### UltimateReplay Namespace

#### Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ReplayAudio</strong></td>
<td>A replay component that is responsible for recording and replaying audio effects that are played during gameplay. Audio can only be replayed when playback is not being reversed.</td>
</tr>
<tr>
<td><strong>ReplayBehaviour</strong></td>
<td>This interface can be implemented by mono behaviour scripts in order to receive replay start and end events. It works in a similar way to the 'Start' or 'Update' method however you must explicitly implement the interface as opposed to using magic methods. This allows for slightly improved performance.</td>
</tr>
<tr>
<td><strong>ReplayControls</strong></td>
<td>Default replay controls used for demonstration and testing. Uses legacy GUI for UI rendering.</td>
</tr>
<tr>
<td><strong>ReplayIgnoreAttribute</strong></td>
<td>Attach this attribute to a class that derives from <strong>ReplayBehaviour</strong> and the replay system will ignore it. This is</td>
</tr>
</tbody>
</table>
useful when you want to receive replay events but don't need to record any data.

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ReplayManager</strong></td>
<td>The main interface for Ultimate Replay and allows full control over object recording and playback.</td>
</tr>
<tr>
<td><strong>ReplayObject</strong></td>
<td>Only one instance of ReplayObject can be added to any game object.</td>
</tr>
<tr>
<td><strong>ReplayState</strong></td>
<td>A ReplayState allows replay objects to serialize and deserialize their data. See IReplaySerialize.</td>
</tr>
<tr>
<td><strong>ReplayTime</strong></td>
<td>This class emulates the behaviour of the Time class in Unity and can be used to modify the playback speed of a replay. There are also delta values that can be used to interpolate between frames where a low record frame rate is used. See ReplayTransform for an example.</td>
</tr>
<tr>
<td><strong>ReplayTransform</strong></td>
<td>Attach this component to a game objects in order to record the objects transform for replays. Only one instance of ReplayTransform can be added to any game object.</td>
</tr>
<tr>
<td><strong>ReplayVarAttribute</strong></td>
<td>Use this attribute on a field to mark it for recording. The type</td>
</tr>
</tbody>
</table>
the field is defined in must inhe\textit{it} from \texttt{ReplayBehaviour} in order for the field to be recorded automatically. Interpolation between field values is also possible where low record rates are used.

\section*{Enumerations}

\begin{table}[h]
\centering
\begin{tabular}{|c|p{10cm}|}
\hline
\textbf{Enumeration} & \textbf{Description} \\
\hline
\texttt{PlaybackDirection} & The playback direction used during replay playback. \\
\hline
\texttt{PlaybackOrigin} & Represents a playback node that can be used to calculate playback offsets. \\
\hline
\texttt{PlaybackState} & The state of the active \texttt{ReplayManager}. \\
\hline
\end{tabular}
\end{table}
PlaybackDirection Enumeration

The playback direction used during replay playback.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

▶ Syntax

```c#
public enum PlaybackDirection
```

▶ Members

<table>
<thead>
<tr>
<th>Member name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forward</td>
<td>0</td>
<td>The replay should be played back in normal mode.</td>
</tr>
<tr>
<td>Backward</td>
<td>1</td>
<td>The replay should be played back in reverse mode.</td>
</tr>
</tbody>
</table>

▶ See Also

Reference  
UltimateReplay Namespace
PlaybackOrigin Enumeration

Represents a playback node that can be used to calculate playback offsets.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

## Syntax

```
public enum PlaybackOrigin
```

## Members

<table>
<thead>
<tr>
<th>Member name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>0</td>
<td>The start of the playback sequence.</td>
</tr>
<tr>
<td>Current</td>
<td>1</td>
<td>The current frame in the playback sequence.</td>
</tr>
<tr>
<td>End</td>
<td>2</td>
<td>The end of the playback sequence.</td>
</tr>
</tbody>
</table>

## See Also

Reference  
UltimateReplay Namespace
PlaybackState Enumeration

The state of the active ReplayManager.

Namespace: UltimateReplay
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

Syntax

```c#
public enum PlaybackState
```

Members

<table>
<thead>
<tr>
<th>Member name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idle</td>
<td>0</td>
<td>The manager is doing nothing.</td>
</tr>
<tr>
<td>Recording</td>
<td>1</td>
<td>The manager is currently recording the scene using the current record settings.</td>
</tr>
<tr>
<td>Recording_Paused</td>
<td>2</td>
<td>The manager is currently paused but is expecting to resume recording.</td>
</tr>
<tr>
<td>Playback</td>
<td>3</td>
<td>The manager is performing playback using the current settings.</td>
</tr>
</tbody>
</table>
The manager is currently paused but is expecting to resume playback.

See Also

Reference
UltimateReplay Namespace
ReplayAudio Class

A replay component that is responsible for recording and replaying audio effects that are played during gameplay. Audio can only be replayed when playback is not being reversed.

Inheritance Hierarchy

```
System
  Object
  Component
    Behaviour
      MonoBehaviour
        UltimateReplay
          ReplayBehaviour
          ReplayAudio
```

Namespace: UltimateReplay
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

Syntax

```
public class ReplayAudio : MonoBehaviour
```

The `ReplayAudio` type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>ReplayAudio</em></td>
<td>Initializes a new instance of the <code>ReplayAudio</code> class</td>
</tr>
</tbody>
</table>
## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Awake</td>
<td>Called by Unity. (Overrids <code>ReplayBehaviourAwake</code>.)</td>
</tr>
<tr>
<td>OnReplayEvent</td>
<td>Called by the replay system when a replay event has occurred. (Overrids <code>ReplayBehaviourOnReplayEvent(ReplayEvent)</code>.)</td>
</tr>
<tr>
<td>Play</td>
<td>You should call this method as a replacement for <code>AudioSource.Play</code> as it will also record the time of the audio event so that it can be replayed later.</td>
</tr>
</tbody>
</table>

## Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>observedAudioSource</td>
<td>The audio source that will be recorded by this <code>ReplayAudio</code>.</td>
</tr>
</tbody>
</table>

## See Also

Reference
UltimateReplay Namespace
ReplayAudio Constructor

Initializes a new instance of the ReplayAudio class

Namespace: UltimateReplay
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

▲ Syntax

```c#
public ReplayAudio()
```

▲ See Also

Reference
ReplayAudio Class
UltimateReplay Namespace
ReplayAudio Fields

The `ReplayAudio` type exposes the following members.

### Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>observedAudioSource</td>
<td>The audio source that will be recorded by this <code>ReplayAudio</code>.</td>
</tr>
</tbody>
</table>

### See Also

Reference
- `ReplayAudio Class`
- `UltimateReplay Namespace`
ReplayAudio.observedAudioSource Field

The audio source that will be recorded by this ReplayAudio.

**Namespace:** UltimateReplay

**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

## Syntax

```c#
public AudioSource observedAudioSource
```

**Field Value**

Type: **AudioSource**

## See Also

Reference

- **ReplayAudio Class**
- **UltimateReplay Namespace**
ReplayAudio Methods

The `ReplayAudio` type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Awake</td>
<td>Called by Unity. (Overrides <code>ReplayBehaviourAwake</code>.)</td>
</tr>
<tr>
<td>OnReplayEvent</td>
<td>Called by the replay system when a replay event has occurred. (Overrides <code>ReplayBehaviourOnReplayEvent</code>.)</td>
</tr>
<tr>
<td>Play</td>
<td>You should call this method as a replacement for <code>AudioSource.Play</code> as it will also record the time of the audio event so that it can be replayed later.</td>
</tr>
</tbody>
</table>

See Also

Reference
- `ReplayAudio Class`
- `UltimateReplay Namespace`
ReplayAudioAwake Method

Called by Unity.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```c#
public override void Awake()
```

### See Also

**Reference**  
ReplayAudio Class  
UltimateReplay Namespace
ReplayAudioOnReplayEvent Method

Called by the replay system when a replay event has occurred.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public override void OnReplayEvent(  
    ReplayEvent replayEvent
)
```

### Parameters

`replayEvent`  
Type: UltimateReplay.Core.ReplayEvent  
The `ReplayEvent` that was triggered

### See Also

Reference  
ReplayAudio Class  
UltimateReplay Namespace
ReplayAudioPlay Method

You should call this method as a replacement for AudioSource.Play as it will also record the time of the audio event so that it can be replayed later.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public void Play()
```

### See Also

Reference  
ReplayAudio Class  
UltimateReplay Namespace
ReplayBehaviour Class

This interface can be implemented by mono behaviour scripts in order to receive replay start and end events. It works in a similar way to the 'Start' or 'Update' method however you must explicitly implement the interface as opposed to using magic methods. This allows for slightly improved performance.

Inheritance Hierarchy

- System
  - Object
    - Component
      - Behaviour
        - MonoBehaviour
          - UltimateReplay.ReplayBehaviour
          - UltimateReplay.ReplayAudio
          - UltimateReplay.ReplayTransform
          - UltimateReplay.Storage.ReplayTarget

Namespace: UltimateReplay
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

Syntax

```csharp
public abstract class ReplayBehaviour : MonoBehaviour, IReplaySerialize
```

The `ReplayBehaviour` type exposes the following members.

Constructors
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReplayBehaviour</td>
<td>Initializes a new instance of the ReplayBehaviour class</td>
</tr>
</tbody>
</table>

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Awake</td>
<td>Called by Unity. Allows the ReplayBehaviour to validate its identity and register itself with the replay system.</td>
</tr>
<tr>
<td>OnReplayDeserialize</td>
<td>Called by the replay system when all replay state information should be deserialized.</td>
</tr>
<tr>
<td>OnReplayEnd</td>
<td>Called by the replay system when playback has ended. You can re-enable game behaviour in this method to allow the gameplay to 'take over'.</td>
</tr>
<tr>
<td>OnReplayEvent</td>
<td>Called by the replay system when an event has been received during playback.</td>
</tr>
<tr>
<td>OnReplayPlayPause</td>
<td>Called by the replay system when playback is about to be paused or resumed.</td>
</tr>
<tr>
<td>OnReplaySerialize</td>
<td>Called by the replay system when all replay state information should be serialized.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>OnReplayStart</strong></td>
<td>Called by the replay system when playback is about to start. You can disable gameplay behaviour that should not run during playback in this method, such as player movement.</td>
</tr>
<tr>
<td><strong>OnReplayUpdate</strong></td>
<td>Called by the replay system every frame while playback is active.</td>
</tr>
<tr>
<td><strong>ReplayDeserializeEvents</strong></td>
<td>Deserializes all active ReplayEvent from the state and dispatches any events to the OnReplayEvent(ReplayEvent) handler.</td>
</tr>
<tr>
<td><strong>ReplayDeserializeVariables</strong></td>
<td>Deserializes all active ReplayVariable from the state.</td>
</tr>
<tr>
<td><strong>ReplayFindVariables</strong></td>
<td>Attempts to find any variables marked with the ReplayVarAttribute so they can be serialized later.</td>
</tr>
<tr>
<td><strong>ReplayInterpolateVariables</strong></td>
<td>Allows all active ReplayVariable to be interpolated between replay frames.</td>
</tr>
<tr>
<td><strong>ReplayRecordEvent(Byte, ReplayState)</strong></td>
<td>Push an event to the replay system for recording.</td>
</tr>
</tbody>
</table>
### Methods

- **ReplayRecordEvent(ReplayEvents, ReplayState)**
  Push an event to the replay system for recording.

- **ReplaySerializeEvents**
  Serializes all awaiting `ReplayEvent` to the state.

- **ReplaySerializeVariables**
  Serializes all active `ReplayVariable` to the state.

- **Reset**
  Called by Unity while in editor mode. Allows the unique id to be generated when the script is attached to an object.

---

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Identity</strong></td>
<td>Get the <code>ReplayIdentity</code> associated with this <code>ReplayBehaviour</code>.</td>
</tr>
<tr>
<td><strong>IsRecording</strong></td>
<td>Returns true if the active replay manager is currently recording the scene. Note: If recording is paused this value will still be true.</td>
</tr>
<tr>
<td><strong>IsReplaying</strong></td>
<td>Returns true if the active replay manager is currently replaying a previous recording. Note: If playback is paused this value will still be true.</td>
</tr>
<tr>
<td><strong>PlaybackDirection</strong></td>
<td>Gets the current <code>PlaybackDirection</code> of replay playback.</td>
</tr>
</tbody>
</table>
See Also

Reference
UltimateReplay Namespace
ReplayBehaviour Constructor

Initializes a new instance of the ReplayBehaviour class

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll)  
**Version:** 1.0.0.0

### Syntax

```csharp
protected ReplayBehaviour()
```

### See Also

Reference  
ReplayBehaviour Class  
UltimateReplay Namespace
ReplayBehaviour Methods

The `ReplayBehaviour` type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Awake</strong></td>
<td>Called by Unity. Allows the <code>ReplayBehaviour</code> to validate its identity and register itself with the replay system.</td>
</tr>
<tr>
<td><strong>OnReplayDeserialize</strong></td>
<td>Called by the replay system when all replay state information should be deserialized.</td>
</tr>
<tr>
<td><strong>OnReplayEnd</strong></td>
<td>Called by the replay system when playback has ended. You can re-enable game behaviour in this method to allow the gameplay to 'take over'.</td>
</tr>
<tr>
<td><strong>OnReplayEvent</strong></td>
<td>Called by the replay system when an event has been received during playback.</td>
</tr>
<tr>
<td><strong>OnReplayPlayPause</strong></td>
<td>Called by the replay system when playback is about to be paused or resumed.</td>
</tr>
<tr>
<td><strong>OnReplaySerialize</strong></td>
<td>Called by the replay system when all replay state information should be serialized.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>OnReplayStart</strong></td>
<td>Called by the replay system when playback is about to start. You can disable game behaviour that should not run during playback in this method, such as player movement.</td>
</tr>
<tr>
<td><strong>OnReplayUpdate</strong></td>
<td>Called by the replay system every frame while playback is active.</td>
</tr>
<tr>
<td><strong>ReplayDeserializeEvents</strong></td>
<td>Deserializes all active <code>ReplayEvent</code> from the state and dispatches any event to the <code>OnReplayEvent(ReplayEvent)</code> handler.</td>
</tr>
<tr>
<td><strong>ReplayDeserializeVariables</strong></td>
<td>Deserializes all active <code>ReplayVariable</code> from the state.</td>
</tr>
<tr>
<td><strong>ReplayFindVariables</strong></td>
<td>Attempts to find any variables marked with the <code>ReplayVarAttribute</code> so that they can be serialized later.</td>
</tr>
<tr>
<td><strong>ReplayInterpolateVariables</strong></td>
<td>Allows all active <code>ReplayVariable</code> to be interpolated between replay frames.</td>
</tr>
<tr>
<td><strong>ReplayRecordEvent(Byte, ReplayState)</strong></td>
<td>Push an event to the replay system for recording.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ReplayRecordEvent(ReplayEvents, ReplayState)</td>
<td>Push an event to the replay system for recording.</td>
</tr>
<tr>
<td>ReplaySerializeEvents</td>
<td>Serializes all awaiting ReplayEvent to the state.</td>
</tr>
<tr>
<td>ReplaySerializeVariables</td>
<td>Serializes all active ReplayVariable to the state.</td>
</tr>
<tr>
<td>Reset</td>
<td>Called by Unity while in editor mode. Allows the unique id to be generated when the script is attached to an object.</td>
</tr>
</tbody>
</table>

See Also

Reference
ReplayBehaviour Class
UltimateReplay Namespace
ReplayBehaviourAwake Method

Called by Unity. Allows the ReplayBehaviour to validate its identity and register its self with the replay system.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

▶ **Syntax**

```csharp
public virtual void Awake()
```

▶ **See Also**

Reference  
ReplayBehaviour Class  
UltimateReplay Namespace
ReplayBehaviourOnReplayDeserialize Method

Called by the replay system when all replay state information should be deserialized.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```csharp
public virtual void OnReplayDeserialize(
    ReplayState state
)
```

**Parameters**

- `state`  
  Type: UltimateReplayReplayState  
  The ReplayState to read the data from

**Implements**

IReplaySerializeOnReplayDeserialize(ReplayState)

**See Also**

**Reference**

ReplayBehaviour Class  
UltimateReplay Namespace
ReplayBehaviour.OnReplayEnd Method

Called by the replay system when playback has ended. You can re-enable game behaviour in this method to allow the gameplay to 'take over'.

Namespace: UltimateReplay
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

Syntax

C#%

```csharp
public virtual void OnReplayEnd()
```

See Also

Reference
ReplayBehaviour Class
UltimateReplay Namespace
ReplayBehaviourOnReplayEvent Method

Called by the replay system when an event has been received during playback.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public virtual void OnReplayEvent(
    ReplayEvent replayEvent
)
```

### Parameters

*replayEvent*
- Type: UltimateReplay.Core.ReplayEvent
  - The event that was received

### See Also

Reference
- ReplayBehaviour Class
- UltimateReplay Namespace
ReplayBehaviourOnReplayPlayPause Method

Called by the replay system when playback is about to be paused or resumed.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```csharp
public virtual void OnReplayPlayPause(
    bool paused
)
```

**Parameters**

*paused*  
Type: SystemBoolean  
True if playback is about to be paused or false if playback is about to be resumed

**See Also**

Reference  
ReplayBehaviour Class  
UltimateReplay Namespace
ReplayBehaviourOnReplaySerialize Method

Called by the replay system when all replay state information should be serialized.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

▲ Syntax

```
public virtual void OnReplaySerialize(  
    ReplayState state
)
```

**Parameters**

`state`  
Type: UltimateReplay.ReplayState  
The ReplayState to write the data to

**Implements**  
IReplaySerializeOnReplaySerialize(ReplayState)

▲ See Also

**Reference**  
ReplayBehaviour Class  
UltimateReplay Namespace
ReplayBehaviourOnReplayStart Method

Called by the replay system when playback is about to start. You can disable game behaviour that should not run during playback in this method, such as player movement.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```c#
public virtual void OnReplayStart()
```

**See Also**

Reference  
ReplayBehaviour Class  
UltimateReplay Namespace
ReplayBehaviour

OnReplayUpdate

Method

Called by the replay system every frame while playback is active.

Namespace: UltimateReplay
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

Syntax

```c#
public virtual void OnReplayUpdate()
```

See Also

Reference

ReplayBehaviour Class
UltimateReplay Namespace
ReplayBehaviour.ReplayDeserializeEvents Method

Deserializes all active ReplayEvent from the state and dispatches any events to the OnReplayEvent(ReplayEvent) handler.

Namespace: UltimateReplay
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

Syntax

```csharp
protected virtual void ReplayDeserializeEvents(
    ReplayState state
)
```

Parameters

- `state`  
  Type: UltimateReplay.ReplayState  
  The ReplayState to read the variable data from

See Also

- Reference
  - ReplayBehaviour Class
  - UltimateReplay Namespace
ReplayBehaviour.ReplayDeserializeVariables Method

Deserializes all active ReplayVariable from the state.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
protected virtual void ReplayDeserializeVariables(ReplayState state)
```

### Parameters

- **state**  
  Type: UltimateReplay.ReplayState  
  The ReplayState to read the variable data from

### See Also

Reference
- ReplayBehaviour Class
- UltimateReplay Namespace
ReplayBehaviour.ReplayFindVariables Method

Attempts to find any variables marked with the ReplayVarAttribute so that they can be serialized later.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
protected virtual void ReplayFindVariables()
```

### See Also

Reference  
ReplayBehaviour Class  
UltimateReplay Namespace
ReplayBehaviour.ReplayInterpolateVariables Method

Allows all active ReplayVariable to be interpolated between replay frames.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
protected virtual void ReplayInterpolateVariables(  
    float delta
)
```

### Parameters

*delta*

Type: System.Single  
The 't' value between frames used for interpolation

### See Also

Reference  
ReplayBehaviour Class  
UltimateReplay Namespace
## ReplayBehaviour.ReplayRecordEvent Method

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ReplayRecordEvent(Byte, ReplayState)</code></td>
<td>Push an event to the replay system for recording.</td>
</tr>
<tr>
<td><code>ReplayRecordEvent(ReplayEvents, ReplayState)</code></td>
<td>Push an event to the replay system for recording.</td>
</tr>
</tbody>
</table>

### See Also

Reference
- [ReplayBehaviour Class](#)
- [UltimateReplay Namespace](#)
ReplayBehaviour.ReplayRecordEvent Method (Byte, ReplayState)

Push an event to the replay system for recording.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

▲ Syntax

```csharp
public void ReplayRecordEvent(
    byte eventId,
    ReplayState state = null
)
```

**Parameters**

- **eventId**
  - Type: System.Byte
  - The event id to uniquely identify the event type

- **state (Optional)**
  - Type: UltimateReplay.ReplayState
  - The state data for the event or null if no state data is required

▲ See Also

**Reference**
- ReplayBehaviour Class
- ReplayRecordEvent Overload
- UltimateReplay Namespace
**ReplayBehaviour**

**ReplayRecordEvent Method (ReplayEvents, ReplayState)**

Push an event to the replay system for recording.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```
public void ReplayRecordEvent(
    ReplayEvents eventID,
    ReplayState state = null
)
```

### Parameters

- **eventID**  
  Type: UltimateReplay.Core.ReplayEvents
  The event id to uniquely identify the event type

- **state (Optional)**  
  Type: UltimateReplay.ReplayState
  The state data for the event or null if no state data is required

### See Also

**Reference**
- ReplayBehaviour Class  
- ReplayRecordEvent Overload  
- UltimateReplay Namespace
ReplayBehaviour.ReplaySerializeEvents Method

Serializes all awaiting ReplayEvent to the state.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```csharp
protected virtual void ReplaySerializeEvents(
    ReplayState state
)
```

**Parameters**

*state*

Type: UltimateReplay.ReplayState  
The ReplayState to write the variable data to

**See Also**

Reference  
ReplayBehaviour Class  
UltimateReplay Namespace
ReplayBehaviour.ReplaySerializeVariables Method

Serializable all active ReplayVariable to the state.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```csharp
protected virtual void ReplaySerializeVariables(ReplayState state)
```

**Parameters**

*state*
- Type: UltimateReplay.ReplayState  
  The ReplayState to write the variable data to

**See Also**

Reference  
ReplayBehaviour Class  
UltimateReplay Namespace
ReplayBehaviourReset Method

Called by Unity while in editor mode. Allows the unique id to be generated when the script is attached to an object.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

⚠️ Syntax

```csharp
public virtual void Reset()
```

⚠️ See Also

Reference  
ReplayBehaviour Class  
UltimateReplay Namespace
ReplayBehaviour Properties

The ReplayBehaviour type exposes the following members.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identity</td>
<td>Get the ReplayIdentity associated with this ReplayBehaviour.</td>
</tr>
<tr>
<td>IsRecording</td>
<td>Returns true if the active replay manager is currently recording the scene. Note: If recording is paused this value will still be true.</td>
</tr>
<tr>
<td>IsReplaying</td>
<td>Returns true if the active replay manager is currently replaying a previous recording. Note: If playback is paused this value will still be true.</td>
</tr>
<tr>
<td>PlaybackDirection</td>
<td>Gets the current PlaybackDirection of replay playback.</td>
</tr>
</tbody>
</table>

See Also

Reference
ReplayBehaviour Class
UltimateReplay Namespace
ReplayBehaviourIdentity Property

Get the ReplayIdentity associated with this ReplayBehaviour.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```csharp
public ReplayIdentity Identity { get; }
```

**Property Value**

Type: ReplayIdentity

**See Also**

Reference  
ReplayBehaviour Class  
UltimateReplay Namespace
ReplayBehaviour.IsRecording Property

Returns true if the active replay manager is currently recording the scene. Note: If recording is paused this value will still be true.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```csharp
public bool IsRecording { get; }
```

**Property Value**

Type: Boolean

**See Also**

Reference  
ReplayBehaviour Class  
UltimateReplay Namespace
ReplayBehaviour.IsReplaying Property

Returns true if the active replay manager is currently replaying a previous recording. Note: If playback is paused this value will still be true.

**Namespace:** UltimateReplay
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```csharp
public bool IsReplaying { get; }
```

**Property Value**

- **Type:** Boolean

**See Also**

- Reference
  - ReplayBehaviour Class
  - UltimateReplay Namespace
ReplayBehaviourPlaybackDirection Property

Gets the current PlaybackDirection of replay playback.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```csharp
public PlaybackDirection PlaybackDirection { get; }
```

**Property Value**

Type: PlaybackDirection

**See Also**

Reference
- ReplayBehaviour Class
- UltimateReplay Namespace
ReplayControls Class

Default replay controls used for demonstration and testing. Uses legacy GUI for UI rendering.

Inheritance Hierarchy

```
System
  Object
  Component
    Behaviour
      MonoBehaviour
        UltimateReplay
        ReplayControls
```

Namespace: UltimateReplay
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

Syntax

```c#
public class ReplayControls : MonoBehaviour
```

The `ReplayControls` type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>!(ReplayControls)</td>
<td>Initializes a new instance of the ReplayControls class</td>
</tr>
</tbody>
</table>

Top
## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Awake</td>
<td>Called by Unity.</td>
</tr>
<tr>
<td>OnGUI</td>
<td>Called by Unity.</td>
</tr>
<tr>
<td>Start</td>
<td>Called by Unity.</td>
</tr>
<tr>
<td>Update</td>
<td>Called by Unity.</td>
</tr>
</tbody>
</table>

## Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>allowPlaybackFreeCam</td>
<td>Should the free cam mode be enabled during playback.</td>
</tr>
<tr>
<td>flySpeed</td>
<td>How fast the free cam can move around the scene.</td>
</tr>
<tr>
<td>lookSpeed</td>
<td>How fast the free cam can look around the scene.</td>
</tr>
</tbody>
</table>

## See Also

Reference
UltimateReplay Namespace
ReplayControls Constructor

Initializes a new instance of the ReplayControls class

Namespace: UltimateReplay
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

> Syntax

```csharp
public ReplayControls()
```

> See Also

Reference
ReplayControls Class
UltimateReplay Namespace
ReplayControls Fields

The `ReplayControls` type exposes the following members.

### Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>allowPlaybackFreeCam</code></td>
<td>Should the free cam mode be enabled during playback.</td>
</tr>
<tr>
<td><code>flySpeed</code></td>
<td>How fast the free cam can move around the scene.</td>
</tr>
<tr>
<td><code>lookSpeed</code></td>
<td>How fast the free cam can look around the scene.</td>
</tr>
</tbody>
</table>

### See Also

Reference

*ReplayControls Class*

*UltimateReplay Namespace*
ReplayControls.allowPlaybackFreeCam Field

Should the free cam mode be enabled during playback.

Namespace: UltimateReplay
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

Syntax

```csharp
public bool allowPlaybackFreeCam
```

Field Value
Type: Boolean

See Also

Reference
ReplayControls Class
UltimateReplay Namespace
ReplayControls.flySpeed Field

How fast the free cam can move around the scene.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```csharp
public float flySpeed
```

**Field Value**  
**Type:** Single

**See Also**

Reference  
ReplayControls Class  
UltimateReplay Namespace
ReplayControlsLookSpeed Field

How fast the free cam can look around the scene.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public float lookSpeed
```

### Field Value

**Type:** Single

### See Also

**Reference**  
ReplayControls Class  
UltimateReplay Namespace
ReplayControls Methods

The ReplayControls type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Awake</td>
<td>Called by Unity.</td>
</tr>
<tr>
<td>OnGUI</td>
<td>Called by unity.</td>
</tr>
<tr>
<td>Start</td>
<td>Called by Unity.</td>
</tr>
<tr>
<td>Update</td>
<td>Called by Unity.</td>
</tr>
</tbody>
</table>

See Also

Reference
ReplayControls Class
UltimateReplay Namespace
ReplayControlsAwake Method

Called by Unity.

Namespace: UltimateReplay
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

Syntax

```csharp
public void Awake()
```

See Also

Reference
ReplayControls Class
UltimateReplay Namespace
ReplayControlsOnGUI Method

Called by unity.

**Namespace:** UltimateReplay

**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```c#
public void OnGUI()
```

### See Also

Reference

ReplayControls Class
UltimateReplay Namespace
ReplayControlsStart Method

Called by Unity.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```c#
public void Start()
```

### See Also

Reference  
ReplayControls Class  
UltimateReplay Namespace
ReplayControlsUpdate Method

Called by Unity.

Namespace: UltimateReplay
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

Syntax

```c#
public void Update()
```

See Also

Reference
ReplayControls Class
UltimateReplay Namespace
ReplayIgnoreAttribute Class

Attach this attribute to a class that derives from ReplayBehaviour and the replay system will ignore it. This is useful when you want to receive replay events but don't need to record any data.

Inheritance Hierarchy

System
  SystemObject
  SystemAttribute
    UltimateReplay
      ReplayIgnoreAttribute

Namespace: UltimateReplay
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

Syntax

C#  Copy

```csharp
public sealed class ReplayIgnoreAttribute : Attribute
```

The ReplayIgnoreAttribute type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReplayIgnoreAttribute</td>
<td>Initializes a new instance of the ReplayIgnoreAttribute class</td>
</tr>
</tbody>
</table>

See Also
Reference

UltimateReplay Namespace
ReplayIgnoreAttribute Constructor

Initializes a new instance of the `ReplayIgnoreAttribute` class

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```
public ReplayIgnoreAttribute()
```

### See Also

Reference
- `ReplayIgnoreAttribute Class`
- `UltimateReplay Namespace`
ReplayManager Class

The main interface for Ultimate Replay and allows full control over object recording and playback.

