Table of Contents

Namespace UltimateReplay

Classes

HexConverter

ReplayAnimator

ReplayAsyncOperation

An awaitable object that is used to report when an async operation has finished.

ReplayAsyncOperation<T>

ReplayAudio

Used to record and replay an AudioSource component

ReplayBehaviour

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.

ReplayBlendShape

ReplayComponentEnabledState

A replay component used to record the enabled state of a behaviour component.

ReplayComponentPreparerAttribute

Use this attribute to register a type as a component preparer. This attribute only works in conjunction with the DefaultReplayPreparer.

ReplayControls

ReplayControls.HighlightButton

ReplayControls.SliderCallback

Helper class used to detect drag start and end events on UI slider control (Seek slider bar).

ReplayEnabledState

A replay component used to record the enabled state of a game object.

ReplayIgnoreAttribute

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 dont need to record any data.

ReplayLineRenderer

Recorder component used to record and replay the Unity line renderer component.

ReplayManager

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

ReplayMaterial

ReplayMaterialChange

ReplayMetadata

Stores all additional non-essential information about a replay. Can be useful to help display information about the replay such as when it was created, or which Unity scene is required for best playback accuracy. You can also derive from this class to add additional custom metadata fields that you would like to save. Note that only primitive types and arrays will be serialized and only reference types that are marked as SerializableAttribute will be saved (Serialization follows standard Unity practices but does not support reference types unless marked as serializable).

ReplayMethodAttribute

Use this attribute to mark a method declared in a ReplayBehaviour script as recordable. The target method must not return a value and must only use primitive parameter types up to a limit of 4 arguments.

ReplayObject

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

ReplayOperation

Represents a dedicated replay operation in progress. Provides access to API's common to both recording and playback operations.

ReplayParentChange

ReplayParticleSystem

A replay component which can be used to record and replay the Unity ParticleSystem.

ReplayParticleSystemV2

ReplayPlaybackOperation

Represents a dedicated playback operation in progress. Provides access to all playback replated API's for a specific playback operation.

ReplayPlaybackOptions

A number of options used to control the playback behaviour.

ReplayPreparerIgnoreAttribute

ReplayRecordOperation

Represents a dedicated record operation in progress. Provides access to all recording related API's for a specific record operation.

ReplayRecordOptions

A number of options that can be used to control the record behaviour.

ReplayRecordableBehaviour

Derive from this base class to create custom recorder components.

ReplayRiggedGeneric

ReplayRiggedHumanoid

ReplayScene

A ReplayScene contains information about all active replay objects.

ReplaySettings

Stores global settings used by the replay system.

ReplayState

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

ReplayTokenSerializeAttribute

Attribute used to mark members as serializable using a text format. The serialized name can be specified via the attribute or the member name will be used if no name is provided.

ReplayTrailRenderer

ReplayTransform

ReplayVarAttribute

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

Structs

ReplayIdentity

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 use to identify prefab instances that are spawned during a replay.

ReplayObject.ReplayObjectReference

Interfaces

IReplaySerialize

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

Enums

PlaybackDirection

The playback direction used during replay playback.

PlaybackEndBehaviour

Used to indicate what should happen when the end of a replay is reached.

PlaybackOrigin

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

PlaybackSeekSnap

The playback seek behaviour that will be used when seeking to a certain time stamp.

RecordAxisFlags

Flags to specify which elements of an axis should be recorded. In supported components, you can also specify that the axis should be interpolated for smoother replays.

RecordFullAxisFlags

RecordPrecision

Specify how much precision is required when serializing a particular value. Use lower precisions where possible to save on storage space and overall performance.

RecordSpace

For transform related replay components, specify whether local or world space should be used for recording.

ReplayAnimator.ReplayIKFlags

ReplayLineRenderer.ReplayLineRendererFlags

Flags used to specify which features are enabled on the recorder.

ReplayMaterial.ReplayMaterialFlags

ReplayMaterialChange.ReplayMaterialChangeFlags

ReplayParticleSystem.ReplayParticleSystemFlags

Replay flags used to determine which component features are enabled.

ReplaySceneMode

The scene state value used to determine which mode a particular scene instance is in.

ReplayTrailRenderer.ReplayTrailRendererFlags

ReplayUpdateMode

The update method used by the replay manager for all recording and replaying samples.

RestoreSceneMode

Class HexConverter

Inheritance

object
HexConverter

Inherited Members

object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
public static class HexConverter
```

Methods

FromHexStringInt32(string)

Declaration

```
public static int FromHexStringInt32(string hex)
```

Parameters

TYPE	NAME	DESCRIPTION
string	hex	

Returns

TYPE	DESCRIPTION
int	

FromHexStringSingle(string)

Declaration

```
public static float FromHexStringSingle(string hex)
```

Parameters

TYPE	NAME	DESCRIPTION
string	hex	

Returns

TYPE	DESCRIPTION
float	

GetHexBytes(string, byte[], int)

Declaration

```
public static void GetHexBytes(string hex, byte[] bytes, int offset)
```

Parameters

TYPE	NAME	DESCRIPTION
string	hex	
byte[]	bytes	
int	offset	

GetHexString(byte[], int, int)

Declaration

```
public static string GetHexString(byte[] bytes, int offset, int length)
```

Parameters

TYPE	NAME	DESCRIPTION
byte[]	bytes	
int	offset	
int	length	

Returns

TYPE	DESCRIPTION
string	

GetHexValue(byte, out char, out char)

Declaration

```
public static void GetHexValue(byte val, out char a, out char b)
```

Parameters

TYPE	NAME	DESCRIPTION
byte	val	
char	a	
char	b	

GetHexValue(char)

Declaration

```
public static int GetHexValue(char hex)
```

Parameters

TYPE	NAME	DESCRIPTION
char	hex	

Returns

TYPE	DESCRIPTION
int	

ToHexString(int)

Declaration

```
public static string ToHexString(int value)
```

Parameters

TYPE	NAME	DESCRIPTION
int	value	

Returns

TYPE	DESCRIPTION
string	

ToHexString(float)

Declaration

```
public static string ToHexString(float value)
```

Parameters

TYPE	NAME	DESCRIPTION
float	value	

Returns

TYPE	DESCRIPTION
string	

Interface IReplaySerialize

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

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
public interface IReplaySerialize
```

Methods

OnReplayDeserialize(ReplayState)

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

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState to read the data from

OnReplaySerialize(ReplayState)

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

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState to write the data to

Enum PlaybackDirection

The playback direction used during replay playback.

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
public enum PlaybackDirection
```

Fields

NAME	DESCRIPTION
Backward	The replay should be played back in reverse mode.
Forward	The replay should be played back in normal mode.

Enum PlaybackEndBehaviour

Used to indicate what should happen when the end of a replay is reached.

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
public enum PlaybackEndBehaviour
```

Fields

NAME	DESCRIPTION
EndPlayback	The playback service should automatically end the replay and trigger and playback end events listeners. The active replay scene will also be reverted to live mode causing physics objects and scripts to be re-activated.
LoopPlayback	The playback service should loop back around to the start of the replay and continue playing. The replay will play indefinitely until ReplayManager.StopPlayback(ref ReplayHandle, bool) is called.
StopPlayback	The playback service should stop the playback and return to the start of the replay. The active replay scene will remain in playback mode and you will need to call ReplayManager.StopPlayback(ref ReplayHandle, bool) manually to end playback.

Enum PlaybackOrigin

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

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
public enum PlaybackOrigin
```

Fields

NAME	DESCRIPTION
Current	The current frame in the playback sequence.
End	The end of the playback sequence.
Start	The start of the playback sequence.

Enum PlaybackSeekSnap

The playback seek behaviour that will be used when seeking to a certain time stamp.

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
public enum PlaybackSeekSnap
```

Fields

NAME	DESCRIPTION
Smooth	The replay system will interpolate between frames if possible when seeking. Seeking will give a smooth seamless transition if replay components support interpolation.
SnapToFrame	The replay system will constrain seeking to snapshot frames. Can give a notchy effect when seeking as the time stamp snaps to the nearest snapshot frame.

Enum RecordAxisFlags

Flags to specify which elements of an axis should be recorded. In supported components, you can also specify that the axis should be interpolated for smoother replays.

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
[Flags]
public enum RecordAxisFlags
```

Fields

NAME	DESCRIPTION
Interpolate	The axis values will be interpolated during playback for smoother replays.
None	No data will be recorded or updated
X	The X component of the transform element should be recorded.
XYZ	All axis of the transform element should be recorded. For rotation elements, full axis rotation will be recorded as quaternion.
XYZInterpolate	All axis of the transform element should be recorded with full interpolation. For rotation elements, full axis rotation will be recorded as quaternion.
Y	The Y component of the transform element should be recorded.
Z	The Z component of the transform element should be recorded.

Enum RecordFullAxisFlags

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
public enum RecordFullAxisFlags
```

Fields

NAME	DESCRIPTION
Interpolate	
None	
XYZ	
XYZInterpolate	

Enum RecordPrecision

Specify how much precision is required when serializing a particular value. Use lower precisions where possible to save on storage space and overall performance.

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
public enum RecordPrecision
```

Fields

NAME	DESCRIPTION
FullPrecision32Bit	Record value in full 32-bit precision, assuming value type is Single.
HalfPrecision16Bit	Record value in half 16-bit precision to reduce space. Generally a floating point value serialize at half precision will remain accurate to roughly 3 decimal places, depending upon usage. Recommended for objects that don't move much, are close to the origin, and not in main focus of the active rendering camera such as player controller.

Enum RecordSpace

For transform related replay components, specify whether local or world space should be used for recording.

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
public enum RecordSpace
```

Fields

NAME	DESCRIPTION
Local	Record the associated transform data using local space. Recommended for child transforms such as bone hierarchies or similar.
World	Record the associated transform data using world space.

Class ReplayAnimator

Inheritance

object
Object
Component
Behaviour
MonoBehaviour
ReplayBehaviour
ReplayRecordableBehaviour
ReplayAnimator

Implements

IReplaySerialize

Inherited Members

ReplayBehaviour.ReplayIdentity
ReplayBehaviour.ReplayObject
ReplayBehaviour.HasPersistentData
ReplayBehaviour.ReplayPersistentData
ReplayBehaviour.Variables
ReplayBehaviour.HasVariables
ReplayBehaviour.IsRecording
ReplayBehaviour.IsRecordingPaused
ReplayBehaviour.IsRecordingOrPaused
ReplayBehaviour.IsReplaying
ReplayBehaviour.IsPlaybackPaused
ReplayBehaviour.IsReplayingOrPaused
ReplayBehaviour.PlaybackTime
ReplayBehaviour.PlaybackTimeNormalized
ReplayBehaviour.PlaybackTimeScale
ReplayBehaviour.PlaybackDirection
ReplayBehaviour.ForceRegenerateIdentity()
ReplayBehaviour.RecordVariable(ReplayVariable)
ReplayBehaviour.RecordEvent(ushort, ReplayState)
ReplayBehaviour.RecordMethodCall(Action)
ReplayBehaviour.RecordMethodCall<T>(Action<T>, T)
ReplayBehaviour.RecordMethodCall<T0, T1>(Action<T0, T1>, T0, T1)
ReplayBehaviour.RecordMethodCall<T0, T1, T2>(Action<T0, T1, T2>, T0, T1, T2)
ReplayBehaviour.RecordMethodCall<T0, T1, T2, T3>(Action<T0, T1, T2, T3>, T0, T1, T2, T3)
MonoBehaviour.IsInvoking()
MonoBehaviour.CancelInvoke()
MonoBehaviour.Invoke(string, float)
MonoBehaviour.InvokeRepeating(string, float, float)
MonoBehaviour.CancelInvoke(string)
MonoBehaviour.IsInvoking(string)
MonoBehaviour.StartCoroutine(string)
MonoBehaviour.StartCoroutine(string, object)
MonoBehaviour.StartCoroutine(IEnumerator)
MonoBehaviour.StartCoroutine_Auto(IEnumerator)
MonoBehaviour.StopCoroutine(IEnumerator)
MonoBehaviour.StopCoroutine(Coroutine)
MonoBehaviour.StopCoroutine(string)

MonoBehaviour.StopAllCoroutines()

MonoBehaviour.print(object)

MonoBehaviour.useGUILayout

MonoBehaviour.runInEditMode

Behaviour.enabled

Behaviour.isActiveAndEnabled

Component.GetComponent(Type)

Component.GetComponent<T>()

Component.TryGetComponent(Type, out Component)

Component.TryGetComponent<T>(out T)

Component.GetComponent(string)

Component.GetComponentInChildren(Type, bool)

Component.GetComponentInChildren(Type)

Component.GetComponentInChildren<T>(bool)

Component.GetComponentInChildren<T>()

Component.GetComponentsInChildren(Type, bool)

Component.GetComponentsInChildren(Type)

Component.GetComponentsInChildren<T>(bool)

Component.GetComponentsInChildren<T>(bool, List<T>)

Component.GetComponentsInChildren<T>()

Component.GetComponentsInChildren<T>(List<T>)

Component.GetComponentInParent(Type, bool)

Component.GetComponentInParent(Type)

Component.GetComponentInParent<T>(bool)

Component.GetComponentInParent<T>()

Component.GetComponentsInParent(Type, bool)

Component.GetComponentsInParent(Type)

Component.GetComponentsInParent<T>(bool)

Component.GetComponentsInParent<T>(bool, List<T>)

Component.GetComponentsInParent<T>()

Component.GetComponents(Type)

Component.GetComponents(Type, List<Component>)

Component.GetComponents<T>(List<T>)

Component.GetComponents<T>()

Component.CompareTag(string)

Component.SendMessageUpwards(string, object, SendMessageOptions)

