelRunningCommandTask (IAsyncCommand asyncCommand)`
Cancels the task of an [IAsyncCommand](#) if that task is currently running.
- `ReadOnlyDictionary< Task, Exception > GetFaultedCommandTasks ()`
Get a dictionary of any faulted tasks and the exceptions that they threw
- `CommandStream (float _historyDepth=Single.PositiveInfinity)`
Creates a new [CommandStream](#)
- `void QueueCommand (ICommand command)`
This adds a [Command](#) to the command Queue
- `void QueueCommands (IEnumerable< ICommand > commands)`
Adds multiple Commands to the queue
- `List< ICommand > DropQueue ()`
This will remove all commands from the [CommandStream](#)'s queue and replace it with a new empty queue.
- `ReadOnlyCollection< ICommand > DropHistory ()`
This will remove all commands from the [CommandStream](#)'s history and replace it with a new empty list.
- `bool TryQueueUndoCommand (IUndoable commandToUndo)`
Attempt to queue's the undo command of a [Command](#) object implementing [IUndoable](#) if that command exists in this [CommandStream](#)'s history
- `void ForceQueueUndoCommand (IUndoable commandToUndo)`
Force the stream to queue's the undo command of a [Command](#) object implementing [IUndoable](#) regardless of whether the command is recorded in this [CommandStream](#)'s history
- `ExecuteCode TryExecuteNext (out ICommand topCommand)`
Attempts to execute the next command in the queue, returns an enum indicating if it was able to or not or if the queue was empty or the command is async and awaiting completion.
- `ExecuteCode TryExecuteNext ()`
Attempts to execute the next command in the queue, returns false if it is empty or the command is [IFailable](#) and would fail.
- `void ExecuteFullQueue ()`
Execute's Commands from the [CommandStream](#) queue until it is empty. Be warned this will not give any indication of commands failing.
- `void ExecuteFullQueue (out List< ICommand > failedCommands)`
Executes Commands from the [CommandStream](#)'s queue until it is empty. Returns the a list of any Commands that failed as an out parameter

Properties

- `bool QueueEmpty` [get]
Gets `commandQueue.Count == 0`
- `float HistoryDepth` [get]
this is the maximum number of commands that will be recorded in the [CommandHistory](#)
- `int HistoryCount` [get]
Gets the number of Commands currently recorded in the [CommandStream](#)'s history.
- `int QueueCount` [get]
Gets the number of Commands currently waiting in the [CommandStream](#)'s queue.

Events

- Action< Exception > **OnTaskFaulted**

This event will be invoked if one of the tasks from an [IAsyncCommand](#) executed by this stream faults. Can be used to throw any exception's caused by tasks rather than storing them on the task object

5.3.1 Detailed Description

This is the object that stores commands to be invoked and executes them when told to by the client. It has no knowledge of the implementation of commands beyond their interfaces.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 CommandStream()

```
SadSapphicGames.CommandPattern.CommandStream.CommandStream (
    float _historyDepth = Single.PositiveInfinity )
```

Creates a new [CommandStream](#)

Parameters

<code>_historyDepth</code>	The depth to which previously executed commands will be recorded. Once this depth is reached the oldest commands will be forgotten first. <remark> To not record history, set to zero. To never forget executed commands, set to positive infinity. </remark>
----------------------------	---

5.3.3 Member Function Documentation

5.3.3.1 CancelRunningCommandTask() [1/2]

```
void SadSapphicGames.CommandPattern.CommandStream.CancelRunningCommandTask (
    IAsyncCommand asyncCommand )
```

Cancels the task of an [IAsyncCommand](#) if that task is currently running.

Parameters

<code>asyncCommand</code>	The IAsyncCommand to cancel the task off
---------------------------	--

5.3.3.2 CancelRunningCommandTask() [2/2]

```
void SadSapphicGames.CommandPattern.CommandStream.CancelRunningCommandTask (
    Task task )
```

Cancels a task if that task is currently running

Parameters

<i>task</i>	The task to cancel
-------------	--------------------

5.3.3.3 DropHistory()

```
ReadOnlyCollection< ICommand > SadSapphicGames.CommandPattern.CommandStream.DropHistory ( )
```

This will remove all commands from the [CommandStream](#)'s history and replace it with a new empty list.

<remark> This can be useful if you need the [CommandStream](#) to record all of its history but also need it to execute an extremely large number of commands without running out of memory. Even if every command is the same object a [CommandStream](#) will run out of memory at 2-3 hundred million commands in its queue or history. </remark>

Returns

The commands in the previous history, in case this information is needed

5.3.3.4 DropQueue()

```
List< ICommand > SadSapphicGames.CommandPattern.CommandStream.DropQueue ( )
```

This will remove all commands from the [CommandStream](#)'s queue and replace it with a new empty queue.

<remark> This can be useful to rearrange the commands in a queue. Simple preform the needed changes on the returned list and re-queue it </remark>

Returns

The commands in the previous queue, in case this information is needed.