Inheritance Hierarchy

- System
  - Object
  - Component
  - Behaviour
    - MonoBehaviour
      - UltimateReplay.UtilMonoSingleton
        - ReplayManager

Namespace: UltimateReplay
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0

Syntax

```csharp
public sealed class ReplayManager : MonoSingleton<
```

The `ReplayManager` type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>🌿 ReplayManager</td>
<td>Initializes a new instance of the ReplayManager class</td>
</tr>
</tbody>
</table>
## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BeginPlayback</td>
<td>Use this method to begin the playback of the recorded object. Before calling this method, use <code>SetPlaybackFrame(Single, PlaybackOrigin)</code> or <code>SetPlaybackFrameNormalize(Single, PlaybackOrigin)</code> to specify the exact location at which playback should begin. The method will run the entire playback gathered and then automatically stop playback on completion if <code>endReplayAfterPlayback</code> is true.</td>
</tr>
<tr>
<td>BeginPlaybackFrame</td>
<td>Use this method to set the current playback at a specific replay frame. This will allow the state of a specific replay frame to be restored but will not continue playback which will provide a freeze frame effect. Before calling this method, use <code>SetPlaybackFrame(Single, PlaybackOrigin)</code> or <code>SetPlaybackFrameNormalize(Single, PlaybackOrigin)</code> to specify the exact location at which the playback frame should be sampled. Use <code>StopPlayback</code> to unfreeze the still frame and return to normal game mode. This method will ignore the value of <code>endReplayAfterPlayback</code> as only a single frame is replayed. As a result, you will need to call <code>StopPlayback</code> to unfreeze the still frame.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>BeginRecording</td>
<td>Use this method to begin sampling the recorded objects in the scene. If recordOnStart is true then this method will be called automatically when the manager is initialized. State information will be recorded via the assigned ReplayTarget (Default: ReplayMemoryTarget). When true, any previous recording data will be discarded.</td>
</tr>
<tr>
<td>DiscardRecording</td>
<td>This method will throw away all recorded data and flush the replay target if necessary. This method can be called at any time. If the manager is currently recording then all data will be discarded and recording will continue. If the manager is currently replaying then all replay data will be discarded and playback will stop.</td>
</tr>
<tr>
<td>FindReplayPrefab</td>
<td>Attempts to find the prefab with the matching name. This is used to restore objects that were destroyed during recording.</td>
</tr>
<tr>
<td>ForceAwake</td>
<td>Override implementation of ForceAwake. Performs exactly the same behaviour. Simply included so that the user does not need to import 'UltimateReplay.Util' to access this method.</td>
</tr>
</tbody>
</table>
| OnDestroy              | Called by Unity. Allows the active...
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>replay manager to cleanup any active recordings.</td>
<td></td>
</tr>
<tr>
<td><strong>OnLevelWasLoaded</strong></td>
<td>Called by Unity. Allows the active replay manager to cleanup recordings when a scene change is made.</td>
</tr>
<tr>
<td><strong>OnValidate</strong></td>
<td>Called by Unity.</td>
</tr>
<tr>
<td><strong>PausePlayback</strong></td>
<td>Use this method to pause replay playback while maintaining the current replay state. See <strong>ResumePlayback</strong> to continue playback.</td>
</tr>
<tr>
<td><strong>PauseRecording</strong></td>
<td>Use this method when you want to pause recording but may continue recording at any point. A good candidate for pausing recording when the user pauses the game and is shown a pause menu. The manager must already be recording; otherwise, this method will have no effect.</td>
</tr>
<tr>
<td><strong>RegisterReplayPrefab</strong></td>
<td>Attempts to register a game object as a prefab so that the replay system is able to spawn or despawn the object as needed. You only need to do this for objects that are likely to be instantiated or destroyed during recording. The replay system must be able to accurately restore the scene state during playback. The specified object must be a prefab; otherwise, an error will be thrown and the object will not be registered. Prefab instances are not accepted.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>ReplayDestroy</strong></td>
<td>Attempts to destroy the specified prefab. <em>OnReplayDestroy</em> will be called if a listener has been registered; otherwise default destruction will be used.</td>
</tr>
<tr>
<td><strong>ReplayInstantiate</strong></td>
<td>Attempts to instantiate the specified prefab. <em>OnReplayInstantiate</em> will be called if a listener has been registered; otherwise default instantiation will be used.</td>
</tr>
<tr>
<td><strong>ResumePlayback</strong></td>
<td>Use this method to resume playback after a previous call to <em>PausePlayback</em> was called. If <em>PausePlayback</em> was not called prior to this method, then the method will have no effect.</td>
</tr>
<tr>
<td><strong>ResumeRecording</strong></td>
<td>Use this method to resume recording after a previous call to <em>PauseRecording</em>. The manager must already be recording; otherwise this method will have no effect.</td>
</tr>
<tr>
<td><strong>SetPlaybackFrame</strong></td>
<td>Use this method to specify where in the replay sequence the playback should start. If the offset does not lie within the bounds of the replay, the value will be clamped to represent either the start or end frame.</td>
</tr>
<tr>
<td><strong>SetPlaybackFrameNormalized</strong></td>
<td>Use this method to specify where in the replay sequence the playback should start. This method accepts normalized offsets values between 0 and 1 and performs validation.</td>
</tr>
</tbody>
</table>
using the value.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>Called by Unity. Allows the active replay manager to initialize.</td>
</tr>
<tr>
<td>StopPlayback</td>
<td>Use this method to stop any active playback. This method will only have an effect if there is an active playback running otherwise it will have no effect.</td>
</tr>
<tr>
<td>StopRecording</td>
<td>Use this method to stop recording after a previous call to <code>BeginRecording(Boolean)</code>. The manager must already be recording otherwise this method will have no effect. This method must be called before attempting to playback otherwise you may get unpredictable results.</td>
</tr>
<tr>
<td>Update</td>
<td>Called by Unity. Allows the active replay manager to update recording or playback.</td>
</tr>
</tbody>
</table>

### Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>endReplayAfterPlayback</code></td>
<td>When true, the replay manager will automatically restore the previous game state after playback has finished. When false, playback will continue to loop forever until a suitable stop is reached.</td>
</tr>
</tbody>
</table>
or pause command is issued manually.

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnReplayDestroy</td>
<td>Called by the replay system whenever it needs to destroy a game object in order to restore a previous scene state. You can add a listener to override the default behaviour which can be useful if you want to handle the destruction manually for purposes such as object pooling.</td>
</tr>
<tr>
<td>OnReplayInstantiate</td>
<td>Called by the replay system whenever it needs to instantiate a prefab for use during playback. You can add a listener to override the default behaviour which can be useful if you want to handle the instantiation manually for purposes such as object pooling.</td>
</tr>
</tbody>
</table>

**prefabs**

A collection of prefabs that may be spawned or destroyed during recording and as a result may need to be spawned or destroyed in order to accurately recreate the replay.

**recordFPS**

The target record framerate of the sampler. The higher this value the higher the memory consumption and
cpu usage will be. You will need to fine tune this value to tradeoff performance or memory usage for replay accuracy.

<table>
<thead>
<tr>
<th>recordOnStart</th>
<th>When true, the manager will automatically begin recording the scene one it is initialized.</th>
</tr>
</thead>
</table>

**Top Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![ ] CurrentPlaybackTime</td>
<td>Get the current playback time in seconds. This value will never be greater than duration.</td>
</tr>
<tr>
<td>![ ] CurrentPlaybackTimeNormalized</td>
<td>Get the current playback time as a normalized value between 0-1. 0 represents the starting frame of the recording and 1 represents the very last frame of the recording.</td>
</tr>
<tr>
<td>![ ] IsPaused</td>
<td>Returns true when the active replay manager is in any paused state. Paused states could include</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td><strong>Playback_Paused</strong> or <strong>Recording_Paused</strong>.</td>
<td></td>
</tr>
<tr>
<td>![IsRecording]</td>
<td><strong>IsRecording</strong></td>
</tr>
<tr>
<td>![IsReplaying]</td>
<td><strong>IsReplaying</strong></td>
</tr>
<tr>
<td>![PlaybackDirection]</td>
<td><strong>PlaybackDirection</strong></td>
</tr>
<tr>
<td>![Preparer]</td>
<td><strong>Preparer</strong></td>
</tr>
<tr>
<td>![Replay]</td>
<td><strong>Replay</strong></td>
</tr>
</tbody>
</table>
This property may be null in a rare case where the active replay manager was destroyed in the same frame as an application quit event was issued. In this situation the replay manager cannot be recreated as it would cause leaked objects.

<table>
<thead>
<tr>
<th>Scene</th>
<th>Get the <code>ReplayScene</code> associated with the replay system.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td>The current replay target that is being used to store the replay data. By default, the replay target is <code>ReplayMemoryTarget</code>.</td>
</tr>
</tbody>
</table>
ReplayManager Constructor

Initializes a new instance of the ReplayManager class

Namespace: UltimateReplay
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

Syntax

```c#
public ReplayManager()
```

See Also

Reference

ReplayManager Class
UltimateReplay Namespace
ReplayManager Fields

The `ReplayManager` type exposes the following members.

## Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>endReplayAfterPlayback</strong></td>
<td>When true, the replay manager will automatically restore the previous game state after playback has finished. When false, playback will continue to loop forever until a suitable stop or pause command is issued manually.</td>
</tr>
<tr>
<td><strong>OnReplayDestroy</strong></td>
<td>Called by the replay system whenever it needs to destroy a game object in order to restore a previous scene state. You can add a listener to override the default behaviour which can be useful if you want to handle the destruction manually for purposes such as object pooling.</td>
</tr>
<tr>
<td><strong>OnReplayInstantiate</strong></td>
<td>Called by the replay system whenever it needs to instantiate a prefab for use during playback. You can add</td>
</tr>
</tbody>
</table>
a listener to override the default behaviour which can be useful if you want to handle the instantiation manually for purposes such as object pooling.

<table>
<thead>
<tr>
<th>prefabs</th>
<th>A collection of prefabs that may be spawned or destroyed during recording and as a result any need to be spawned or destroyed in order to accurately recreate the replay.</th>
</tr>
</thead>
<tbody>
<tr>
<td>recordFPS</td>
<td>The target record framerate of the sampler. The higher this value the higher the memory consumption and cpu usage will be. You will need to fine tune this value to tradeoff performance or memory usage for replay accuracy.</td>
</tr>
<tr>
<td>recordOnStart</td>
<td>When true, the manager will automatically begin recording the scene one it is initialized.</td>
</tr>
</tbody>
</table>
ReplayManager.endReplayAfterPlayback Field

When true, the replay manager will automatically restore the previous game state after playback has finished. When false, playback will continue to loop forever until a suitable stop or pause command is issued manually.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```c#
public bool endReplayAfterPlayback
```

### Field Value

Type: Boolean

### See Also

Reference  
ReplayManager Class  
UltimateReplay Namespace
ReplayManagerOnReplayDestroy Field

Called by the replay system whenever it needs to destroy a game object in order to restore a previous scene state. You can add a listener to override the default behaviour which can be useful if you want to handle the destruction manually for purposes such as object pooling.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```csharp
public static Action<GameObject> OnReplayDestroy
```

**Field Value**

Type: `Action<GameObject>`

**See Also**

Reference

ReplayManager Class  
UltimateReplay Namespace
ReplayManager_OnReplayInstantiate Field

Called by the replay system whenever it needs to instantiate a prefab for use during playback. You can add a listener to override the default behaviour which can be useful if you want to handle the instantiation manually for purposes such as object pooling.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public static Func<GameObject, Vector3, Quaternion, GameObject>
```

### Field Value

Type: `Func<GameObject, Vector3, Quaternion, GameObject>`

### See Also

Reference  
ReplayManager Class  
UltimateReplay Namespace
ReplayManager::prefabs Field

A collection of prefabs that may be spawned or destroyed during recording and as a result any need to be spawned or destroyed in order to accurately recreate the replay.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```c#
public GameObject[] prefabs
```

**Field Value**

Type: `GameObject`

### See Also

**Reference**
- ReplayManager Class
- UltimateReplay Namespace
ReplayManager.recordFPS Field

The target record framerate of the sampler. The higher this value the higher the memory consumption and cpu usage will be. You will need to fine tune this value to tradeoff performance or memory usage for replay accuracy.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```c#
public int recordFPS
```

### Field Value

Type: `Int32`

### See Also

Reference  
ReplayManager Class  
UltimateReplay Namespace
ReplayManager.recordOnStart Field

When true, the manager will automatically begin recording the scene one it is initialized.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public bool recordOnStart
```

### Field Value

Type: **Boolean**

### See Also

**Reference**  
ReplayManager Class  
UltimateReplay Namespace
ReplayManager Methods

The `ReplayManager` type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| ![s] BeginPlayback          | Use this method to begin the playback of the recorded objects. SetPlaybackFrame(Single, PlaybackOrigin) or SetPlaybackFrameNormalize(PlaybackOrigin) before calling this method to specify the exact location at which playback should begin.
| ![s] BeginPlaybackFrame    | Use this method to set the current playback at a specific replay frame. This will allow the state of a specific replay frame to be restored but will not continue playback which will provide a freeze frame effect. SetPlaybackFrame(Single, PlaybackOrigin) or SetPlaybackFrameNormalize(PlaybackOrigin) before calling this method to specify the exact location at which the playback frame should be sampled. Use StopPlayback to stop playback if necessary. |
unfreeze the still frame and return to normal game mode. This method will ignore the value of `endReplayAfterPlayback` as only a single frame is replayed. As a result, you will need to call `StopPlayback` when you want to end the playback frame.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BeginRecording</strong></td>
<td>Use this method to begin sampling the recorded objects in the scene. If <code>recordOnStart</code> is true then this method will be called automatically when the manager is initialized. State information will be recorded via the assigned <code>ReplayTarget</code> (Default <code>ReplayMemoryTarget</code>). When <code>true</code>, any previous recording data will be discarded.</td>
</tr>
<tr>
<td><strong>DiscardRecording</strong></td>
<td>This method will throw away all the recorded data and flush the replay target if necessary. This method can be called at any time. If the manager is currently recording then all previous data will be discarded and recording will continue. If the manager is currently replaying then all replay data will be discarded and playback will stop.</td>
</tr>
<tr>
<td><strong>FindReplayPrefab</strong></td>
<td>Attempts to find the prefab with the matching name. This is used to restore objects that were destroyed during recording.</td>
</tr>
<tr>
<td><strong>ForceAwake</strong></td>
<td>Override implementation of <code>ForceAwake</code>. Performs exactly the same actions as the inherited method.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>OnDestroy</td>
<td>Called by Unity. Allows the active replay manager to cleanup any active recordings.</td>
</tr>
<tr>
<td>OnLevelWasLoaded</td>
<td>Called by Unity. Allows the active replay manager to cleanup recordings when a scene change is made.</td>
</tr>
<tr>
<td>OnValidate</td>
<td>Called by Unity.</td>
</tr>
<tr>
<td>PausePlayback</td>
<td>Use this method to pause replay playback while maintaining the current replay state. See ResumePlayback to continue playback.</td>
</tr>
<tr>
<td>PauseRecording</td>
<td>Use this method when you want to pause recording but may continue recording at any point. A good candidate for pausing recording is when the user pauses the game and is shown a pause menu. The manager must already be recording, otherwise this method will have no effect.</td>
</tr>
<tr>
<td>RegisterReplayPrefab</td>
<td>Attempts to register a game object as a prefab so that the replay system is able to spawn or despawn the object as needed. You only need to do this for objects that are likely to be instantiated or destroyed during recording. The replay system will then...</td>
</tr>
</tbody>
</table>
be able to accurately restore scene state during playback. The specified object must be a prefab; otherwise an error will be thrown and the object will not be registered. Prefab instances are not accepted.

<p>| S | ReplayDestroy | Attempts to destroy the specified prefab. OnReplayDestroy will be called if a listener has been registered and otherwise default destruction will be used. |
| S | ReplayInstantiate | Attempts to instantiate the specified prefab. OnReplayInstantiate will be called if a listener has been registered and otherwise default instantiation will be used. |
| S | ResumePlayback | Use this method to resume playback after a previous call to PausePlayback was called. If PausePlayback was not called prior to this method then the method will have no effect. |
| S | ResumeRecording | Use this method to resume recording after a previous call to PauseRecording. The manager must already be recording otherwise this method will have no effect. |
| S | SetPlaybackFrame | Use this method to specify where in the replay sequence the playback should start. If the offset does not lie within the bounds of the replay the value will be clamped to represent either the start or end frame. |</p>
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SetPlaybackFrameNormalized</strong></td>
<td>Use this method to specify where in the replay sequence the playback should start. This method accepts normalized offsets values between 0 and 1 and performs validation before using the value.</td>
</tr>
<tr>
<td><strong>Start</strong></td>
<td>Called by Unity. Allows the active replay manager to initialize.</td>
</tr>
<tr>
<td><strong>StopPlayback</strong></td>
<td>Use this method to stop any active playback. This method will only have an effect if there is an active playback running otherwise it will have no effect.</td>
</tr>
<tr>
<td><strong>StopRecording</strong></td>
<td>Use this method to stop recording after a previous call to BeginRecording(Boolean). The manager must already be recording otherwise this method will have no effect. This method must be called before attempting to playback, otherwise you may get unpredictable results.</td>
</tr>
<tr>
<td><strong>Update</strong></td>
<td>Called by Unity. Allows the active replay manager to update recording or playback.</td>
</tr>
</tbody>
</table>

**See Also**

Reference
ReplayManager Class
UltimateReplay Namespace
ReplayManagerBeginPlayback Method

Use this method to begin the playback of the recorded objects. Use `SetPlaybackFrame(Single, PlaybackOrigin)` or `SetPlaybackFrameNormalized(Single, PlaybackOrigin)` before calling this method to specify the exact location at which playback should begin. This method will run the entire playback gathered and then automatically stop playback on completion if `endReplayAfterPlayback` is true.

**Namespace**: UltimateReplay  
**Assembly**: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

▲ Syntax

```csharp
public static void BeginPlayback(
    bool fromStart = true,
    PlaybackDirection direction = PlaybackDirection
)
```

**Parameters**

`fromStart (Optional)`  
Type: `SystemBoolean`  
When true, the replay will be played from the first frame recorded

`direction (Optional)`  
Type: `UltimateReplayPlaybackDirection`  
The direction that the replay should be played
See Also

Reference
ReplayManager Class
UltimateReplay Namespace
ReplayManagerBeginPlaybackFrame Method

Use this method to set the current playback at a specific replay frame. This will allow the state of a specific replay frame to be restored but will not continue playback which will provide a freeze frame effect. Use `SetPlaybackFrame(Single, PlaybackOrigin)` or `SetPlaybackFrameNormalized(Single, PlaybackOrigin)` before calling this method to specify the exact location at which the playback frame should be sampled. Use `StopPlayback` to unfreeze the still frame and return to normal game mode. This method will ignore the value of `endReplayAfterPlayback` as only a single frame is replayed. As a result you will need to call `StopPlayback` when you want to end the playback frame.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```csharp
public static void BeginPlaybackFrame()
```

**See Also**

Reference  
**ReplayManager Class**  
**UltimateReplay Namespace**
ReplayManagerBeginRecording Method

Use this method to begin sampling the recorded objects in the scene. If recordOnStart is true then this method will be called automatically when the manager is initialized. Any state information will be recored via the assigned ReplayTarget (Default ReplayMemoryTarget). When true, any previous recording data will be discarded.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public static void BeginRecording(
    bool cleanRecording = true
)
```

### Parameters

*cleanRecording* (Optional)

Type: SystemBoolean


### See Also

Reference

- ReplayManager Class
- UltimateReplay Namespace
ReplayManagerDiscardRecording Method

This method will throw away any recorded data and flush the replay target if necessary. This method can be called at any time. If the manager is currently recording then all previous data will be discarded and recording will continue. If the manager is currently replaying then all replay data will be discarded and playback will stop.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public static void DiscardRecording()
```

### See Also

**Reference**  
ReplayManager Class  
UltimateReplay Namespace
ReplayManager.FindReplayPrefab Method

Attempts to find the prefab with the matching name. This is used to restore objects that were destroyed during recording.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public static GameObject FindReplayPrefab(
    string prefabName
)
```

### Parameters

`prefabName`  
Type: `System.String`  
The name of the prefab to locate

### Return Value

Type: `GameObject`  
The matching prefab or null if no matching prefab was found

### See Also

**Reference**  
ReplayManager Class  
UltimateReplay Namespace
ReplayManagerForceAwake Method

Override implementation of ForceAwake. Performs exactly the same behaviour. Simply included so that the user does not need to import 'UltimateReplay.Util' to access the method.

Namespace: UltimateReplay
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

 Syntax

```
public static void ForceAwake()
```

See Also

Reference
ReplayManager Class
UltimateReplay Namespace
ReplayManagerOnDestroy Method

Called by Unity. Allows the active replay manager to cleanup any active recordings.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

## Syntax

```csharp
public void OnDestroy()
```

## See Also

Reference  
ReplayManager Class  
UltimateReplay Namespace
ReplayManagerOnLevelWasLoaded Method

Called by Unity. Allows the active replay manager to cleanup recordings when a scene change is made.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public void OnLevelWasLoaded(
    int index
)
```

### Parameters

- **index**  
  Type: `System.Int32`  
  The level id

### See Also

Reference  
ReplayManager Class  
UltimateReplay Namespace
ReplayManagerOnValidate Method

Called by Unity.

Namespace: UltimateReplay
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

Syntax

```csharp
public void OnValidate()
```

See Also

Reference
ReplayManager Class
UltimateReplay Namespace
ReplayManagerPausePlayback Method

Use this method to pause replay playback while maintaining the current replay state. See ResumePlayback to continue a playback.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public static void PausePlayback()
```

### See Also

Reference  
ReplayManager Class  
UltimateReplay Namespace
ReplayManagerPauseRecording Method

Use this method when you want to pause recording but may continue recording at any point. A good candidate for pausing recording is when the user pauses the game and is shown a pause menu. The manager must already be recording otherwise this method will have no effect.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public static void PauseRecording()
```

### See Also

Reference  
ReplayManager Class  
UltimateReplay Namespace
ReplayManagerRegisterReplayPrefab Method

Attempts to register a game object as a prefab so that the replay system is able to spawn or despawn the object as needed. You only need to do this for objects that are likely to be either instantiated or destroyed during recording. The replay system will then be able to accurately restore the scene state during playback. The specified object must be a prefab otherwise an error will be thrown and the object will not be registered. Prefab instances are not accepted.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public static void RegisterReplayPrefab(
    GameObject prefab
)
```

**Parameters**

`prefab`

Type: `GameObject`  
The prefab object to register with the replay system

### See Also

**Reference**  
ReplayManager Class  
UltimateReplay Namespace
ReplayManagerReplayDestroy Method

Attempts to destroy the specified prefab. OnReplayDestroy will be called if a listener has been registered otherwise default destruction will be used.