Component.SendMessageUpwards(string, object)

Component.SendMessageUpwards(string)

Component.SendMessageUpwards(string, SendMessageOptions)

Component.SendMessage(string, object)

Component.SendMessage(string)

Component.SendMessage(string, object, SendMessageOptions)

Component.SendMessage(string, SendMessageOptions)

Component.BroadcastMessage(string, object, SendMessageOptions)

Component.BroadcastMessage(string, object)

Component.BroadcastMessage(string)

Component.BroadcastMessage(string, SendMessageOptions)

Component.transform

Component.gameObject

Component.tag

Object.GetInstanceID()

Object.GetHashCode()

Object.Equals(object)
Object.Instantiate(Object, Vector3, Quaternion)
Object.Instantiate(Object, Vector3, Quaternion, Transform)
Object.Instantiate(Object)
Object.Instantiate(Object, Transform)
Object.Instantiate(Object, Transform, bool)
Object.Instantiate<T>(T)
Object.Instantiate<T>(T, Vector3, Quaternion)
Object.Instantiate<T>(T, Vector3, Quaternion, Transform)
Object.Instantiate<T>(T, Transform)
Object.Instantiate<T>(T, Transform, bool)
Object.Destroy(Object, float)
Object.Destroy(Object)
Object.DestroyImmediate(Object, bool)
Object.DestroyImmediate(Object)
Object.FindObjectsOfType(Type)
Object.FindObjectsOfType(Type, bool)
Object.DontDestroyOnLoad(Object)
Object.DestroyObject(Object, float)
Object.DestroyObject(Object)
Object.FindSceneObjectsOfType(Type)
Object.FindObjectsOfTypeIncludingAssets(Type)
Object.FindObjectsOfType<T>()
Object.FindObjectsOfType<T>(bool)
Object.FindObjectOfType<T>()
Object.FindObjectOfType<T>(bool)
Object.FindObjectsOfTypeAll(Type)
Object.FindObjectOfType(Type)
Object.FindObjectOfType(Type, bool)
Object.ToString()
Object.name
Object.hideFlags
object.Equals(object, object)
object.GetType()
object.ReferenceEquals(object, object)

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
[DisallowMultipleComponent]
public sealed class ReplayAnimator : ReplayRecordableBehaviour, IReplaySerialize
```

Fields

observedAnimator

Declaration

```
public Animator observedAnimator
```

Field Value

TYPE	DESCRIPTION
Animator	

Properties

Formatter

An optional ReplayFormatter that is used to serialize a particular component. Providing a formatter via his property can greatly reduce the amount of data that a ReplayObject needs to store.

Declaration

```
public override ReplayFormatter Formatter { get; }
```

Property Value

TYPE	DESCRIPTION
ReplayFormatter	

Overrides

ReplayRecordableBehaviour.Formatter

Interpolate

Declaration

```
public bool Interpolate { get; set; }
```

Property Value

TYPE	DESCRIPTION
bool	

InterpolateParameters

Declaration

```
public bool InterpolateParameters { get; set; }
```

Property Value

TYPE	DESCRIPTION
bool	

RecordPrecision

Declaration

```
public RecordPrecision RecordPrecision { get; set; }
```

Property Value

TYPE	DESCRIPTION
RecordPrecision	

ReplayIKPositionTargets

Declaration

```
public bool ReplayIKPositionTargets { get; set; }
```

Property Value

TYPE	DESCRIPTION
bool	

ReplayIKRotationTargets

Declaration

```
public bool ReplayIKRotationTargets { get; set; }
```

Property Value

TYPE	DESCRIPTION
bool	

ReplayIKWeights

Declaration

```
public bool ReplayIKWeights { get; set; }
```

Property Value

TYPE	DESCRIPTION
bool	

ReplayParameters

Declaration

```
public bool ReplayParameters { get; set; }
```

Property Value

TYPE	DESCRIPTION
bool	

Methods

Awake()

Called by Unity.

Declaration

```
protected override void Awake()
```

Overrides

ReplayBehaviour.Awake()

OnReplayDeserialize(ReplayState)

Called by the replay system when replay data should be deserialized and restored.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState containing the previously recorded data

Overrides

ReplayRecordableBehaviour.OnReplayDeserialize(ReplayState)

OnReplayEnd()

Called by the replay system when playback will end.

Declaration

```
protected override void OnReplayEnd()
```

Overrides

ReplayBehaviour.OnReplayEnd()

OnReplayPlayPause(bool)

Called by the replay system when playback will be paused or resumed.

Declaration

```
protected override void OnReplayPlayPause(bool paused)
```

Parameters

TYPE	NAME	DESCRIPTION
bool	paused	True if playback is pausing or false if it is resuming

Overrides

ReplayBehaviour.OnReplayPlayPause(bool)

OnReplayReset()

Called by the replay system when preserved data should be reset.

Declaration

```
protected override void OnReplayReset()
```

Overrides

ReplayBehaviour.OnReplayReset()

OnReplaySerialize(ReplayState)

Called by the replay system when recorded data should be captured and serialized.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState used to store the recorded data

Overrides

ReplayRecordableBehaviour.OnReplaySerialize(ReplayState)

OnReplayStart()

Called by the replay system when playback is about to begin.

Declaration

```
protected override void OnReplayStart()
```

Overrides

ReplayBehaviour.OnReplayStart()

OnReplayUpdate(float)

Called by the replay system when the playback will be updated. Use this method to perform interpolation and smoothing processes.

Declaration

```
protected override void OnReplayUpdate(float t)
```

Parameters

TYPE	NAME	DESCRIPTION
float	t	The delta value from 0-1 between current replay snapshots

Overrides

ReplayBehaviour.OnReplayUpdate(float)

Reset()

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

Declaration

```
protected override void Reset()
```

Overrides

ReplayBehaviour.Reset()

Implements

IReplaySerialize

Enum ReplayAnimator.ReplayIKFlags

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
[Flags]
public enum ReplayAnimator.ReplayIKFlags
```

Fields

NAME	DESCRIPTION
None	
Position	
Rotation	
Weights	

Class ReplayAsyncOperation

An awaitable object that is used to report when an async operation has finished.

Inheritance

object
CustomYieldInstruction
ReplayAsyncOperation
ReplayAsyncOperation<T>

Implements

IEnumerator

Inherited Members

CustomYieldInstruction.MoveNext()
CustomYieldInstruction.Reset()
CustomYieldInstruction.Current
object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
public class ReplayAsyncOperation : CustomYieldInstruction, IEnumerator
```

Properties

Error

Declaration

```
public string Error { get; }
```

Property Value

TYPE	DESCRIPTION
string	

IsDone

Check whether the associated async operation has finished.

Declaration

```
public bool IsDone { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

Progress

Declaration

```
public float Progress { get; }
```

Property Value

TYPE	DESCRIPTION
float	

Success

Check whether the associated async operation was successful.

Declaration

```
public bool Success { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

keepWaiting

Returns true if the associated async operation is not yet completed.

Declaration

```
public override bool keepWaiting { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

Overrides

UnityEngine.CustomYieldInstruction.keepWaiting

Implements

IEnumerator

Class ReplayAsyncOperation<T>

Inheritance

object
CustomYieldInstruction
ReplayAsyncOperation
ReplayAsyncOperation<T>

Implements

IEnumerator

Inherited Members

ReplayAsyncOperation.keepWaiting
ReplayAsyncOperation.IsDone
ReplayAsyncOperation.Success
ReplayAsyncOperation.Progress
ReplayAsyncOperation.Error
CustomYieldInstruction.MoveNext()
CustomYieldInstruction.Reset()
CustomYieldInstruction.Current
object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
public sealed class ReplayAsyncOperation<T> : ReplayAsyncOperation, IEnumerator
```

Type Parameters

NAME	DESCRIPTION
T	

Properties

Result

Declaration

```
public T Result { get; }
```

Property Value

TYPE	DESCRIPTION
T	

Implements

IEnumerator

Class ReplayAudio

Used to record and replay an AudioSource component

Inheritance

object
Object
Component
Behaviour
MonoBehaviour
ReplayBehaviour
ReplayRecordableBehaviour
ReplayAudio

Implements

IReplaySerialize

Inherited Members

ReplayRecordableBehaviour.Formatter
ReplayRecordableBehaviour.OnDestroy()
ReplayBehaviour.ReplayIdentity
ReplayBehaviour.ReplayObject
ReplayBehaviour.HasPersistentData
ReplayBehaviour.ReplayPersistentData
ReplayBehaviour.Variables
ReplayBehaviour.HasVariables
ReplayBehaviour.IsRecording
ReplayBehaviour.IsRecordingPaused
ReplayBehaviour.IsRecordingOrPaused
ReplayBehaviour.IsReplaying
ReplayBehaviour.IsPlaybackPaused
ReplayBehaviour.IsReplayingOrPaused
ReplayBehaviour.PlaybackTime
ReplayBehaviour.PlaybackTimeNormalized
ReplayBehaviour.PlaybackTimeScale
ReplayBehaviour.PlaybackDirection
ReplayBehaviour.OnEnable()
ReplayBehaviour.OnDisable()
ReplayBehaviour.OnReplayStart()
ReplayBehaviour.OnReplayEnd()
ReplayBehaviour.OnReplayCapture()
ReplayBehaviour.OnReplaySpawned(Vector3, Quaternion)
ReplayBehaviour.ForceRegenerateIdentity()
ReplayBehaviour.RecordVariable(ReplayVariable)
ReplayBehaviour.RecordEvent(ushort, ReplayState)
ReplayBehaviour.RecordMethodCall(Action)
ReplayBehaviour.RecordMethodCall<T>(Action<T>, T)
ReplayBehaviour.RecordMethodCall<T0, T1>(Action<T0, T1>, T0, T1)
ReplayBehaviour.RecordMethodCall<T0, T1, T2>(Action<T0, T1, T2>, T0, T1, T2)
ReplayBehaviour.RecordMethodCall<T0, T1, T2, T3>(Action<T0, T1, T2, T3>, T0, T1, T2, T3)
MonoBehaviour.IsInvoking()
MonoBehaviour.CancelInvoke()
MonoBehaviour.Invoke(string, float)

MonoBehaviour.InvokeRepeating(string, float, float)
MonoBehaviour.CancelInvoke(string)
MonoBehaviour.IsInvoking(string)
MonoBehaviour.StartCoroutine(string)
MonoBehaviour.StartCoroutine(string, object)
MonoBehaviour.StartCoroutine(IEnumerator)
MonoBehaviour.StartCoroutine_Auto(IEnumerator)
MonoBehaviour.StopCoroutine(IEnumerator)
MonoBehaviour.StopCoroutine(Coroutine)
MonoBehaviour.StopCoroutine(string)
MonoBehaviour.StopAllCoroutines()
MonoBehaviour.print(object)
MonoBehaviour.useGUILayout
MonoBehaviour.runInEditMode
Behaviour.enabled
Behaviour.isActiveAndEnabled
Component.GetComponent(Type)
Component.GetComponent<T>()
Component.TryGetComponent(Type, out Component)
Component.TryGetComponent<T>(out T)
Component.GetComponent(string)
Component.GetComponentInChildren(Type, bool)
Component.GetComponentInChildren(Type)
Component.GetComponentInChildren<T>(bool)
Component.GetComponentInChildren<T>()
Component.GetComponentsInChildren(Type, bool)
Component.GetComponentsInChildren(Type)
Component.GetComponentsInChildren<T>(bool)
Component.GetComponentsInChildren<T>(bool, List<T>)
Component.GetComponentsInChildren<T>()
Component.GetComponentsInChildren<T>(List<T>)
Component.GetComponentInParent(Type, bool)
Component.GetComponentInParent(Type)
Component.GetComponentInParent<T>(bool)
Component.GetComponentInParent<T>()
Component.GetComponentsInParent(Type, bool)
Component.GetComponentsInParent(Type)
Component.GetComponentsInParent<T>(bool)
Component.GetComponentsInParent<T>(bool, List<T>)
Component.GetComponentsInParent<T>()
Component.GetComponents(Type)
Component.GetComponents(Type, List<Component>)
Component.GetComponents<T>(List<T>)
Component.GetComponents<T>()
Component.CompareTag(string)
Component.SendMessageUpwards(string, object, SendMessageOptions)
Component.SendMessageUpwards(string, object)
Component.SendMessageUpwards(string)
Component.SendMessageUpwards(string, SendMessageOptions)
Component.SendMessage(string, object)
Component.SendMessage(string)
Component.SendMessage(string, object, SendMessageOptions)

Component.SendMessage(string, SendMessageOptions)
Component.BroadcastMessage(string, object, SendMessageOptions)
Component.BroadcastMessage(string, object)
Component.BroadcastMessage(string)
Component.BroadcastMessage(string, SendMessageOptions)
Component.transform
Component.gameObject
Component.tag
Object.GetInstanceID()
Object.GetHashCode()
Object.Equals(object)
Object.Instantiate(Object, Vector3, Quaternion)
Object.Instantiate(Object, Vector3, Quaternion, Transform)
Object.Instantiate(Object)
Object.Instantiate(Object, Transform)
Object.Instantiate(Object, Transform, bool)
Object.Instantiate<T>(T)
Object.Instantiate<T>(T, Vector3, Quaternion)
Object.Instantiate<T>(T, Vector3, Quaternion, Transform)
Object.Instantiate<T>(T, Transform)
Object.Instantiate<T>(T, Transform, bool)
Object.Destroy(Object, float)
Object.Destroy(Object)
Object.DestroyImmediate(Object, bool)
Object.DestroyImmediate(Object)
Object.FindObjectsOfType(Type)
Object.FindObjectsOfType(Type, bool)
Object.DontDestroyOnLoad(Object)
Object.DestroyObject(Object, float)
Object.DestroyObject(Object)
Object.FindSceneObjectsOfType(Type)
Object.FindObjectsOfTypeIncludingAssets(Type)
Object.FindObjectsOfType<T>()
Object.FindObjectsOfType<T>(bool)
Object.FindObjectOfType<T>()
Object.FindObjectOfType<T>(bool)
Object.FindObjectsOfTypeAll(Type)
Object.FindObjectOfType(Type)
Object.FindObjectOfType(Type, bool)
Object.ToString()
Object.name
Object.hideFlags
object.Equals(object, object)
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
public class ReplayAudio : ReplayRecordableBehaviour, IReplaySerialize
```

Fields

observedAudio

The AudioSource component that will be observed during recording and used for playback during replays. Only a single AudioClip is supported and should be assigned to the AudioSource.