5.3.3.5 ExecuteFullQueue()

```
void SadSapphicGames.CommandPattern.CommandStream.ExecuteFullQueue (
    out List< ICommand > failedCommands )
```

Executes Commands from the [CommandStream](#)'s queue until it is empty. Returns the a list of any Commands that failed as an out parameter

Parameters

<i>failedCommands</i>	A list of any Commands in the Queue that failed to execute
-----------------------	--

5.3.3.6 ForceQueueUndoCommand()

```
void SadSapphicGames.CommandPattern.CommandStream.ForceQueueUndoCommand (
    IUndoable commandToUndo )
```

Force the stream to queue's the undo command of a [Command](#) object implementing [IUndoable](#) regardless of whether the command is recorded in this [CommandStream](#)'s history

<remark>This is equivalent to passing the result of [IUndoable.GetUndoCommand\(\)](#) into [CommandStream.QueueCommand\(Command command\)](#) directly</remark>

Parameters

<i>commandToUndo</i>	The IUndoable Command to queue the undo-Command of
----------------------	--

5.3.3.7 GetCommandHistory()

```
ReadOnlyCollection< ICommand > SadSapphicGames.CommandPattern.CommandStream.GetCommandHistory
( )
```

Get the [CommandStream](#)'s history of executed Commands.

Returns

The history of executed commands, null if history is not recorded.

5.3.3.8 GetCommandQueue()

```
ReadOnlyCollection< ICommand > SadSapphicGames.CommandPattern.CommandStream.GetCommandQueue (
)
```

Get the queue of commands to be executed by the command stream.

Returns

The queue of commands the commandStream will execute.

5.3.3.9 GetFaultedCommandTasks()

```
ReadOnlyDictionary< Task, Exception > SadSapphicGames.CommandPattern.CommandStream.GetFaultedCommandTasks ( )
```

Get a dictionary of any faulted tasks and the exceptions that they threw

Returns

A read only copy of the dictionary of faulted tasks and their exceptions

5.3.3.10 GetRunningCommandTasks()

```
ReadOnlyCollection< Task > SadSapphicGames.CommandPattern.CommandStream.GetRunningCommandTasks ( )
```

Get the Tasks being being run by AsyncCommands executed by this [CommandStream](#)

Returns

The tasks that are currently being run

5.3.3.11 QueueCommand()

```
void SadSapphicGames.CommandPattern.CommandStream.QueueCommand (
    ICommand command )
```

This adds a [Command](#) to the command Queue

Parameters

<i>command</i>	The Command to be Queued
----------------	--

5.3.3.12 QueueCommands()

```
void SadSapphicGames.CommandPattern.CommandStream.QueueCommands (
    IEnumerable< ICommand > commands )
```

Adds multiple Commands to the queue

Parameters

<i>commands</i>	The commands to be queued
-----------------	---------------------------

5.3.3.13 TryExecuteNext() [1/2]

`ExecuteCode` SadSapphicGames.CommandPattern.CommandStream.TryExecuteNext ()

Attempts to execute the next command in the queue, returns false if it is empty or the command is `IFailable` and would fail.

Returns

False if the command queue is empty, or the next command would fail. True otherwise.

5.3.3.14 TryExecuteNext() [2/2]

`ExecuteCode` SadSapphicGames.CommandPattern.CommandStream.TryExecuteNext (
 out `ICommand` *topCommand*)

Attempts to execute the next command in the queue, returns an enum indicating if it was able to or not or if the queue was empty or the command is async and awaiting completion.

Parameters

<i>topCommand</i>	The command that was next in the queue, null if the queue was empty
-------------------	---

Returns

An `ExecuteCode` enum value indicating what happened when attempting to execute the next command

5.3.3.15 TryQueueUndoCommand()

`bool` SadSapphicGames.CommandPattern.CommandStream.TryQueueUndoCommand (
 `IUndoable` *commandToUndo*)

Attempt to queue's the undo command of a `Command` object implementing `IUndoable` if that command exists in this `CommandStream`'s history

Parameters

<i>commandToUndo</i>	The <code>IUndoable Command</code> to try and queue the undo-Command of
----------------------	---

Returns

Wether the undo command was queued

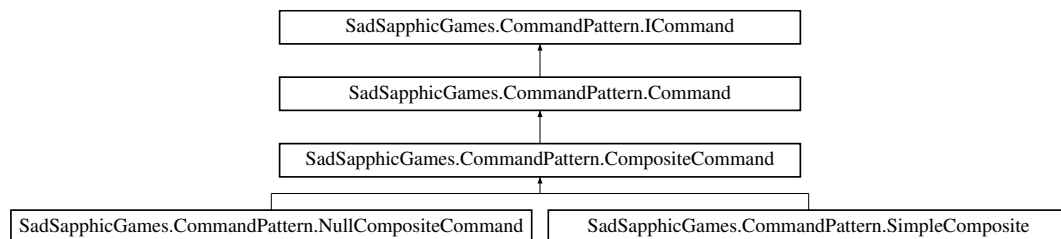
The documentation for this class was generated from the following file:

- C:/Users/Layla/Documents/My Docs/GameDev/CommandPatternDevelopment/Assets/Packages/CommandPattern/Runtime/Commands/CommandStream.cs

5.4 SadSapphicGames.CommandPattern.CompositeCommand Class Reference

A [Command](#) that is composed of multiple child commands, all of which are executed together and leave one record in the [CommandStream](#)'s history. <remark> For more information on this type of object seek external documentation on the composite design pattern </remark>

Inheritance diagram for SadSapphicGames.CommandPattern.CompositeCommand:



Public Member Functions

- override void [Execute](#) ()
Queues all of the child commands into the internal [CommandStream](#) and attempts to invoke all of them. Will throw an exception if one of its children fails after attempting to revert all its executed commands.