Namespace: UltimateReplay  
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

⚠️ Syntax

```c#
public static void ReplayDestroy(
    GameObject go
)
```

Parameters

*go*  
Type: GameObject  
The game object to destroy

⚠️ See Also

Reference  
ReplayManager Class  
UltimateReplay Namespace
ReplayManager.ReplayInstantiate Method

Attempts to instantiate the specified prefab. `OnReplayInstantiate` will be called if a listener has been registered otherwise default instantiation will be used.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```csharp
public static GameObject ReplayInstantiate(
    GameObject prefab,
    Vector3 position,
    Quaternion rotation
)
```

**Parameters**

`prefab`  
Type: `GameObject`  
The prefab to instantiate

`position`  
Type: `Vector3`  
The position to spawn the prefab at

`rotation`  
Type: `Quaternion`  
The rotation to spawn the prefab with

**Return Value**

Type: `GameObject`
The new instance of the specified prefab or null if an error occurred

See Also

Reference
ReplayManager Class
UltimateReplay Namespace
ReplayManagerResumePlayback Method

Use this method to resume playback after a previous call to PausePlayback was called. If PausePlayback was not called prior to this method then the method will have no effect.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

Syntax

```csharp
public static void ResumePlayback()
```

See Also

Reference  
ReplayManager Class  
UltimateReplay Namespace
ReplayManagerResumeRecording Method

Use this method to resume recording after a previous call to **PauseRecording**. The manager must already be recording otherwise this method will have no effect.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

⚠ Syntax

```csharp
public static void ResumeRecording()
```

⚠ See Also

Reference  
ReplayManager Class  
UltimateReplay Namespace
ReplayManager.SetPlaybackFrame Method

Use this method to specify where in the replay sequence the playback should start. If the offset does not lie within the bounds of the replay then the value will be clamped to represent either the start or end frame.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public static void SetPlaybackFrame(
    float offset,
    PlaybackOrigin origin = PlaybackOrigin.Start
)
```

### Parameters

- **offset**
  - Type: `System.Single`
  - The amount of time in seconds to offset the playback

- **origin** *(Optional)*
  - Type: `UltimateReplayPlaybackOrigin`
  - The playback node to take the offset from. If `End` is specified then the offset value will be used as a negative offset

### See Also

Reference
ReplayManager Class
UltimateReplay Namespace
ReplayManager.SetPlaybackFrameNormalized Method

Use this method to specify where in the replay sequence the playback should start. This method accepts normalized offsets values between 0 and 1 and performs validation before using the value.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```csharp
public static void SetPlaybackFrameNormalized(
    float normalizedOffset,
    PlaybackOrigin origin = PlaybackOrigin.Start
)
```

**Parameters**

- **normalizedOffset**  
  Type: System.Single  
  The normalized value representing the offset from the specified origin to start the playback from

- **origin (Optional)**  
  Type: UltimateReplayPlaybackOrigin  
  The playback node to take the offset from. If End is specified then the offset value will be used as a negative offset

**See Also**

Reference
ReplayManagerStart Method

Called by Unity. Allows the active replay manager to initialize.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public void Start()
```

### See Also

Reference  
- **ReplayManager Class**  
- **UltimateReplay Namespace**
ReplayManagerStopPlayback Method

Use this method to stop any active playback. This method will only have an effect if there is an active playback running otherwise it will have no effect.

Namespace: UltimateReplay
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

⚠️ Syntax

```csharp
public static void StopPlayback()
```

⚠️ See Also

Reference
ReplayManager Class
UltimateReplay Namespace
ReplayManagerStopRecording Method

Use this method to stop recording after a previous call to BeginRecording(Boolean). The manager must already be recording otherwise this method will have no effect. This method must be called before attempting to playback otherwise you may get unpredictable results.

Namespace: UltimateReplay
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

Syntax

C#

```csharp
public static void StopRecording()
```

See Also

Reference
ReplayManager Class
UltimateReplay Namespace
ReplayManagerUpdate Method

Called by Unity. Allows the active replay manager to update recording or playback.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public void Update()
```

### See Also

Reference  
ReplayManager Class  
UltimateReplay Namespace
# ReplayManager Properties

The `ReplayManager` type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![s] CurrentPlaybackTime</td>
<td>Get the current playback time in seconds. This value will never be greater than <code>duration</code>.</td>
</tr>
<tr>
<td>![s] CurrentPlaybackTimeNormalized</td>
<td>Get the current playback time as a normalized value between 0-1. 0 represents the starting frame of the recording and 1 represents the very last frame of the recording.</td>
</tr>
<tr>
<td>![s] IsPaused</td>
<td>Returns true when the active replay manager is in any paused state. Paused states could include <code>Playback_Paused</code> or <code>Recording_Paused</code>.</td>
</tr>
<tr>
<td>![s] IsRecording</td>
<td>Returns true if the manager is currently recording.</td>
</tr>
</tbody>
</table>
recording the scene. Note: If recording is paused this value will still be true.

<table>
<thead>
<tr>
<th></th>
<th>IsReplaying</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>Returns true if the manager is currently playing back previously recorded replay data. Note: if playback is paused this value will still be true.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>PlaybackDirection</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>Gets the current PlaybackDirection of replay playback.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>s</th>
<th>Preparer</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>Access the current IReplayPreparer that the active replay manager will use to prepare game objects for replay. By default a DefaultReplayPreparer is used.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>s</th>
<th>Replay</th>
</tr>
</thead>
</table>
| s | Get the active replay manager in the scene. If no replay manager could be found then one will be created with default settings. This property may be null in a rare case where the active replay manager was destroyed in the same
frame as an application quit event was issued. In this situation the replay manager cannot be recreated as it would cause leaked objects.

<table>
<thead>
<tr>
<th>Scene</th>
<th>Get the <strong>ReplayScene</strong> associated with the replay system.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td>The current replay target that is being used to store the replay data. By default, the replay target is <strong>ReplayMemoryTarget</strong>.</td>
</tr>
</tbody>
</table>

**Top**

**See Also**

**Reference**
- ReplayManager Class
- UltimateReplay Namespace
ReplayManagerCurrentPlaybackTime Property

Get the current playback time in seconds. This value will never be greater than duration.

Namespace: UltimateReplay
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

Syntax

C#

```csharp
public static float CurrentPlaybackTime { get; }
```

Property Value
Type: Single

See Also

Reference
ReplayManager Class
UltimateReplay Namespace
ReplayManagerCurrentPlaybackTimeNormalized Property

Get the current playback time as a normalized value between 0-1. 0 represents the starting frame of the recording and 1 represents the very last frame of the recording.

Namespace: UltimateReplay
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

Syntax

```csharp
public static float CurrentPlaybackTimeNormalized
```

Property Value

Type: Single

See Also

Reference

ReplayManager Class
UltimateReplay Namespace
ReplayManagerIsPaused Property

Returns true when the active replay manager is in any paused state. Paused states could include Playback_Paused or Recording_Paused.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public static bool IsPaused { get; }
```

Property Value  
**Type:** Boolean

### See Also

**Reference**  
ReplayManager Class  
UltimateReplay Namespace
ReplayManager.IsRecording Property

Returns true if the manager is currently recording the scene. Note: If recording is paused this value will still be true.

Namespace: UltimateReplay
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0

Syntax

```csharp
public static bool IsRecording { get; }
```

Property Value
Type: Boolean

See Also

Reference
ReplayManager Class
UltimateReplay Namespace
ReplayManager.IsReplaying Property

Returns true if the manager is currently playing back previously recorded replay data. Note: if playback is paused this value will still be true.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public static bool IsReplaying { get; }
```

**Property Value**

**Type:** Boolean

### See Also

Reference

*ReplayManager Class*  
*UltimateReplay Namespace*
ReplayManagerPlaybackDirection Property

Gets the current PlaybackDirection of replay playback.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public static PlaybackDirection PlaybackDirection
```

### Property Value

**Type:** PlaybackDirection

### See Also

Reference  
ReplayManager Class  
UltimateReplay Namespace
ReplayManagerPreparer Property

Access the current `IReplayPreparer` that the active replay manager will use to prepare game objects for replay. By default a `DefaultReplayPreparer` is used.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```
public static IReplayPreparer Preparer {
  get;
  set;
}
```

### Property Value

Type: `IReplayPreparer`

### See Also

- Reference
  - `ReplayManager Class`
  - `UltimateReplay Namespace`
ReplayManagerReplay Property

Get the active replay manager in the scene. If no replay manager could be found then one will be created with default settings. This property may be null in a rare case where the active replay manager was destroyed in the same frame as an application quit event was issued. In this situation the replay manager cannot be recreated as it would cause leaked objects.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

▲ Syntax

```c#
public static ReplayManager Replay { get; }
```

Property Value
Type: ReplayManager

▲ See Also

Reference
ReplayManager Class  
UltimateReplay Namespace
ReplayManagerScene Property

Get the ReplayScene associated with the replay system.

Namespace: UltimateReplay
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

Syntax

```c#
public static ReplayScene Scene { get; }
```

Property Value
Type: ReplayScene

See Also

Reference
ReplayManager Class
UltimateReplay Namespace
ReplayManagerTarget Property

The current replay target that is being used to store the replay data. By default, the replay target is `ReplayMemoryTarget`.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```csharp
public static ReplayTarget Target { get; set; }
```

**Property Value**

Type: `ReplayTarget`

**See Also**

Reference
- ReplayManager Class
- UltimateReplay Namespace
ReplayObject Class

Only one instance of ReplayObject can be added to any game object.

Inheritance Hierarchy

System
  Object
  Component
    Behaviour
      MonoBehaviour
        UltimateReplay.ReplayObject

Namespace: UltimateReplay
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

Syntax

```csharp
public sealed class ReplayObject : MonoBehaviour, IReplaySerialize
```

The ReplayObject type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>ReplayObject</em></td>
<td>Initializes a new instance of the ReplayObject class</td>
</tr>
</tbody>
</table>
## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Awake</td>
<td>Called by Unity.</td>
</tr>
<tr>
<td>OnDestroy</td>
<td>Called by Unity.</td>
</tr>
<tr>
<td>OnReplayDeserializer</td>
<td>Called by the replay system when this <code>ReplayObject</code> should deserialize its replay data.</td>
</tr>
<tr>
<td>OnReplaySerialize</td>
<td>Called by the replay system when this <code>ReplayObject</code> should serialize its replay data.</td>
</tr>
<tr>
<td>RebuildComponentList</td>
<td>Forces the object to refresh its list of observed components. Observed components are components which inherit from <code>ReplayBehaviour</code> and exist on either this game object or a child of this game object.</td>
</tr>
</tbody>
</table>

## Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>observedComponents</td>
<td>An array of <code>ReplayBehaviour</code> components that this object will serialize during recording. Dynamically adding replay components during recording is not supported.</td>
</tr>
</tbody>
</table>
## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsPrefab</td>
<td>Returns true when this game object is a prefab asset. Returns false when this game object is a scene object or prefab instance.</td>
</tr>
<tr>
<td>PrefabIdentity</td>
<td>Get the prefab associated with this ReplayObject.</td>
</tr>
<tr>
<td>ReplayIdentity</td>
<td>Get the unique ReplayIdentity for this ReplayObject.</td>
</tr>
</tbody>
</table>

## See Also

Reference
UltimateReplay Namespace
ReplayObject Constructor

Initializes a new instance of the ReplayObject class

Namespace: UltimateReplay
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

Syntax

```csharp
public ReplayObject()
```

See Also

Reference
ReplayObject Class
UltimateReplay Namespace
ReplayObject Fields

The `ReplayObject` type exposes the following members.

### Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>observedComponents</code></td>
<td>An array of <code>ReplayBehaviour</code> components that this object will serialize during recording. Dynamically adding replay components during recording is not supported.</td>
</tr>
</tbody>
</table>

See Also

Reference

- `ReplayObject Class`
- `UltimateReplay Namespace`
ReplayObjectobservedComponents Field

An array of ReplayBehaviour components that this object will serialize during recording. Dynamically adding replay components during recording is not supported.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

⚠️ Syntax

```c#
public ReplayBehaviour[] observedComponents
```

Field Value  
Type: ReplayBehaviour

⚠️ See Also

Reference  
ReplayObject Class  
UltimateReplay Namespace
ReplayObject Methods

The **ReplayObject** type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Awake</td>
<td>Called by Unity.</td>
</tr>
<tr>
<td>OnDestroy</td>
<td>Called by Unity.</td>
</tr>
<tr>
<td>OnReplayDeserialize</td>
<td>Called by the replay system when this <strong>ReplayObject</strong> should deserialize its replay data.</td>
</tr>
<tr>
<td>OnReplaySerialize</td>
<td>Called by the replay system when this <strong>ReplayObject</strong> should serialize its replay data.</td>
</tr>
<tr>
<td>RebuildComponentList</td>
<td>Forces the object to refresh its list of observed components. Observed components are components which inherit from <strong>ReplayBehaviour</strong> and exist on either this game object or a child of this game object.</td>
</tr>
</tbody>
</table>

**Top**

**See Also**

Reference
**ReplayObject Class**
UltimateReplay Namespace
ReplayObjectAwake Method

Called by Unity.

**Namespace:** UltimateReplay

**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```
public void Awake()
```

### See Also

Reference

ReplayObject Class

UltimateReplay Namespace
ReplayObject OnDestroy Method

Called by Unity.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0  
(1.0.0.0)

### Syntax

```csharp
public void OnDestroy()
```

### See Also

Reference  
ReplayObject Class  
UltimateReplay Namespace
ReplayObjectOnReplayDeserialize Method

Called by the replay system when this ReplayObject should deserialize its replay data.

Namespace: UltimateReplay
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0

Syntax

```csharp
public void OnReplayDeserialize(ReplayState state)
```

Parameters

*state*  
Type: UltimateReplayReplayState  
The ReplayState to deserialize the data from

Implements

IReplaySerializeOnReplayDeserialize(ReplayState)

See Also

Reference
ReplayObject Class
UltimateReplay Namespace
ReplayObjectOnReplaySerialize Method

Called by the replay system when this `ReplayObject` should serialize its replay data.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```c#
public void OnReplaySerialize(
    ReplayState state
)
```

**Parameters**

- `state`  
  Type: `UltimateReplay.ReplayState`  
  The `ReplayState` to serialize the data to

**Implements**

`IReplaySerializeOnReplaySerialize(ReplayState)`

**See Also**

- Reference
  - `ReplayObject Class`
  - `UltimateReplay Namespace`
ReplayObjectRebuildComponentList Method

Forces the object to refresh its list of observed components. Observed components are components which inherit from ReplayBehaviour and exist on either this game object or a child of this game object.

Namespace: UltimateReplay
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

Syntax

```c#
public void RebuildComponentList()
```

See Also

Reference
ReplayObject Class
UltimateReplay Namespace
ReplayObject Properties

The ReplayObject type exposes the following members.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsPrefab</td>
<td>Returns true when this game object is a prefab asset. Returns false when this game object is a scene object or prefab instance.</td>
</tr>
<tr>
<td>PrefabIdentity</td>
<td>Get the prefab associated with this ReplayObject.</td>
</tr>
<tr>
<td>ReplayIdentity</td>
<td>Get the unique ReplayIdentity for this ReplayObject.</td>
</tr>
</tbody>
</table>

See Also

Reference
ReplayObject Class
UltimateReplay Namespace
ReplayObjectIsPrefab Property

Returns true when this game object is a prefab asset. Returns false when this game object is a scene object or prefab instance.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```csharp
public bool IsPrefab { get; }
```

**Property Value**

Type: Boolean

**See Also**

Reference  
ReplayObject Class  
UltimateReplay Namespace
ReplayObjectPrefabIdentity Property

Get the prefab associated with this ReplayObject.

Namespace: UltimateReplay
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

Syntax

```csharp
public string PrefabIdentity { get; }
```

Property Value
Type: String

See Also

Reference
ReplayObject Class
UltimateReplay Namespace
ReplayObject.ReplayIdentity Property

Get the unique ReplayIdentity for this ReplayObject.

Namespace: UltimateReplay
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

Syntax

```csharp
public ReplayIdentity ReplayIdentity { get; }
```

Property Value
Type: ReplayIdentity

See Also

Reference
ReplayObject Class
UltimateReplay Namespace
ReplayState Class

A ReplayState allows replay objects to serialize and deserialize their data. See IReplaySerialize.

Inheritance Hierarchy

System
  Object
  UltimateReplay
    ReplayState

Namespace: UltimateReplay
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0

Syntax

C#

```csharp
public sealed class ReplayState
```

The ReplayState type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReplayState</td>
<td>Create an empty ReplayState that can be written to.</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clear</td>
<td>Clears all buffered data from</td>
</tr>
</tbody>
</table>
this **ReplayState** and resets its state.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read16</td>
<td>Read a short from the state.</td>
</tr>
<tr>
<td>Read32</td>
<td>Read an int from the state.</td>
</tr>
<tr>
<td>ReadBool</td>
<td>Read a bool from the state.</td>
</tr>
<tr>
<td>ReadByte</td>
<td>Read a byte from the state.</td>
</tr>
<tr>
<td>ReadBytes(Int32)</td>
<td>Read a byte array from the state.</td>
</tr>
<tr>
<td>ReadBytes(Byte, Int32, Int32)</td>
<td>Fill a byte array with data from the state.</td>
</tr>
<tr>
<td>ReadColor</td>
<td>Read a colour from the state.</td>
</tr>
<tr>
<td>ReadColor32</td>
<td>Read a colour32 from the state.</td>
</tr>
<tr>
<td>ReadFloat</td>
<td>Read a float from the state.</td>
</tr>
<tr>
<td>ReadFloatLowPrecision</td>
<td>Attempts to read a low precision float. You should only use this method when the value is relatively small (less than 65000).</td>
</tr>
<tr>
<td>ReadIdentity</td>
<td>Read a <strong>ReplayIdentity</strong> from the state.</td>
</tr>
<tr>
<td>ReadQuat</td>
<td>Read a quaternion from the state.</td>
</tr>
<tr>
<td>ReadState</td>
<td>Read the specified amount of bytes as a new <strong>ReplayState</strong>.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ReadString</td>
<td>Read a string from the state.</td>
</tr>
<tr>
<td>ReadVec2</td>
<td>Read a vector2 from the state.</td>
</tr>
<tr>
<td>ReadVec3</td>
<td>Read a vector3 from the state.</td>
</tr>
<tr>
<td>ReadVec4</td>
<td>Read a vector4 from the state.</td>
</tr>
<tr>
<td>ToArray</td>
<td>Get the <code>ReplayState</code> data as a byte array.</td>
</tr>
<tr>
<td>TryReadObject</td>
<td>Attempts to read an object state from this <code>ReplayState</code>.</td>
</tr>
<tr>
<td>TryWriteObject</td>
<td>Attempts to write an object to this <code>ReplayState</code>. This method may write extra meta data for deserialization purposes which may cause excessive storage size. Use one of the <code>Write(Byte)</code> methods if the type is known at compile time.</td>
</tr>
<tr>
<td>Write(Boolean)</td>
<td>Write a bool to the state.</td>
</tr>
<tr>
<td>Write(Byte)</td>
<td>Write a byte to the state.</td>
</tr>
<tr>
<td>Write(Byte)</td>
<td>Write a byte array to the state.</td>
</tr>
<tr>
<td>Write(Int16)</td>
<td>Write a short to the state.</td>
</tr>
<tr>
<td>Write(Int32)</td>
<td>Write an int to the state.</td>
</tr>
<tr>
<td>Write(Single)</td>
<td>Write a float to the state.</td>
</tr>
<tr>
<td>Write(String)</td>
<td>Write a string to the state.</td>
</tr>
<tr>
<td>Write(Color)</td>
<td>Write a colour to the state.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>Write(Color32)</td>
<td>Write a colour32 value to the state.</td>
</tr>
<tr>
<td>Write(Quaternion)</td>
<td>Write a quaternion to the state.</td>
</tr>
<tr>
<td>Write(Vector2)</td>
<td>Write a vector2 to the state.</td>
</tr>
<tr>
<td>Write(Vector3)</td>
<td>Write a vector3 to the state.</td>
</tr>
<tr>
<td>Write(Vector4)</td>
<td>Write a vector4 to the state.</td>
</tr>
<tr>
<td>Write(ReplayIdentity)</td>
<td>Write the specified replay identity to this ReplayState.</td>
</tr>
<tr>
<td>Write(ReplayState)</td>
<td>Write the entire contents of a ReplayState to this ReplayState. All bytes will be appended.</td>
</tr>
<tr>
<td>Write(Byte, Int32, Int32)</td>
<td>Write a byte array to the state using an offset position and length.</td>
</tr>
<tr>
<td>WriteLowPrecision</td>
<td>Attempts to write a 32 bit float value as a low precision 16 bit representation. You should only use this method when the value is relatively small (less than 65000). Accuracy may be lost by storing low precision values.</td>
</tr>
</tbody>
</table>

**Top Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
CanRead  Returns true if the state contains any more data.

EndRead  Returns true if the read pointer is at the end of the buffered data or false if there is still data to be read.

Size  Returns the size of the object state in bytes.

See Also

Reference
UltimateReplay Namespace
ReplayState Constructor

Create an empty ReplayState that can be written to.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

⚠️ **Syntax**

```c#
public ReplayState()
```

⚠️ **See Also**

Reference  
ReplayState Class  
UltimateReplay Namespace
ReplayState Methods

The ReplayState type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clear</td>
<td>Clears all buffered data from this ReplayState and resets its state.</td>
</tr>
<tr>
<td>Read16</td>
<td>Read a short from the state.</td>
</tr>
<tr>
<td>Read32</td>
<td>Read an int from the state.</td>
</tr>
<tr>
<td>ReadBool</td>
<td>Read a bool from the state.</td>
</tr>
<tr>
<td>ReadByte</td>
<td>Read a byte from the state.</td>
</tr>
<tr>
<td>ReadBytes(Int32)</td>
<td>Read a byte array from the state.</td>
</tr>
<tr>
<td>ReadBytes(Byte, Int32, Int32)</td>
<td>Fill a byte array with data from the state.</td>
</tr>
<tr>
<td>ReadColor</td>
<td>Read a colour from the state.</td>
</tr>
<tr>
<td>ReadColor32</td>
<td>Read a colour32 from the state.</td>
</tr>
<tr>
<td>ReadFloat</td>
<td>Read a float from the state.</td>
</tr>
<tr>
<td>ReadFloatLowPrecision</td>
<td>Attempts to read a low precision float. You should</td>
</tr>
</tbody>
</table>
only use this method when the value is relatively small (less than 65000).

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReadIdentity</td>
<td>Read a <code>ReplayIdentity</code> from the state.</td>
</tr>
<tr>
<td>ReadQuat</td>
<td>Read a quaternion from the state.</td>
</tr>
<tr>
<td>ReadState</td>
<td>Read the specified amount of bytes as a new <code>ReplayState</code>.</td>
</tr>
<tr>
<td>ReadString</td>
<td>Read a string from the state.</td>
</tr>
<tr>
<td>ReadVec2</td>
<td>Read a vector2 from the state.</td>
</tr>
<tr>
<td>ReadVec3</td>
<td>Read a vector3 from the state.</td>
</tr>
<tr>
<td>ReadVec4</td>
<td>Read a vector4 from the state.</td>
</tr>
<tr>
<td>ToArray</td>
<td>Get the <code>ReplayState</code> data as a byte array.</td>
</tr>
<tr>
<td>TryReadObject</td>
<td>Attempts to read an object state from this <code>ReplayState</code>.</td>
</tr>
<tr>
<td>TryWriteObject</td>
<td>Attempts to write an object to this <code>ReplayState</code>. This method may write extra meta data for deserialization purposes which may cause excessive storage size. Use one of the <code>Write(Byte)</code> methods if the type is known at compile time.</td>
</tr>
<tr>
<td>Write(Boolean)</td>
<td>Write a bool to the state.</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Write(Byte)</td>
<td>Write a byte to the state.</td>
</tr>
<tr>
<td>Write(Byte)</td>
<td>Write a byte array to the state.</td>
</tr>
<tr>
<td>Write(Int16)</td>
<td>Write a short to the state.</td>
</tr>
<tr>
<td>Write(Int32)</td>
<td>Write an int to the state.</td>
</tr>
<tr>
<td>Write(Single)</td>
<td>Write a float to the state.</td>
</tr>
<tr>
<td>Write(String)</td>
<td>Write a string to the state.</td>
</tr>
<tr>
<td>Write(Color)</td>
<td>Write a colour to the state.</td>
</tr>
<tr>
<td>Write(Color32)</td>
<td>Write a colour32 value to the state.</td>
</tr>
<tr>
<td>Write(Quaternion)</td>
<td>Write a quaternion to the state.</td>
</tr>
<tr>
<td>Write(Vector2)</td>
<td>Write a vector2 to the state.</td>
</tr>
<tr>
<td>Write(Vector3)</td>
<td>Write a vector3 to the state.</td>
</tr>
<tr>
<td>Write(Vector4)</td>
<td>Write a vector4 to the state.</td>
</tr>
<tr>
<td>Write(ReplayIdentity)</td>
<td>Write the specified replay identity to this <code>ReplayState</code>.</td>
</tr>
<tr>
<td>Write(ReplayState)</td>
<td>Write the entire contents of a <code>ReplayState</code> to this <code>ReplayState</code>. All bytes will be appended.</td>
</tr>
<tr>
<td>Write(Byte, Int32, Int32)</td>
<td>Write a byte array to the state using an offset position and length.</td>
</tr>
<tr>
<td>WriteLowPrecision</td>
<td>Attempts to write a 32 bit float.</td>
</tr>
</tbody>
</table>
value as a low precision 16 bit representation. You should only use this method when the value is relatively small (less than 65000). Accuracy may be lost by storing low precision values.

See Also

Reference
ReplayState Class
UltimateReplay Namespace
ReplayState Clear Method

Clears all buffered data from this ReplayState and resets its state.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```
public void Clear()
```

### See Also

Reference  
ReplayState Class  
UltimateReplay Namespace
ReplayStateRead16 Method

Read a short from the state.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public short Read16()
```

### Return Value

**Type:** Int16  
Short value

### See Also

**Reference**  
ReplayState Class  
UltimateReplay Namespace
ReplayStateRead32 Method

Read an int from the state.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```c#
public int Read32()
```

### Return Value

Type: `Int32`  
Int value

### See Also

**Reference**  
ReplayState Class  
UltimateReplay Namespace
ReplayStateReadBool Method

Read a bool from the state.