Declaration

```
public AudioSource observedAudio
```

Field Value

TYPE	DESCRIPTION
AudioSource	

Methods

Awake()

Called by Unity.

Declaration

```
protected override void Awake()
```

Overrides

ReplayBehaviour.Awake()

OnReplayDeserialize(ReplayState)

Called by the replay system when the replay component should deserialize previously recorded data.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState to read from

Overrides

ReplayRecordableBehaviour.OnReplayDeserialize(ReplayState)

OnReplayEvent(ushort, ReplayState)

Called by the replay system when an event occurs.

Declaration

```
protected override void OnReplayEvent(ushort eventID, ReplayState eventData)
```

Parameters

TYPE	NAME	DESCRIPTION
ushort	eventID	

TYPE	NAME	DESCRIPTION
ReplayState	eventData	

Overrides

ReplayBehaviour.OnReplayEvent(ushort, ReplayState)

OnReplayPlayPause(bool)

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

Declaration

```
protected override void OnReplayPlayPause(bool paused)
```

Parameters

TYPE	NAME	DESCRIPTION
bool	paused	True if the replay system is paused or false if it is resuming

Overrides

ReplayBehaviour.OnReplayPlayPause(bool)

OnReplayReset()

Caled by the replay system when the component should reset any persistent data.

Declaration

```
protected override void OnReplayReset()
```

Overrides

ReplayBehaviour.OnReplayReset()

OnReplaySerialize(ReplayState)

Called by the replay system when the replay component should serialize its recorded data.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState to write to

Overrides

ReplayRecordableBehaviour.OnReplaySerialize(ReplayState)

OnReplayUpdate(float)

Called by the replay system during playback mode.

Declaration

```
protected override void OnReplayUpdate(float t)
```

Parameters

TYPE	NAME	DESCRIPTION
float	t	

Overrides

ReplayBehaviour.OnReplayUpdate(float)

Reset()

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

Declaration

```
protected override void Reset()
```

Overrides

ReplayBehaviour.Reset()

Implements

IReplaySerialize

Class ReplayBehaviour

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

object
Object
Component
Behaviour
MonoBehaviour
ReplayBehaviour
ReplayRecordableBehaviour

Inherited Members

MonoBehaviour.IsInvoking()
MonoBehaviour.CancelInvoke()
MonoBehaviour.Invoke(string, float)
MonoBehaviour.InvokeRepeating(string, float, float)
MonoBehaviour.CancelInvoke(string)
MonoBehaviour.IsInvoking(string)
MonoBehaviour.StartCoroutine(string)
MonoBehaviour.StartCoroutine(string, object)
MonoBehaviour.StartCoroutine(IEnumerator)
MonoBehaviour.StartCoroutine_Auto(IEnumerator)
MonoBehaviour.StopCoroutine(IEnumerator)
MonoBehaviour.StopCoroutine(Coroutine)
MonoBehaviour.StopCoroutine(string)
MonoBehaviour.StopAllCoroutines()
MonoBehaviour.print(object)
MonoBehaviour.useGUILayout
MonoBehaviour.runInEditMode
Behaviour.enabled
Behaviour.isActiveAndEnabled
Component.GetComponent(Type)
Component.GetComponent<T>()
Component.TryGetComponent(Type, out Component)
Component.TryGetComponent<T>(out T)
Component.GetComponent(string)
Component.GetComponentInChildren(Type, bool)
Component.GetComponentInChildren(Type)
Component.GetComponentInChildren<T>(bool)
Component.GetComponentInChildren<T>()
Component.GetComponentsInChildren(Type, bool)
Component.GetComponentsInChildren(Type)
Component.GetComponentsInChildren<T>(bool)
Component.GetComponentsInChildren<T>(bool, List<T>)
Component.GetComponentsInChildren<T>()
Component.GetComponentsInChildren<T>(List<T>)
Component.GetComponentInParent(Type, bool)
Component.GetComponentInParent(Type)
Component.GetComponentInParent<T>(bool)

Component.GetComponentInParent<T>()
Component.GetComponentsInParent(Type, bool)
Component.GetComponentsInParent(Type)
Component.GetComponentsInParent<T>(bool)
Component.GetComponentsInParent<T>(bool, List<T>)
Component.GetComponentsInParent<T>()
Component.GetComponents(Type)
Component.GetComponents(Type, List<Component>)
Component.GetComponents<T>(List<T>)
Component.GetComponents<T>()
Component.CompareTag(string)
Component.SendMessageUpwards(string, object, SendMessageOptions)
Component.SendMessageUpwards(string, object)
Component.SendMessageUpwards(string)
Component.SendMessageUpwards(string, SendMessageOptions)
Component.SendMessage(string, object)
Component.SendMessage(string)
Component.SendMessage(string, object, SendMessageOptions)
Component.SendMessage(string, SendMessageOptions)
Component.BroadcastMessage(string, object, SendMessageOptions)
Component.BroadcastMessage(string, object)
Component.BroadcastMessage(string)
Component.BroadcastMessage(string, SendMessageOptions)
Component.transform
Component.gameObject
Component.tag
Object.GetInstanceID()
Object.GetHashCode()
Object.Equals(object)
Object.Instantiate(Object, Vector3, Quaternion)
Object.Instantiate(Object, Vector3, Quaternion, Transform)
Object.Instantiate(Object)
Object.Instantiate(Object, Transform)
Object.Instantiate(Object, Transform, bool)
Object.Instantiate<T>(T)
Object.Instantiate<T>(T, Vector3, Quaternion)
Object.Instantiate<T>(T, Vector3, Quaternion, Transform)
Object.Instantiate<T>(T, Transform)
Object.Instantiate<T>(T, Transform, bool)
Object.Destroy(Object, float)
Object.Destroy(Object)
Object.DestroyImmediate(Object, bool)
Object.DestroyImmediate(Object)
Object.FindObjectsOfType(Type)
Object.FindObjectsOfType(Type, bool)
Object.DontDestroyOnLoad(Object)
Object.DestroyObject(Object, float)
Object.DestroyObject(Object)
Object.FindSceneObjectsOfType(Type)
Object.FindObjectsOfTypeIncludingAssets(Type)
Object.FindObjectsOfType<T>()
Object.FindObjectsOfType<T>(bool)

Object.FindObjectOfType<T>()
Object.FindObjectOfType<T>(bool)
Object.FindObjectsOfTypeAll(Type)
Object.FindObjectOfType(Type)
Object.FindObjectOfType(Type, bool)
Object.ToString()
Object.name
Object.hideFlags
object.Equals(object, object)
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
public abstract class ReplayBehaviour : MonoBehaviour
```

Properties

HasPersistentData

Declaration

```
public bool HasPersistentData { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

HasVariables

Returns a value indicating whether this ReplayObject has any ReplayVariable.

Declaration

```
public bool HasVariables { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

IsPlaybackPaused

Declaration

```
public bool IsPlaybackPaused { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

IsRecording

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

Declaration

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

Property Value

TYPE	DESCRIPTION
bool	

IsRecordingOrPaused

Declaration

```
public bool IsRecordingOrPaused { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

IsRecordingPaused

Declaration

```
public bool IsRecordingPaused { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

IsReplaying

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

Declaration

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

Property Value

TYPE	DESCRIPTION
bool	

IsReplayingOrPaused

Declaration

```
public bool IsReplayingOrPaused { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

PlaybackDirection

Gets the current PlaybackDirection of replay playback.

Declaration

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

Property Value

TYPE	DESCRIPTION
PlaybackDirection	

PlaybackTime

Get the current playback time in seconds. This ReplayBehaviour must be attached to an object that is currently being replayed for this value to be valid.

Declaration

```
public float PlaybackTime { get; }
```

Property Value

TYPE	DESCRIPTION
float	

PlaybackTimeNormalized

Declaration

```
public float PlaybackTimeNormalized { get; }
```

Property Value

TYPE	DESCRIPTION
float	

PlaybackTimeScale

Declaration

```
public float PlaybackTimeScale { get; }
```

Property Value

TYPE	DESCRIPTION
float	

ReplayIdentity

Get the Core.ReplayIdentity associated with this ReplayBehaviour.

Declaration

```
public ReplayIdentity ReplayIdentity { get; set; }
```

Property Value

TYPE	DESCRIPTION
ReplayIdentity	

ReplayObject

Get the managing ReplayObject.

Declaration

```
public ReplayObject ReplayObject { get; }
```

Property Value

TYPE	DESCRIPTION
ReplayObject	

ReplayPersistentData

Declaration

```
public ReplayState ReplayPersistentData { get; }
```

Property Value

TYPE	DESCRIPTION
ReplayState	

Variables

Get all ReplayVariable associated with this ReplayBehaviour.

Declaration

```
public IList<ReplayVariable> Variables { get; }
```

Property Value

TYPE	DESCRIPTION
IList<ReplayVariable>	

Methods

Awake()

Called by Unity.

Declaration

```
protected virtual void Awake()
```

ForceRegenerateIdentity()

Force the ReplayIdentity of this component to be regenerated.

Declaration

```
public void ForceRegenerateIdentity()
```

OnDestroy()

Declaration

```
protected virtual void OnDestroy()
```

OnDisable()

Called by Unity. Be sure to call this base method when overriding otherwise replay events will not be received.

Declaration

```
protected virtual void OnDisable()
```

OnEnable()

Called by Unity. Be sure to call this base method when overriding otherwise replay events will not be received.

Declaration

```
protected virtual void OnEnable()
```

OnReplayCapture()

Called by the replay system when non-recordable components should submit data to be recorded to the managing replay object. This method is ideal for recording variables, events, methods calls and similar. Update can be used instead however 'OnReplayCapture' is guarenteed to be called during the same frame that replay recordable data is serialized.

Declaration

```
protected virtual void OnReplayCapture()
```

OnReplayEnd()

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'

Declaration

```
protected virtual void OnReplayEnd()
```

OnReplayEvent(ushort, ReplayState)

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

Declaration

```
protected virtual void OnReplayEvent(ushort eventID, ReplayState eventData)
```

Parameters

TYPE	NAME	DESCRIPTION
ushort	eventID	
ReplayState	eventData	

OnReplayPlayPause(bool)

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

Declaration

```
protected virtual void OnReplayPlayPause(bool paused)
```

Parameters

TYPE	NAME	DESCRIPTION
bool	paused	True if playback is about to be paused or false if plyabck is about to be resumed

OnReplayReset()

Called by the replay system during playback when cached values should be reset to safe default to avoid glitches or inaccuracies in the playback.

Declaration

```
protected virtual void OnReplayReset()
```

OnReplaySpawned(Vector3, Quaternion)

Called by the replay system when the object has been spawned from a prefab instance during playback.

Declaration

```
protected virtual void OnReplaySpawned(Vector3 position, Quaternion rotation)
```

Parameters

TYPE	NAME	DESCRIPTION
Vector3	position	
Quaternion	rotation	

OnReplayStart()

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.

Declaration

```
protected virtual void OnReplayStart()
```

OnReplayUpdate(float)

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

Declaration

```
protected virtual void OnReplayUpdate(float t)
```

Parameters

TYPE	NAME	DESCRIPTION
float	t	A normalized value representing the progress between replay snapshots. Use this value for interpolation or similar smoothing passes

RecordEvent(ushort, ReplayState)

Record a replay event on the current record frame.

Declaration

```
public void RecordEvent(ushort eventID, ReplayState eventData = null)
```

Parameters

TYPE	NAME	DESCRIPTION
ushort	eventID	A unique event ID value used to identify the event type
ReplayState	eventData	A replay state containing data associated with the event

RecordMethodCall(Action)

Record a method call. Note that this will also cause the target method to be invoked immediatley.

Declaration

```
public void RecordMethodCall(Action method)
```

Parameters

TYPE	NAME	DESCRIPTION
Action	method	The delegate method to record

RecordMethodCall<T>(Action<T>, T)

Record a method call. Note that this will also cause the target method to be invoked immediatley.

Declaration

```
public void RecordMethodCall<T>(Action<T> method, T arg)
```

Parameters

TYPE	NAME	DESCRIPTION
Action<T>	method	The delegate method to record
T	arg	The first argument for the target method

Type Parameters

NAME	DESCRIPTION
T	The parameter type of the first method parameter

RecordMethodCall<T0, T1>(Action<T0, T1>, T0, T1)

Record a method call. Note that this will also cause the target method to be invoked immediatley.

Declaration

```
public void RecordMethodCall<T0, T1>(Action<T0, T1> method, T0 arg0, T1 arg1)
```

Parameters

TYPE	NAME	DESCRIPTION
Action<T0, T1>	method	The delegate method to record
T0	arg0	The first argument for the target method
T1	arg1	The second argument for the target method

Type Parameters

NAME	DESCRIPTION
T0	The parameter type of the first method parameter
T1	The parameter type of the second method parameter

RecordMethodCall<T0, T1, T2>(Action<T0, T1, T2>, T0, T1, T2)

Record a method call. Note that this will also cause the target method to be invoked immediately.

Declaration