Protected Member Functions

- virtual void [AddChild](#) (ICommand childCommand)
Adds a [Command](#) to this objects children

Protected Attributes

- List< [ICommand](#) > **subCommands** = new List<[ICommand](#)>()
The child Commands that will be executed upon executing this object
- [CommandStream](#) **internalStream** = new [CommandStream](#)()
An internal [CommandStream](#) to provide more control of the execution of the subCommands of the Composite

Properties

- int **ChildCount** [get]
Number of child Commands included in this object

5.4.1 Detailed Description

A [Command](#) that is composed of multiple child commands, all of which are executed together and leave one record in the [CommandStream](#)'s history. <remark> For more information on this type of object seek external documentation on the composite design pattern </remark>

5.4.2 Member Function Documentation

5.4.2.1 AddChild()

```
virtual void SadSapphicGames.CommandPattern.CompositeCommand.AddChild (
    ICommand childCommand ) [protected], [virtual]
```

Adds a [Command](#) to this objects children

Parameters

<i>childCommand</i>	The Command to be added to the objects children
---------------------	---

5.4.2.2 Execute()

```
override void SadSapphicGames.CommandPattern.CompositeCommand.Execute ( ) [virtual]
```

Queues all of the child commands into the internal [CommandStream](#) and attempts to invoke all of them. Will throw an exception if one of its children fails after attempting to revert all its executed commands.

Be aware that if you override this method you will bypass the implemented failsafe's for children of the [CompositeCommand](#) failing such as attempting to undo executed commands

Exceptions

<i>IrreversibleCompositeFailureException</i>	Indicates one of the children of the CompositeCommand failed and it executed one or more commands that cannot be undone. TryExecuteNext will catch this exception and throw it upwards
<i>ReversibleCompositeFailureException</i>	Indicates one of the children of the CompositeCommand failed but it was able to undo all of its executed commands. TryExecuteNext will catch this exception and handle it by returning false.

Implements [SadSapphicGames.CommandPattern.Command](#).

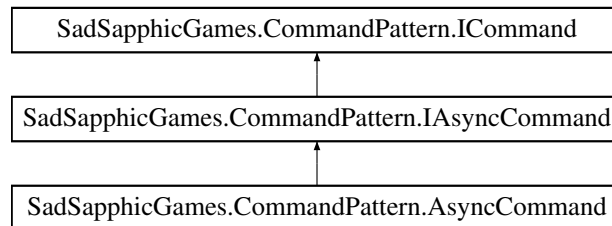
The documentation for this class was generated from the following file:

- C:/Users/Layla/Documents/My Docs/GameDev/CommandPatternDevelopment/Assets/Packages/CommandPattern/Runtime/Commands/CompositeCommand.cs

5.5 SadSapphicGames.CommandPattern.IAsyncCommand Interface Reference

This Interface of the [AsyncCommand](#) abstract class, It is strongly recommended you use that class rather than implement this yourself unless you are very familiar with asynchronous programming

Inheritance diagram for SadSapphicGames.CommandPattern.IAsyncCommand:



Public Member Functions

- abstract Task [ExecuteAsync](#) ()

This is where the logic of executing the command should be placed for an [AsyncCommand](#), Execute should just store the return in CommandTask and setup the OnTaskCompleted method. Remember to make this method async as that isn't considered part of its signature.

Properties

- Task [CommandTask](#) [get]

This should get the asynchronous task returned by ExecuteAsync after it reaches its first await

- CancellationTokenSource [CancellationTokensource](#) [get]

This can be used to Cancel the task after it has been started.

Events

- Action **OnTaskCompleted**

This event should be invoked when CommandTask is successfully completed.

- Action **OnTaskCanceled**

This event should be invoked when CommandTask is cancelled.

- Action< Exception > **OnTaskFaulted**

This event should be invoked when CommandTask throws an exception

5.5.1 Detailed Description

This Interface of the [AsyncCommand](#) abstract class, It is strongly recommended you use that class rather than implement this yourself unless you are very familiar with asynchronous programming

5.5.2 Member Function Documentation

5.5.2.1 ExecuteAsync()

```
abstract Task SadSapphicGames.CommandPattern.IAsyncCommand.ExecuteAsync ( ) [pure virtual]
```

This is where the logic of executing the command should be placed for an [AsyncCommand](#), Execute should just store the return in CommandTask and setup the OnTaskCompleted method. Remember to make this method async as that isn't considered part of its signature.

Returns

The Task representing the completion of the method after it reaches its first await statement

Implemented in [SadSapphicGames.CommandPattern.AsyncCommand](#).

5.5.3 Property Documentation

5.5.3.1 CancellationTokenSource

```
CancellationTokenSource SadSapphicGames.CommandPattern.IAsyncCommand.CancellationTokenSource  
[get]
```

This can be used to Cancel the task after it has been started.

Implemented in [SadSapphicGames.CommandPattern.AsyncCommand](#).