**Namespace:** UltimateReplay
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```c#
public bool ReadBool()
```

### Return Value

Type: **Boolean**

Boo value

### See Also

Reference

ReplayState Class
UltimateReplay Namespace
ReplayState ReadByte Method

Read a byte from the state.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```csharp
public byte ReadByte()
```

**Return Value**

Type: **Byte**  
Byte value

**See Also**

Reference  
ReplayState Class  
UltimateReplay Namespace
ReplayStateReadStream Method

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ReadStream(Int32)</code></td>
<td>Read a byte array from the state.</td>
</tr>
<tr>
<td><code>ReadStream(Byte, Int32, Int32)</code></td>
<td>Fill a byte array with data from the state.</td>
</tr>
</tbody>
</table>

See Also

Reference
- ReplayState Class
- UltimateReplay Namespace
ReplayStateReadBytes Method (Int32)

Read a byte array from the state.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```c#
public byte[] ReadBytes(
    int amount
)
```

**Parameters**

*amount*  
Type: SystemInt32  
The number of bytes to read

**Return Value**

Type: Byte  
Byte array value

**See Also**

Reference  
ReplayState Class  
ReadBytes Overload  
UltimateReplay Namespace
ReplayStateReadBytes Method (Byte, Int32, Int32)

Fill a byte array with data from the state.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```c#
public void ReadBytes(
    byte[] buffer,
    int offset,
    int amount
)
```

### Parameters

- **buffer**
  - Type: `SystemByte`  
  - The byte array to store data in

- **offset**
  - Type: `SystemInt32`  
  - The index offset to start filling the buffer at

- **amount**
  - Type: `SystemInt32`  
  - The number of bytes to read

### See Also

Reference
ReplayState Class
ReadBytes Overload
UltimateReplay Namespace
ReplayStateReadColor Method

Read a colour from the state.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public Color ReadColor()
```

### Return Value

**Type:** Color  
Colour value

### See Also

Reference

- [ReplayState Class](#)
- [UltimateReplay Namespace](#)
ReplayStateReadColor32 Method

Read a colour32 from the state.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```csharp
public Color32 ReadColor32()
```

**Return Value**

Type: **Color32**  
Colour32 value

**See Also**

Reference  
ReplayState Class  
UltimateReplay Namespace
ReplayStateReadFloat Method

Read a float from the state.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public float ReadFloat()
```

### Return Value

**Type:** Single  
Float value

### See Also

**Reference**  
ReplayState Class  
UltimateReplay Namespace
ReplayStateReadFloatLowPrecision Method

Attempts to read a low precision float. You should only use this method when the value is relatively small (less than 65000).

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```csharp
public float ReadFloatLowPrecision()
```

**Return Value**

Type: Single  
float value

**See Also**

Reference  
ReplayState Class  
UltimateReplay Namespace
ReplayStateReadIdentity Method

Read a ReplayIdentity from the state.

Namespace: UltimateReplay
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

Syntax

```c#
public ReplayIdentity ReadIdentity()
```

Return Value

Type: ReplayIdentity
Identity value

See Also

Reference
ReplayState Class
UltimateReplay Namespace
**ReplayStateReadQuat Method**

Read a quaternion from the state.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

⚠️ **Syntax**

```csharp
public Quaternion ReadQuat()
```

**Return Value**

Type: Quaternion  
Quaternion value

⚠️ **See Also**

Reference

- ReplayState Class
- UltimateReplay Namespace
ReplayState.ReadState Method

Read the specified amount of bytes as a new ReplayState.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

⚠️ Syntax

```csharp
public ReplayState ReadState(int bytes)
```

**Parameters**

`bytes`  
Type: System.Int32  
The amount of bytes to read into the state

**Return Value**

Type: ReplayState  
A new ReplayState containing the specified number of bytes

⚠️ See Also

Reference
- ReplayState Class  
- UltimateReplay Namespace
ReplayState.ReadString Method

Read a string from the state

Namespace: UltimateReplay
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

Syntax

```
public string ReadString()
```

Return Value

Type: String
string value

See Also

Reference
ReplayState Class
UltimateReplay Namespace
ReplayStateReadVec2 Method

Read a vector2 from the state.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```c#
public Vector2 ReadVec2()
```

**Return Value**

Type: `Vector2`  
Vector2 value

### See Also

**Reference**

- [ReplayState Class](#)
- [UltimateReplay Namespace](#)
ReplayStateReadVec3 Method

Read a vector3 from the state.

Namespace: UltimateReplay
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

Syntax

```csharp
public Vector3 ReadVec3()
```

Return Value

Type: Vector3
Vector3 value

See Also

Reference

ReplayState Class
UltimateReplay Namespace
ReplayStateReadVec4 Method

Read a vector4 from the state.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

⚠️ Syntax

```csharp
public Vector4 ReadVec4()
```

**Return Value**

Type: **Vector4**  
Vector4 value

⚠️ See Also

Reference  
ReplayState Class  
UltimateReplay Namespace
ReplayStateToArray Method

Get the ReplayState data as a byte array.

Namespace: UltimateReplay
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

Syntax

```c#
public byte[] ToArray()
```

Return Value

Type: Byte
A byte array of data

See Also

Reference

ReplayState Class
UltimateReplay Namespace
ReplayState.TryReadObject Method

Attempts to read an object state from this ReplayState.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```csharp
public Object TryReadObject()
```

**Return Value**

Type: Object

The state data for the object

**See Also**

Reference

ReplayState Class  
UltimateReplay Namespace
ReplayState TryWriteObject Method

Attempts to write an object to this ReplayState. This method may write extra meta data for deserialization purposes which may cause excessive storage size. Use one of the Write(Byte) methods if the type is known at compile time.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```csharp
public void TryWriteObject(
    Object value
)
```

**Parameters**

`value`  
Type: `System.Object`  
The object to write to the state

**See Also**

Reference  
ReplayState Class  
UltimateReplay Namespace
# ReplayState\textbf{Write} Method

## Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write(Boolean)</td>
<td>Write a bool to the state.</td>
</tr>
<tr>
<td>Write(Byte)</td>
<td>Write a byte to the state.</td>
</tr>
<tr>
<td>Write(Byte)</td>
<td>Write a byte array to the state.</td>
</tr>
<tr>
<td>Write(Int16)</td>
<td>Write a short to the state.</td>
</tr>
<tr>
<td>Write(Int32)</td>
<td>Write an int to the state.</td>
</tr>
<tr>
<td>Write(Single)</td>
<td>Write a float to the state.</td>
</tr>
<tr>
<td>Write(String)</td>
<td>Write a string to the state.</td>
</tr>
<tr>
<td>Write(Color)</td>
<td>Write a colour to the state.</td>
</tr>
<tr>
<td>Write(Color32)</td>
<td>Write a colour32 value to the state.</td>
</tr>
<tr>
<td>Write(Quaternion)</td>
<td>Write a quaternion to the state.</td>
</tr>
<tr>
<td>Write(Vector2)</td>
<td>Write a vector2 to the state.</td>
</tr>
<tr>
<td>Write(Vector3)</td>
<td>Write a vector3 to the state.</td>
</tr>
<tr>
<td>Write(Vector4)</td>
<td>Write a vector4 to the state.</td>
</tr>
<tr>
<td>Write(ReplayIdentity)</td>
<td>Write the specified replay identity to this ReplayState.</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Write(ReplayState)</td>
<td>Write the entire contents of a ReplayState to this ReplayState. All bytes will be appended.</td>
</tr>
<tr>
<td>Write(Byte, Int32, Int32)</td>
<td>Write a byte array to the state using an offset position and length.</td>
</tr>
</tbody>
</table>

See Also

Reference
- ReplayState Class
- UltimateReplay Namespace
ReplayStateWrite Method (Boolean)

Write a bool to the state.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

```csharp
public void Write(
    bool value
)
```

### Parameters

**value**
- Type: SystemBoolean
- bool value

### See Also

Reference
- ReplayState Class
- Write Overload
- UltimateReplay Namespace
ReplayStateWrite Method (Byte)

Write a byte to the state.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

⚠️ Syntax

```csharp
public void Write(
    byte value
)
```

Parameters

`value`

- Type: `SystemByte`  
  Byte value

⚠️ See Also

Reference

- ReplayState Class
- Write Overload
- UltimateReplay Namespace
ReplayStateWrite Method (Byte)

Write a byte array to the state.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public void Write(byte[] bytes)
```

### Parameters

**bytes**
- Type: `System.Byte`
- Byte array value

### See Also

Reference
- ReplayState Class  
- Write Overload  
- UltimateReplay Namespace
ReplayStateWrite Method (Int16)

Write a short to the state.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```c#
public void Write(
    short value
)
```

### Parameters

**value**
- Type: `System.Int16`
- Short value

### See Also

Reference
- ReplayState Class
- Write Overload
- UltimateReplay Namespace
ReplayStateWrite Method (Int32)

Write an int to the state.

Namespace: UltimateReplay
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

Syntax

```c#
public void Write(
    int value
)
```

Parameters

- **value**
  - Type: SystemInt32
  - Int value

See Also

Reference

- ReplayState Class
- Write Overload
- UltimateReplay Namespace
ReplayStateWrite Method (Single)

Write a float to the state.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```c#
public void Write(
    float value
)
```

### Parameters

- **value**
  - Type: System.Single
  - Float value

### See Also

- **Reference**
  - ReplayState Class
  - Write Overload
  - UltimateReplay Namespace
ReplayStateWrite Method (String)

Write a string to the state.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

*Syntax*

```csharp
public void Write(  
    string value  
)
```

**Parameters**

`value`

Type: `System.String`  
string value

*See Also*

Reference  
ReplayState Class  
Write Overload  
UltimateReplay Namespace
ReplayStateWrite Method (Color)

Write a colour to the state.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```csharp
public void Write(
    Color value
)
```

**Parameters**

*value*

Type: **Color**  
Colour value

**See Also**

Reference

ReplayState Class  
Write Overload  
UltimateReplay Namespace
ReplayStateWrite Method (Color32)

Write a colour32 value to the state.

Namespace: UltimateReplay
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0
(1.0.0.0)

Syntax

```c#
public void Write(
    Color32 value
)
```

Parameters

`value`
- Type: `Color32`
  Colour32 value

See Also

Reference
- ReplayState Class
- Write Overload
- UltimateReplay Namespace
ReplayStateWrite Method (Quaternion)

Write a quaternion to the state.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0

(1.0.0.0)

### Syntax

```csharp
public void Write(
    Quaternion value
)
```

### Parameters

- **value**
  - Type: Quaternion
  - Quaternion value

### See Also

- Reference
  - ReplayState Class
  - Write Overload
  - UltimateReplay Namespace
ReplayStateWrite Method (Vector2)

Write a vector2 to the state.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

▲ Syntax

```c#
public void Write(
    Vector2 value
)
```

**Parameters**

`value`
- Type: `Vector2`
- `Vector2` value

▲ See Also

**Reference**
- ReplayState Class
- Write Overload
- UltimateReplay Namespace
ReplayStateWrite Method (Vector3)

Write a vector3 to the state.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

⚠️ Syntax

```csharp
public void Write(
    Vector3 value
)
```

Parameters

- **value**
  - Type: **Vector3**  
  - Vector3 value

⚠️ See Also

**Reference**
- ReplayState Class
- Write Overload
- UltimateReplay Namespace
ReplayStateWrite Method (Vector4)

Write a vector4 to the state.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```c#
public void Write(
    Vector4 value
)
```

### Parameters

- **value**
  - Type: **Vector4**
    - Vector4 value

### See Also

- Reference
  - ReplayState Class
  - Write Overload
  - UltimateReplay Namespace
ReplayStateWrite Method (ReplayIdentity)

Write the specified replay identity to this ReplayState.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public void Write(
    ReplayIdentity identity
)
```

### Parameters

**identity**  
Type: UltimateReplay.Core.ReplayIdentity  
The identity to write

### See Also

Reference  
ReplayState Class  
Write Overload  
UltimateReplay Namespace
ReplayStateWrite Method (ReplayState)

Write the entire contents of a ReplayState to this ReplayState. All bytes will be appended.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```csharp
public void Write(  
    ReplayState other
)
```

**Parameters**

*other*
- Type: UltimateReplay.ReplayState
- The other state to append

**See Also**

**Reference**
- ReplayState Class
- Write Overload
- UltimateReplay Namespace
ReplayStateWrite Method (Byte, Int32, Int32)

Write a byte array to the state using an offset position and length.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

⚠️ Syntax

```
public void Write(
    byte[] bytes,
    int offset,
    int length
)
```

**Parameters**

- **bytes**  
  Type: SystemByte  
  Byte array value

- **offset**  
  Type: SystemInt32  
  The start index to read data from the array

- **length**  
  Type: SystemInt32  
  The amount of data to read

⚠️ See Also

Reference
ReplayState Class
Write Overload
UltimateReplay Namespace
ReplayStateWriteLowPrecision Method

Attempts to write a 32 bit float value as a low precision 16 bit representation. You should only use this method when the value is relatively small (less than 65000). Accuracy may be lost by storing low precision values.

Namespace: UltimateReplay
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

Syntax

```c#
public void WriteLowPrecision(
    float value
)
```

Parameters

- `value`
  - Type: System.Single
  - float value

See Also

Reference
- ReplayState Class
- UltimateReplay Namespace
ReplayState Properties

The `ReplayState` type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanRead</td>
<td>Returns true if the state contains any more data.</td>
</tr>
<tr>
<td>EndRead</td>
<td>Returns true if the read pointer is at the end of the buffered data or false if there is still data to be read.</td>
</tr>
<tr>
<td>Size</td>
<td>Returns the size of the object state in bytes.</td>
</tr>
</tbody>
</table>

See Also

Reference

- ReplayState Class
- UltimateReplay Namespace
ReplayStateCanRead Property

Returns true if the state contains any more data.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0

### Syntax

```csharp
public bool CanRead {
    get;
}
```

### Property Value

Type: Boolean

### See Also

Reference  
ReplayState Class  
UltimateReplay Namespace
ReplayStateEndRead Property

Returns true if the read pointer is at the end of the buffered data or false if there is still data to be read.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public bool EndRead { get; }
```

### Property Value

**Type:** Boolean

### See Also

**Reference**  
ReplayState Class  
UltimateReplay Namespace
ReplayStateSize Property

Returns the size of the object state in bytes.

Namespace: UltimateReplay
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0
(1.0.0.0)

Syntax

```csharp
public int Size { get; }
```

Property Value
Type: Int32

See Also

Reference
ReplayState Class
UltimateReplay Namespace
ReplayTime Class

This class emulates the behaviour of the Time class in Unity and can be used to modify the playback speed of a replay. There are also delta values that can be used to interpolate between frames where a low record frame rate is used. See ReplayTransform for an example.

▶ Inheritance Hierarchy

System

Object

UltimateReplay

ReplayTime

Namespace: UltimateReplay
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

▶ Syntax

```csharp
public static class ReplayTime
```

The ReplayTime type exposes the following members.

▶ Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetCorrectedTimeValueString</td>
<td>Gets the current time as a float and converts it to minutes and seconds formatted as a string.</td>
</tr>
</tbody>
</table>
### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![s]</td>
<td>Delta</td>
</tr>
<tr>
<td>![s]</td>
<td>Time</td>
</tr>
<tr>
<td>![s]</td>
<td>TimeScale</td>
</tr>
</tbody>
</table>

### See Also

Reference

UltimateReplay Namespace
ReplayTime Methods

The `ReplayTime` type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>GetCorrectedTimeValueString</code></td>
<td>Gets the current time as a float and converts it to minutes and seconds formatted as a string.</td>
</tr>
</tbody>
</table>

See Also

Reference

- `ReplayTime Class`
- `UltimateReplay Namespace`
ReplayTime.GetCorrectedTimeValueString Method

Gets the current time as a float and converts it to minutes and seconds formatted as a string.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0  
(1.0.0.0)

**Syntax**

```csharp
public static string GetCorrectedTimeValueString(
    float timeValue
)
```

**Parameters**

- `timeValue`
  - Type: `System.Single`
  - The time value input, for example: Time.time

**Return Value**

- Type: `String`
  - A formatted time string

**See Also**

**Reference**

- `ReplayTime Class`
- `UltimateReplay Namespace`
ReplayTime Properties

The ReplayTime type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![s] Delta</td>
<td>Represents a delta between current replay frames. This normalized value can be used to interpolate smoothly between replay states where a low record rate is used. Note: this value is not the actual delta time but a value representing the transition progress between replay frames.</td>
</tr>
<tr>
<td>![s] Time</td>
<td>Get the current replay playback time.</td>
</tr>
<tr>
<td>![s] TimeScale</td>
<td>The time scale value used during replay playback. This value is ignored during replay recording.</td>
</tr>
</tbody>
</table>

**See Also**

Reference

ReplayTime Class

UltimateReplay Namespace
ReplayTimeDelta Property

Represents a delta between current replay frames. This normalized value can be used to interpolate smoothly between replay states where a low record rate is used. Note: this value is not the actual delta time but a value representing the transition progress between replay frames.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```csharp
public static float Delta { get; }
```

**Property Value**

Type: Single

**See Also**

Reference  
ReplayTime Class  
UltimateReplay Namespace
ReplayTime Time Property

Get the current replay playback time.

Namespace: UltimateReplay
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

⚠️ Syntax

```c#
public static float Time { get; }
```

Property Value
Type: Single

⚠️ See Also

Reference
ReplayTime Class
UltimateReplay Namespace
ReplayTime TimeScale Property

The time scale value used during replay playback. This value is ignored during replay recording.

Namespace: UltimateReplay
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

Syntax

```
public static float TimeScale { get; set; }
```

Property Value
Type: Single

See Also

Reference
ReplayTime Class
UltimateReplay Namespace
ReplayTransform Class

Attach this component to a game objects in order to record the objects transform for replays. Only one instance of `ReplayTransform` can be added to any game object.

Inheritance Hierarchy

```
System
  Object
  Component
    Behaviour
      MonoBehaviour
      UltimateReplay
        ReplayBehaviour
          ReplayTransform
```

Namespace: UltimateReplay
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

Syntax

```c#
public class ReplayTransform : MonoBehaviour
```

The `ReplayTransform` type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ReplayTransform</code></td>
<td>Initializes a new instance of the <code>ReplayTransform</code> class</td>
</tr>
</tbody>
</table>

Top
## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Awake</strong></td>
<td>Called by Unity. (Overrides <code>ReplayBehaviourAwake</code>.)</td>
</tr>
<tr>
<td><strong>OnReplayDeserialize</strong></td>
<td>Called by the relay system when the object should return to a previous state. (Overrides <code>ReplayBehaviourOnReplayDeserialize</code>.)</td>
</tr>
<tr>
<td><strong>OnReplaySerialize</strong></td>
<td>Called by the replay system when the object should be recorded. (Overrides <code>ReplayBehaviourOnReplaySerialize</code>.)</td>
</tr>
<tr>
<td><strong>OnReplayUpdate</strong></td>
<td>Called during playback and allows the transform to be interpolated to provide a smooth replay even if lower record rates are used. (Overrides <code>ReplayBehaviourOnReplayUpdate</code>.)</td>
</tr>
</tbody>
</table>

## Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>interpolate</td>
<td>Should the transform be interpolated between states. This is recommended when low record rates are used as without interpolation the playback can seem jumpy.</td>
</tr>
<tr>
<td>recordPosition</td>
<td>Should the position value of the transform be recorded.</td>
</tr>
<tr>
<td>recordRotation</td>
<td>Should the rotation value of the transform be recorded.</td>
</tr>
<tr>
<td>recordScale</td>
<td>Should the scale value of the transform be recorded.</td>
</tr>
</tbody>
</table>

See Also

Reference
UltimateReplay Namespace
ReplayTransform Constructor

Initializes a new instance of the `ReplayTransform` class

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```csharp
public ReplayTransform()
```

**See Also**

Reference
- `ReplayTransform Class`
- `UltimateReplay Namespace`
# ReplayTransform Fields

The `ReplayTransform` type exposes the following members.

## Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>interpolate</code></td>
<td>Should the transform be interpolated between states. This is recommended when low record rates are used as without interpolation the playback can seem jumpy.</td>
</tr>
<tr>
<td><code>recordPosition</code></td>
<td>Should the position value of the transform be recorded.</td>
</tr>
<tr>
<td><code>recordRotation</code></td>
<td>Should the rotation value of the transform be recorded.</td>
</tr>
<tr>
<td><code>recordScale</code></td>
<td>Should the scale value of the transform be recorded.</td>
</tr>
</tbody>
</table>

## See Also

Reference
- `ReplayTransform Class`
- `UltimateReplay Namespace`
ReplayTransformInterpolate Field

Should the transform be interpolated between states. This is recommended when low record rates are used as without interpolation the playback can seem jumpy.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public bool interpolate
```

### Field Value

Type: **Boolean**

### See Also

Reference
- **ReplayTransform Class**
- **UltimateReplay Namespace**
ReplayTransformrecordPosition Field

Should the position value of the transform be recorded.

**Namespace**: UltimateReplay  
**Assembly**: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```c#
public bool recordPosition
```

**Field Value**

Type: Boolean

**See Also**

Reference

ReplayTransform Class
UltimateReplay Namespace
ReplayTransform recordRotation Field

Should the rotation value of the transform be recorded.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

▶ Syntax

```
public bool recordRotation
```

Field Value  
Type: **Boolean**

▶ See Also

Reference  
ReplayTransform Class  
UltimateReplay Namespace
ReplayTransform recordScale Field

Should the scale value of the transform be recorded.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```
public bool recordScale
```

### Field Value

Type: **Boolean**

### See Also

Reference

ReplayTransform Class  
UltimateReplay Namespace
The `ReplayTransform` type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Awake</strong></td>
<td>Called by Unity. (Overrides <code>ReplayBehaviourAwake</code>.)</td>
</tr>
<tr>
<td><strong>OnReplayDeserialize</strong></td>
<td>Called by the relay system when the object should return to a previous state. (Overrides <code>ReplayBehaviourOnReplayDeserialize</code>.)</td>
</tr>
<tr>
<td><strong>OnReplaySerialize</strong></td>
<td>Called by the replay system when the object should be recorded. (Overrides <code>ReplayBehaviourOnReplaySerialize</code>.)</td>
</tr>
<tr>
<td><strong>OnReplayUpdate</strong></td>
<td>Called during playback and allows the transform to be interpolated to provide a smooth replay even if lower record rates are used. (Overrides <code>ReplayBehaviourOnReplayUpdate</code>.)</td>
</tr>
</tbody>
</table>

Top

### See Also

Reference
- `ReplayTransform Class`
- `UltimateReplay Namespace`
ReplayTransformAwake Method

Called by Unity.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

▶ Syntax

```c#
public override void Awake()
```

▶ See Also

Reference  
ReplayTransform Class  
UltimateReplay Namespace
ReplayTransformOnReplayDeserialize Method

Called by the relay system when the object should return to a previous state.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public override void OnReplayDeserialize(ReplayState state)
```

### Parameters

- **state**  
  Type: UltimateReplay.ReplayState  
  The state object to deserialize the transform from

### Implements

IReplaySerializeOnReplayDeserialize(ReplayState)

### See Also

Reference  
ReplayTransform Class  
UltimateReplay Namespace
ReplayTransformOnReplaySerialize Method

Called by the replay system when the object should be recorded.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public override void OnReplaySerialize(
    ReplayState state
)
```

### Parameters

*state*

- Type: UltimateReplayReplayState  
  - The state object to serialize the transform into

### Implements

IReplaySerializeOnReplaySerialize(ReplayState)

### See Also

**Reference**  
ReplayTransform Class  
UltimateReplay Namespace
ReplayTransformOnReplayUpdate Method

Called during playback and allows the transform to be interpolated to provide a smooth replay even if lower record rates are used.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```
public override void OnReplayUpdate()
```

### See Also

Reference
- ReplayTransform Class
- UltimateReplay Namespace
ReplayVarAttribute Class

Use this attribute on a field to mark it for recording. The type the field is defined in must inherit from ReplayBehaviour in order for the field to be recorded automatically. Interpolation between field values is also possible where low record rates are used.

Inheritance Hierarchy

System Object System Attribute
  UltimateReplay ReplayVarAttribute

Namespace: UltimateReplay
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

Syntax

C#

```csharp
public sealed class ReplayVarAttribute : Attribute
```

The ReplayVarAttribute type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReplayVarAttribute</td>
<td>Create a new ReplayVarAttribute for a field.</td>
</tr>
</tbody>
</table>

Fields
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>interpolate</td>
<td>Should the value of the field be interpolated between frames or should the value snap to the exact frame value. Most built-in types support interpolation such as Byte and Single. Basic Unity types such as Vector2 and Color also support interpolation.</td>
</tr>
</tbody>
</table>

See Also

Reference
UltimateReplay Namespace
ReplayVarAttribute Constructor

Create a new ReplayVarAttribute for a field.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```c#
public ReplayVarAttribute(
    bool interpolated = true
)
```

**Parameters**

`interpolated` *(Optional)*
- Type: `SystemBoolean`  
  - Should the field value be interpolated between frames

**See Also**

Reference
- ReplayVarAttribute Class  
- UltimateReplay Namespace
ReplayVarAttribute Fields

The `ReplayVarAttribute` type exposes the following members.

### Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>interpolate</td>
<td>Should the value of the field be interpolated between frames or should the value snap to the exact frame value. Most built-in types support interpolation such as <code>Byte</code> and <code>Single</code>. Basic Unity types such as <code>Vector2</code> and <code>Color</code> also support interpolation.</td>
</tr>
</tbody>
</table>

### See Also

Reference
- `ReplayVarAttribute Class`
- `UltimateReplay Namespace`
ReplayVarAttribute interpolate Field

Should the value of the field be interpolated between frames or should the value snap to the exact frame value. Most built-in types support interpolation such as `Byte` and `Single`. Basic Unity types such as `Vector2` and `Color` also support interpolation.