```
public void RecordMethodCall<T0, T1, T2>(Action<T0, T1, T2> method, T0 arg0, T1 arg1, T2 arg2)
```

Parameters

TYPE	NAME	DESCRIPTION
Action<T0, T1, T2>	method	The delegate method to record
T0	arg0	The first argument for the target method

TYPE	NAME	DESCRIPTION
T1	arg1	The second argument for the target method
T2	arg2	The third argument for the target method

Type Parameters

NAME	DESCRIPTION
T0	The parameter type of the first method parameter
T1	The parameter type of the second method parameter
T2	The parameter type of the third method parameter

RecordMethodCall<T0, T1, T2, T3>(Action<T0, T1, T2, T3>, T0, T1, T2, T3)

Record a method call. Note that this will also cause the target method to be invoked immediately.

Declaration

```
public void RecordMethodCall<T0, T1, T2, T3>(Action<T0, T1, T2, T3> method, T0 arg0, T1 arg1, T2 arg2, T3
arg3)
```

Parameters

TYPE	NAME	DESCRIPTION
Action<T0, T1, T2, T3>	method	The delegate method to record
T0	arg0	The first argument for the target method
T1	arg1	The second argument for the target method
T2	arg2	The third argument for the target method
T3	arg3	The fourth argument for the target method

Type Parameters

NAME	DESCRIPTION

NAME	DESCRIPTION
T0	The parameter type of the first method parameter
T1	The parameter type of the second method parameter
T2	The parameter type of the third method parameter
T3	The parameter type of the fourth method parameter

RecordVariable(ReplayVariable)

Record the value of the specified ReplayVariable. Should only be called when IsRecording is true. The variable data will be recorded for a single frame. To order to record a variable over time, simply call this method every frame.

Declaration

```
public void RecordVariable(ReplayVariable variable)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayVariable	variable	

Reset()

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

Declaration

```
protected virtual void Reset()
```

Class ReplayBlendShape

Inheritance

object
Object
Component
Behaviour
MonoBehaviour
ReplayBehaviour
ReplayRecordableBehaviour
ReplayBlendShape

Implements

IReplaySerialize

Inherited Members

ReplayRecordableBehaviour.Formatter
ReplayRecordableBehaviour.OnDestroy()
ReplayBehaviour.ReplayIdentity
ReplayBehaviour.ReplayObject
ReplayBehaviour.HasPersistentData
ReplayBehaviour.ReplayPersistentData
ReplayBehaviour.Variables
ReplayBehaviour.HasVariables
ReplayBehaviour.IsRecording
ReplayBehaviour.IsRecordingPaused
ReplayBehaviour.IsRecordingOrPaused
ReplayBehaviour.IsReplaying
ReplayBehaviour.IsPlaybackPaused
ReplayBehaviour.IsReplayingOrPaused
ReplayBehaviour.PlaybackTime
ReplayBehaviour.PlaybackTimeNormalized
ReplayBehaviour.PlaybackTimeScale
ReplayBehaviour.PlaybackDirection
ReplayBehaviour.Awake()
ReplayBehaviour.OnEnable()
ReplayBehaviour.OnDisable()
ReplayBehaviour.OnReplayStart()
ReplayBehaviour.OnReplayEnd()
ReplayBehaviour.OnReplayPlayPause(bool)
ReplayBehaviour.OnReplayCapture()
ReplayBehaviour.OnReplayEvent(ushort, ReplayState)
ReplayBehaviour.OnReplaySpawned(Vector3, Quaternion)
ReplayBehaviour.ForceRegenerateIdentity()
ReplayBehaviour.RecordVariable(ReplayVariable)
ReplayBehaviour.RecordEvent(ushort, ReplayState)
ReplayBehaviour.RecordMethodCall(Action)
ReplayBehaviour.RecordMethodCall<T>(Action<T>, T)
ReplayBehaviour.RecordMethodCall<T0, T1>(Action<T0, T1>, T0, T1)
ReplayBehaviour.RecordMethodCall<T0, T1, T2>(Action<T0, T1, T2>, T0, T1, T2)
ReplayBehaviour.RecordMethodCall<T0, T1, T2, T3>(Action<T0, T1, T2, T3>, T0, T1, T2, T3)
MonoBehaviour.IsInvoking()
MonoBehaviour.CancelInvoke()

MonoBehaviour.Invoke(string, float)
MonoBehaviour.InvokeRepeating(string, float, float)
MonoBehaviour.CancelInvoke(string)
MonoBehaviour.IsInvoking(string)
MonoBehaviour.StartCoroutine(string)
MonoBehaviour.StartCoroutine(string, object)
MonoBehaviour.StartCoroutine(IEnumerator)
MonoBehaviour.StartCoroutine_Auto(IEnumerator)
MonoBehaviour.StopCoroutine(IEnumerator)
MonoBehaviour.StopCoroutine(Coroutine)
MonoBehaviour.StopCoroutine(string)
MonoBehaviour.StopAllCoroutines()
MonoBehaviour.print(object)
MonoBehaviour.useGUILayout
MonoBehaviour.runInEditMode
Behaviour.enabled
Behaviour.isActiveAndEnabled
Component.GetComponent(Type)
Component.GetComponent<T>()
Component.TryGetComponent(Type, out Component)
Component.TryGetComponent<T>(out T)
Component.GetComponent(string)
Component.GetComponentInChildren(Type, bool)
Component.GetComponentInChildren(Type)
Component.GetComponentInChildren<T>(bool)
Component.GetComponentInChildren<T>()
Component.GetComponentsInChildren(Type, bool)
Component.GetComponentsInChildren(Type)
Component.GetComponentsInChildren<T>(bool)
Component.GetComponentsInChildren<T>(bool, List<T>)
Component.GetComponentsInChildren<T>()
Component.GetComponentsInChildren<T>(List<T>)
Component.GetComponentInParent(Type, bool)
Component.GetComponentInParent(Type)
Component.GetComponentInParent<T>(bool)
Component.GetComponentInParent<T>()
Component.GetComponentsInParent(Type, bool)
Component.GetComponentsInParent(Type)
Component.GetComponentsInParent<T>(bool)
Component.GetComponentsInParent<T>(bool, List<T>)
Component.GetComponentsInParent<T>()
Component.GetComponents(Type)
Component.GetComponents(Type, List<Component>)
Component.GetComponents<T>(List<T>)
Component.GetComponents<T>()
Component.CompareTag(string)
Component.SendMessageUpwards(string, object, SendMessageOptions)
Component.SendMessageUpwards(string, object)
Component.SendMessageUpwards(string)
Component.SendMessageUpwards(string, SendMessageOptions)
Component.SendMessage(string, object)
Component.SendMessage(string)

Component.SendMessage(string, object, SendMessageOptions)

Component.SendMessage(string, SendMessageOptions)

Component.BroadcastMessage(string, object, SendMessageOptions)

Component.BroadcastMessage(string, object)

Component.BroadcastMessage(string)

Component.BroadcastMessage(string, SendMessageOptions)

Component.transform

Component.gameObject

Component.tag

Object.GetInstanceID()

Object.GetHashCode()

Object.Equals(object)

Object.Instantiate(Object, Vector3, Quaternion)

Object.Instantiate(Object, Vector3, Quaternion, Transform)

Object.Instantiate(Object)

Object.Instantiate(Object, Transform)

Object.Instantiate(Object, Transform, bool)

Object.Instantiate<T>(T)

Object.Instantiate<T>(T, Vector3, Quaternion)

Object.Instantiate<T>(T, Vector3, Quaternion, Transform)

Object.Instantiate<T>(T, Transform)

Object.Instantiate<T>(T, Transform, bool)

Object.Destroy(Object, float)

Object.Destroy(Object)

Object.DestroyImmediate(Object, bool)

Object.DestroyImmediate(Object)

Object.FindObjectsOfType(Type)

Object.FindObjectsOfType(Type, bool)

Object.DontDestroyOnLoad(Object)

Object.DestroyObject(Object, float)

Object.DestroyObject(Object)

Object.FindSceneObjectsOfType(Type)

Object.FindObjectsOfTypeIncludingAssets(Type)

Object.FindObjectsOfType<T>()

Object.FindObjectsOfType<T>(bool)

Object.FindObjectOfType<T>()

Object.FindObjectOfType<T>(bool)

Object.FindObjectsOfTypeAll(Type)

Object.FindObjectOfType(Type)

Object.FindObjectOfType(Type, bool)

Object.ToString()

Object.name

Object.hideFlags

object.Equals(object, object)

object.GetType()

object.MemberwiseClone()

object.ReferenceEquals(object, object)

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
public class ReplayBlendShape : ReplayRecordableBehaviour, IReplaySerialize
```

Fields

interpolate

Declaration

```
public bool interpolate
```

Field Value

TYPE	DESCRIPTION
bool	

observedSkinnedMeshRenderer

Declaration

```
public SkinnedMeshRenderer observedSkinnedMeshRenderer
```

Field Value

TYPE	DESCRIPTION
SkinnedMeshRenderer	

Methods

OnReplayDeserialize(ReplayState)

Called by the replay system when the recorder component should deserialize any necessary data during playback.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState containing the recorded data

Overrides

ReplayRecordableBehaviour.OnReplayDeserialize(ReplayState)

OnReplayReset()

Called by the replay system during playback when cached values should be reset to safe default to avoid glitches or inaccuracies in the playback.

Declaration

```
protected override void OnReplayReset()
```

Overrides

ReplayBehaviour.OnReplayReset()

OnReplaySerialize(ReplayState)

Called by the replay system when the recorder component should serialize and necessary data during recording.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState used to store the serialized data

Overrides

ReplayRecordableBehaviour.OnReplaySerialize(ReplayState)

OnReplayUpdate(float)

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

Declaration

```
protected override void OnReplayUpdate(float t)
```

Parameters

TYPE	NAME	DESCRIPTION
float	t	A normalized value representing the progress between replay snapshots. Use this value for interpolation or similar smoothing passes

Overrides

ReplayBehaviour.OnReplayUpdate(float)

Reset()

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

Declaration

```
protected override void Reset()
```

Overrides

ReplayBehaviour.Reset()

Start()

Declaration