5.5.3.2 CommandTask

```
Task SadSapphicGames.CommandPattern.IAsyncCommand.CommandTask [get]
```

This should get the asynchronous task returned by ExecuteAsync after it reaches its first await

Implemented in [SadSapphicGames.CommandPattern.AsyncCommand](#).

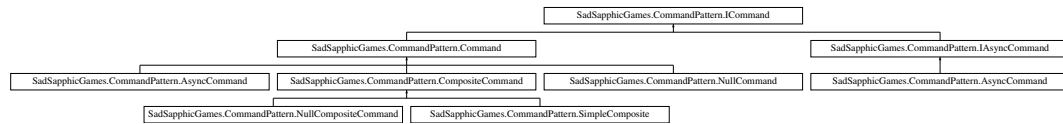
The documentation for this interface was generated from the following file:

- C:/Users/Layla/Documents/My Docs/GameDev/CommandPatternDevelopment/Assets/Packages/CommandPattern/Runtime/Interfaces/IAsyncCommand.cs

5.6 SadSapphicGames.CommandPattern.ICommand Interface Reference

This is the Interface of the [Command](#) abstract class, unless you are defining your own base type for commands you should probably inherit from [Command](#) over this

Inheritance diagram for SadSapphicGames.CommandPattern.ICommand:



Public Member Functions

- abstract void [Execute](#) ()
Executes the function of the command

5.6.1 Detailed Description

This is the Interface of the [Command](#) abstract class, unless you are defining your own base type for commands you should probably inherit from [Command](#) over this

5.6.2 Member Function Documentation

5.6.2.1 Execute()

```
abstract void SadSapphicGames.CommandPattern.ICommand.Execute ( ) [pure virtual]
```

Executes the function of the command

Implemented in [SadSapphicGames.CommandPattern.AsyncCommand](#), [SadSapphicGames.CommandPattern.Command](#), [SadSapphicGames.CommandPattern.CompositeCommand](#), and [SadSapphicGames.CommandPattern.NullCommand](#).

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

- C:/Users/Layla/Documents/My Docs/GameDev/CommandPatternDevelopment/Assets/Packages/CommandPattern/Runtime/Interfaces/ICommand.cs

5.7 SadSapphicGames.CommandPattern.IFailable Interface Reference

Interface implemented by commands that could fail to execute. Commands that would fail do not have their execute method invoked and are not recorded in the [CommandStream](#)'s history

Public Member Functions

- bool [WouldFail](#) ()

Determine if the implementing command would be able to be executed or if it would fail

5.7.1 Detailed Description

Interface implemented by commands that could fail to execute. Commands that would fail do not have their execute method invoked and are not recorded in the [CommandStream](#)'s history

5.7.2 Member Function Documentation

5.7.2.1 WouldFail()

```
bool SadSapphicGames.CommandPattern.IFailable.WouldFail ( )
```

Determine if the implementing command would be able to be executed or if it would fail

Returns

True if the implementing command would fail, false if it would execute successfully.

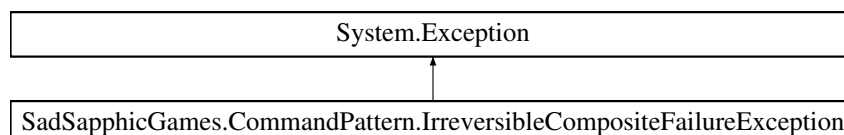
The documentation for this interface was generated from the following file:

- C:/Users/Layla/Documents/My Docs/GameDev/CommandPatternDevelopment/Assets/Packages/CommandPattern/Runtime/Interfaces/IFailable.cs

5.8 SadSapphicGames.CommandPattern.IrreversibleCompositeFailureException Class Reference

An exception that indicates a [CompositeCommand](#) is executed but one of its children failed and the composite cannot undo its executed commands

Inheritance diagram for SadSapphicGames.CommandPattern.IrreversibleCompositeFailureException:



Public Member Functions

- **IrreversibleCompositeFailureException** ([ICommand](#) failedCommand, List< [ICommand](#) > irreversibleCommands)

Public Attributes

- readonly [ICommand](#) **failedCommand**
The child command that failed
- readonly List< [ICommand](#) > **irreversibleCommands**
The executed commands that could not be undone

5.8.1 Detailed Description

An exception that indicates a [CompositeCommand](#) is executed but one of its children failed and the composite cannot undo its executed commands

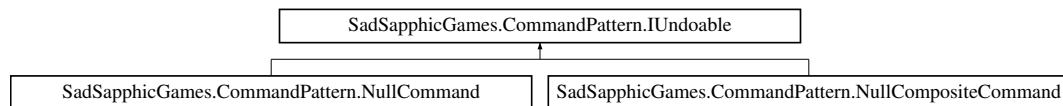
The documentation for this class was generated from the following file:

- C:/Users/Layla/Documents/My Docs/GameDev/CommandPatternDevelopment/Assets/Packages/CommandPattern/Runtime/Commands/CompositeCommand.cs

5.9 SadSapphicGames.CommandPattern.IUndoable Interface Reference

Indicates a command can be undone

Inheritance diagram for SadSapphicGames.CommandPattern.IUndoable:



Public Member Functions

- [ICommand](#) **GetUndoCommand** ()
Creates a command to revert the changes of the implementing command

5.9.1 Detailed Description

Indicates a command can be undone

5.9.2 Member Function Documentation

5.9.2.1 GetUndoCommand()

`ICommand SadSapphicGames.CommandPattern.IUndoable.GetUndoCommand ()`

Creates a command to revert the changes of the implementing command

Returns

a command that reverts the implementing command

Implemented in [SadSapphicGames.CommandPattern.NullCommand](#), and [SadSapphicGames.CommandPattern.NullCompositeCommand](#)

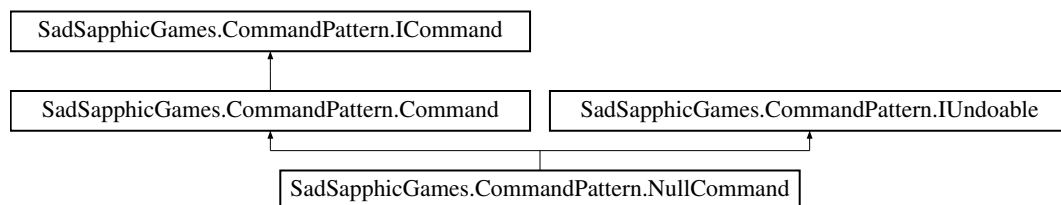
The documentation for this interface was generated from the following file:

- C:/Users/Layla/Documents/My Docs/GameDev/CommandPatternDevelopment/Assets/Packages/CommandPattern/Runtime/Interfaces/IUndoable.cs

5.10 SadSapphicGames.CommandPattern.NullCommand Class Reference

A [Command](#) that does nothing

Inheritance diagram for SadSapphicGames.CommandPattern.NullCommand:



Public Member Functions

- override void [Execute](#) ()
Does Nothing
- [ICommand GetUndoCommand](#) ()
Since the command doesn't do anything it returns itself

5.10.1 Detailed Description

A [Command](#) that does nothing

5.10.2 Member Function Documentation

5.10.2.1 Execute()

```
override void SadSapphicGames.CommandPattern.NullCommand.Execute ( ) [virtual]
```

Does Nothing

Implements [SadSapphicGames.CommandPattern.Command](#).

5.10.2.2 GetUndoCommand()

```
ICommand SadSapphicGames.CommandPattern.NullCommand.GetUndoCommand ( )
```

Since the command doesn't do anything it returns itself

Returns

The same null command object

Implements [SadSapphicGames.CommandPattern.IUndoable](#).

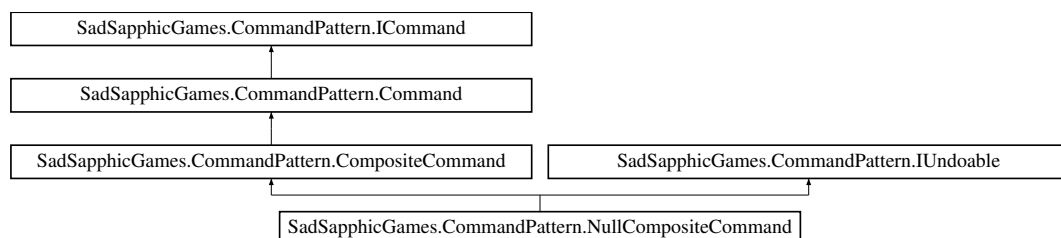
The documentation for this class was generated from the following file:

- C:/Users/Layla/Documents/My Docs/GameDev/CommandPatternDevelopment/Assets/Packages/CommandPattern/Runtime/Commands/NullCommand.cs

5.11 SadSapphicGames.CommandPattern.NullCompositeCommand Class Reference

Like the [NullCommand](#) this is a composite command that does nothing, multiple times.

Inheritance diagram for SadSapphicGames.CommandPattern.NullCompositeCommand:



Public Member Functions

- [NullCompositeCommand](#) (int size)
Creates a [NullCompositeCommand](#) composed of multiple [NullCommands](#)
- [ICommand GetUndoCommand](#) ()
Like the [NullCommand](#) it is composed of, a [NullCompositeCommand](#) is its own undo-command

Additional Inherited Members

5.11.1 Detailed Description

Like the [NullCommand](#) this is a composite command that does nothing, multiple times.

5.11.2 Constructor & Destructor Documentation

5.11.2.1 NullCompositeCommand()

```
SadSapphicGames.CommandPattern.NullCompositeCommand.NullCompositeCommand (
    int size )
```

Creates a [NullCompositeCommand](#) composed of multiple NullCommands

Parameters

<i>size</i>	How many NullCommands to include in the composite
-------------	---

5.11.3 Member Function Documentation

5.11.3.1 GetUndoCommand()

```
ICommand SadSapphicGames.CommandPattern.NullCompositeCommand.GetUndoCommand ( )
```

Like the [NullCommand](#) it is composed of, a [NullCompositeCommand](#) is its own undo-command

Returns

This object

Implements [SadSapphicGames.CommandPattern.IUndoable](#).