**Namespace:** UltimateReplay  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```
c# Copy

public bool interpolate
```

### Field Value

Type: `Boolean`

### See Also

**Reference**  
ReplayVarAttribute Class  
UltimateReplay Namespace
# UltimateReplay.Scripting Reference

## UltimateReplay.Core Namespace

### Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DefaultReplayPreparer</td>
<td>The default IReplayPreparer used by Ultimate Replay to prepare game objects for gameplay and playback.</td>
</tr>
<tr>
<td>ReplayIdentity</td>
<td>A replay identity is an essential component in the Ultimate Replay system and is used to identify replay objects between sessions. Replay identities are assigned at edit time where possible and will never change values. Replay identities are also used to identify prefab instances that are spawned during a replay.</td>
</tr>
<tr>
<td>ReplayScene</td>
<td>A ReplayScene contains information about all active replay objects.</td>
</tr>
<tr>
<td>ReplayVariable</td>
<td>Represents a variable that can be recorded using the replay system in order to replay script animations or similar during playback.</td>
</tr>
</tbody>
</table>
### Structures

<table>
<thead>
<tr>
<th>Structure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReplayEvent</td>
<td>Used to mark key recording events that will be triggered during playback. Good candidates would be to trigger audio effects or similar.</td>
</tr>
</tbody>
</table>

### Interfaces

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IReplayPreparer</td>
<td>A preparer is used by Ultimate Replay to prepare any replay objects for either gameplay mode or playback mode. In order for game systems such as physics and scripts to not affect playback, replay objects must be prepared in some way to disable these systems while playback is enabled. The appropriate prepare method will be called by the replay system when objects need to either enter playback mode or return to gameplay mode.</td>
</tr>
<tr>
<td>IReplaySerialize</td>
<td>This class should be implemented when you want to serialize custom replay data. This could really be an interface but it needs to be a class to be assignable in the inspector.</td>
</tr>
</tbody>
</table>

### Enumerations
<table>
<thead>
<tr>
<th>Enumeration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ReplayEvents</strong></td>
<td>Common events identifiers used to record ReplayEvent with the replay system.</td>
</tr>
</tbody>
</table>
DefaultReplayPreparer Class

The default IReplayPreparer used by Ultimate Replay to prepare game objects for gameplay and playback.

### Inheritance Hierarchy

System\Object  UltimateReplay.Core\DefaultReplayPreparer

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```c#
public class DefaultReplayPreparer : IReplayPreparer
```

The DefaultReplayPreparer type exposes the following members.

### Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DefaultReplayPreparer</td>
<td>Initializes a new instance of the DefaultReplayPreparer class</td>
</tr>
</tbody>
</table>

### Methods
<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PrepareComponentForGameplay</strong></td>
<td>Prepare the specified component for gameplay mode.</td>
</tr>
<tr>
<td><strong>PrepareComponentForPlayback</strong></td>
<td>Prepare the specified component for playback mode.</td>
</tr>
<tr>
<td><strong>PrepareForGameplay</strong></td>
<td>Prepare the specified replay object for gameplay mode.</td>
</tr>
<tr>
<td><strong>PrepareForPlayback</strong></td>
<td>Prepare the specified replay object for playback mode.</td>
</tr>
</tbody>
</table>

**See Also**

**Reference**

*UltimateReplay.Core Namespace*
DefaultReplayPreparer Constructor

Initializes a new instance of the DefaultReplayPreparer class

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

## Syntax

```c#
public DefaultReplayPreparer()
```

## See Also

**Reference**
- DefaultReplayPreparer Class  
- UltimateReplay.Core Namespace
DefaultReplayPreparer Methods

The DefaultReplayPreparer type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PrepareComponentForGameplay</td>
<td>Prepare the specified component for gameplay mode.</td>
</tr>
<tr>
<td>PrepareComponentForPlayback</td>
<td>Prepare the specified component for playback mode.</td>
</tr>
<tr>
<td>PrepareForGameplay</td>
<td>Prepare the specified replay object for gameplay mode.</td>
</tr>
<tr>
<td>PrepareForPlayback</td>
<td>Prepare the specified replay object for playback mode.</td>
</tr>
</tbody>
</table>
DefaultReplayPreparerPrepareComponentForGameplay Method

Prepare the specified component for gameplay mode.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public virtual void PrepareComponentForGameplay(
  Component component
)
```

### Parameters

- **component**
  - Type: `Component`
  - The component to prepare

### See Also

Reference  
DefaultReplayPreparer Class  
UltimateReplay.Core Namespace
DefaultReplayPreparerPrepareComponentForPlayback Method

Prepare the specified component for playback mode.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public virtual void PrepareComponentForPlayback(
  Component component
)
```

### Parameters

**component**

Type: **Component**  
The component to prepare

### See Also

Reference
- DefaultReplayPreparer Class
- UltimateReplay.Core Namespace
DefaultReplayPreparerPrepareForGameplay Method

Prepare the specified replay object for gameplay mode.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```csharp
public virtual void PrepareForGameplay(
    ReplayObject replayObject
)
```

**Parameters**

- **replayObject**  
  - Type: UltimateReplay.ReplayObject  
  - The replay object to prepare

**Implements**

- IReplayPreparerPrepareForGameplay(ReplayObject)

**See Also**

- Reference
- DefaultReplayPreparer Class
- UltimateReplay.Core Namespace
DefaultReplayPreparerPrepareForPlayback Method

Prepare the specified replay object for playback mode.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```csharp
public virtual void PrepareForPlayback(  
    ReplayObject replayObject  
)
```

**Parameters**

*replayObject*

Type: UltimateReplay.ReplayObject  
The replay object to prepare

**Implements**

IReplayPreparerPrepareForPlayback(ReplayObject)

**See Also**

Reference  
DefaultReplayPreparer Class  
UltimateReplay.Core Namespace
IReplayPreparer Interface

A preparer is used by Ultimate Replay to prepare any replay objects for either gameplay mode or playback mode. In order for game systems such as physics and scripts to not affect playback, replay objects must be prepared in some way to disable these systems while playback is enabled. The appropriate prepare method will be called by the replay system when objects need to either enter playback mode or return to gameplay mode.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0

### Syntax

```c#
public interface IReplayPreparer
```

The **IReplayPreparer** type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PrepareForGameplay</td>
<td>Prepares the specified replay object for gameplay. The implementing type should restore all game systems that affect the replay object so that the object is in its original state. This method will be called for each replay object that must be prepared.</td>
</tr>
</tbody>
</table>
PrepareForPlayback

Prepares the specified replay object for playback. The implementing type should ensure that all game systems likely to affect the replay object during playback are suitable disabled in order to avoid glitching or unpredicted behaviour. This method will be called for each replay object that must be prepared.

See Also

Reference
UltimateReplay.Core Namespace
IReplayPreparer Methods

The IReplayPreparer type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PrepareForGameplay</td>
<td>Prepares the specified replay object for gameplay. The implementing type should restore all game systems that affect the replay object so that the object is in its original state. This method will be called for each replay object that must be prepared.</td>
</tr>
<tr>
<td>PrepareForPlayback</td>
<td>Prepares the specified replay object for playback. The implementing type should ensure that all game systems likely to affect the replay object during playback are suitable disabled in order to avoid glitching or unpredicted behaviour. This method will be called for each replay object that must be prepared.</td>
</tr>
</tbody>
</table>

See Also

Top
Reference
IReplayPreparer Interface
UltimateReplay.Core Namespace
IReplayPreparerPrepareForGameplay Method

Prepares the specified replay object for gameplay. The implementing type should restore all game systems that affect the replay object so that the object is in its original state. This method will be called for each replay object that must be prepared.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```csharp
void PrepareForGameplay(
    ReplayObject replayObject
)
```

**Parameters**

`replayObject`  
Type: UltimateReplay.ReplayObject  
The replay object to prepare

**See Also**

Reference  
IReplayPreparer Interface  
UltimateReplay.Core Namespace
IReplayPreparerPrepareForPlayback Method

Prepares the specified replay object for playback. The implementing type should ensure that all game systems likely to affect the replay object during playback are suitable disabled in order to avoid glitching or unpredicted behaviour. This method will be called for each replay object that must be prepared.

Namespace: UltimateReplay.Core
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0

Syntax

```c#
void PrepareForPlayback(
    ReplayObject replayObject
)
```

Parameters

replayObject
- Type: UltimateReplay.ReplayObject
  - The replay object that should be prepared

See Also

Reference
IReplayPreparer Interface
UltimateReplay.Core Namespace
IReplaySerialize Interface

This class should be implemented when you want to serialize custom replay data. This should really be an interface but it needs to be a class to be assignable in the inspector.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public interface IReplaySerialize
```

The `IReplaySerialize` type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnReplayDeserialize</td>
<td>Called by the replay system when all replay state data should be deserialized.</td>
</tr>
<tr>
<td>OnReplaySerialize</td>
<td>Called by the replay system when all replay state data should be serialized.</td>
</tr>
</tbody>
</table>

**See Also**

Reference
UltimateReplay.Core Namespace
The IReplaySerialize type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnReplayDeserialize</td>
<td>Called by the replay system when all replay state data should be deserialized.</td>
</tr>
<tr>
<td>OnReplaySerialize</td>
<td>Called by the replay system when all replay state data should be serialized.</td>
</tr>
</tbody>
</table>

See Also

Reference

IReplaySerialize Interface
UltimateReplay.Core Namespace
IReplaySerializeOnReplayDeserialize Method

Called by the replay system when all replay state data should be deserialized.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```csharp
void OnReplayDeserialize(
    ReplayState state
)
```

**Parameters**

`state`  
Type: UltimateReplay.ReplayState  
The `ReplayState` to read the data from

**See Also**

Reference  
IReplaySerialize Interface  
UltimateReplay.Core Namespace
IReplaySerialize

OnReplaySerialize Method

Called by the replay system when all replay state data should be serialized.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```csharp
void OnReplaySerialize(
    ReplayState state
)
```

**Parameters**

*state*
  
  Type: UltimateReplay.ReplayState  
  The ReplayState to write the data to

**See Also**

Reference
  
  IReplaySerialize Interface  
  UltimateReplay.Core Namespace
ReplayEvent Structure

Used to mark key recording events that will be triggered during playback. Good candidates would be to trigger audio effects or similar.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

## Syntax

```csharp
public struct ReplayEvent : IReplaySerialize
```

The `ReplayEvent` type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![arrow] OnReplayDeserialize</td>
<td>Called by the replay system when all replay state information should be deserialized.</td>
</tr>
<tr>
<td>![arrow] OnReplaySerialize</td>
<td>Called by the replay system when all replay state information should be serialized.</td>
</tr>
</tbody>
</table>

## Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>eventData</strong></td>
<td>The replay state data associated with this <strong>ReplayEvent</strong>. The event data should contain no more than 255 bytes to ensure that the data is serialized correctly.</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>eventID</strong></td>
<td>A unique event identifier used to distinguish between different replay events.</td>
</tr>
</tbody>
</table>

**See Also**

Reference

**UltimateReplay.Core Namespace**
ReplayEvent Fields

The **ReplayEvent** type exposes the following members.

### Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>eventData</strong></td>
<td>The replay state data associated with this <strong>ReplayEvent</strong>. The event data should contain no more than 255 bytes to ensure that the data is serialized correctly.</td>
</tr>
<tr>
<td><strong>eventID</strong></td>
<td>A unique event identifier used to distinguish between different replay events.</td>
</tr>
</tbody>
</table>

**Top**

**See Also**

Reference
- **ReplayEvent Structure**
- **UltimateReplay.Core Namespace**
ReplayEvent.eventData Field

The replay state data associated with this ReplayEvent. The event data should contain no more than 255 bytes to ensure that the data is serialized correctly.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public ReplayState eventData
```

### Field Value

Type: ReplayState

### See Also

Reference

- ReplayEvent Structure
- UltimateReplay.Core Namespace
ReplayEventeventID Field

A unique event identifier used to distinguish between different replay events.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```c#
public byte eventID
```

### Field Value

Type: **Byte**

### See Also

Reference

- **ReplayEvent Structure**
- **UltimateReplay.Core Namespace**
ReplayEvent Methods

The `ReplayEvent` type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>OnReplayDeserialize</code></td>
<td>Called by the replay system when all replay state information should be deserialized.</td>
</tr>
<tr>
<td><code>OnReplaySerialize</code></td>
<td>Called by the replay system when all replay state information should be serialized.</td>
</tr>
</tbody>
</table>

See Also

Reference
- `ReplayEvent Structure`
- `UltimateReplay.Core Namespace`
ReplayEventOnReplayDeserialize Method

Called by the replay system when all replay state information should be deserialized.

Namespace: UltimateReplay.Core
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

Syntax

```csharp
public void OnReplayDeserialize(
    ReplayState state
)
```

Parameters

- **state**
  - Type: UltimateReplay.ReplayState
  - The ReplayState to read the data from

Implements

IReplaySerializeOnReplayDeserialize(ReplayState)

See Also

Reference
ReplayEvent Structure
UltimateReplay.Core Namespace
ReplayEventOnReplaySerialize Method

Called by the replay system when all replay state information should be serialized.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public void OnReplaySerialize(
    ReplayState state
)
```

### Parameters

**state**
Type: UltimateReplay.ReplayState  
The ReplayState to write the data to

### Implements

IReplaySerializeOnReplaySerialize(ReplayState)

### See Also

Reference
- ReplayEvent Structure  
- UltimateReplay.Core Namespace
ReplayEvents Enumeration

Common events identifiers used to record ReplayEvent with the replay system.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

⚠️ Syntax

```c#
public enum ReplayEvents
```

⚠️ Members

<table>
<thead>
<tr>
<th>Member name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ObjectSpawn</td>
<td>1</td>
<td>Object instantiated event.</td>
</tr>
<tr>
<td>ObjectDespawn</td>
<td>2</td>
<td>Object destroyed event.</td>
</tr>
<tr>
<td>PlaySound</td>
<td>3</td>
<td>Play audio clip event.</td>
</tr>
<tr>
<td>PlayMusic</td>
<td>4</td>
<td>Play audio music event.</td>
</tr>
<tr>
<td>ParticleStart</td>
<td>5</td>
<td>Start particle system event.</td>
</tr>
<tr>
<td>ParticleEnd</td>
<td>6</td>
<td>Stop particle system event.</td>
</tr>
</tbody>
</table>

⚠️ See Also

Reference  
UltimateReplay.Core Namespace
ReplayIdentity Class

A replay identity is an essential component in the Ultimate Replay system and is used to identify replay objects between sessions. Replay identities are assigned at edit time where possible and will never change values. Replay identities are also used to identify prefab instances that are spawned during a replay.

Inheritance Hierarchy

System\Object  UltimateReplay.Core\ReplayIdentity

Namespace: UltimateReplay.Core
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

Syntax

```csharp
[SerializableAttribute]
public sealed class ReplayIdentity : IEquatable<ReplayIdentity>
```

The `ReplayIdentity` type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![ReplayIdentity]</td>
<td>Create a new <code>ReplayIdentity</code>.</td>
</tr>
</tbody>
</table>

Top

Methods
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals(Object)</td>
<td>Override implementation. (Overrides Object.Equals(Object).)</td>
</tr>
<tr>
<td>Equals(ReplayIdentity)</td>
<td>IEquateable implementation.</td>
</tr>
<tr>
<td>Generate</td>
<td>Generates a unique ReplayIdentity.</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Override implementation. (Overrides Object.GetHashCode.)</td>
</tr>
<tr>
<td>IsUnique(Int32)</td>
<td>Returns true if the specified id is unique or false if not.</td>
</tr>
<tr>
<td>IsUnique(ReplayIdentity)</td>
<td>Returns true if the specified ReplayIdentity is unique or false if not.</td>
</tr>
<tr>
<td>ToString</td>
<td>Override implementation. (Overrides Object.ToString.)</td>
</tr>
</tbody>
</table>

**Top**

**Operators**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equality</td>
<td>Override equals operator.</td>
</tr>
<tr>
<td>(Int16 to ReplayIdentity)</td>
<td>Implicit short conversion (16 bit identity only).</td>
</tr>
<tr>
<td>(ReplayIdentity to Int16)</td>
<td>Implicit int conversion (32 bit identity only).</td>
</tr>
</tbody>
</table>
Inequality Override not-equals operator.

### Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>byteSize</code></td>
<td>Get the size in bytes of a <code>ReplayIdentity</code> representation.</td>
</tr>
<tr>
<td><code>unassignedIdentity</code></td>
<td>Get the default value for a <code>ReplayIdentity</code> id which equates to an unassigned identity.</td>
</tr>
</tbody>
</table>

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>IsAssigned</code></td>
<td>Returns true if the identity has been generated or false if it has not.</td>
</tr>
</tbody>
</table>

### See Also

Reference

*UltimateReplay.Core Namespace*
ReplayIdentity Constructor

Create a new ReplayIdentity.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```c#
public ReplayIdentity()
```

### See Also

Reference
- ReplayIdentity Class
- UltimateReplay.Core Namespace
ReplayIdentity Fields

The `ReplayIdentity` type exposes the following members.

### Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>byteSize</code></td>
<td>Get the size in bytes of a <code>ReplayIdentity</code> representation.</td>
</tr>
<tr>
<td><code>unassignedIdentity</code></td>
<td>Get the default value for a <code>ReplayIdentity</code> id which equates to an unassigned identity.</td>
</tr>
</tbody>
</table>

See Also

Reference

*ReplayIdentity Class*

*UltimateReplay.Core Namespace*
ReplayIdentity.byteSize Field

Get the size in bytes of a ReplayIdentity representation.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```
public static readonly int byteSize
```

### Field Value

Type: `Int32`

### See Also

Reference  
ReplayIdentity Class  
UltimateReplay.Core Namespace
ReplayIdentity\unassignedIdentity Field

Get the default value for a ReplayIdentity id which equates to an unassigned identity.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

## Syntax

```c#
public const int unassignedIdentity
```

## Field Value

Type: Int32

## See Also

Reference  
ReplayIdentity Class  
UltimateReplay.Core Namespace
ReplayIdentity Methods

The ReplayIdentity type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals(Object)</td>
<td>Override implementation. (Overrides Object.Equals(Object).)</td>
</tr>
<tr>
<td>Equals(ReplayIdentity)</td>
<td>IEquatable implementation.</td>
</tr>
<tr>
<td>Generate</td>
<td>Generates a unique ReplayIdentity.</td>
</tr>
<tr>
<td>GetHashCode</td>
<td>Override implementation. (Overrides Object.GetHashCode.)</td>
</tr>
<tr>
<td>IsUnique(Int32)</td>
<td>Returns true if the specified id is unique or false if not.</td>
</tr>
<tr>
<td>IsUnique(ReplayIdentity)</td>
<td>Returns true if the specified ReplayIdentity is unique or false if not.</td>
</tr>
<tr>
<td>ToString</td>
<td>Override implementation. (Overrides Object.ToString.)</td>
</tr>
</tbody>
</table>

See Also
Reference
ReplayIdentity Class
UltimateReplay.Core Namespace
ReplayIdentity Equals Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equals(Object)</td>
<td>Override implementation. (Overrides ObjectEquals(Object).)</td>
</tr>
<tr>
<td>Equals(ReplayIdentity)</td>
<td>IEquateable implementation.</td>
</tr>
</tbody>
</table>

See Also

Reference

ReplayIdentity Class
UltimateReplay.Core Namespace
ReplayIdentity Equals Method (Object)

Override implementation.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```csharp
public override bool Equals(
    Object obj
)
```

**Parameters**

`obj`  
Type: `SystemObject`  
The object to compare against

**Return Value**

Type: `Boolean`  


**See Also**

Reference  
`ReplayIdentity Class`  
`Equals Overload`  
`UltimateReplay.Core Namespace`
ReplayIdentity Equals Method (ReplayIdentity)

IEquatable implementation.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll)  
**Version:** 1.0.0.0 (1.0.0.0)

▲ Syntax

```c#
public bool Equals(  
    ReplayIdentity obj
)
```

▲ Parameters

**obj**
Type: UltimateReplay.Core.ReplayIdentity  
The ReplayIdentity to compare against

▲ Return Value

Type: Boolean  

▲ Implements

IEquatableTEquals(T)

▲ See Also

Reference  
ReplayIdentity Class  
Equals Overload
UltimateReplay.Core Namespace
ReplayIdentity.Generate Method

Generates a unique ReplayIdentity.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public void Generate()
```

### See Also

**Reference**
- **ReplayIdentity Class**
- **UltimateReplay.Core Namespace**
ReplayIdentity GetHashCode Method

Override implementation.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public override int GetHashCode()
```

### Return Value

**Type:** Int32  

### See Also

**Reference**  
ReplayIdentity Class  
UltimateReplay.Core Namespace
ReplayIdentityIsUnique Method

Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>s IsUnique(Int32)</td>
<td>Returns true if the specified id is unique or false if not.</td>
</tr>
<tr>
<td>s IsUnique(ReplayIdentity)</td>
<td>Returns true if the specified ReplayIdentity is unique or false if not.</td>
</tr>
</tbody>
</table>

See Also

Reference
ReplayIdentity Class
UltimateReplay.Core Namespace
ReplayIdentity.IsUnique Method (Int32)

Returns true if the specified id is unique or false if not.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

## Syntax

```csharp
public static bool IsUnique(
    int id
)
```

### Parameters

**id**
- Type: System.Int32  
The int id to check

### Return Value

Type: Boolean  


## See Also

**Reference**
- ReplayIdentity Class  
- IsUnique Overload  
- UltimateReplay.Core Namespace
ReplayIdentityIsUnique Method (ReplayIdentity)

Returns true if the specified ReplayIdentity is unique or false if not.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```csharp
public static bool IsUnique(  
    ReplayIdentity id  
)
```

**Parameters**

*id*

Type: UltimateReplay.Core.ReplayIdentity  
The identity to check

**Return Value**

Type: Boolean


**See Also**

Reference

ReplayIdentity Class  
IsUnique Overload  
UltimateReplay.Core Namespace
ReplayIdentityToString Method

Override implementation.

Namespace: UltimateReplay.Core
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

Syntax

```c#
public override string ToString()
```

Return Value

Type: String


See Also

Reference

ReplayIdentity Class
UltimateReplay.Core Namespace
ReplayIdentity Operators and Type Conversions

The ReplayIdentity type exposes the following members.

Operators

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equality</td>
<td>Override equals operator.</td>
</tr>
<tr>
<td>(Int16 to ReplayIdentity)</td>
<td>Implicit short conversion (16 bit identity only).</td>
</tr>
<tr>
<td>(ReplayIdentity to Int16)</td>
<td>Implicit int conversion (32 bit identity only).</td>
</tr>
<tr>
<td>Inequality</td>
<td>Override not-equals operator.</td>
</tr>
</tbody>
</table>

See Also

Reference
ReplayIdentity Class
UltimateReplay.Core Namespace
ReplayIdentityEquality Operator

Override equals operator.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

## Syntax

```csharp
public static bool operator ==(  
    ReplayIdentity a,  
    ReplayIdentity b  
)
```

### Parameters

**a**  
Type: UltimateReplay.Core.ReplayIdentity  
First ReplayIdentity

**b**  
Type: UltimateReplay.Core.ReplayIdentity  
Second ReplayIdentity

### Return Value

Type: Boolean

[Missing <returns> documentation for  

## See Also

Reference  
ReplayIdentity Class  
UltimateReplay.Core Namespace
## ReplayIdentity Conversion Operators

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![ReplayIdentity to Int16] ![ReplayIdentity] ![Int16]</td>
<td>Implicit int conversion (32 bit identity only).</td>
</tr>
</tbody>
</table>

### See Also

- [ReplayIdentity Class](#)
- [UltimateReplay.Core Namespace](#)
ReplayIdentity Conversion (Int16 to ReplayIdentity)

Implicit short conversion (16 bit identity only).

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public static implicit operator ReplayIdentity (short identity)
```

### Parameters

- **identity**
  - Type: `System.Int16`
  - The identity to convert

### Return Value

- Type: `ReplayIdentity`


### See Also

- Reference
  - ReplayIdentity Class
  - Overload
  - UltimateReplay.Core Namespace
ReplayIdentity Conversion (ReplayIdentity to Int16)

Implicit int conversion (32 bit identity only).