```
public void Start()
```

Implements

IReplaySerialize

Class ReplayComponentEnabledState

A replay component used to record the enabled state of a behaviour component.

Inheritance

object
Object
Component
Behaviour
MonoBehaviour
ReplayBehaviour
ReplayRecordableBehaviour
ReplayComponentEnabledState

Implements

IReplaySerialize

Inherited Members

ReplayRecordableBehaviour.OnDestroy()
ReplayBehaviour.ReplayIdentity
ReplayBehaviour.ReplayObject
ReplayBehaviour.HasPersistentData
ReplayBehaviour.ReplayPersistentData
ReplayBehaviour.Variables
ReplayBehaviour.HasVariables
ReplayBehaviour.IsRecording
ReplayBehaviour.IsRecordingPaused
ReplayBehaviour.IsRecordingOrPaused
ReplayBehaviour.IsReplaying
ReplayBehaviour.IsPlaybackPaused
ReplayBehaviour.IsReplayingOrPaused
ReplayBehaviour.PlaybackTime
ReplayBehaviour.PlaybackTimeNormalized
ReplayBehaviour.PlaybackTimeScale
ReplayBehaviour.PlaybackDirection
ReplayBehaviour.Awake()
ReplayBehaviour.OnEnable()
ReplayBehaviour.OnDisable()
ReplayBehaviour.OnReplayStart()
ReplayBehaviour.OnReplayEnd()
ReplayBehaviour.OnReplayPlayPause(bool)
ReplayBehaviour.OnReplayReset()
ReplayBehaviour.OnReplayCapture()
ReplayBehaviour.OnReplayUpdate(float)
ReplayBehaviour.OnReplayEvent(ushort, ReplayState)
ReplayBehaviour.OnReplaySpawned(Vector3, Quaternion)
ReplayBehaviour.ForceRegenerateIdentity()
ReplayBehaviour.RecordVariable(ReplayVariable)
ReplayBehaviour.RecordEvent(ushort, ReplayState)
ReplayBehaviour.RecordMethodCall(Action)
ReplayBehaviour.RecordMethodCall<T>(Action<T>, T)
ReplayBehaviour.RecordMethodCall<T0, T1>(Action<T0, T1>, T0, T1)
ReplayBehaviour.RecordMethodCall<T0, T1, T2>(Action<T0, T1, T2>, T0, T1, T2)

ReplayBehaviour.RecordMethodCall<T0, T1, T2, T3>(Action<T0, T1, T2, T3>, T0, T1, T2, T3)
MonoBehaviour.IsInvoking()
MonoBehaviour.CancelInvoke()
MonoBehaviour.Invoke(string, float)
MonoBehaviour.InvokeRepeating(string, float, float)
MonoBehaviour.CancelInvoke(string)
MonoBehaviour.IsInvoking(string)
MonoBehaviour.StartCoroutine(string)
MonoBehaviour.StartCoroutine(string, object)
MonoBehaviour.StartCoroutine(IEnumerator)
MonoBehaviour.StartCoroutine_Auto(IEnumerator)
MonoBehaviour.StopCoroutine(IEnumerator)
MonoBehaviour.StopCoroutine(Coroutine)
MonoBehaviour.StopCoroutine(string)
MonoBehaviour.StopAllCoroutines()
MonoBehaviour.print(object)
MonoBehaviour.useGUILayout
MonoBehaviour.runInEditMode
Behaviour.enabled
Behaviour.isActiveAndEnabled
Component.GetComponent(Type)
Component.GetComponent<T>()
Component.TryGetComponent(Type, out Component)
Component.TryGetComponent<T>(out T)
Component.GetComponent(string)
Component.GetComponentInChildren(Type, bool)
Component.GetComponentInChildren(Type)
Component.GetComponentInChildren<T>(bool)
Component.GetComponentInChildren<T>()
Component.GetComponentsInChildren(Type, bool)
Component.GetComponentsInChildren(Type)
Component.GetComponentsInChildren<T>(bool)
Component.GetComponentsInChildren<T>(bool, List<T>)
Component.GetComponentsInChildren<T>()
Component.GetComponentsInChildren<T>(List<T>)
Component.GetComponentInParent(Type, bool)
Component.GetComponentInParent(Type)
Component.GetComponentInParent<T>(bool)
Component.GetComponentInParent<T>()
Component.GetComponentsInParent(Type, bool)
Component.GetComponentsInParent(Type)
Component.GetComponentsInParent<T>(bool)
Component.GetComponentsInParent<T>(bool, List<T>)
Component.GetComponentsInParent<T>()
Component.GetComponents(Type)
Component.GetComponents(Type, List<Component>)
Component.GetComponents<T>(List<T>)
Component.GetComponents<T>()
Component.CompareTag(string)
Component.SendMessageUpwards(string, object, SendMessageOptions)
Component.SendMessageUpwards(string, object)
Component.SendMessageUpwards(string)

Component.SendMessageUpwards(string, SendMessageOptions)
Component.SendMessage(string, object)
Component.SendMessage(string)
Component.SendMessage(string, object, SendMessageOptions)
Component.SendMessage(string, SendMessageOptions)
Component.BroadcastMessage(string, object, SendMessageOptions)
Component.BroadcastMessage(string, object)
Component.BroadcastMessage(string)
Component.BroadcastMessage(string, SendMessageOptions)
Component.transform
Component.gameObject
Component.tag
Object.GetInstanceID()
Object.GetHashCode()
Object.Equals(object)
Object.Instantiate(Object, Vector3, Quaternion)
Object.Instantiate(Object, Vector3, Quaternion, Transform)
Object.Instantiate(Object)
Object.Instantiate(Object, Transform)
Object.Instantiate(Object, Transform, bool)
Object.Instantiate<T>(T)
Object.Instantiate<T>(T, Vector3, Quaternion)
Object.Instantiate<T>(T, Vector3, Quaternion, Transform)
Object.Instantiate<T>(T, Transform)
Object.Instantiate<T>(T, Transform, bool)
Object.Destroy(Object, float)
Object.Destroy(Object)
Object.DestroyImmediate(Object, bool)
Object.DestroyImmediate(Object)
Object.FindObjectsOfType(Type)
Object.FindObjectsOfType(Type, bool)
Object.DontDestroyOnLoad(Object)
Object.DestroyObject(Object, float)
Object.DestroyObject(Object)
Object.FindSceneObjectsOfType(Type)
Object.FindObjectsOfTypeIncludingAssets(Type)
Object.FindObjectsOfType<T>()
Object.FindObjectsOfType<T>(bool)
Object.FindObjectOfType<T>()
Object.FindObjectOfType<T>(bool)
Object.FindObjectsOfTypeAll(Type)
Object.FindObjectOfType(Type)
Object.FindObjectOfType(Type, bool)
Object.ToString()
Object.name
Object.hideFlags
object.Equals(object, object)
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
public class ReplayComponentEnabledState : ReplayRecordableBehaviour, IReplaySerialize
```

Fields

observedComponent

The behaviour component that will have its enabled state recorded and replayed.

Declaration

```
public Behaviour observedComponent
```

Field Value

TYPE	DESCRIPTION
Behaviour	

Properties

Formatter

Get the formatter for this replay component.

Declaration

```
public override ReplayFormatter Formatter { get; }
```

Property Value

TYPE	DESCRIPTION
ReplayFormatter	

Overrides

ReplayRecordableBehaviour.Formatter

Methods

OnReplayDeserialize(ReplayState)

Called by the replay system when the component should deserialize previously recorded data.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState to read from

Overrides

ReplayRecordableBehaviour.OnReplayDeserialize(ReplayState)

OnReplaySerialize(ReplayState)

Called by the replay system when the component should serialize its recorded data.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState to write to

Overrides

ReplayRecordableBehaviour.OnReplaySerialize(ReplayState)

Reset()

Reset this replay component.

Declaration

```
protected override void Reset()
```

Overrides

ReplayBehaviour.Reset()

Implements

IReplaySerialize

Class ReplayComponentPreparerAttribute

Use this attribute to register a type as a component preparer. This attribute only works in conjunction with the DefaultReplayPreparer.

Inheritance

object
Attribute
ReplayComponentPreparerAttribute

Implements

_Attribute

Inherited Members

Attribute.Equals(object)
Attribute.GetCustomAttribute(Assembly, Type)
Attribute.GetCustomAttribute(Assembly, Type, bool)
Attribute.GetCustomAttribute(MemberInfo, Type)
Attribute.GetCustomAttribute(MemberInfo, Type, bool)
Attribute.GetCustomAttribute(Module, Type)
Attribute.GetCustomAttribute(Module, Type, bool)
Attribute.GetCustomAttribute(ParameterInfo, Type)
Attribute.GetCustomAttribute(ParameterInfo, Type, bool)
Attribute.GetCustomAttributes(Assembly)
Attribute.GetCustomAttributes(Assembly, bool)
Attribute.GetCustomAttributes(Assembly, Type)
Attribute.GetCustomAttributes(Assembly, Type, bool)
Attribute.GetCustomAttributes(MemberInfo)
Attribute.GetCustomAttributes(MemberInfo, bool)
Attribute.GetCustomAttributes(MemberInfo, Type)
Attribute.GetCustomAttributes(MemberInfo, Type, bool)
Attribute.GetCustomAttributes(Module)
Attribute.GetCustomAttributes(Module, bool)
Attribute.GetCustomAttributes(Module, Type)
Attribute.GetCustomAttributes(Module, Type, bool)
Attribute.GetCustomAttributes(ParameterInfo)
Attribute.GetCustomAttributes(ParameterInfo, bool)
Attribute.GetCustomAttributes(ParameterInfo, Type)
Attribute.GetCustomAttributes(ParameterInfo, Type, bool)
Attribute.GetHashCode()
Attribute.IsDefaultAttribute()
Attribute.IsDefined(Assembly, Type)
Attribute.IsDefined(Assembly, Type, bool)
Attribute.IsDefined(MemberInfo, Type)
Attribute.IsDefined(MemberInfo, Type, bool)
Attribute.IsDefined(Module, Type)
Attribute.IsDefined(Module, Type, bool)
Attribute.IsDefined(ParameterInfo, Type)
Attribute.IsDefined(ParameterInfo, Type, bool)
Attribute.Match(object)
Attribute.TypeId
object.Equals(object, object)
object.GetType()

object.ReferenceEquals(object, object)
object.ToString()

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
[AttributeUsage(AttributeTargets.Class, AllowMultiple = false)]
public sealed class ReplayComponentPreparerAttribute : Attribute, _Attribute
```

Constructors

ReplayComponentPreparerAttribute(Type, int)

Declaration

```
public ReplayComponentPreparerAttribute(Type componentType, int priority = 100)
```

Parameters

TYPE	NAME	DESCRIPTION
Type	componentType	
int	priority	

Fields

componentType

Declaration

```
public Type componentType
```

Field Value

TYPE	DESCRIPTION
Type	

priority

Declaration

```
public int priority
```

Field Value

TYPE	DESCRIPTION
int	

Implements

_Attribute

Class ReplayControls

Inheritance

object

Object

Component

Behaviour

MonoBehaviour

ReplayControls

Inherited Members

MonoBehaviour.IsInvoking()

MonoBehaviour.CancelInvoke()

MonoBehaviour.Invoke(string, float)

MonoBehaviour.InvokeRepeating(string, float, float)

MonoBehaviour.CancelInvoke(string)

MonoBehaviour.IsInvoking(string)

MonoBehaviour.StartCoroutine(string)

MonoBehaviour.StartCoroutine(string, object)

MonoBehaviour.StartCoroutine(IEnumerator)

MonoBehaviour.StartCoroutine_Auto(IEnumerator)

MonoBehaviour.StopCoroutine(IEnumerator)

MonoBehaviour.StopCoroutine(Coroutine)

MonoBehaviour.StopCoroutine(string)

MonoBehaviour.StopAllCoroutines()

MonoBehaviour.print(object)

MonoBehaviour.useGUILayout

MonoBehaviour.runInEditMode

Behaviour.enabled

Behaviour.isActiveAndEnabled

Component.GetComponent(Type)

Component.GetComponent<T>()

Component.TryGetComponent(Type, out Component)

Component.TryGetComponent<T>(out T)

Component.GetComponent(string)

Component.GetComponentInChildren(Type, bool)

Component.GetComponentInChildren(Type)

Component.GetComponentInChildren<T>(bool)

Component.GetComponentInChildren<T>()

Component.GetComponentsInChildren(Type, bool)

Component.GetComponentsInChildren(Type)

Component.GetComponentsInChildren<T>(bool)

Component.GetComponentsInChildren<T>(bool, List<T>)

Component.GetComponentsInChildren<T>()

Component.GetComponentsInChildren<T>(List<T>)

Component.GetComponentInParent(Type, bool)

Component.GetComponentInParent(Type)

Component.GetComponentInParent<T>(bool)

Component.GetComponentInParent<T>()

Component.GetComponentsInParent(Type, bool)

Component.GetComponentsInParent(Type)

Component.GetComponentsInParent<T>(bool)

Component.GetComponentsInParent<T>(bool, List<T>)
Component.GetComponentsInParent<T>()
Component.GetComponents(Type)
Component.GetComponents(Type, List<Component>)
Component.GetComponents<T>(List<T>)
Component.GetComponents<T>()
Component.CompareTag(string)
Component.SendMessageUpwards(string, object, SendMessageOptions)
Component.SendMessageUpwards(string, object)
Component.SendMessageUpwards(string)
Component.SendMessageUpwards(string, SendMessageOptions)
Component.SendMessage(string, object)
Component.SendMessage(string)
Component.SendMessage(string, object, SendMessageOptions)
Component.SendMessage(string, SendMessageOptions)
Component.BroadcastMessage(string, object, SendMessageOptions)
Component.BroadcastMessage(string, object)
Component.BroadcastMessage(string)
Component.BroadcastMessage(string, SendMessageOptions)
Component.transform
Component.gameObject
Component.tag
Object.GetInstanceID()
Object.GetHashCode()
Object.Equals(object)
Object.Instantiate(Object, Vector3, Quaternion)
Object.Instantiate(Object, Vector3, Quaternion, Transform)
Object.Instantiate(Object)
Object.Instantiate(Object, Transform)
Object.Instantiate(Object, Transform, bool)
Object.Instantiate<T>(T)
Object.Instantiate<T>(T, Vector3, Quaternion)
Object.Instantiate<T>(T, Vector3, Quaternion, Transform)
Object.Instantiate<T>(T, Transform)
Object.Instantiate<T>(T, Transform, bool)
Object.Destroy(Object, float)
Object.Destroy(Object)
Object.DestroyImmediate(Object, bool)
Object.DestroyImmediate(Object)
Object.FindObjectsOfType(Type)
Object.FindObjectsOfType(Type, bool)
Object.DontDestroyOnLoad(Object)
Object.DestroyObject(Object, float)
Object.DestroyObject(Object)
Object.FindSceneObjectsOfType(Type)
Object.FindObjectsOfTypeIncludingAssets(Type)
Object.FindObjectsOfType<T>()
Object.FindObjectsOfType<T>(bool)
Object.FindObjectOfType<T>()
Object.FindObjectOfType<T>(bool)
Object.FindObjectsOfTypeAll(Type)
Object.FindObjectOfType(Type)

Object.FindObjectOfType(Type, bool)
Object.ToString()
Object.name
Object.hideFlags
object.Equals(object, object)
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
public class ReplayControls : MonoBehaviour
```

Fields

playback

Declaration

```
protected ReplayPlaybackOperation playback
```

Field Value

TYPE	DESCRIPTION
ReplayPlaybackOperation	

record

Declaration

```
protected ReplayRecordOperation record
```

Field Value

TYPE	DESCRIPTION
ReplayRecordOperation	

recordFileName

Declaration

```
[Tooltip("The name of the replay file to save when 'recordToFile' is enabled")]
public string recordFileName
```

Field Value

TYPE	DESCRIPTION
string	

recordOnStart

Declaration

```
[Header("Options")]
[Tooltip("Should recording start as soon as the replay controls have loaded")]
public bool recordOnStart
```

Field Value

TYPE	DESCRIPTION
bool	

recordToFile

Declaration

```
[Tooltip("Replays will be saved to file when enabled or will be stored in memory when disabled")]
public bool recordToFile
```

Field Value

TYPE	DESCRIPTION
bool	

storage

Declaration

```
protected ReplayStorage storage
```

Field Value

TYPE	DESCRIPTION
ReplayStorage	

Properties

IsLive

Declaration

```
public bool IsLive { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

IsRecording

Declaration

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

Property Value

TYPE	DESCRIPTION
bool	

IsReplaying

Declaration

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

Property Value

TYPE	DESCRIPTION
bool	

Methods

Awake()

Declaration

```
protected virtual void Awake()
```

OnDestroy()

Declaration

```
protected virtual void OnDestroy()
```

ReplayBeginPlayback()

Declaration

```
public virtual void ReplayBeginPlayback()
```

ReplayBeginRecording()

Declaration

```
public virtual void ReplayBeginRecording()
```

ReplayGoLive()

Declaration

```
public virtual void ReplayGoLive()
```

ReplayStartPlayback()

Declaration

```
protected virtual void ReplayStartPlayback()
```

ReplayStartRecording()

Declaration

```
protected virtual void ReplayStartRecording()
```

ReplayStopPlayback()

Declaration

```
protected virtual void ReplayStopPlayback()
```

ReplayStopRecording()

Declaration

```
protected virtual void ReplayStopRecording()
```

SeekPlayback(float)

Declaration

```
public void SeekPlayback(float value)
```

Parameters

TYPE	NAME	DESCRIPTION
float	value	

SetPlaybackSpeed(float)

Declaration

```
public void SetPlaybackSpeed(float value)
```

Parameters

TYPE	NAME	DESCRIPTION
float	value	

Start()

Declaration

```
protected virtual void Start()
```

TogglePlaybackDirection()

Declaration

```
public void TogglePlaybackDirection()
```

TogglePlaybackLooped()

Declaration

```
public void TogglePlaybackLooped()
```

TogglePlaybackPaused()

Declaration

```
public void TogglePlaybackPaused()
```

TogglePlaybackSpeedMenu()

Declaration

```
public void TogglePlaybackSpeedMenu()
```

Update()

Declaration

```
protected virtual void Update()
```

Class ReplayControls.HighlightButton

Inheritance

object
ReplayControls.HighlightButton

Inherited Members

object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
[Serializable]
public class ReplayControls.HighlightButton
```

Fields

button

Declaration

```
public Button button
```

Field Value

TYPE	DESCRIPTION
Button	

highlight

Declaration

```
public Image highlight
```

Field Value

TYPE	DESCRIPTION
Image	

Class ReplayControls.SliderCallback

Helper class used to detect drag start and end events on UI slider control (Seek slider bar).