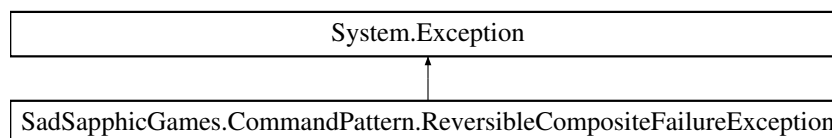
The documentation for this class was generated from the following file:

- C:/Users/Layla/Documents/My Docs/GameDev/CommandPatternDevelopment/Assets/Packages/CommandPattern/Runtime/Commands/NullCompositeCommand.cs

5.12 SadSapphicGames.CommandPattern.ReversibleCompositeFailureException Class Reference

An exception that indicates a [CompositeCommand](#) is executed but one of its children failed, however the composite was able to undo the commands it had executed

Inheritance diagram for SadSapphicGames.CommandPattern.ReversibleCompositeFailureException:



Public Member Functions

- **ReversibleCompositeFailureException** ([ICommand](#) [failedCommand](#))

Public Attributes

- readonly [ICommand](#) **failedCommand**
The child command that failed

5.12.1 Detailed Description

An exception that indicates a [CompositeCommand](#) is executed but one of its children failed, however the composite was able to undo the commands it had executed

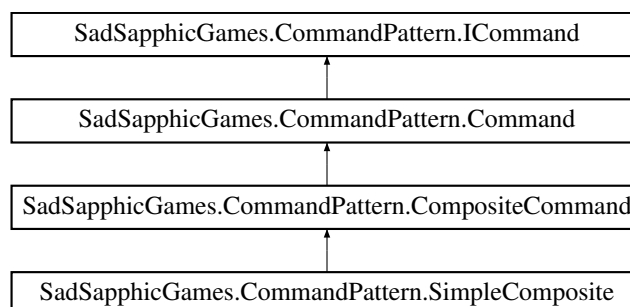
The documentation for this class was generated from the following file:

- C:/Users/Layla/Documents/My Docs/GameDev/CommandPatternDevelopment/Assets/Packages/CommandPattern/Runtime/Commands/CompositeCommand.cs

5.13 SadSapphicGames.CommandPattern.SimpleComposite Class Reference

A [CompositeCommand](#) created from a collection of [Command](#)'s that cannot fail

Inheritance diagram for SadSapphicGames.CommandPattern.SimpleComposite:



Public Member Functions

- [SimpleComposite](#) (IEnumerable< [Command](#) > [subCommands](#))
Creates a [SimpleComposite](#) from a collection of [Command](#)'s that cannot fail

Additional Inherited Members

5.13.1 Detailed Description

A [CompositeCommand](#) created from a collection of [Command](#)'s that cannot fail

5.13.2 Constructor & Destructor Documentation

5.13.2.1 SimpleComposite()

```
SadSapphicGames.CommandPattern.SimpleComposite.SimpleComposite (
    IEnumerable< Command > subCommands )
```

Creates a [SimpleComposite](#) from a collection of [Command](#)'s that cannot fail

Parameters

subCommands	The collection of unfailable Commands to be included in the composite
-----------------------------	---

Exceptions

System.ArgumentException	One or more of the Commands included in the argument implement IFailable
--	--

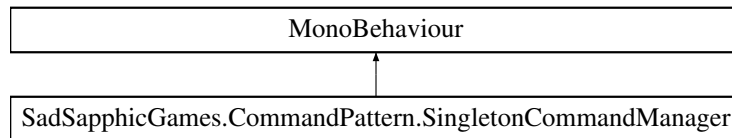
The documentation for this class was generated from the following file:

- C:/Users/Layla/Documents/My Docs/GameDev/CommandPatternDevelopment/Assets/Packages/CommandPattern/Runtime/Commands/SimpleComposite.cs

5.14 SadSapphicGames.CommandPattern.SingletonCommandManager Class Reference

A singleton manager for a single-stream, out of the box implementation of the [Command](#) pattern. For more complicated implementations create your own [CommandStream](#) wrappers. Does not drop Commands from its CommandHistory. Executes the next command in the [CommandStream](#) every frame.

Inheritance diagram for SadSapphicGames.CommandPattern.SingletonCommandManager:



Public Member Functions

- void **ToggleCommandExecution** ()
Turns command execution off if its on and on if its off
- void **ToggleCommandExecution** (bool onoff)
Turns command execution off or on
- ReadOnlyCollection< [ICommand](#) > **GetCommandHistory** ()
Get the underlying [CommandStream](#)'s history
- ReadOnlyCollection< Task > **GetRunningCommandTasks** ()
Get the currently uncompleted tasks from executed AsyncCommands
- ReadOnlyCollection< [ICommand](#) > **DropCommandHistory** ()
Empties the history of the internal [CommandStream](#) and replaces it with an empty one.
- void **QueueCommand** ([ICommand](#) command)
Queue's a [Command](#) into the CommandManager's [CommandStream](#)
- void **QueueCommands** (IEnumerable< [ICommand](#) > commands)
Queue's multiple commands into the CommandManager's [CommandStream](#)
- bool **TryQueueUndoCommand** ([IUndoable](#) commandToUndo)
Queue the undo-command of a [Command](#) implementing [IUndoable](#) into the [CommandStream](#)
- void **ForceQueueUndoCommand** ([IUndoable](#) commandToUndo)
Forces the internal [CommandStream](#) to queue and [IUndoable](#) commands undo command