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```
public static implicit operator short (  
    ReplayIdentity identity  
)
```

**Parameters**

- **identity**
  - Type: UltimateReplay.Core.ReplayIdentity
  - The identity to convert

**Return Value**

- Type: Int16

**See Also**

**Reference**

- ReplayIdentity Class
- Overload
- UltimateReplay.Core Namespace
ReplayIdentityInequality Operator

Override not-equals operator.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

⚠️ Syntax

```c#
public static bool operator !=(
    ReplayIdentity a,
    ReplayIdentity b
)
```

Parameters

- **a**
  - Type: UltimateReplay.Core.ReplayIdentity  
  - First ReplayIdentity

- **b**
  - Type: UltimateReplay.Core.ReplayIdentity  
  - Second ReplayIdentity

Return Value

Type: Boolean

[Missing <returns> documentation for  

⚠️ See Also

Reference

ReplayIdentity Class  
UltimateReplay.Core Namespace
ReplayIdentity Properties

The ReplayIdentity type exposes the following members.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IsAssigned</td>
<td>Returns true if the identity has been generated or false if it has not.</td>
</tr>
</tbody>
</table>

See Also

Reference

ReplayIdentity Class
UltimateReplay.Core Namespace
ReplayIdentity.IsAssigned Property

Returns true if the identity has been generated or false if it has not.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0  
(1.0.0.0)

▲ Syntax

```
public bool IsAssigned { get; }
```

**Property Value**

Type: Boolean

▲ See Also

Reference
ReplayIdentity Class
UltimateReplay.Core Namespace
ReplayScene Class

A ReplayScene contains information about all active replay objects.

⚠️ Inheritance Hierarchy

System
Object
UltimateReplay.Core
ReplayScene

Namespace: UltimateReplay.Core
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

⚠️ Syntax

C#

```csharp
public sealed class ReplayScene
```

The ReplayScene type exposes the following members.

⚠️ Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>⚠️ ReplayScene</td>
<td>Initializes a new instance of the ReplayScene class</td>
</tr>
</tbody>
</table>

⚠️ Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>⚠️ RecordSnapshot</td>
<td>Take a snapshot of the current replay scene using the</td>
</tr>
</tbody>
</table>
specified timestamp.

- **RegisterReplayObject**
  Registers a replay object with the replay system so that it can be recorded for playback. Typically all `ReplayObject` will auto register when they 'Awake' meaning that you will not need to manually register objects.

- **RestoreSnapshot**
  Restore the scene to the state described by the specified snapshot.

- **UnregisterReplayObject**
  Unregisters a replay object from the replay system so that it will no longer be recorded for playback. Typically all `ReplayObject` will auto un-register when they are destroyed so you will normally not need to un-register a replay object.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActiveReplayObjects</td>
<td>Get a collection of all game objects that are registered with the replay system.</td>
</tr>
<tr>
<td>ReplayEnabled</td>
<td>Enable or disable the replay scene in preparation for playback or live mode. When true, all</td>
</tr>
</tbody>
</table>
replay objects will be prepared for playback causing certain components or scripts to be disabled to prevent interference from game systems. A prime candidate would be the Rigidbody component which could cause a replay object to be affected by gravity and as a result deviate from its intended position. When false, all replay objects will be returned to their 'Live' state when all game systems will be reactivated.

See Also

Reference
UltimateReplay.Core Namespace
ReplayScene Constructor

Initializes a new instance of the `ReplayScene` class

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

⚠️ Syntax

```c#
public ReplayScene()
```

⚠️ See Also

Reference  
`ReplayScene Class`  
`UltimateReplay.Core Namespace`
# ReplayScene Methods

The `ReplayScene` type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>RecordSnapshot</code></td>
<td>Take a snapshot of the current replay scene using the specified timestamp.</td>
</tr>
<tr>
<td><code>RegisterReplayObject</code></td>
<td>Registers a replay object with the replay system so that it can be recorded for playback. Typically all <code>ReplayObject</code> will auto register when they 'Awake' meaning that you will not need to manually register objects.</td>
</tr>
<tr>
<td><code>RestoreSnapshot</code></td>
<td>Restore the scene to the state described by the specified snapshot.</td>
</tr>
<tr>
<td><code>UnregisterReplayObject</code></td>
<td>Unregisters a replay object from the replay system so that it will no longer be recorded for playback. Typically all <code>ReplayObject</code> will auto unregister when they are destroyed so you will normally not need to un-register a replay object.</td>
</tr>
</tbody>
</table>
See Also

Reference
ReplayScene Class
UltimateReplay.Core Namespace
ReplaySceneRecordSnapshot Method

Take a snapshot of the current replay scene using the specified timestamp.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```csharp
public ReplaySnapshot RecordSnapshot(float timeStap)
```

**Parameters**

`timeStamp`
- Type: **System.Single**
  - The timestamp for the frame indicating its position in the playback sequence

**Return Value**

- Type: **ReplaySnapshot**
  - A new snapshot of the current replay scene

**See Also**

- Reference
  - ReplayScene Class
  - UltimateReplay.Core Namespace
ReplaySceneRegisterReplayObject Method

Registers a replay object with the replay system so that it can be recorded for playback. Typically all ReplayObject will auto register when they 'Awake' meaning that you will not need to manually register objects.

Namespace: UltimateReplay.Core
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

Syntax

```
public void RegisterReplayObject(
    ReplayObject replayObject
)
```

Parameters

`replayObject`
Type: UltimateReplay.ReplayObject
The ReplayObject to register

See Also

Reference
ReplayScene Class
UltimateReplay.Core Namespace
ReplaySceneRestoreSnapshot Method

Restore the scene to the state described by the specified snapshot.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

▶ Syntax

```csharp
public void RestoreSnapshot(
    ReplaySnapshot snapshot
)
```

Parameters

**snapshot**  
Type: UltimateReplay.Storage.ReplaySnapshot  
The snapshot to restore

▶ See Also

Reference  
ReplayScene Class  
UltimateReplay.Core Namespace
ReplaySceneUnregisterReplayObject Method

Unregisters a replay object from the replay system so that it will no longer be recorded for playback. Typically all ReplayObject will auto un-register when they are destroyed so you will normally not need to un-register a replay object.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```csharp
public void UnregisterReplayObject(
    ReplayObject replayObject
)
```

**Parameters**

*replayObject*  
Type: UltimateReplay.ReplayObject

**See Also**

Reference  
ReplayScene Class  
UltimateReplay.Core Namespace
ReplayScene Properties

The *ReplayScene* type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActiveReplayObjects</td>
<td>Get a collection of all game objects that are registered with the replay system.</td>
</tr>
<tr>
<td>ReplayEnabled</td>
<td>Enable or disable the replay scene in preparation for playback or live mode. When true, all replay objects will be prepared for playback causing certain components or scripts to be disabled to prevent interference from game systems. A prime candidate would be the RigidBody component which could cause a replay object to be affected by gravity and as a result deviate from its intended position. When false, all replay objects will be returned to their 'Live' state when all game systems will be reactivated.</td>
</tr>
</tbody>
</table>
Reference
ReplayScene Class
UltimateReplay.Core Namespace
ReplaySceneActiveReplayObjects Property

Get a collection of all game objects that are registered with the replay system.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public HashSet<ReplayObject> ActiveReplayObjects {
}
```

### Property Value

**Type:** HashSet<ReplayObject>

### See Also

Reference
- ReplayScene Class
- UltimateReplay.Core Namespace
ReplayScene.ReplayEnabled Property

Enable or disable the replay scene in preparation for playback or live mode. When true, all replay objects will be prepared for playback causing certain components or scripts to be disabled to prevent interference from game systems. A prime candidate would be the RigidBody component which could cause a replay object to be affected by gravity and as a result deviate from its intended position. When false, all replay objects will be returned to their 'Live' state when all game systems will be reactivated.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public bool ReplayEnabled { get; set; }
```

### Property Value

Type: Boolean

### See Also

Reference  
ReplayScene Class  
UltimateReplay.Core Namespace
ReplayVariable Class

Represents a variable that can be recorded using the replay system in order to replay script animations or similar during playback.

Inheritance Hierarchy

System\Object  UltimateReplay.Core\ReplayVariable

Namespace: UltimateReplay.Core
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0

Syntax

```csharp
public sealed class ReplayVariable : IReplaySerializable.
```

The ReplayVariable type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReplayVariable</td>
<td>Create a new ReplayVariable.</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanInterpolate</td>
<td>Returns true if the</td>
</tr>
</tbody>
</table>
specified type can be interpolated by the replay system.

<table>
<thead>
<tr>
<th>Interpolate</th>
<th>Attempts to interpolate the ReplayVariable value using the values from the last and next frame.</th>
</tr>
</thead>
<tbody>
<tr>
<td>InterpolateByte</td>
<td>Default interpolator for byte.</td>
</tr>
<tr>
<td>InterpolateColor</td>
<td>Default interpolator for Color.</td>
</tr>
<tr>
<td>InterpolateColor32</td>
<td>Default interpolator for Color32.</td>
</tr>
<tr>
<td>InterpolateDouble</td>
<td>Default interpolator for double.</td>
</tr>
<tr>
<td>InterpolateFloat</td>
<td>Default interpolator for float.</td>
</tr>
<tr>
<td>InterpolateInt</td>
<td>Default interpolator for int.</td>
</tr>
<tr>
<td>InterpolateLong</td>
<td>Default interpolator for long.</td>
</tr>
<tr>
<td>InterpolateQuat</td>
<td>Default interpolator for Quaternion.</td>
</tr>
<tr>
<td>InterpolateShort</td>
<td>Default interpolator for short.</td>
</tr>
<tr>
<td>InterpolateValue</td>
<td>Attempts to interpolate the ReplayVariable value</td>
</tr>
</tbody>
</table>
using the values from the last and next frame. In order for interpolation to succeed, the last and next values must be of the same type.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InterpolateVec2</td>
<td>Default interpolator for Vector2.</td>
</tr>
<tr>
<td>InterpolateVec3</td>
<td>Default interpolator for Vector3.</td>
</tr>
<tr>
<td>InterpolateVec4</td>
<td>Default interpolator for Vector4.</td>
</tr>
<tr>
<td>OnReplayDeserialize</td>
<td>Called by the replay system when the variable should be deserialized.</td>
</tr>
<tr>
<td>OnReplaySerialize</td>
<td>Called by the replay system when the variable should be serialized.</td>
</tr>
<tr>
<td>RegisterCustomInterpolatorT</td>
<td>Allows a custom interpolation method to be registered so that unsupported variable types can be interpolated automatically.</td>
</tr>
<tr>
<td>UpdateValueRange</td>
<td>Sets the current interpolation range for the ReplayVariable value.</td>
</tr>
</tbody>
</table>
## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute</td>
<td>Get the <code>ReplayVarAttribute</code> associated with this <code>ReplayVariable</code>.</td>
</tr>
<tr>
<td>gameObject</td>
<td>Get the game object that this <code>ReplayVariable</code> is attached to.</td>
</tr>
<tr>
<td>IsInterpolated</td>
<td>Returns true if this <code>ReplayVariable</code> should be interpolated between frames.</td>
</tr>
<tr>
<td>IsInterpolationSupported</td>
<td>Returns true if this <code>ReplayVariable</code> supports interpolation. Interpolation can only be supported if the variable type has a registered interpolator.</td>
</tr>
<tr>
<td>Name</td>
<td>Get the name of this <code>ReplayVariable</code>.</td>
</tr>
<tr>
<td>Value</td>
<td>The current value for this <code>ReplayVariable</code>.</td>
</tr>
</tbody>
</table>
ReplayVariable Constructor

Create a new ReplayVariable.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

▲ Syntax

```csharp
public ReplayVariable(
    ReplayBehaviour owner,
    FieldInfo field,
    ReplayVarAttribute attribute
)
```

**Parameters**

`owner`
- Type: UltimateReplay.ReplayBehaviour  
  The ReplayBehaviour that this ReplayVariable is defined in

`field`
- Type: System.Reflection.FieldInfo  
  The field info for the variable field

`attribute`
- Type: UltimateReplay.ReplayVarAttribute  
  The ReplayVarAttribute for the field

▲ See Also

Reference  
ReplayVariable Class  
UltimateReplay.Core Namespace
ReplayVariable Methods

The `ReplayVariable` type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CanInterpolate</td>
<td>Returns true if the specified type can be interpolated by the replay system.</td>
</tr>
<tr>
<td>Interpolate</td>
<td>Attempts to interpolate the <code>ReplayVariable</code> value using the values from the last and next frame.</td>
</tr>
<tr>
<td>InterpolateByte</td>
<td>Default interpolator for byte.</td>
</tr>
<tr>
<td>InterpolateColor</td>
<td>Default interpolator for Color.</td>
</tr>
<tr>
<td>InterpolateColor32</td>
<td>Default interpolator for Color32.</td>
</tr>
<tr>
<td>InterpolateDouble</td>
<td>Default interpolator for double.</td>
</tr>
<tr>
<td>InterpolateFloat</td>
<td>Default interpolator for float.</td>
</tr>
<tr>
<td>InterpolateInt</td>
<td>Default interpolator for int.</td>
</tr>
<tr>
<td><strong>s</strong></td>
<td>InterpolateLong</td>
</tr>
<tr>
<td><strong>s</strong></td>
<td>InterpolateQuat</td>
</tr>
<tr>
<td><strong>s</strong></td>
<td>InterpolateShort</td>
</tr>
<tr>
<td><strong>s</strong></td>
<td>InterpolateValue</td>
</tr>
<tr>
<td><strong>s</strong></td>
<td>InterpolateVec2</td>
</tr>
<tr>
<td><strong>s</strong></td>
<td>InterpolateVec3</td>
</tr>
<tr>
<td><strong>s</strong></td>
<td>InterpolateVec4</td>
</tr>
<tr>
<td></td>
<td>OnReplayDeserialize</td>
</tr>
<tr>
<td></td>
<td>OnReplaySerialize</td>
</tr>
<tr>
<td></td>
<td><strong>RegisterCustomInterpolatorT</strong></td>
</tr>
<tr>
<td>---</td>
<td>-------------------------------</td>
</tr>
<tr>
<td></td>
<td><strong>UpdateValueRange</strong></td>
</tr>
</tbody>
</table>
ReplayVariableCanInterpolate Method

Returns true if the specified type can be interpolated by the replay system.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```csharp
public static bool CanInterpolate(
    Type type
)
```

**Parameters**

*type*
- Type: `SystemType`  
- The system type to check for interpolation support

**Return Value**

Type: `Boolean`  
True if interpolation is supported or false if it is not

**See Also**

Reference  
ReplayVariable Class  
UltimateReplay.Core Namespace
ReplayVariableInterpolate Method

Attempts to interpolate the ReplayVariable value using the values from the last and next frame.

**Namespace:** UltimateReplay.Core
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public void Interpolate(
    float delta
)
```

### Parameters

*delta*

Type: System.Single
The normalized delta representing the progression from the last frame to the next frame

### See Also

Reference
ReplayVariable Class
UltimateReplay.Core Namespace
ReplayVariableInterpolateByte Method

Default interpolator for byte.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll)  
Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```c#
public static Object InterpolateByte(
    Object last,
    Object next,
    float delta
)
```

**Parameters**

*last*  
Type: SystemObject  
Last value

*next*  
Type: SystemObject  
Next value

*delta*  
Type: SystemSingle  
Interpolation delta

**Return Value**  
Type: Object  
The interpolated byte value
See Also

Reference
ReplayVariable Class
UltimateReplay.Core Namespace
ReplayVariableInterpolateColor Method

Default interpolator for Color.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```csharp
public static Object InterpolateColor(
    Object last,
    Object next,
    float delta
)
```

**Parameters**

- **last**  
  Type: SystemObject  
  Last value

- **next**  
  Type: SystemObject  
  Next value

- **delta**  
  Type: SystemSingle  
  Interpolation delta

**Return Value**  
Type: Object  
The interpolated Color value
See Also

Reference
ReplayVariable Class
UltimateReplay.Core Namespace
ReplayVariableInterpolateColor32 Method

Default interpolator for Color32.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

▲ Syntax

```csharp
public static Object InterpolateColor32(
    Object last,
    Object next,
    float delta
)
```

Parameters

*last*  
Type: `SystemObject`  
Last value

*next*  
Type: `SystemObject`  
Next value

*delta*  
Type: `SystemSingle`  
Interpolation delta

Return Value  
Type: `Object`  
The interpolated Color32 value
See Also

Reference
ReplayVariable Class
UltimateReplay.Core Namespace
ReplayVariableInterpolateDouble Method

Default interpolator for double.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```csharp
public static Object InterpolateDouble(
    Object last,
    Object next,
    float delta
)
```

**Parameters**

*last*
- Type: SystemObject  
  Last value

*next*
- Type: SystemObject  
  Next value

*delta*
- Type: SystemSingle  
  Interpolation delta

**Return Value**

Type: Object  
The interpolated double value
See Also

Reference
ReplayVariable Class
UltimateReplay.Core Namespace
ReplayVariableInterpolateFloat Method

Default interpolator for float.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

## Syntax

```c#
public static Object InterpolateFloat(
    Object last,
    Object next,
    float delta
)
```

### Parameters

*last*
- Type: SystemObject  
  Last value

*next*
- Type: SystemObject  
  Next value

*delta*
- Type: SystemSingle  
  Interpolation delta

### Return Value

Type: Object  
The interpolated float value
See Also

Reference
ReplayVariable Class
UltimateReplay.Core Namespace
ReplayVariableInterpolateInt Method

Default interpolator for int.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public static Object InterpolateInt(  
    Object last,  
    Object next,  
    float delta
)
```

### Parameters

**last**
- Type: `SystemObject`  
  Last value

**next**
- Type: `SystemObject`  
  Next value

**delta**
- Type: `SystemSingle`  
  Interpolation delta

### Return Value

Type: `Object`  
The interpolated int value
See Also

Reference
ReplayVariable Class
UltimateReplay.Core Namespace
ReplayVariableInterpolateLong Method

Default interpolator for long.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```c#
public static Object InterpolateLong(
    Object last,
    Object next,
    float delta
)
```

**Parameters**

- **last**
  - Type: SystemObject
  - Last value

- **next**
  - Type: SystemObject
  - Next value

- **delta**
  - Type: SystemSingle
  - Interpolation delta

**Return Value**

- Type: Object
  - The interpolated long value
See Also

Reference
ReplayVariable Class
UltimateReplay.Core Namespace
ReplayVariableInterpolateQuat Method

Default interpolator for Quaternion.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

▲ Syntax

```csharp
public static Object InterpolateQuat(  
    Object last,  
    Object next,  
    float delta  
)
```

Parameters

*last*
- Type: `SystemObject`
- Last value

*next*
- Type: `SystemObject`
- Next value

*delta*
- Type: `SystemSingle`
- Interpolation delta

Return Value

Type: `Object`

The interpolated Quaternion value
See Also

Reference
ReplayVariable Class
UltimateReplay.Core Namespace
ReplayVariableInterpolateShort Method

Default interpolator for short.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

⚠️ Syntax

```csharp
public static Object InterpolateShort(
    Object last,
    Object next,
    float delta
)
```

Parameters

- **last**
  - Type: SystemObject
  - Last value

- **next**
  - Type: SystemObject
  - Next value

- **delta**
  - Type: SystemSingle
  - Interpolation delta

Return Value

Type: Object

The interpolated short value
See Also

Reference
ReplayVariable Class
UltimateReplay.Core Namespace
ReplayVariableInterpolateValue Method

Attempts to interpolate the ReplayVariable value using the values from the last and next frame. In order for interpolation to succeed, the last and next values must be of the same type.

Namespace: UltimateReplay.Core
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

Syntax

```c#
public static Object InterpolateValue(
    Object last,
    Object next,
    float delta
)
```

Parameters

last
- Type: SystemObject
- The value of the variable in the last frame

next
- Type: SystemObject
- The value of the variable in the next frame

delta
- Type: SystemSingle
- The normalized delta representing the progression from the last frame to the next frame

Return Value
Type: Object
The interpolated value result or null if interpolation is not supported for the type

See Also

Reference
ReplayVariable Class
UltimateReplay.Core Namespace
ReplayVariableInterpolateVec2 Method

Default interpolator for Vector2.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```csharp
public static Object InterpolateVec2(  
    Object last,  
    Object next,  
    float delta
)
```

**Parameters**

*last*
- Type: SystemObject  
- Last value

*next*
- Type: SystemObject  
- Next value

*delta*
- Type: SystemSingle  
- Interpolation delta

**Return Value**

Type: Object  
- The interpolated Vector2 value
See Also

Reference
ReplayVariable Class
UltimateReplay.Core Namespace
ReplayVariableInterpolateVec3 Method

Default interpolator for Vector3.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

 harms

**Syntax**

```csharp
public static Object InterpolateVec3(
    Object last,
    Object next,
    float delta
)
```

**Parameters**

*last*  
Type: SystemObject  
Last value

*next*  
Type: SystemObject  
Next value

*delta*  
Type: SystemSingle  
Interpolation delta

**Return Value**  
Type: Object  
The interpolated Vector3 value
See Also

Reference
ReplayVariable Class
UltimateReplay.Core Namespace
ReplayVariableInterpolateVec4 Method

Default interpolator for Vector4.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```csharp
public static Object InterpolateVec4(  
    Object last,  
    Object next,  
    float delta  
)
```

**Parameters**

*last*
Type: SystemObject  
Last value

*next*
Type: SystemObject  
Next value

*delta*
Type: SystemSingle  
Interpolation delta

**Return Value**

Type: Object  
The interpolated Vector4 value
See Also

Reference
ReplayVariable Class
UltimateReplay.Core Namespace
ReplayVariableOnReplayDeserialize Method

Called by the replay system when the variable should be deserialized.

**Namespace:** UltimateReplay.Core
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```c#
public void OnReplayDeserialize(
    ReplayState state
)
```

**Parameters**

`state`
- Type: UltimateReplay.ReplayState
- The `ReplayState` to deserialize the data from

**Implements**

`IReplaySerializeOnReplayDeserialize(ReplayState)`

**See Also**

**Reference**
- `ReplayVariable Class`
- `UltimateReplay.Core Namespace`
ReplayVariableOnReplaySerialize Method

Called by the replay system when the variable should be serialized.

**Namespace:** UltimateReplay.Core
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```c#
public void OnReplaySerialize(ReplayState state)
```

**Parameters**

`state`
- Type: `UltimateReplay.ReplayState`
- The `ReplayState` to serialize the data into

**Implements**

`IReplaySerializeOnReplaySerialize(ReplayState)`

**See Also**

**Reference**
- `ReplayVariable Class`
- `UltimateReplay.Core Namespace`
ReplayVariableRegisterCustomInterpolator Method

Allows a custom interpolation method to be registered so that unsupported variable types can be interpolated automatically.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public static void RegisterCustomInterpolator<T>(
    Func<Object, Object, float, Object> interpolatorFunc
)
```

### Parameters

**interpolatorFunc**
- Type: `System.Func<Object, Object, float, Object>`
  - The interpolation method to invoke when interpolation of the custom type is required

### Type Parameters

**T**
- The type of variable that the custom interpolation should be used for

### See Also

**Reference**
- `ReplayVariable Class`
- `UltimateReplay.Core Namespace`
ReplayVariableUpdateValueRange Method

Sets the current interpolation range for the ReplayVariable value.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

▲ Syntax

```csharp
public void UpdateValueRange(
    Object last,
    Object next
)
```

**Parameters**

- **last**
  - Type: SystemObject
  - The value of the variable in the last frame

- **next**
  - Type: SystemObject
  - The value of the variable in the next frame

▲ See Also

Reference

- ReplayVariable Class
- UltimateReplay.Core Namespace
# ReplayVariable Properties

The `ReplayVariable` type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute</td>
<td>Get the <code>ReplayVarAttribute</code> associated with this <code>ReplayVariable</code>.</td>
</tr>
<tr>
<td>gameObject</td>
<td>Get the game object that this <code>ReplayVariable</code> is attached to.</td>
</tr>
<tr>
<td>IsInterpolated</td>
<td>Returns true if this <code>ReplayVariable</code> should be interpolated between frames.</td>
</tr>
<tr>
<td>IsInterpolationSupported</td>
<td>Returns true if this <code>ReplayVariable</code> supports interpolation. Interpolation can only be supported if the variable type has a registered interpolator.</td>
</tr>
<tr>
<td>Name</td>
<td>Get the name of this <code>ReplayVariable</code>.</td>
</tr>
<tr>
<td>Value</td>
<td>The current value for this <code>ReplayVariable</code>.</td>
</tr>
</tbody>
</table>
See Also

Reference

ReplayVariable Class
UltimateReplay.Core Namespace
ReplayVariableAttribute Property

Get the ReplayVarAttribute associated with this ReplayVariable.

**Namespace:** UltimateReplay.Core

**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```c#
public ReplayVarAttribute Attribute {
    get;
}
```

**Property Value**

**Type:** ReplayVarAttribute

### See Also

**Reference**

ReplayVariable Class
UltimateReplay.Core Namespace
ReplayVariable gameObject Property

Get the game object that this ReplayVariable is attached to.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

⚠️ **Syntax**

```csharp
public GameObject gameObject { get; }
```

**Property Value**  
Type: **GameObject**

⚠️ **See Also**

Reference  
ReplayVariable Class  
UltimateReplay.Core Namespace
ReplayVariable.IsInterpolated Property

Returns true if this ReplayVariable should be interpolated between frames.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public bool IsInterpolated { get; }
```

**Property Value**

Type: Boolean

### See Also

Reference

ReplayVariable Class  
UltimateReplay.Core Namespace
ReplayVariable.IsInterpolationSupported Property

Returns true if this ReplayVariable supports interpolation. Interpolation can only be supported if the variable type has a registered interpolator.

Namespace: UltimateReplay.Core
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```csharp
public bool IsInterpolationSupported { get; }
```

Property Value
Type: Boolean

**See Also**

Reference
ReplayVariable Class
UltimateReplay.Core Namespace
ReplayVariable Name Property

Get the name of this ReplayVariable.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0

## Syntax

```c#
public string Name { get; }
```

### Property Value

Type: String

### See Also

- **Reference**
  - ReplayVariable Class
  - UltimateReplay.Core Namespace
ReplayVariableValue Property

The current value for this ReplayVariable.

**Namespace:** UltimateReplay.Core  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

▶ **Syntax**

```
public Object Value { get; set; }
```

Property Value  
Type: Object

▶ **See Also**

Reference  
ReplayVariable Class  
UltimateReplay.Core Namespace
## UltimateReplay.Demo Namespace

### Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AudioDemo</td>
<td>A script using in the audio demo to show the ReplayAudio component working.</td>
</tr>
<tr>
<td>CubeSpawner</td>
<td>A demo script used in the stress test scene which spawns a large number of cubes.</td>
</tr>
</tbody>
</table>
AudioDemo Class

A script using in the audio demo to show the ReplayAudio component working.

▲ Inheritance Hierarchy

[SystemObject] Object
  Component
    Behaviour
      MonoBehaviour
        UltimateReplay.DemoAudioDemo

Namespace: UltimateReplay.Demo
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

▲ Syntax

```c#
public class AudioDemo : MonoBehaviour
```

The AudioDemo type exposes the following members.

▲ Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>🌟 AudioDemo</td>
<td>Initializes a new instance of the AudioDemo class</td>
</tr>
</tbody>
</table>
## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnGUI</td>
<td>Called by Unity.</td>
</tr>
<tr>
<td>Update</td>
<td>Called by Unity.</td>
</tr>
</tbody>
</table>

## Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>replayAudio</td>
<td>The <code>ReplayAudio</code> script.</td>
</tr>
</tbody>
</table>

## See Also

Reference

*UltimateReplay.Demo Namespace*
AudioDemo Constructor

Initializes a new instance of the **AudioDemo** class

**Namespace:** UltimateReplay.Demo  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```c#
public AudioDemo()
```

### See Also

**Reference**
- AudioDemo Class
- UltimateReplay.Demo Namespace
AudioDemo Fields

The **AudioDemo** type exposes the following members.

### Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>replayAudio</td>
<td>The <strong>ReplayAudio</strong> script.</td>
</tr>
</tbody>
</table>

### See Also

Reference

- **AudioDemo Class**
- **UltimateReplay.Demo Namespace**
AudioDemo
toreplayAudio Field

The ReplayAudio script.

**Namespace:** UltimateReplay.Demo  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```c#
public ReplayAudio replayAudio
```

### Field Value

Type: ReplayAudio

### See Also

Reference  
AudioDemo Class  
UltimateReplay.Demo Namespace
AudioDemo Methods

The AudioDemo type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnGUI</td>
<td>Called by Unity.</td>
</tr>
<tr>
<td>Update</td>
<td>Called by Unity.</td>
</tr>
</tbody>
</table>

See Also

Reference

AudioDemo Class
UltimateReplay.Demo Namespace
AudioDemoOnGUI Method

Called by Unity.