Inheritance

object
Object
Component
Behaviour
MonoBehaviour
ReplayControls.SliderCallback

Implements

IBeginDragHandler
IEndDragHandler
IEventSystemHandler

Inherited Members

MonoBehaviour.IsInvoking()
MonoBehaviour.CancelInvoke()
MonoBehaviour.Invoke(string, float)
MonoBehaviour.InvokeRepeating(string, float, float)
MonoBehaviour.CancelInvoke(string)
MonoBehaviour.IsInvoking(string)
MonoBehaviour.StartCoroutine(string)
MonoBehaviour.StartCoroutine(string, object)
MonoBehaviour.StartCoroutine(IEnumerator)
MonoBehaviour.StartCoroutine_Auto(IEnumerator)
MonoBehaviour.StopCoroutine(IEnumerator)
MonoBehaviour.StopCoroutine(Coroutine)
MonoBehaviour.StopCoroutine(string)
MonoBehaviour.StopAllCoroutines()
MonoBehaviour.print(object)
MonoBehaviour.useGUILayout
MonoBehaviour.runInEditMode
Behaviour.enabled
Behaviour.isActiveAndEnabled
Component.GetComponent(Type)
Component.GetComponent<T>()
Component.TryGetComponent(Type, out Component)
Component.TryGetComponent<T>(out T)
Component.GetComponent(string)
Component.GetComponentInChildren(Type, bool)
Component.GetComponentInChildren(Type)
Component.GetComponentInChildren<T>(bool)
Component.GetComponentInChildren<T>()
Component.GetComponentsInChildren(Type, bool)
Component.GetComponentsInChildren(Type)
Component.GetComponentsInChildren<T>(bool)
Component.GetComponentsInChildren<T>(bool, List<T>)
Component.GetComponentsInChildren<T>()
Component.GetComponentsInChildren<T>(List<T>)
Component.GetComponentInParent(Type, bool)

Component.GetComponentInParent(Type)
Component.GetComponentInParent<T>(bool)
Component.GetComponentInParent<T>()
Component.GetComponentsInParent(Type, bool)
Component.GetComponentsInParent(Type)
Component.GetComponentsInParent<T>(bool)
Component.GetComponentsInParent<T>(bool, List<T>)
Component.GetComponentsInParent<T>()
Component.GetComponents(Type)
Component.GetComponents(Type, List<Component>)
Component.GetComponents<T>(List<T>)
Component.GetComponents<T>()
Component.CompareTag(string)
Component.SendMessageUpwards(string, object, SendMessageOptions)
Component.SendMessageUpwards(string, object)
Component.SendMessageUpwards(string)
Component.SendMessageUpwards(string, SendMessageOptions)
Component.SendMessage(string, object)
Component.SendMessage(string)
Component.SendMessage(string, object, SendMessageOptions)
Component.SendMessage(string, SendMessageOptions)
Component.BroadcastMessage(string, object, SendMessageOptions)
Component.BroadcastMessage(string, object)
Component.BroadcastMessage(string)
Component.BroadcastMessage(string, SendMessageOptions)
Component.transform
Component.gameObject
Component.tag
Object.GetInstanceID()
Object.GetHashCode()
Object.Equals(object)
Object.Instantiate(Object, Vector3, Quaternion)
Object.Instantiate(Object, Vector3, Quaternion, Transform)
Object.Instantiate(Object)
Object.Instantiate(Object, Transform)
Object.Instantiate(Object, Transform, bool)
Object.Instantiate<T>(T)
Object.Instantiate<T>(T, Vector3, Quaternion)
Object.Instantiate<T>(T, Vector3, Quaternion, Transform)
Object.Instantiate<T>(T, Transform)
Object.Instantiate<T>(T, Transform, bool)
Object.Destroy(Object, float)
Object.Destroy(Object)
Object.DestroyImmediate(Object, bool)
Object.DestroyImmediate(Object)
Object.FindObjectsOfType(Type)
Object.FindObjectsOfType(Type, bool)
Object.DontDestroyOnLoad(Object)
Object.DestroyObject(Object, float)
Object.DestroyObject(Object)
Object.FindSceneObjectsOfType(Type)
Object.FindObjectsOfTypeIncludingAssets(Type)

Object.FindObjectsOfType<T>()
Object.FindObjectsOfType<T>(bool)
Object.FindObjectOfType<T>()
Object.FindObjectOfType<T>(bool)
Object.FindObjectsOfTypeAll(Type)
Object.FindObjectOfType(Type)
Object.FindObjectOfType(Type, bool)
Object.ToString()
Object.name
Object.hideFlags
object.Equals(object, object)
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
public class ReplayControls.SliderCallback : MonoBehaviour, IBeginDragHandler, IEndDragHandler,
IEventSystemHandler
```

Fields

isDragging

Declaration

```
public bool isDragging
```

Field Value

TYPE	DESCRIPTION
bool	

Methods

OnBeginDrag(PointerEventData)

Declaration

```
public void OnBeginDrag(PointerEventData eventData)
```

Parameters

TYPE	NAME	DESCRIPTION
PointerEventData	eventData	

OnEndDrag(PointerEventData)

Declaration

```
public void OnEndDrag(PointerEventData eventData)
```

Parameters

TYPE	NAME	DESCRIPTION
PointerEventData	eventData	

Implements

UnityEngine.EventSystems.IBeginDragHandler
UnityEngine.EventSystems.IEndDragHandler
UnityEngine.EventSystems.IEventSystemHandler

Class ReplayEnabledState

A replay component used to record the enabled state of a game object.

Inheritance

object
Object
Component
Behaviour
MonoBehaviour
ReplayBehaviour
ReplayRecordableBehaviour
ReplayEnabledState

Implements

IReplaySerialize

Inherited Members

ReplayBehaviour.ReplayIdentity
ReplayBehaviour.ReplayObject
ReplayBehaviour.HasPersistentData
ReplayBehaviour.ReplayPersistentData
ReplayBehaviour.Variables
ReplayBehaviour.HasVariables
ReplayBehaviour.IsRecording
ReplayBehaviour.IsRecordingPaused
ReplayBehaviour.IsRecordingOrPaused
ReplayBehaviour.IsReplaying
ReplayBehaviour.IsPlaybackPaused
ReplayBehaviour.IsReplayingOrPaused
ReplayBehaviour.PlaybackTime
ReplayBehaviour.PlaybackTimeNormalized
ReplayBehaviour.PlaybackTimeScale
ReplayBehaviour.PlaybackDirection
ReplayBehaviour.ForceRegenerateIdentity()
ReplayBehaviour.RecordVariable(ReplayVariable)
ReplayBehaviour.RecordEvent(ushort, ReplayState)
ReplayBehaviour.RecordMethodCall(Action)
ReplayBehaviour.RecordMethodCall<T>(Action<T>, T)
ReplayBehaviour.RecordMethodCall<T0, T1>(Action<T0, T1>, T0, T1)
ReplayBehaviour.RecordMethodCall<T0, T1, T2>(Action<T0, T1, T2>, T0, T1, T2)
ReplayBehaviour.RecordMethodCall<T0, T1, T2, T3>(Action<T0, T1, T2, T3>, T0, T1, T2, T3)
MonoBehaviour.IsInvoking()
MonoBehaviour.CancelInvoke()
MonoBehaviour.Invoke(string, float)
MonoBehaviour.InvokeRepeating(string, float, float)
MonoBehaviour.CancelInvoke(string)
MonoBehaviour.IsInvoking(string)
MonoBehaviour.StartCoroutine(string)
MonoBehaviour.StartCoroutine(string, object)
MonoBehaviour.StartCoroutine(IEnumerator)
MonoBehaviour.StartCoroutine_Auto(IEnumerator)
MonoBehaviour.StopCoroutine(IEnumerator)

MonoBehaviour.StopCoroutine(Coroutine)
MonoBehaviour.StopCoroutine(string)
MonoBehaviour.StopAllCoroutines()
MonoBehaviour.print(object)
MonoBehaviour.useGUILayout
MonoBehaviour.runInEditMode
Behaviour.enabled
Behaviour.isActiveAndEnabled
Component.GetComponent(Type)
Component.GetComponent<T>()
Component.TryGetComponent(Type, out Component)
Component.TryGetComponent<T>(out T)
Component.GetComponent(string)
Component.GetComponentInChildren(Type, bool)
Component.GetComponentInChildren(Type)
Component.GetComponentInChildren<T>(bool)
Component.GetComponentInChildren<T>()
Component.GetComponentsInChildren(Type, bool)
Component.GetComponentsInChildren(Type)
Component.GetComponentsInChildren<T>(bool)
Component.GetComponentsInChildren<T>(bool, List<T>)
Component.GetComponentsInChildren<T>()
Component.GetComponentsInChildren<T>(List<T>)
Component.GetComponentInParent(Type, bool)
Component.GetComponentInParent(Type)
Component.GetComponentInParent<T>(bool)
Component.GetComponentInParent<T>()
Component.GetComponentsInParent(Type, bool)
Component.GetComponentsInParent(Type)
Component.GetComponentsInParent<T>(bool)
Component.GetComponentsInParent<T>(bool, List<T>)
Component.GetComponentsInParent<T>()
Component.GetComponents(Type)
Component.GetComponents(Type, List<Component>)
Component.GetComponents<T>(List<T>)
Component.GetComponents<T>()
Component.CompareTag(string)
Component.SendMessageUpwards(string, object, SendMessageOptions)
Component.SendMessageUpwards(string, object)
Component.SendMessageUpwards(string)
Component.SendMessageUpwards(string, SendMessageOptions)
Component.SendMessage(string, object)
Component.SendMessage(string)
Component.SendMessage(string, object, SendMessageOptions)
Component.SendMessage(string, SendMessageOptions)
Component.BroadcastMessage(string, object, SendMessageOptions)
Component.BroadcastMessage(string, object)
Component.BroadcastMessage(string)
Component.BroadcastMessage(string, SendMessageOptions)
Component.transform
Component.gameObject
Component.tag

Object.GetInstanceID()

Object.GetHashCode()

Object.Equals(object)

Object.Instantiate(Object, Vector3, Quaternion)

Object.Instantiate(Object, Vector3, Quaternion, Transform)

Object.Instantiate(Object)

Object.Instantiate(Object, Transform)

Object.Instantiate(Object, Transform, bool)

Object.Instantiate<T>(T)

Object.Instantiate<T>(T, Vector3, Quaternion)

Object.Instantiate<T>(T, Vector3, Quaternion, Transform)

Object.Instantiate<T>(T, Transform)

Object.Instantiate<T>(T, Transform, bool)

Object.Destroy(Object, float)

Object.Destroy(Object)

Object.DestroyImmediate(Object, bool)

Object.DestroyImmediate(Object)

Object.FindObjectsOfType(Type)

Object.FindObjectsOfType(Type, bool)

Object.DontDestroyOnLoad(Object)

Object.DestroyObject(Object, float)

Object.DestroyObject(Object)

Object.FindSceneObjectsOfType(Type)

Object.FindObjectsOfTypeIncludingAssets(Type)

Object.FindObjectsOfType<T>()

Object.FindObjectsOfType<T>(bool)

Object.FindObjectOfType<T>()

Object.FindObjectOfType<T>(bool)

Object.FindObjectsOfTypeAll(Type)

Object.FindObjectOfType(Type)

Object.FindObjectOfType(Type, bool)

Object.ToString()

Object.name

Object.hideFlags

object.Equals(object, object)

object.GetType()

object.ReferenceEquals(object, object)

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
[DisallowMultipleComponent]
public sealed class ReplayEnabledState : ReplayRecordableBehaviour, IReplaySerialize
```

Properties

Formatter

Get the formatter for this replay component.