Public Attributes

- int **maximumHistoryDepth** = -1
The value that will be used in the internal [CommandStream](#)'s constructor, set to negative to record all history

Properties

- static [SingletonCommandManager](#) **Instance** [get]
The singleton instance of the CommandManger.
- int **HistoryCount** [get]
The number of Commands recorded by the CommandManager's [CommandStream](#)
- float **HistoryDepth** [get]
The depth to which the CommandManager's [CommandStream](#) records its history
- int **QueueCount** [get]
The Number of commands queued in the CommandManager's [CommandStream](#)
- bool **QueueEmpty** [get]
Wether or not the CommandManger's [CommandStream](#) has an empty queue

5.14.1 Detailed Description

A singleton manager for a single-stream, out of the box implementation of the [Command](#) pattern. For more complicated implementations create your own [CommandStream](#) wrappers. Does not drop Commands from its CommandHistory. Executes the next command in the [CommandStream](#) every frame.

5.14.2 Member Function Documentation

5.14.2.1 DropCommandHistory()

```
ReadOnlyCollection< ICommand > SadSapphicGames.CommandPattern.SingletonCommandManager.Drop↵
CommandHistory ( )
```

Empties the history of the internal [CommandStream](#) and replaces it with an empty one.

Returns

The old history of the internal [CommandStream](#)

5.14.2.2 ForceQueueUndoCommand()

```
void SadSapphicGames.CommandPattern.SingletonCommandManager.ForceQueueUndoCommand (
    IUndoable commandToUndo )
```

Forces the internal [CommandStream](#) to queue and [IUndoable](#) commands undo command

Parameters

<i>commandToUndo</i>	The IUndoable command to undo
----------------------	---

5.14.2.3 GetCommandHistory()

```
ReadOnlyCollection< ICommand > SadSapphicGames.CommandPattern.SingletonCommandManager.Get↵
CommandHistory ( )
```

Get the underlying [CommandStream](#)'s history

Returns

A [ReadOnlyCollection](#) of all the commands executed by the [CommandManager](#)'s [CommandStream](#)

5.14.2.4 GetRunningCommandTasks()

```
ReadOnlyCollection< Task > SadSapphicGames.CommandPattern.SingletonCommandManager.GetRunning↵
CommandTasks ( )
```

Get the currently uncompleted tasks from executed [AsyncCommands](#)

Returns

A [ReadOnlyCollection](#) of uncompleted tasks from executed [AsyncCommands](#)

5.14.2.5 QueueCommand()

```
void SadSapphicGames.CommandPattern.SingletonCommandManager.QueueCommand (
    ICommand command )
```

Queue's a [Command](#) into the CommandManager's [CommandStream](#)

Parameters

<i>command</i>	The Command to be Queued
----------------	--

5.14.2.6 QueueCommands()

```
void SadSapphicGames.CommandPattern.SingletonCommandManager.QueueCommands (
    IEnumerable< ICommand > commands )
```

Queue's multiple commands into the CommandManager's [CommandStream](#)

Parameters

<i>commands</i>	The collection of commands to be Queued
-----------------	---

5.14.2.7 ToggleCommandExecution()

```
void SadSapphicGames.CommandPattern.SingletonCommandManager.ToggleCommandExecution (
    bool onoff )
```

Turns command execution off or on

Parameters

<i>onoff</i>	if false stops the execution of commands, if true enables it
--------------	--

5.14.2.8 TryQueueUndoCommand()

```
bool SadSapphicGames.CommandPattern.SingletonCommandManager.TryQueueUndoCommand (
    IUndoable commandToUndo )
```

Queue the undo-command of a [Command](#) implementing [IUndoable](#) into the [CommandStream](#)

Parameters

<i>commandToUndo</i>	The IUndoable Command to queue an undo-command for
----------------------	--

Returns

Whether the undo command was allowed to be queued

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

- C:/Users/Layla/Documents/My Docs/GameDev/CommandPatternDevelopment/Assets/Packages/CommandPattern/Runtime/Monobehaviours/SingletonCommandManager.cs