**Namespace:** UltimateReplay.Demo  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```c#
public void OnGUI()
```

### See Also

Reference  
AudioDemo Class  
UltimateReplay.Demo Namespace
AudioDemo Update Method

Called by Unity.

**Namespace:** UltimateReplay.Demo  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

⚠️ Syntax

```c#
public void Update()
```

⚠️ See Also

Reference

AudioDemo Class  
UltimateReplay.Demo Namespace
CubeSpawner Class

A demo script used in the stress test scene which spawns a large number of cubes.

Inheritance Hierarchy

- System
  - Object
    - Component
      - Behaviour
        - MonoBehaviour
          - UltimateReplay.DemoCubeSpawner

Namespace: UltimateReplay.Demo
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0

Syntax

```csharp
public class CubeSpawner : MonoBehaviour
```

The `CubeSpawner` type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>🌟 CubeSpawner</td>
<td>Initializes a new instance of the <code>CubeSpawner</code> class</td>
</tr>
</tbody>
</table>

Top
## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>Called by Unity.</td>
</tr>
</tbody>
</table>

## Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>explosiveForce</td>
<td>The amount of force that is initially given to the spawning cubes.</td>
</tr>
<tr>
<td>spawnAmount</td>
<td>The amount of objects to spawn into the scene.</td>
</tr>
<tr>
<td>spawnCubes</td>
<td>An array of prefabs used to randomly spawn objects.</td>
</tr>
<tr>
<td>spawnHeight</td>
<td>The maximum height that an object can be spawned from the center.</td>
</tr>
<tr>
<td>spawnRange</td>
<td>The maximum distance that an object can be spawned from the center.</td>
</tr>
</tbody>
</table>

## See Also

Reference
UltimateReplay.Demo Namespace
CubeSpawner Constructor

Initializes a new instance of the CubeSpawner class

Namespace: UltimateReplay.Demo  
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

Syntax

```csharp
public CubeSpawner()
```

See Also

Reference
CubeSpawner Class  
UltimateReplay.Demo Namespace
CubeSpawner Fields

The CubeSpawner type exposes the following members.

### Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>explosiveForce</td>
<td>The amount of force that is initially given to the spawning cubes.</td>
</tr>
<tr>
<td>spawnAmount</td>
<td>The amount of objects to spawn into the scene.</td>
</tr>
<tr>
<td>spawnCubes</td>
<td>An array of prefabs used to randomly spawn objects.</td>
</tr>
<tr>
<td>spawnHeight</td>
<td>The maximum height that an object can be spawned from the center.</td>
</tr>
<tr>
<td>spawnRange</td>
<td>The maximum distance that an object can be spawned from the center.</td>
</tr>
</tbody>
</table>

See Also

Reference
CubeSpawner Class
UltimateReplay.Demo Namespace
CubeSpawner.explosiveForce Field

The amount of force that is initially given to the spawning cubes.

**Namespace:** UltimateReplay.Demo  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```c#
public float explosiveForce
```

### Field Value

Type: Single

### See Also

- Reference
  - CubeSpawner Class
  - UltimateReplay.Demo Namespace
CubeSpawnerspawnAmount Field

The amount of objects to spawn into the scene.

**Namespace:** UltimateReplay.Demo

**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```csharp
public int spawnAmount
```

**Field Value**

Type: **Int32**

**See Also**

Reference

CubeSpawner Class
UltimateReplay.Demo Namespace
CubeSpawnerspawnCubes Field

An array of prefabs used to randomly spawn objects.

**Namespace:** UltimateReplay.Demo  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

## Syntax

```csharp
public GameObject[] spawnCubes
```

## Field Value

Type: **GameObject**

## See Also

Reference  
**CubeSpawner Class**  
**UltimateReplay.Demo Namespace**
CubeSpawnerspawnHeight Field

The maximum height that an object can be spawned from the center.

**Namespace:** UltimateReplay.Demo  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0  
(1.0.0.0)

### Syntax

```
public float spawnHeight
```

### Field Value

Type: Single

### See Also

Reference  
CubeSpawner Class  
UltimateReplay.Demo Namespace
CubeSpawner.spawnRange Field

The maximum distance that an object can be spawned from the center.

Namespace: UltimateReplay.Demo
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

Syntax

```c#
public float spawnRange
```

Field Value
Type: Single

See Also

Reference
CubeSpawner Class
UltimateReplay.Demo Namespace
CubeSpawner Methods

The `CubeSpawner` type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>🚀</td>
<td>Start</td>
</tr>
</tbody>
</table>

See Also

Reference

- `CubeSpawner Class`
- `UltimateReplay.Demo Namespace`
CubeSpawnerStart Method

Called by Unity.

**Namespace:** UltimateReplay.Demo  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public IEnumerator Start()
```

### Return Value

Type: `IEnumerator`  

### See Also

Reference  
CubeSpawner Class  
UltimateReplay.Demo Namespace
## UltimateReplay.Storage Namespace

### Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compression</td>
<td>Compression utility using the GZip compression algorithm.</td>
</tr>
<tr>
<td>ReplayMemoryTarget</td>
<td>Represents a memory storage buffer where replay data can be stored for game sessions. The buffer can be used as a continuous rolling buffer of a fixed size where a fixed amount of playback footage is recorded and then overwritten by new data as it is received.</td>
</tr>
<tr>
<td>ReplaySnapshot</td>
<td>A frame state is a snapshot of a replay frame that is indexed based on its time stamp. By sequencing multiple frame states you can create the replay effect.</td>
</tr>
<tr>
<td>ReplayTarget</td>
<td>Represents and abstract storage device capable of holding recorded state data for playback at a later date. Depending upon implementation, a <a href="#">ReplayTarget</a> is used.</td>
</tr>
</tbody>
</table>
may be volatile or non-volatile.

## Structures

<table>
<thead>
<tr>
<th>Structure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReplayInitialData</td>
<td>Represents the initial settings of a newly spawned replay object. When a game object is instantiated it must be given an initial position and rotation.</td>
</tr>
</tbody>
</table>

## Enumerations

<table>
<thead>
<tr>
<th>Enumeration</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompressionLevel</td>
<td>The amount of compression to apply to a data stream.</td>
</tr>
<tr>
<td>ReplayInitialDataFlags</td>
<td>Represents initial data that may be stored by an object.</td>
</tr>
<tr>
<td>ReplayTargetTask</td>
<td>Represents a task that can be issued to a ReplayTarget.</td>
</tr>
</tbody>
</table>
Compression Class

Compression utility using the GZip compression algorithm.

▲ Inheritance Hierarchy

[SystemObject]   UltimateReplay.StorageCompression

**Namespace:** UltimateReplay.Storage  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

▲ Syntax

```csharp
public static class Compression
```

The **Compression** type exposes the following members.

▲ Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>🚀 CompressData</td>
<td>Compress a data stream using the GZip compression algorithm.</td>
</tr>
<tr>
<td>🚀 DecompressData</td>
<td>Decompress a data stream using the GZip compression algorithm.</td>
</tr>
</tbody>
</table>

Top

▲ See Also

Reference
UltimateReplay.Storage Namespace
Compression Methods

The **Compression** type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CompressData</td>
<td>Compress a data stream using the GZip compression algorithm.</td>
</tr>
<tr>
<td>DecompressData</td>
<td>Decompress a data stream using the GZip compression algorithm.</td>
</tr>
</tbody>
</table>

### See Also

Reference
- Compression Class
- UltimateReplay.Storage Namespace
Compression

**Compression CompressData Method**

Compress a data stream using the GZip compression algorithm.

**Namespace:** UltimateReplay.Storage  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```c#
public static byte[] CompressData(
    byte[] data,
    CompressionLevel level = CompressionLevel.Optimal
)
```

### Parameters

- **data**
  - Type: `System.Byte`
  - The input data to compress

- **level (Optional)**
  - Type: `UltimateReplay.Storage.CompressionLevel`
  - The target compression level to use

### Return Value

- **Type:** Byte
  - The compressed data

### See Also

Reference
Compression Class
UltimateReplay.Storage Namespace
CompressionDecompressData Method

Decompress a data stream using the GZip compression algorithm.

**Namespace:** UltimateReplay.Storage  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public static byte[] DecompressData(
    byte[] data,
    CompressionLevel level = CompressionLevel.Optimal
)
```

### Parameters

**data**
- Type: `System.Byte`  
The input data to decompress

**level (Optional)**
- Type: `UltimateReplay.Storage.CompressionLevel`  
The target compression level to use

### Return Value

Type: `Byte`  
The decompressed data

### See Also

Reference
Compression Class
UltimateReplay.Storage Namespace
CompressionLevel Enumeration

The amount of compression to apply to a data stream.

**Namespace:** UltimateReplay.Storage  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

C#

```csharp
public enum CompressionLevel
```

### Members

<table>
<thead>
<tr>
<th>Member name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>0</td>
<td>No compression is applied and all data is left unchanged.</td>
</tr>
<tr>
<td>Optimal</td>
<td>1</td>
<td>All data is compressed to the optimal level.</td>
</tr>
</tbody>
</table>

### See Also

Reference
UltimateReplay.Storage Namespace
ReplayInitialData Structure

Represents the initial settings of a newly spawned replay object. When a game object is instantiated it must be given an initial position and rotation.

**Namespace:** UltimateReplay.Storage  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```c#
public struct ReplayInitialData
```

The `ReplayInitialData` type exposes the following members.

**Fields**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>position</td>
<td>Initial position data.</td>
</tr>
<tr>
<td>rotation</td>
<td>Initial rotation data.</td>
</tr>
<tr>
<td>scale</td>
<td>Initial scale data.</td>
</tr>
</tbody>
</table>

**See Also**

Reference

UltimateReplay.Storage Namespace
ReplayInitialData Fields

The `ReplayInitialData` type exposes the following members.

### Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>position</td>
<td>Initial position data.</td>
</tr>
<tr>
<td>rotation</td>
<td>Initial rotation data.</td>
</tr>
<tr>
<td>scale</td>
<td>Initial scale data.</td>
</tr>
</tbody>
</table>

See Also

Reference

- [ReplayInitialData Structure](#)
- [UltimateReplay.Storage Namespace](#)
ReplayInitialData position Field

Initial position data.

**Namespace:** UltimateReplay.Storage  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

▶ Syntax

```c#
public Vector3 position
```

Field Value  
Type: `Vector3`

▶ See Also

Reference  
ReplayInitialData Structure  
UltimateReplay.Storage Namespace
ReplayInitialData rotation Field

Initial rotation data.

**Namespace:** UltimateReplay.Storage  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

🔗 Syntax

```c#
public Quaternion rotation
```

Field Value  
Type: **Quaternion**

🔗 See Also

Reference  
ReplayInitialData Structure  
UltimateReplay.Storage Namespace
ReplayInitialData scale Field

Initial scale data.

**Namespace:** UltimateReplay.Storage  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

⚠️ Syntax

```c#  
public Vector3 scale  
```

Field Value  
Type: Vector3

⚠️ See Also

Reference  
ReplayInitialData Structure  
UltimateReplay.Storage Namespace
ReplayInitialDataFlags Enumeration

Represents initial data that may be stored by an object.

**Namespace:** UltimateReplay.Storage  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```
public enum ReplayInitialDataFlags

[FlagsAttribute]
```

### Members

<table>
<thead>
<tr>
<th>Member name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>0</td>
<td>No initial data is stored.</td>
</tr>
<tr>
<td>Position</td>
<td>1</td>
<td>Initial position is recorded.</td>
</tr>
<tr>
<td>Rotation</td>
<td>2</td>
<td>Initial rotation is recorded.</td>
</tr>
<tr>
<td>Scale</td>
<td>4</td>
<td>Initial scale is recorded.</td>
</tr>
</tbody>
</table>

### See Also

Reference  
UltimateReplay.Storage Namespace
ReplayMemoryTarget Class

Represents a memory storage buffer where replay data can be stored for game sessions. The buffer can be used as a continuous rolling buffer of a fixed size where a fixed amount of playback footage is recorded and then overwritten by new data as it is received.

Inheritance Hierarchy

```
System
   Object
   Component
      Behaviour
         MonoBehaviour
            UltimateReplay.ReplayBehaviour
            UltimateReplay.Storage.ReplayTarget
            UltimateReplay.Storage.ReplayMemoryTarget
```

Namespace: UltimateReplay.Storage
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

Syntax

```
[SerializableAttribute]
public class ReplayMemoryTarget : ReplayTarget
```

The `ReplayMemoryTarget` type exposes the following members.

Constructors

```
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ReplayMemoryTarget</code></td>
<td>Initializes a new instance of the</td>
</tr>
</tbody>
</table>
```
ReplayMemoryTarget class

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PrepareTarget</td>
<td>Clears all state information for the current recording essentially restoring the memory to its initial state. (Overrides ReplayTargetPrepareTarget(ReplayTarget))</td>
</tr>
<tr>
<td>RecordSnapshot</td>
<td>Store a replay snapshot in the replay target. A new snapshot causes the internal buffer to 'overflow' then the recording clip will be wrapped so that the recording duration is no more than recordSeconds. (Overrides ReplayTargetRecordSnapshot(ReplaySnapshot))</td>
</tr>
<tr>
<td>RestoreSnapshot</td>
<td>Recall a snapshot from the replay target based on the specified replay offset. (Overrides ReplayTargetRestoreSnapshot(Single)).</td>
</tr>
</tbody>
</table>

Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>recordSeconds</td>
<td>The amount of time in seconds of recording that should be kept in memory before it is discarded. The time is measured backwards from the current time to give a rolling buffer of</td>
</tr>
</tbody>
</table>
the last 'n' seconds. This is useful in situations where you are only need the previous few seconds of gameplay to be recorded for example: A kill cam. If this value is set to 0 then the internal buffer will not be wrapped at all. You should take extra care when using an unconstrained buffer as there is potential to run into an OutOfMemoryException, especially on mobile platforms where memory is at a premium.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Duration</strong></td>
<td>The amount of time in seconds that the recording lasts. Usually this value will be equal to recordSeconds however it will take atleas the amount of recordSeconds to initially fill the buffer before it wraps around. (Overrides ReplayTargetDuration.)</td>
</tr>
<tr>
<td><strong>MemorySize</strong></td>
<td>Get the amount of size in bytes that this memory target requires for all state data. This size does not include internal structures used to store the data but exclusivley contains game state sizes. (Overrides ReplayTargetMemorySize.)</td>
</tr>
</tbody>
</table>

### See Also
Reference
UltimateReplay.Storage Namespace
ReplayMemoryTarget Constructor

Initializes a new instance of the ReplayMemoryTarget class

Namespace: UltimateReplay.Storage
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

Syntax

```c#
public ReplayMemoryTarget()
```

See Also

Reference
ReplayMemoryTarget Class
UltimateReplay.Storage Namespace
## ReplayMemoryTarget Fields

The `ReplayMemoryTarget` type exposes the following members.

### Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>recordSeconds</td>
<td>The amount of time in seconds of recording that should be kept in memory before it is discarded. The time is measured backwards from the current time to give a rolling buffer of the last 'n' seconds. This is useful in situations where you are only need the previous few seconds of gameplay to be recorded for example: A kill cam. If this value is set to 0 then the internal buffer will not be wrapped at all. You should take extra care when using an unconstrained buffer as there is potential to run into an <code>OutOfMemoryException</code>, especially on mobile platforms where memory is at a premium.</td>
</tr>
</tbody>
</table>

### See Also

Reference

- `ReplayMemoryTarget` Class
- `UltimateReplay.Storage` Namespace
ReplayMemoryTargetrecordSeconds Field

The amount of time in seconds of recording that should be kept in memory before it is discarded. The time is measured backwards from the current time to give a rolling buffer of the last 'n' seconds. This is useful in situations where you are only need the previous few seconds of gameplay to be recorded for example: A kill cam. If this value is set to 0 then the internal buffer will not be wrapped at all. You should take extra care when using an unconstrained buffer as there is potential to run into an OutOfMemoryException, especially on mobile platforms where memory is at a premium.

**Namespace:** UltimateReplay.Storage  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```
public float recordSeconds
```

Field Value  
Type: Single

### See Also

Reference  
ReplayMemoryTarget Class  
UltimateReplay.Storage Namespace
ReplayMemoryTarget Methods

The `ReplayMemoryTarget` type exposes the following members.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>PrepareTarget</code></td>
<td>Clears all state information for the current recording essentially restoring the memory to its initial state. (Overrides <code>ReplayTargetPrepareTarget(ReplayTarget)</code></td>
</tr>
<tr>
<td><code>RecordSnapshot</code></td>
<td>Store a replay snapshot in the replay target. A new snapshot causes the internal buffer to 'overflow' then the recording clip will be wrapped so that the recording duration is no more than <code>recordSeconds</code>. (Overrides <code>ReplayTargetRecordSnapshot(ReplaySnapshot)</code></td>
</tr>
<tr>
<td><code>RestoreSnapshot</code></td>
<td>Recall a snapshot from the replay target based on the specified replay offset. (Overrides <code>ReplayTargetRestoreSnapshot(Single)</code>.)</td>
</tr>
</tbody>
</table>

### See Also

Reference
- `ReplayMemoryTarget Class`
- `UltimateReplay.Storage Namespace`
ReplayMemoryTargetPrepareTarget Method

Clears all state information for the current recording essentially restoring the memory target to its initial state.

**Namespace:** UltimateReplay.Storage  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public override void PrepareTarget(  
    ReplayTargetTask mode
)
```

**Parameters**

- `mode`
  
  Type: UltimateReplay.Storage.ReplayTargetTask
  

### See Also

Reference

ReplayMemoryTarget Class  
UltimateReplay.Storage Namespace
ReplayMemoryTargetRecordSnapshot Method

Store a replay snapshot in the replay target. If the new snapshot causes the internal buffer to 'overflow' then the recoding clip will be wrapped so that the recording duration is no more than recordSeconds.

Namespace: UltimateReplay.Storage
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0

Syntax

```csharp
public override void RecordSnapshot(
    ReplaySnapshot state
)
```

Parameters

state
   Type: UltimateReplay.Storage.ReplaySnapshot
   The snapshot to store

See Also

Reference
ReplayMemoryTarget Class
UltimateReplay.Storage Namespace
ReplayMemoryTargetRestoreSnapshot Method

Recall a snapshot from the replay target based on the specified replay offset.

**Namespace:** UltimateReplay.Storage  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```csharp
public override ReplaySnapshot RestoreSnapshot(
    float offset
)
```

**Parameters**

*offset*

Type: `System.Single`  
The offset pointing to the individual snapshot to recall

**Return Value**

Type: `ReplaySnapshot`  
The replay snapshot at the specified offset

**See Also**

Reference  
ReplayMemoryTarget Class  
UltimateReplay.Storage Namespace
ReplayMemoryTarget Properties

The `ReplayMemoryTarget` type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Duration</strong></td>
<td>The amount of time in seconds that the recording lasts. Usually this value will be equal to <code>recordSeconds</code> however it will take at least the amount of <code>recordSeconds</code> to initially fill the buffer before it wraps around. (Overrides <code>ReplayTargetDuration</code>.)</td>
</tr>
<tr>
<td><strong>MemorySize</strong></td>
<td>Get the amount of size in bytes that this memory target requires for all state data. This size does not include internal structures used to store the data but exclusively contains game state sizes. (Overrides <code>ReplayTargetMemorySize</code>.)</td>
</tr>
</tbody>
</table>

See Also

Reference

- `ReplayMemoryTarget Class`
- `UltimateReplay.Storage Namespace`
ReplayMemoryTargetDuration Property

The amount of time in seconds that the recording lasts. Usually this value will be equal to recordSeconds however it will take at least the amount of recordSeconds to initially fill the buffer before it wraps around.

**Namespace:** UltimateReplay.Storage  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```c# Copy
public override float Duration { get; }
```

Property Value

Type: Single

**See Also**

Reference

- ReplayMemoryTarget Class
- UltimateReplay.Storage Namespace
ReplayMemoryTarget.MemorySize Property

Get the amount of size in bytes that this memory target requires for all state data. This size does not include internal structures used to store the data but exclusively contains game state sizes.

Namespace: UltimateReplay.Storage
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

Syntax

C# Copy

```csharp
public override int MemorySize { get; }
```

Property Value
Type: Int32

See Also

Reference
ReplayMemoryTarget Class
UltimateReplay.Storage Namespace
ReplaySnapshot Class

A frame state is a snapshot of a replay frame that is indexed based on its time stamp. By sequencing multiple frame states you can create the replay effect.

Inheritance Hierarchy

System\Object  UltimateReplay.Storage\ReplaySnapshot

Namespace: UltimateReplay.Storage
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

Syntax

```c#
[SerializableAttribute]
public sealed class ReplaySnapshot : IReplaySerializer,
IReplayDataSerializer
```

The `ReplaySnapshot` type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReplaySnapshot</td>
<td>Create a new snapshot with the specified time stamp.</td>
</tr>
</tbody>
</table>

Methods
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnReplayDataDeserialize</td>
<td>Called by the replay system when this ReplaySnapshot should be deserialized from binary.</td>
</tr>
<tr>
<td>OnReplayDataSerialize</td>
<td>Called by the replay system when this ReplaySnapshot should be serialized to binary.</td>
</tr>
<tr>
<td>OnReplayDeserialize</td>
<td>Called by the replay system when this ReplaySnapshot should be deserialized.</td>
</tr>
<tr>
<td>OnReplaySerialize</td>
<td>Called by the replay system when this ReplaySnapshot should be serialized.</td>
</tr>
<tr>
<td>RecordInitialReplayObjectData</td>
<td>Attempts to record the initial information about a newly created replay object. The initial information will only be sorted if the data is not equal to the default value.</td>
</tr>
<tr>
<td>RecordSnapshot</td>
<td>Registers the specified replay state with this snapshot. The specified identity</td>
</tr>
</tbody>
</table>
is used during playback to ensure that the replay objects receive the correct state to deserialize.

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset</td>
<td>Clears all state information from the snapshot but keeps the time stamp.</td>
</tr>
<tr>
<td>RestoreInitialReplayObjectData</td>
<td>Attempts to get the initial data for the replay object with the specified ReplayIdentity. A ReplayInitialData will be returned containing the initial state for the replay object.</td>
</tr>
<tr>
<td>RestoreReplayObjects</td>
<td>Attempts to restore any replay objects that were spawned or despawned during this snapshot.</td>
</tr>
<tr>
<td>RestoreSnapshot</td>
<td>Attempts to recall the state information for the specified replay object identity. If the identity does not exist in the scene then the return value will be null.</td>
</tr>
</tbody>
</table>
## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>Get the size in bytes of the snapshot data.</td>
</tr>
<tr>
<td>TimeStamp</td>
<td>The time stamp for this snapshot. The time stamp is used to identify the snapshot location in the sequence.</td>
</tr>
</tbody>
</table>

## See Also

Reference

UltimateReplay.Storage Namespace
ReplaySnapshot Constructor

Create a new snapshot with the specified time stamp.

**Namespace:** UltimateReplay.Storage  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public ReplaySnapshot(
    float timeTimestamp
)
```

**Parameters**

timeStamp  
Type: System.Single  
The time stamp to give to this snapshot

### See Also

Reference  
ReplaySnapshot Class  
UltimateReplay.Storage Namespace
ReplaySnapshot Methods

The ReplaySnapshot type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnReplayDataDeserialize</td>
<td>Called by the replay system when this ReplaySnapshot should be deserialized from binary.</td>
</tr>
<tr>
<td>OnReplayDataSerialize</td>
<td>Called by the replay system when this ReplaySnapshot should be serialized to binary.</td>
</tr>
<tr>
<td>OnReplayDeserialize</td>
<td>Called by the replay system when this ReplaySnapshot should be deserialized.</td>
</tr>
<tr>
<td>OnReplaySerialize</td>
<td>Called by the replay system when this ReplaySnapshot should be serialized.</td>
</tr>
<tr>
<td>RecordInitialReplayObjectData</td>
<td>Attempts to record the initial information about a newly created object.</td>
</tr>
</tbody>
</table>
replay object. The initial information will only be stored if the data is not equal to the default value.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>RecordSnapshot</code></td>
<td>Registers the specified replay state with this snapshot. The specified identity is used during playback to ensure that the replay objects receive the correct state to deserialize.</td>
</tr>
<tr>
<td><code>Reset</code></td>
<td>Clears all state information from the snapshot but keeps the timestamp.</td>
</tr>
<tr>
<td><code>RestoreInitialReplayObjectData</code></td>
<td>Attempts to get the initial data for the replay object with the specified <code>ReplayIdentity</code>. A <code>ReplayInitialData</code> will be returned containing the initial state for the replay object.</td>
</tr>
<tr>
<td><code>RestoreReplayObjects</code></td>
<td>Attempts to restore any replay objects that were spawned or despawned during this snapshot.</td>
</tr>
</tbody>
</table>
**RestoreSnapshot**

Attempts to recall the state information for the specified replay object identity. If the identity does not exist in the scene then the return value will be null.

**See Also**

Reference

- *ReplaySnapshot Class*
- *UltimateReplay.Storage Namespace*
ReplaySnapshotOnReplayDataDeserialize Method

Called by the replay system when this ReplaySnapshot should be deserialized from binary.

**Namespace:** UltimateReplay.Storage  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```
public void OnReplayDataDeserialize(
    BinaryReader reader
)
```

**Parameters**

*reader*
  
  Type: System.IO.BinaryReader  
  The binary stream to read the data from

**See Also**

Reference  
ReplaySnapshot Class  
UltimateReplay.Storage Namespace
ReplaySnapshotOnReplayDataSerialize Method

Called by the replay system when this ReplaySnapshot should be serialized to binary.