Declaration

```
public override ReplayFormatter Formatter { get; }
```

Property Value

TYPE	DESCRIPTION
ReplayFormatter	

Overrides

ReplayRecordableBehaviour.Formatter

Methods

OnReplayDeserialize(ReplayState)

Called by the replay system when replay data should be restored.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState containing the previously recorded data

Overrides

ReplayRecordableBehaviour.OnReplayDeserialize(ReplayState)

OnReplaySerialize(ReplayState)

Called by the replay system when recorded data should be captured.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState used to store the recorded data

Overrides

ReplayRecordableBehaviour.OnReplaySerialize(ReplayState)

Implements

IReplaySerialize

Struct ReplayIdentity

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 use to identify prefab instances that are spawned during a replay.

Implements

IEquatable<ReplayIdentity>
IReplaySerialize
IReplayStreamSerialize

Inherited Members

object.Equals(object, object)
object.GetType()
object.ReferenceEquals(object, object)

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
[Serializable]
public struct ReplayIdentity : IEquatable<ReplayIdentity>, IReplaySerialize, IReplayStreamSerialize
```

Constructors

ReplayIdentity(uint)

Create a new instance with the specified id value.

Declaration

```
public ReplayIdentity(uint id)
```

Parameters

TYPE	NAME	DESCRIPTION
uint	id	The id value to give this identity

ReplayIdentity(ReplayIdentity)

Declaration

```
public ReplayIdentity(ReplayIdentity other)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayIdentity	other	

Fields

byteSize

Get the number of bytes that this object uses to represent its id data.

Declaration

```
public static readonly int byteSize
```

Field Value

TYPE	DESCRIPTION
int	

invalid

Declaration

```
public static readonly ReplayIdentity invalid
```

Field Value

TYPE	DESCRIPTION
ReplayIdentity	

Properties

ID

Declaration

```
public int ID { get; }
```

Property Value

TYPE	DESCRIPTION
int	

IsValid

Returns true if this id is not equal to unassignedID.

Declaration

```
public bool IsValid { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

Methods

Equals(object)

Override implementation.

Declaration

```
public override bool Equals(object obj)
```

Parameters

TYPE	NAME	DESCRIPTION
object	obj	The object to compare against

Returns

TYPE	DESCRIPTION
bool	

Overrides

ValueType.Equals(object)

Equals(ReplayIdentity)

IEquateable implementation.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayIdentity	obj	The ReplayIdentity to compare against

Returns

TYPE	DESCRIPTION
bool	

GetHashCode()

Override implementation.

Declaration

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

Returns

TYPE	DESCRIPTION
int	

Overrides

ValueType.GetHashCode()

IsIdentityUnique(in ReplayIdentity)

Declaration

```
public static bool IsIdentityUnique(in ReplayIdentity identity)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayIdentity	identity	

Returns

TYPE	DESCRIPTION
bool	

RegisterIdentity(ReplayIdentity)

Declaration

```
public static void RegisterIdentity(ReplayIdentity identity)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayIdentity	identity	

ToString()

Override implementation.

Declaration

```
public override string ToString()
```

Returns

TYPE	DESCRIPTION
string	

Overrides

ValueType.ToString()

UnregisterIdentity(ReplayIdentity)

Declaration

```
public static void UnregisterIdentity(ReplayIdentity identity)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayIdentity	identity	

Operators

operator ==(ReplayIdentity, ReplayIdentity)

Override equals operator.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayIdentity	a	First ReplayIdentity
ReplayIdentity	b	Second ReplayIdentity

Returns

TYPE	DESCRIPTION
bool	

operator !=(ReplayIdentity, ReplayIdentity)

Override not-equals operator.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayIdentity	a	First ReplayIdentity
ReplayIdentity	b	Second ReplayIdentity

Returns

TYPE	DESCRIPTION
bool	

Implements

IEquatable<T>
IReplaySerialize
IReplayStreamSerialize

Class ReplayIgnoreAttribute

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 dont need to record any data.

Inheritance

object
Attribute
ReplayIgnoreAttribute

Implements

_Attribute

Inherited Members

Attribute.Equals(object)
Attribute.GetCustomAttribute(Assembly, Type)
Attribute.GetCustomAttribute(Assembly, Type, bool)
Attribute.GetCustomAttribute(MemberInfo, Type)
Attribute.GetCustomAttribute(MemberInfo, Type, bool)
Attribute.GetCustomAttribute(Module, Type)
Attribute.GetCustomAttribute(Module, Type, bool)
Attribute.GetCustomAttribute(ParameterInfo, Type)
Attribute.GetCustomAttribute(ParameterInfo, Type, bool)
Attribute.GetCustomAttributes(Assembly)
Attribute.GetCustomAttributes(Assembly, bool)
Attribute.GetCustomAttributes(Assembly, Type)
Attribute.GetCustomAttributes(Assembly, Type, bool)
Attribute.GetCustomAttributes(MemberInfo)
Attribute.GetCustomAttributes(MemberInfo, bool)
Attribute.GetCustomAttributes(MemberInfo, Type)
Attribute.GetCustomAttributes(MemberInfo, Type, bool)
Attribute.GetCustomAttributes(Module)
Attribute.GetCustomAttributes(Module, bool)
Attribute.GetCustomAttributes(Module, Type)
Attribute.GetCustomAttributes(Module, Type, bool)
Attribute.GetCustomAttributes(ParameterInfo)
Attribute.GetCustomAttributes(ParameterInfo, bool)
Attribute.GetCustomAttributes(ParameterInfo, Type)
Attribute.GetCustomAttributes(ParameterInfo, Type, bool)
Attribute.GetHashCode()
Attribute.IsDefaultAttribute()
Attribute.IsDefined(Assembly, Type)
Attribute.IsDefined(Assembly, Type, bool)
Attribute.IsDefined(MemberInfo, Type)
Attribute.IsDefined(MemberInfo, Type, bool)
Attribute.IsDefined(Module, Type)
Attribute.IsDefined(Module, Type, bool)
Attribute.IsDefined(ParameterInfo, Type)
Attribute.IsDefined(ParameterInfo, Type, bool)
Attribute.Match(object)
Attribute.TypeId
object.Equals(object, object)
object.GetType()

object.ReferenceEquals(object, object)

object.ToString()

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
[AttributeUsage(AttributeTargets.Class)]
public sealed class ReplayIgnoreAttribute : Attribute, _Attribute
```

Implements

_Attribute

Class ReplayLineRenderer

Recorder component used to record and replay the Unity line renderer component.

Inheritance

object
Object
Component
Behaviour
MonoBehaviour
ReplayBehaviour
ReplayRecordableBehaviour
ReplayLineRenderer

Implements

IReplaySerialize

Inherited Members

ReplayRecordableBehaviour.Formatter
ReplayRecordableBehaviour.OnDestroy()
ReplayBehaviour.ReplayIdentity
ReplayBehaviour.ReplayObject
ReplayBehaviour.HasPersistentData
ReplayBehaviour.ReplayPersistentData
ReplayBehaviour.Variables
ReplayBehaviour.HasVariables
ReplayBehaviour.IsRecording
ReplayBehaviour.IsRecordingPaused
ReplayBehaviour.IsRecordingOrPaused
ReplayBehaviour.IsReplaying
ReplayBehaviour.IsPlaybackPaused
ReplayBehaviour.IsReplayingOrPaused
ReplayBehaviour.PlaybackTime
ReplayBehaviour.PlaybackTimeNormalized
ReplayBehaviour.PlaybackTimeScale
ReplayBehaviour.PlaybackDirection
ReplayBehaviour.Awake()
ReplayBehaviour.OnEnable()
ReplayBehaviour.OnDisable()
ReplayBehaviour.OnReplayStart()
ReplayBehaviour.OnReplayEnd()
ReplayBehaviour.OnReplayPlayPause(bool)
ReplayBehaviour.OnReplayCapture()
ReplayBehaviour.OnReplayEvent(ushort, ReplayState)
ReplayBehaviour.OnReplaySpawned(Vector3, Quaternion)
ReplayBehaviour.ForceRegenerateIdentity()
ReplayBehaviour.RecordVariable(ReplayVariable)
ReplayBehaviour.RecordEvent(ushort, ReplayState)
ReplayBehaviour.RecordMethodCall(Action)
ReplayBehaviour.RecordMethodCall<T>(Action<T>, T)
ReplayBehaviour.RecordMethodCall<T0, T1>(Action<T0, T1>, T0, T1)
ReplayBehaviour.RecordMethodCall<T0, T1, T2>(Action<T0, T1, T2>, T0, T1, T2)
ReplayBehaviour.RecordMethodCall<T0, T1, T2, T3>(Action<T0, T1, T2, T3>, T0, T1, T2, T3)

MonoBehaviour.IsInvoking()
MonoBehaviour.CancelInvoke()
MonoBehaviour.Invoke(string, float)
MonoBehaviour.InvokeRepeating(string, float, float)
MonoBehaviour.CancelInvoke(string)
MonoBehaviour.IsInvoking(string)
MonoBehaviour.StartCoroutine(string)
MonoBehaviour.StartCoroutine(string, object)
MonoBehaviour.StartCoroutine(IEnumerator)
MonoBehaviour.StartCoroutine_Auto(IEnumerator)
MonoBehaviour.StopCoroutine(IEnumerator)
MonoBehaviour.StopCoroutine(Coroutine)
MonoBehaviour.StopCoroutine(string)
MonoBehaviour.StopAllCoroutines()
MonoBehaviour.print(object)
MonoBehaviour.useGUILayout
MonoBehaviour.runInEditMode
Behaviour.enabled
Behaviour.isActiveAndEnabled
Component.GetComponent(Type)
Component.GetComponent<T>()
Component.TryGetComponent(Type, out Component)
Component.TryGetComponent<T>(out T)
Component.GetComponent(string)
Component.GetComponentInChildren(Type, bool)
Component.GetComponentInChildren(Type)
Component.GetComponentInChildren<T>(bool)
Component.GetComponentInChildren<T>()
Component.GetComponentsInChildren(Type, bool)
Component.GetComponentsInChildren(Type)
Component.GetComponentsInChildren<T>(bool)
Component.GetComponentsInChildren<T>(bool, List<T>)
Component.GetComponentsInChildren<T>()
Component.GetComponentsInChildren<T>(List<T>)
Component.GetComponentInParent(Type, bool)
Component.GetComponentInParent(Type)
Component.GetComponentInParent<T>(bool)
Component.GetComponentInParent<T>()
Component.GetComponentsInParent(Type, bool)
Component.GetComponentsInParent(Type)
Component.GetComponentsInParent<T>(bool)
Component.GetComponentsInParent<T>(bool, List<T>)
Component.GetComponentsInParent<T>()
Component.GetComponents(Type)
Component.GetComponents(Type, List<Component>)
Component.GetComponents<T>(List<T>)
Component.GetComponents<T>()
Component.CompareTag(string)
Component.SendMessageUpwards(string, object, SendMessageOptions)
Component.SendMessageUpwards(string, object)
Component.SendMessageUpwards(string)
Component.SendMessageUpwards(string, SendMessageOptions)

Component.SendMessage(string, object)
Component.SendMessage(string)
Component.SendMessage(string, object, SendMessageOptions)
Component.SendMessage(string, SendMessageOptions)
Component.BroadcastMessage(string, object, SendMessageOptions)
Component.BroadcastMessage(string, object)
Component.BroadcastMessage(string)
Component.BroadcastMessage(string, SendMessageOptions)
Component.transform
Component.gameObject
Component.tag
Object.GetInstanceID()
Object.GetHashCode()
Object.Equals(object)
Object.Instantiate(Object, Vector3, Quaternion)
Object.Instantiate(Object, Vector3, Quaternion, Transform)
Object.Instantiate(Object)
Object.Instantiate(Object, Transform)
Object.Instantiate(Object, Transform, bool)
Object.Instantiate<T>(T)
Object.Instantiate<T>(T, Vector3, Quaternion)
Object.Instantiate<T>(T, Vector3, Quaternion, Transform)
Object.Instantiate<T>(T, Transform)
Object.Instantiate<T>(T, Transform, bool)
Object.Destroy(Object, float)
Object.Destroy(Object)
Object.DestroyImmediate(Object, bool)
Object.DestroyImmediate(Object)
Object.FindObjectsOfType(Type)
Object.FindObjectsOfType(Type, bool)
Object.DontDestroyOnLoad(Object)
Object.DestroyObject(Object, float)
Object.DestroyObject(Object)
Object.FindSceneObjectsOfType(Type)
Object.FindObjectsOfTypeIncludingAssets(Type)
Object.FindObjectsOfType<T>()
Object.FindObjectsOfType<T>(bool)
Object.FindObjectOfType<T>()
Object.FindObjectOfType<T>(bool)
Object.FindObjectsOfTypeAll(Type)
Object.FindObjectOfType(Type)
Object.FindObjectOfType(Type, bool)
Object.ToString()
Object.name
Object.hideFlags
object.Equals(object, object)
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
public class ReplayLineRenderer : ReplayRecordableBehaviour, IReplaySerialize
```

Fields

observedLineRenderer

Declaration

```
public LineRenderer observedLineRenderer
```

Field Value

TYPE	DESCRIPTION
LineRenderer	

updateFlags

Declaration

```
[HideInInspector]
public ReplayLineRenderer.ReplayLineRendererFlags updateFlags
```

Field Value

TYPE	DESCRIPTION
ReplayLineRenderer.ReplayLineRendererFlags	

Methods

OnReplayDeserialize(ReplayState)

Called by the replay system when the recorder component should deserialize any necessary data during playback.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState containing the recorded data

Overrides

ReplayRecordableBehaviour.OnReplayDeserialize(ReplayState)

OnReplayReset()

Called by the replay system during playback when cached values should be reset to safe default to avoid glitches or inaccuracies in the playback.