Index

AddChild
SadSapphicGames.CommandPattern.CompositeCommand, 22
20
AwaitingCompletion
SadSapphicGames.CommandPattern, 8
CancellationTokenSource
SadSapphicGames.CommandPattern.AsyncCommand, 11
SadSapphicGames.CommandPattern.IAsyncCommand, 22
CancelRunningCommandTask
SadSapphicGames.CommandPattern.CommandStream, 14
CommandStream
SadSapphicGames.CommandPattern.CommandStream, 14
CommandTask
SadSapphicGames.CommandPattern.AsyncCommand, 11
SadSapphicGames.CommandPattern.IAsyncCommand, 22
CompositeFailure
SadSapphicGames.CommandPattern, 8
DropCommandHistory
SadSapphicGames.CommandPattern.SingletonCommandManager, 32
DropHistory
SadSapphicGames.CommandPattern.CommandStream, 15
DropQueue
SadSapphicGames.CommandPattern.CommandStream, 15
Execute
SadSapphicGames.CommandPattern.AsyncCommand, 10
SadSapphicGames.CommandPattern.Command, 12
SadSapphicGames.CommandPattern.CompositeCommand, 20
SadSapphicGames.CommandPattern.ICommand, 23
SadSapphicGames.CommandPattern.NullCommand, 26
ExecuteAsync
SadSapphicGames.CommandPattern.AsyncCommand, 10
SadSapphicGames.CommandPattern.IAsyncCommand, ExecuteCode
SadSapphicGames.CommandPattern, 8
ExecuteFullQueue
SadSapphicGames.CommandPattern.CommandStream, 15
Failure
SadSapphicGames.CommandPattern, 8
ForceQueueUndoCommand
SadSapphicGames.CommandPattern.CommandStream, 16
SadSapphicGames.CommandPattern.SingletonCommandManager, 32
GetCommandHistory
SadSapphicGames.CommandPattern.CommandStream, 16
SadSapphicGames.CommandPattern.SingletonCommandManager, 32
GetCommandQueue
SadSapphicGames.CommandPattern.CommandStream, 16
GetFaultedCommandTasks
SadSapphicGames.CommandPattern.CommandStream, 16
GetRunningCommandTasks
SadSapphicGames.CommandPattern.CommandStream, 17
SadSapphicGames.CommandPattern.SingletonCommandManager, 32
GetUndoCommand
SadSapphicGames.CommandPattern.IUndoable, 25
SadSapphicGames.CommandPattern.NullCommand, 27
SadSapphicGames.CommandPattern.NullCompositeCommand, 28
SadSapphicGames.CommandPattern.NullCompositeCommand, 28
SadSapphicGames.CommandPattern.NullCompositeCommand, 28
QueueCommand
SadSapphicGames.CommandPattern.CommandStream, 17
SadSapphicGames.CommandPattern.SingletonCommandManager, 32
QueueCommands

- SadSapphicGames.CommandPattern.CommandStream, GetUndoCommand, 27
- 17
- SadSapphicGames.CommandPattern.SingletonCommandManager, 33
- QueueEmpty
 - SadSapphicGames.CommandPattern, 8
- SadSapphicGames, 7
- SadSapphicGames.CommandPattern, 7
 - AwaitingCompletion, 8
 - CompositeFailure, 8
 - ExecuteCode, 8
 - Failure, 8
 - QueueEmpty, 8
 - Success, 8
- SadSapphicGames.CommandPattern.AsyncCommand, 9
 - CancellationTokenSource, 11
 - CommandTask, 11
 - Execute, 10
 - ExecuteAsync, 10
- SadSapphicGames.CommandPattern.Command, 12
 - Execute, 12
- SadSapphicGames.CommandPattern.CommandStream, 12
 - CancelRunningCommandTask, 14
 - CommandStream, 14
 - DropHistory, 15
 - DropQueue, 15
 - ExecuteFullQueue, 15
 - ForceQueueUndoCommand, 16
 - GetCommandHistory, 16
 - GetCommandQueue, 16
 - GetFaultedCommandTasks, 16
 - GetRunningCommandTasks, 17
 - QueueCommand, 17
 - QueueCommands, 17
 - TryExecuteNext, 18
 - TryQueueUndoCommand, 18
- SadSapphicGames.CommandPattern.CompositeCommand, 19
 - AddChild, 20
 - Execute, 20
- SadSapphicGames.CommandPattern.IAsyncCommand, 21
 - CancellationTokenSource, 22
 - CommandTask, 22
 - ExecuteAsync, 22
- SadSapphicGames.CommandPattern.ICommand, 23
 - Execute, 23
- SadSapphicGames.CommandPattern.IFailable, 23
 - WouldFail, 24
- SadSapphicGames.CommandPattern.IrreversibleCompositeFailureException, 24
- SadSapphicGames.CommandPattern.IUndoable, 25
 - GetUndoCommand, 25
- SadSapphicGames.CommandPattern.NullCommand, 26
 - Execute, 26
- SadSapphicGames.CommandPattern.NullCompositeCommand, 27
 - GetUndoCommand, 28
 - NullCompositeCommand, 28
- SadSapphicGames.CommandPattern.ReversibleCompositeFailureException, 29
- SadSapphicGames.CommandPattern.SimpleComposite, 29
 - SimpleComposite, 30
- SadSapphicGames.CommandPattern.SingletonCommandManager, 30
 - DropCommandHistory, 32
 - ForceQueueUndoCommand, 32
 - GetCommandHistory, 32
 - GetRunningCommandTasks, 32
 - QueueCommand, 32
 - QueueCommands, 33
 - ToggleCommandExecution, 33
 - TryQueueUndoCommand, 33
- SimpleComposite
 - SadSapphicGames.CommandPattern.SimpleComposite, 30
- Success
 - SadSapphicGames.CommandPattern, 8
- ToggleCommandExecution
 - SadSapphicGames.CommandPattern.SingletonCommandManager, 33
- TryExecuteNext
 - SadSapphicGames.CommandPattern.CommandStream, 18
- TryQueueUndoCommand
 - SadSapphicGames.CommandPattern.CommandStream, 18
 - SadSapphicGames.CommandPattern.SingletonCommandManager, 33
- WouldFail
 - SadSapphicGames.CommandPattern.IFailable, 24