**Namespace**: UltimateReplay.Storage  
**Assembly**: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public void OnReplayDataSerialize(
    BinaryWriter writer
)
```

### Parameters

*writer*

Type: System.IO.BinaryWriter

The binary stream to write the data to

### See Also

Reference

- ReplaySnapshot Class
- UltimateReplay.Storage Namespace
ReplaySnapshotOnReplayDeserialize Method

Called by the replay system when this ReplaySnapshot should be deserialized.

**Namespace:** UltimateReplay.Storage  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public void OnReplayDeserialize(
    ReplayState state
)
```

### Parameters

- **state**  
  Type: UltimateReplay.ReplayState  
  The ReplayState to read the data from

### Implements

IReplaySerializeOnReplayDeserialize(ReplayState)

### See Also

**Reference**

ReplaySnapshot Class  
UltimateReplay.Storage Namespace
ReplaySnapshotOnReplaySerialize Method

Called by the replay system when this ReplaySnapshot should be serialized.

Namespace: UltimateReplay.Storage
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

Syntax

```c#
public void OnReplaySerialize(
    ReplayState state
)
```

Parameters

`state`
Type: UltimateReplay.ReplayState
The ReplayState to write the data to

Implements

IReplaySerializeOnReplaySerialize(ReplayState)

See Also

Reference
ReplaySnapshot Class
UltimateReplay.Storage Namespace
ReplaySnapshotRecordInitialReplay Method

Attempts to record the initial information about a newly created replay object. The initial information will only be stored if the data is not equal to the default value.

**Namespace:** UltimateReplay.Storage  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

## Syntax

```csharp
public void RecordInitialReplayObjectData(
    ReplayIdentity identity,
    Vector3 position,
    Quaternion rotation,
    Vector3 scale
)
```

### Parameters

- **identity**  
  Type: UltimateReplay.Core.ReplayIdentity  
  The object to store the initial data for

- **position**  
  Type: Vector3  
  The initial position of the object

- **rotation**  
  Type: Quaternion  
  The initial rotation of the object

- **scale**
Type: **Vector3**
The initial scale of the object

**See Also**

Reference

- ReplaySnapshot Class
- UltimateReplay.Storage Namespace
ReplaySnapshotRecordSnapshot Method

Registers the specified replay state with this snapshot. The specified identity is used during playback to ensure that the replay objects receive the correct state to deserialize.

Namespace: UltimateReplay.Storage
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

Syntax

```c#
public void RecordSnapshot(
    ReplayIdentity identity,
    ReplayState state
)
```

Parameters

- **identity**
  - Type: UltimateReplay.Core.ReplayIdentity
  - The identity of the object that was serialized

- **state**
  - Type: UltimateReplay.Replay.ReplayState
  - The state data for the object

See Also

Reference
- ReplaySnapshot Class
- UltimateReplay.Storage Namespace
ReplaySnapshotReset Method

Clears all state information from the snapshot but keeps the time stamp.

**Namespace:** UltimateReplay.Storage  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

▲ Syntax

```csharp
public void Reset()
```

▲ See Also

Reference  
ReplaySnapshot Class  
UltimateReplay.Storage Namespace
ReplaySnapshot.RestoreInitialReplayObjectData Method

Attempts to get the initial data for the replay object with the specified ReplayIdentity. A ReplayInitialData will be returned containing the initial state for the replay object.

Namespace: UltimateReplay.Storage
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

▸ Syntax

```c#
public ReplayInitialData RestoreInitialReplayObjectData(ReplayIdentity identity)
```

Parameters

identity
Type: UltimateReplay.Core.ReplayIdentity
The identity of the object to get the initial data for

Return Value

Type: ReplayInitialData
A ReplayInitialData containing initial values

▸ See Also

Reference
ReplaySnapshot Class
UltimateReplay.Storage Namespace
ReplaySnapshot RestoreReplayObjects Method

Attempts to restore any replay objects that were spawned or despawned during this snapshot.

**Namespace:** UltimateReplay.Storage  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public void RestoreReplayObjects(
    ReplayScene scene
)
```

### Parameters

**scene**
- Type: UltimateReplay.Core.ReplayScene

### See Also

Reference
- ReplaySnapshot Class
- UltimateReplay.Storage Namespace
ReplaySnapshotRestoreSnapshot Method

Attempts to recall the state information for the specified replay object identity. If the identity does not exist in the scene then the return value will be null.

Namespace: UltimateReplay.Storage
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

Syntax

```csharp
public ReplayState RestoreSnapshot(
    ReplayIdentity identity
)
```

Parameters

identity
Type: UltimateReplay.Core.ReplayIdentity
The identity of the object to deserialize

Return Value
Type: ReplayState
The state information for the specified identity or null if the identity does not exist

See Also

Reference
ReplaySnapshot Class
UltimateReplay.Storage Namespace
ReplaySnapshot Properties

The `ReplaySnapshot` type exposes the following members.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image.png" alt="Image" /> <strong>Size</strong></td>
<td>Get the size in bytes of the snapshot data.</td>
</tr>
<tr>
<td><img src="image.png" alt="Image" /> <strong>TimeStamp</strong></td>
<td>The time stamp for this snapshot. The time stamp is used to identify the snapshot location in the sequence.</td>
</tr>
</tbody>
</table>

### See Also

Reference

- `ReplaySnapshot Class`
- `UltimateReplay.Storage Namespace`
ReplaySnapshotSize Property

Get the size in bytes of the snapshot data.

**Namespace:** UltimateReplay.Storage  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

⚠️ Syntax

```csharp
public int Size { get; }
```

Property Value

Type: Int32

⚠️ See Also

Reference

ReplaySnapshot Class
UltimateReplay.Storage Namespace
ReplaySnapshotTimeStamp Property

The time stamp for this snapshot. The time stamp is used to identify the snapshot location in the sequence.

**Namespace:** UltimateReplay.Storage  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```
public float TimeStamp { get; }
```

Property Value

Type: **Single**

**See Also**

Reference  
ReplaySnapshot Class  
UltimateReplay.Storage Namespace
ReplayTarget Class

Represents and abstract storage device capable of holding recorded state data for playback at a later date. Depending upon implementation, a ReplayTarget may be volatile or non-volatile.

Inheritance Hierarchy


Namespace: UltimateReplay.Storage
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0
(1.0.0.0)

Syntax

C#

```
[SerializableAttribute]
public abstract class ReplayTarget : ReplayBehaviour
```

The ReplayTarget type exposes the following members.

Constructors

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>🌱 ReplayTarget</td>
<td>Initializes a new instance of the</td>
</tr>
</tbody>
</table>
ReplayTarget class

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PrepareTarget</td>
<td>Called by the recording system to notify the active ReplayTarget of an upcoming event.</td>
</tr>
<tr>
<td>RecordSnapshot</td>
<td>Store a replay snapshot in the replay target.</td>
</tr>
<tr>
<td>RestoreSnapshot</td>
<td>Recall a snapshot from the replay target based on the specified replay offset.</td>
</tr>
</tbody>
</table>

Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>duration</td>
<td>The amount of time in seconds that the current recording data lasts. If no data exists then the duration will default to a length of 0.</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duration</td>
<td>The amount of time in seconds that this recording lasts.</td>
</tr>
</tbody>
</table>
MemorySize  Get the total amount of bytes that this replay uses.

Top

See Also

Reference
UltimateReplay.Storage Namespace
ReplayTarget Constructor

Initializes a new instance of the `ReplayTarget` class

**Namespace:** UltimateReplay.Storage  
**Assembly:** UltimateReplay (in UltimateReplay.dll)  
**Version:** 1.0.0.0 (1.0.0.0)

Syntax

```csharp
protected ReplayTarget()
```

See Also

Reference  
`ReplayTarget Class`  
`UltimateReplay.Storage Namespace`
ReplayTarget Fields

The ReplayTarget type exposes the following members.

## Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>duration</td>
<td>The amount of time in seconds that the current recording data lasts. If no data exists then the duration will default to a length of 0.</td>
</tr>
</tbody>
</table>

## See Also

Reference
- ReplayTarget Class
- UltimateReplay.Storage Namespace
ReplayTarget.duration Field

The amount of time in seconds that the current recording data lasts. If no data exists then the duration will default to a length of 0.

**Namespace:** UltimateReplay.Storage  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

▲ Syntax

```c#
protected float duration
```

Field Value
Type: Single

▲ See Also

Reference
ReplayTarget Class  
UltimateReplay.Storage Namespace
ReplayTarget Methods

The `ReplayTarget` type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>PrepareTarget</code></td>
<td>Called by the recording system to notify the active <code>ReplayTarget</code> of an upcoming event.</td>
</tr>
<tr>
<td><code>RecordSnapshot</code></td>
<td>Store a replay snapshot in the replay target.</td>
</tr>
<tr>
<td><code>RestoreSnapshot</code></td>
<td>Recall a snapshot from the replay target based on the specified replay offset.</td>
</tr>
</tbody>
</table>

Top

### See Also

**Reference**
- `ReplayTarget Class`
- `UltimateReplay.Storage Namespace`
ReplayTargetPrepareTarget Method

Called by the recording system to notify the active ReplayTarget of an upcoming event.

**Namespace:** UltimateReplay.Storage  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

## Syntax

```c#
public abstract void PrepareTarget(
    ReplayTargetTask mode
)
```

### Parameters

- **mode**  
  Type: UltimateReplay.StorageReplayTargetTask  
  The ReplayTargetTask that the target should prepare for

## See Also

- **Reference**  
  - ReplayTarget Class  
  - UltimateReplay.Storage Namespace
ReplayTargetRecordSnapshot Method

Store a replay snapshot in the replay target.

**Namespace:** UltimateReplay.Storage  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```c#
public abstract void RecordSnapshot(
    ReplaySnapshot state
)
```

### Parameters

- **state**
  - Type: UltimateReplay.Storage.ReplaySnapshot
  - The snapshot to store

### See Also

Reference
- ReplayTarget Class
- UltimateReplay.Storage Namespace
ReplayTarget.RestoreSnapshot Method

Recall a snapshot from the replay target based on the specified replay offset.

**Namespace:** UltimateReplay.Storage  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public abstract ReplaySnapshot RestoreSnapshot(float offset)
```

### Parameters

**offset**  
Type: **System.Single**  
The offset pointing to the individual snapshot to recall

### Return Value

Type: **ReplaySnapshot**  
The replay snapshot at the specified offset

### See Also

Reference  
ReplayTarget Class  
UltimateReplay.Storage Namespace
ReplayTarget Properties

The ReplayTarget type exposes the following members.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duration</td>
<td>The amount of time in seconds that this recording lasts.</td>
</tr>
<tr>
<td>MemorySize</td>
<td>Get the total amount of bytes that this replay uses.</td>
</tr>
</tbody>
</table>

See Also

Reference
ReplayTarget Class
UltimateReplay.Storage Namespace
ReplayTargetDuration Property

The amount of time in seconds that this recording lasts.

**Namespace:** UltimateReplay.Storage  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```
public abstract float Duration { get; }
```

### Property Value

**Type:** Single

### See Also

**Reference**

- ReplayTarget Class
- UltimateReplay.Storage Namespace
ReplayTargetMemorySize Property

Get the total amount of bytes that this replay uses.

**Namespace:** UltimateReplay.Storage

**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0

(1.0.0.0)

### Syntax

```c#
public abstract int MemorySize { get; }
```

### Property Value

Type: Int32

### See Also

Reference

ReplayTarget Class

UltimateReplay.Storage Namespace
ReplayTargetTask Enumeration

Represents a task that can be issued to a ReplayTarget.

Namespace: UltimateReplay.Storage
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

## Syntax

```c#
public enum ReplayTargetTask
```

## Members

<table>
<thead>
<tr>
<th>Member name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commit</td>
<td>0</td>
<td>The replay target should commit all data currently in memory to its end destination. Similar to a flush method.</td>
</tr>
<tr>
<td>Discard</td>
<td>1</td>
<td>The replay target should discard any recorded data.</td>
</tr>
<tr>
<td>PrepareWrite</td>
<td>2</td>
<td>The replay target should prepare for subsequent write requests.</td>
</tr>
<tr>
<td>PrepareRead</td>
<td>3</td>
<td>The replay target should prepare for subsequent read requests.</td>
</tr>
</tbody>
</table>
See Also

Reference
UltimateReplay.Storage Namespace
## Classes

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BitConverterNonAlloc</td>
<td>Custom implementation of the BitConverter class that does not make any allocations. This is important as the methods may be called thousands of times per second.</td>
</tr>
<tr>
<td>MonoSingletonT</td>
<td>Singleton implementation using MonoBehaviour as a base class. A dedicated owner object is created so that the singleton instance can be attached to it.</td>
</tr>
</tbody>
</table>
BitConverterNonAlloc Class

Custom implementation of the BitConverter class that does not make any allocations. This is important as the methods may be called thousands of times per second.

Inheritance Hierarchy

System
Object
UltimateReplay.Util

BitConverterNonAlloc

Namespace: UltimateReplay.Util
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

Syntax

```csharp
public static class BitConverterNonAlloc
```

The BitConverterNonAlloc type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>GetBool</code></td>
<td>Retreive a 8-bit bool from the specified byte array.</td>
</tr>
<tr>
<td><code>GetBytes(Byte, Boolean)</code></td>
<td>Store an 8-bit bool into the specified byte array.</td>
</tr>
<tr>
<td><code>GetBytes(Byte, Int16)</code></td>
<td>Store a 16 bit int into the specified byte array. The buffer to store the int which must have a size of 2 or greaterThe</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GetBytes(Byte, Int32)</td>
<td>Store a 32-bit int into the specified byte array.</td>
</tr>
<tr>
<td>GetBytes(Byte, Single)</td>
<td>Store a 32-bit float into the specified byte array.</td>
</tr>
<tr>
<td>GetFloat</td>
<td>Retreive a 32-bit float from the specified byte array.</td>
</tr>
<tr>
<td>GetInt</td>
<td>Retreive a 32-bit int from the specified byte array.</td>
</tr>
<tr>
<td>GetShort</td>
<td>Retreive a 16-bit int from the specified byte array.</td>
</tr>
</tbody>
</table>

See Also

Reference
UltimateReplay.Util Namespace
BitConverterNonAlloc Methods

The BitConverterNonAlloc type exposes the following members.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetBool</td>
<td>Retreive a 8-bit bool from the specified byte array.</td>
</tr>
<tr>
<td>GetBytes(Boolean)</td>
<td>Store an 8-bit bool into the specified byte array.</td>
</tr>
<tr>
<td>GetBytes(Int16)</td>
<td>Store a 16 bit int into the specified byte array.</td>
</tr>
<tr>
<td></td>
<td>The buffer to store the int which must have a size of 2 or greaterThe short value to store</td>
</tr>
<tr>
<td>GetBytes(Int32)</td>
<td>Store a 32-bit int into the specified byte array.</td>
</tr>
<tr>
<td>GetBytes(Single)</td>
<td>Store a 32-bit float into the specified byte array.</td>
</tr>
<tr>
<td>GetFloat</td>
<td>Retreive a 32-bit float from the specified byte array.</td>
</tr>
<tr>
<td>GetInt</td>
<td>Retreive a 32-bit int from the specified byte array.</td>
</tr>
<tr>
<td>GetShort</td>
<td>Retreive a 16-bit int from the specified byte array.</td>
</tr>
</tbody>
</table>
See Also

Reference

BitConverterNonAlloc Class
UltimateReplay.Util Namespace
BitConverterNonAllocGetBool Method

Retrive a 8-bit bool from the specified byte array.

**Namespace:** UltimateReplay.Util  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```c#
public static bool GetBool(
    byte[] buffer
)
```

**Parameters**

*buffer*

Type: `SystemByte`  
The buffer to retrieve the bool from which must have a size of 1 or greater

**Return Value**

Type: `Boolean`  
The unpacked bool value

**See Also**

Reference  
BitConverterNonAlloc Class  
UltimateReplay.Util Namespace
## BitConverterNonAllocGetBytes Method

### Overload List

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetBytes(Byte, Boolean)</td>
<td>Store an 8-bit bool into the specified byte array.</td>
</tr>
<tr>
<td>GetBytes(Byte, Int16)</td>
<td>Store a 16 bit int into the specified byte array. The buffer to store the int which must have a size of 2 or greater The short value to store</td>
</tr>
<tr>
<td>GetBytes(Byte, Int32)</td>
<td>Store a 32-bit int into the specified byte array.</td>
</tr>
<tr>
<td>GetBytes(Byte, Single)</td>
<td>Store a 32-bit float into the specified byte array.</td>
</tr>
</tbody>
</table>

### See Also

Reference
- BitConverterNonAlloc Class
- UltimateReplay.Util Namespace
BitConverterNonAlloc.GetBytes Method (Byte, Boolean)

Store an 8-bit bool into the specified byte array.

**Namespace:** UltimateReplay.Util  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

## Syntax

```csharp
public static void GetBytes(
    byte[] buffer,
    bool value
)
```

### Parameters

**buffer**  
Type: System.Byte  
The buffer to store the bool which must have a size of 1 or greater

**value**  
Type: System.Boolean  
The bool value to store

## See Also

**Reference**  
BitConverterNonAlloc Class  
GetBytes Overload  
UltimateReplay.Util Namespace
BitConverterNonAlloc.GetBytes Method (Byte, Int16)

Store a 16 bit int into the specified byte array. The buffer to store the int which must have a size of 2 or greater. The short value to store.

**Namespace:** UltimateReplay.Util
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public static void GetBytes(
    byte[] buffer,
    short value
)
```

### Parameters

**buffer**
Type: `System.Byte`


**value**
Type: `System.Int16`


### See Also

Reference

BitConverterNonAlloc Class
getBytes Overload
UltimateReplay.Util Namespace
BitConverterNonAllocGetBytes Method (Byte, Int32)

Store a 32-bit int into the specified byte array.

**Namespace:** UltimateReplay.Util  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public static void GetBytes(
    byte[] buffer,
    int value
)
```

### Parameters

**buffer**
- Type: **SystemByte**  
  - The buffer to store the int which must have a size of 4 or greater

**value**
- Type: **SystemInt32**  
  - The int value to store

### See Also

**Reference**
- BitConverterNonAlloc Class  
- GetBytes Overload  
- UltimateReplay.Util Namespace
BitConverterNonAllocGetBytes Method (Byte, Single)

Store a 32-bit float into the specified byte array.

**Namespace:** UltimateReplay.Util

**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

**Syntax**

```c#
public static void GetBytes(
    byte[] buffer,
    float value
)
```

**Parameters**

- **buffer**
  - Type: System.Byte
  - The buffer to store the float which must have a size of 4 or greater

- **value**
  - Type: System.Single
  - The float value to store

**See Also**

- BitConverterNonAlloc Class
- GetBytes Overload
- UltimateReplay.Util Namespace
BitConverterNonAllocGetFloat Method

Retreive a 32-bit float from the specified byte array.

**Namespace:** UltimateReplay.Util  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```c#
public static float GetFloat(
    byte[] buffer
)
```

### Parameters

**buffer**  
Type: `SystemByte`  
The buffer to retreive the float from which must have a size of 4 or greater

### Return Value

Type: `Single`  
The unpacked float value

### See Also

Reference  
BitConverterNonAlloc Class  
UltimateReplay.Util Namespace
BitConverterNonAllocGetInt Method

Retrieve a 32-bit int from the specified byte array.

**Namespace:** UltimateReplay.Util  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

▲ Syntax

```csharp
public static int GetInt(  
    byte[] buffer
)
```

**Parameters**

*buffer*  
Type: SystemByte  
The buffer to retrieve the int from which must have a size of 4 or greater

**Return Value**

Type: Int32  
The unpacked int value

▲ See Also

**Reference**

BitConverterNonAlloc Class  
UltimateReplay.Util Namespace
BitConverterNonAllocGetShort Method

Retreive a 16-bit int from the specified byte array.

**Namespace:** UltimateReplay.Util
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0

Syntax

```c#
public static short GetShort(
    byte[] buffer
)
```

Parameters

`buffer`
Type: `System.Byte`
The buffer to retrieve the short from which must have a size of 2 or greater

Return Value

Type: `Int16`
The unpacked short value

See Also

Reference

BitConverterNonAlloc Class
UltimateReplay.Util Namespace
MonoSingleton\texttt{T} Class

Singleton implementation using \texttt{MonoBehaviour} as a base class. A
dedicated owner object is created so that the singleton instance can be
attached to it.

\section*{Inheritance Hierarchy}

\texttt{System}\rightarrow \texttt{Object} \\
\hspace{1cm} \texttt{Component} \\
\hspace{2cm} \texttt{Behaviour} \\
\hspace{3cm} \texttt{MonoBehaviour} \\
\hspace{4cm} \texttt{UltimateReplay.UtilMonoSingleton\texttt{T}} \\
\hspace{5cm} \texttt{UltimateReplay.ReplayManager}

\textbf{Namespace:} \texttt{UltimateReplay.Util} \\
\textbf{Assembly:} \texttt{UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0} \\
(1.0.0.0)

\section*{Syntax}

\begin{verbatim}
public abstract class MonoSingleton\texttt{T} : MonoBehaviour
where T : MonoSingleton\texttt{T}
\end{verbatim}

Type Parameters
\texttt{T}

The generic type that the singleton is wrapping

The \texttt{MonoSingleton\texttt{T}} type exposes the following members.

\section*{Constructors}
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MonoSingletonT</td>
<td>Initializes a new instance of the MonoSingletonT class</td>
</tr>
</tbody>
</table>

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Awake</td>
<td>Called by Unity. Allows the singleton to find and claim any scene instances.</td>
</tr>
<tr>
<td>ForceAwake</td>
<td>Attempts to forcefully create the singleton instance if one does not already exist.</td>
</tr>
<tr>
<td>OnApplicationQuit</td>
<td>Called by Unity. Allows the singleton to prevent recreation of the instance when the game is about to quit.</td>
</tr>
</tbody>
</table>

### Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>active</td>
<td>The active singleton instance or null if no instance has been created yet. Use Active to guarentee that the instance is non-null.</td>
</tr>
</tbody>
</table>

### Properties
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td>Get the active singleton instance. If not instance exists then one is created.</td>
</tr>
<tr>
<td>IsDisposing</td>
<td>Returns true if the singleton is currently disposing. This will only occur when the game is about to quit.</td>
</tr>
</tbody>
</table>

See Also

Reference
UltimateReplay.Util Namespace
MonoSingleton<T> Constructor

Initializes a new instance of the MonoSingletonT class

Namespace: UltimateReplay.Util
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

Syntax

```csharp
protected MonoSingleton()
```

See Also

Reference
MonoSingletonT Class
UltimateReplay.Util Namespace
MonoSingletonT Fields

The MonoSingletonT generic type exposes the following members.

Fields

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>active</td>
<td>The active singleton instance or null if no instance has been created yet. Use Active to guarantee that the instance is non-null.</td>
</tr>
</tbody>
</table>

See Also

Reference
MonoSingletonT Class
UltimateReplay.Util Namespace
MonoSingletonTactive Field

The active singleton instance or null if no instance has been created yet. Use Active to guarantee that the instance is non-null.

**Namespace:** UltimateReplay.Util  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```c#
protected static T active
```

Field Value

Type: $T$

### See Also

Reference

MonoSingletonT Class  
UltimateReplay.Util Namespace
The `MonoSingletonT` generic type exposes the following members.

## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Awake</td>
<td>Called by Unity. Allows the singleton to find and claim any scene instances.</td>
</tr>
<tr>
<td>ForceAwake</td>
<td>Attempts to forcefully create the singleton instance if one does not already exist.</td>
</tr>
<tr>
<td>OnApplicationQuit</td>
<td>Called by Unity. Allows the singleton to prevent recreation of the instance when the game is about to quit.</td>
</tr>
</tbody>
</table>

See Also

Reference

- `MonoSingletonT Class`
- `UltimateReplay.Util Namespace`
MonoSingletonT Awake Method

Called by Unity. Allows the singleton to find and claim any scene instances.

Namespace: UltimateReplay.Util
Assembly: UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

▶ Syntax

```c#
public virtual void Awake()
```

▶ See Also

Reference
MonoSingletonT Class
UltimateReplay.Util Namespace
MonoSingletonT ForceAwake Method

Attempts to forcefully create the singleton instance if one does not already exist.

**Namespace:** UltimateReplay.Util  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

## Syntax

```csharp
public static T ForceAwake()
```

Return Value

Type: `T`  
The active singleton instance

## See Also

Reference

MonoSingletonT Class  
UltimateReplay.Util Namespace
MonoSingletonT OnApplicationQuit Method

Called by Unity. Allows the singleton to prevent recreation of the instance when the game is about to quit.

**Namespace:** UltimateReplay.Util  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public void OnApplicationQuit()
```

### See Also

Reference  
MonoSingletonT Class  
UltimateReplay.Util Namespace
MonoSingleton\(T\) Properties

The `MonoSingleton\(T\)` generic type exposes the following members.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td>Get the active singleton instance. If not instance exists then one is created.</td>
</tr>
<tr>
<td>IsDisposing</td>
<td>Returns true if the singleton is currently disposing. This will only occur when the game is about to quit.</td>
</tr>
</tbody>
</table>

### See Also

Reference

- `MonoSingleton\(T\)` Class
- `UltimateReplay.Util` Namespace
MonoSingleton\text{T}\text{Active} Property

Get the active singleton instance. If not instance exists then one is created.

\textbf{Namespace:} UltimateReplay.Util  
\textbf{Assembly:} UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

\textbf{Syntax}

\begin{verbatim}
public static T Active { get; }
\end{verbatim}

\textbf{Property Value}

Type: \text{T}

\textbf{See Also}

Reference

MonoSingleton\text{T} Class  
UltimateReplay.Util Namespace
MonoSingletonTIsDisposing Property

Returns true if the singleton is currently disposing. This will only occur when the game is about to quit.

**Namespace:** UltimateReplay.Util  
**Assembly:** UltimateReplay (in UltimateReplay.dll) Version: 1.0.0.0 (1.0.0.0)

### Syntax

```csharp
public static bool IsDisposing { get; }
```

**Property Value**

Type: *Boolean*

### See Also

**Reference**

MonoSingletonT Class  
UltimateReplay.Util Namespace