Declaration

```
protected override void OnReplayReset()
```

Overrides

ReplayBehaviour.OnReplayReset()

OnReplaySerialize(ReplayState)

Called by the replay system when the recorder component should serialize and necessary data during recording.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState used to store the serialized data

Overrides

ReplayRecordableBehaviour.OnReplaySerialize(ReplayState)

OnReplayUpdate(float)

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

Declaration

```
protected override void OnReplayUpdate(float t)
```

Parameters

TYPE	NAME	DESCRIPTION
float	t	A normalized value representing the progress between replay snapshots. Use this value for interpolation or similar smoothing passes

Overrides

ReplayBehaviour.OnReplayUpdate(float)

Reset()

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

Declaration

```
protected override void Reset()
```

Overrides

ReplayBehaviour.Reset()

Start()

Declaration

```
public void Start()
```

Implements

IReplaySerialize

Enum ReplayLineRenderer.ReplayLineRendererFlags

Flags used to specify which features are enabled on the recorder.

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
[Flags]
public enum ReplayLineRenderer.ReplayLineRendererFlags
```

Fields

NAME	DESCRIPTION
Interpolate	interpolation will be used during playback to create smoother results.
None	No additional features.

Class ReplayManager

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

Inheritance

object
Object
Component
Behaviour
MonoBehaviour
ReplayManager

Inherited Members

MonoBehaviour.IsInvoking()
MonoBehaviour.CancelInvoke()
MonoBehaviour.Invoke(string, float)
MonoBehaviour.InvokeRepeating(string, float, float)
MonoBehaviour.CancelInvoke(string)
MonoBehaviour.IsInvoking(string)
MonoBehaviour.StartCoroutine(string)
MonoBehaviour.StartCoroutine(string, object)
MonoBehaviour.StartCoroutine(IEnumerator)
MonoBehaviour.StartCoroutine_Auto(IEnumerator)
MonoBehaviour.StopCoroutine(IEnumerator)
MonoBehaviour.StopCoroutine(Coroutine)
MonoBehaviour.StopCoroutine(string)
MonoBehaviour.StopAllCoroutines()
MonoBehaviour.print(object)
MonoBehaviour.useGUILayout
MonoBehaviour.runInEditMode
Behaviour.enabled
Behaviour.isActiveAndEnabled
Component.GetComponent(Type)
Component.GetComponent<T>()
Component.TryGetComponent(Type, out Component)
Component.TryGetComponent<T>(out T)
Component.GetComponent(string)
Component.GetComponentInChildren(Type, bool)
Component.GetComponentInChildren(Type)
Component.GetComponentInChildren<T>(bool)
Component.GetComponentInChildren<T>()
Component.GetComponentsInChildren(Type, bool)
Component.GetComponentsInChildren(Type)
Component.GetComponentsInChildren<T>(bool)
Component.GetComponentsInChildren<T>(bool, List<T>)
Component.GetComponentsInChildren<T>()
Component.GetComponentsInChildren<T>(List<T>)
Component.GetComponentInParent(Type, bool)
Component.GetComponentInParent(Type)
Component.GetComponentInParent<T>(bool)
Component.GetComponentInParent<T>()
Component.GetComponentsInParent(Type, bool)
Component.GetComponentsInParent(Type)

Component.GetComponentsInParent<T>(bool)
Component.GetComponentsInParent<T>(bool, List<T>)
Component.GetComponentsInParent<T>()
Component.GetComponents(Type)
Component.GetComponents(Type, List<Component>)
Component.GetComponents<T>(List<T>)
Component.GetComponents<T>()
Component.CompareTag(string)
Component.SendMessageUpwards(string, object, SendMessageOptions)
Component.SendMessageUpwards(string, object)
Component.SendMessageUpwards(string)
Component.SendMessageUpwards(string, SendMessageOptions)
Component.SendMessage(string, object)
Component.SendMessage(string)
Component.SendMessage(string, object, SendMessageOptions)
Component.SendMessage(string, SendMessageOptions)
Component.BroadcastMessage(string, object, SendMessageOptions)
Component.BroadcastMessage(string, object)
Component.BroadcastMessage(string)
Component.BroadcastMessage(string, SendMessageOptions)
Component.transform
Component.gameObject
Component.tag
Object.GetInstanceID()
Object.GetHashCode()
Object.Equals(object)
Object.Instantiate(Object, Vector3, Quaternion)
Object.Instantiate(Object, Vector3, Quaternion, Transform)
Object.Instantiate(Object)
Object.Instantiate(Object, Transform)
Object.Instantiate(Object, Transform, bool)
Object.Instantiate<T>(T)
Object.Instantiate<T>(T, Vector3, Quaternion)
Object.Instantiate<T>(T, Vector3, Quaternion, Transform)
Object.Instantiate<T>(T, Transform)
Object.Instantiate<T>(T, Transform, bool)
Object.Destroy(Object, float)
Object.Destroy(Object)
Object.DestroyImmediate(Object, bool)
Object.DestroyImmediate(Object)
Object.FindObjectsOfType(Type)
Object.FindObjectsOfType(Type, bool)
Object.DontDestroyOnLoad(Object)
Object.DestroyObject(Object, float)
Object.DestroyObject(Object)
Object.FindSceneObjectsOfType(Type)
Object.FindObjectsOfTypeIncludingAssets(Type)
Object.FindObjectsOfType<T>()
Object.FindObjectsOfType<T>(bool)
Object.FindObjectOfType<T>()
Object.FindObjectOfType<T>(bool)
Object.FindObjectsOfTypeAll(Type)

Object.FindObjectOfType(Type)
Object.FindObjectOfType(Type, bool)
Object.ToString()
Object.name
Object.hideFlags
object.Equals(object, object)
object.GetType()
object.ReferenceEquals(object, object)

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
public sealed class ReplayManager : MonoBehaviour
```

Fields

manualStateUpdate

Should manual state update be enabled? If you set this value to true, you will then be responsible for update all replay and record operations by manually calling UpdateState(float).

Declaration

```
public static bool manualStateUpdate
```

Field Value

TYPE	DESCRIPTION
bool	

Properties

IsRecordingAny

Returns a value indicating if one or more recording operations are running.

Declaration

```
public static bool IsRecordingAny { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

IsReplayingAny

Returns a value indicating if one or more replay operations are running.

Declaration

```
public static bool IsReplayingAny { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

Settings

Get or load the replay settings from the current project. This is the replay settings editable via `Tools -> Ultimate Replay 3.0 -> Settings` and can be edited from code if required.

Declaration

```
public static ReplaySettings Settings { get; }
```

Property Value

TYPE	DESCRIPTION
ReplaySettings	

Methods

AddReplayObjectToPlaybackOperation(ReplayPlaybackOperation, ReplayObject)

Declaration

```
public static void AddReplayObjectToPlaybackOperation(ReplayPlaybackOperation playbackOperation, ReplayObject
replayObject)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayPlaybackOperation	playbackOperation	
ReplayObject	replayObject	

AddReplayObjectToPlaybackOperation(ReplayPlaybackOperation, GameObject)

Declaration

```
public static void AddReplayObjectToPlaybackOperation(ReplayPlaybackOperation playbackOperation, GameObject
gameObject)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayPlaybackOperation	playbackOperation	
GameObject	gameObject	

AddReplayObjectToPlaybackScenes(ReplayObject)

Declaration

```
public static void AddReplayObjectToPlaybackScenes(ReplayObject playbackObject)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayObject	playbackObject	

AddReplayObjectToPlaybackScenes(GameObject)

Declaration

```
public static void AddReplayObjectToPlaybackScenes(GameObject gameObject)
```

Parameters

TYPE	NAME	DESCRIPTION
GameObject	gameObject	

AddReplayObjectToRecordOperation(ReplayRecordOperation, ReplayObject)

Declaration

```
public static void AddReplayObjectToRecordOperation(ReplayRecordOperation recordOperation, ReplayObject
replayObject)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayRecordOperation	recordOperation	
ReplayObject	replayObject	

AddReplayObjectToRecordOperation(ReplayRecordOperation, GameObject)

Declaration

```
public static void AddReplayObjectToRecordOperation(ReplayRecordOperation recordOperation, GameObject
gameObject)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayRecordOperation	recordOperation	
GameObject	gameObject	

AddReplayObjectToRecordScenes(ReplayObject)

Declaration

```
public static void AddReplayObjectToRecordScenes(ReplayObject replayObject)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayObject	replayObject	

AddReplayObjectToRecordScenes(GameObject)

Declaration

```
public static void AddReplayObjectToRecordScenes(GameObject gameObject)
```

Parameters

TYPE	NAME	DESCRIPTION
GameObject	gameObject	

BeginPlayback(ReplayStorage, ReplayObject, ReplayObject, IReplayPreparer, ReplayPlaybackOptions, RestoreSceneMode)

Start a new playback operation with the specified parameters. The recorded data from the specified storage will be replayed onto the specified `playbackObject` and the `recordedObject` must be provided

Declaration

```
public static ReplayPlaybackOperation BeginPlayback(ReplayStorage storage, ReplayObject recordedObject,
ReplayObject playbackObject, IReplayPreparer preparer = null, ReplayPlaybackOptions playbackOptions = null,
RestoreSceneMode restoreSceneMode = RestoreSceneMode.RestoreState)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayStorage	storage	The storage where the replay is stored
ReplayObject	recordedObject	The single replay object that should be replayed
ReplayObject	playbackObject	
IReplayPreparer	preparer	
ReplayPlaybackOptions	playbackOptions	
RestoreSceneMode	restoreSceneMode	

Returns

TYPE	DESCRIPTION
ReplayPlaybackOperation	

Exceptions

TYPE	CONDITION
ArgumentNullException	
InvalidOperationException	

BeginPlayback(ReplayStorage, ReplayObject, IReplayPreparer, ReplayPlaybackOptions, RestoreSceneMode)

Declaration

```
public static ReplayPlaybackOperation BeginPlayback(ReplayStorage storage, ReplayObject playbackObject,
IReplayPreparer preparer = null, ReplayPlaybackOptions playbackOptions = null, RestoreSceneMode
restoreSceneMode = RestoreSceneMode.RestoreState)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayStorage	storage	
ReplayObject	playbackObject	
IReplayPreparer	preparer	
ReplayPlaybackOptions	playbackOptions	
RestoreSceneMode	restoreSceneMode	

Returns

TYPE	DESCRIPTION
ReplayPlaybackOperation	

BeginPlayback(ReplayStorage, ReplayScene, ReplayPlaybackOptions, RestoreSceneMode)

Declaration

```
public static ReplayPlaybackOperation BeginPlayback(ReplayStorage storage, ReplayScene playbackScene = null,
ReplayPlaybackOptions playbackOptions = null, RestoreSceneMode restoreReplayScene =
RestoreSceneMode.RestoreState)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayStorage	storage	
ReplayScene	playbackScene	
ReplayPlaybackOptions	playbackOptions	
RestoreSceneMode	restoreReplayScene	

Returns

TYPE	DESCRIPTION
ReplayPlaybackOperation	

Exceptions

TYPE	CONDITION
ArgumentNullException	
NotSupportedException	

BeginRecording(ReplayStorage, ReplayObject, bool, ReplayRecordOptions)

Start a new recording operation capturing only the specified replay object with the specified parameters.

Declaration

```
public static ReplayRecordOperation BeginRecording(ReplayStorage storage, ReplayObject recordObject, bool
cleanRecording = true, ReplayRecordOptions recordOptions = null)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayStorage	storage	The ReplayStorage that replay data should be saved to
ReplayObject	recordObject	The ReplayObject that should be sampled during recording
bool	cleanRecording	Should the recording start from scratch
ReplayRecordOptions	recordOptions	The ReplayRecordOptions used to control the record behaviour. Pass null if the global record options should be used

Returns

TYPE	DESCRIPTION
ReplayRecordOperation	A ReplayRecordOperation object that allows control over the new recording operation

Exceptions

TYPE	CONDITION
ArgumentNullException	The specified replay storage is null
ArgumentNullException	The specified replay object is null
AccessViolationException	The specified storage target is in use by another replay operation
NotSupportedException	The specified storage is not writable

BeginRecording(ReplayStorage, ReplayScene, bool, ReplayRecordOptions)

Start a new recording operation with the specified parameters.

Declaration

```
public static ReplayRecordOperation BeginRecording(ReplayStorage storage, ReplayScene recordScene = null, bool
cleanRecording = true, ReplayRecordOptions recordOptions = null)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayStorage	storage	
ReplayScene	recordScene	The ReplayScene that should be sampled during recording. Pass null to use all ReplayObject in the active unity scene
bool	cleanRecording	Should the recording start from scratch
ReplayRecordOptions	recordOptions	The ReplayRecordOptions used to control the record behaviour. Pass null if the global record options should be used

Returns

TYPE	DESCRIPTION
ReplayRecordOperation	A ReplayRecordOperation object that allows control over the new recording operation

Exceptions

TYPE	CONDITION
ArgumentNullException	The specified replay storage is null
AccessViolationException	The specified storage target is in use by another replay operation
NotSupportedException	The specified storage is not writable

FindReplayPrefab(string)

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
string	prefabName	

Returns

TYPE	DESCRIPTION
GameObject	

ForceAwake()

Declaration

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

RegisterReplayPrefab(GameObject)

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
GameObject	prefab	

RemoveReplayObjectFromPlaybackOperation(ReplayPlaybackOperation, ReplayObject)

Declaration

```
public static void RemoveReplayObjectFromPlaybackOperation(ReplayPlaybackOperation playbackOperation,
ReplayObject replayObject)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayPlaybackOperation	playbackOperation	
ReplayObject	replayObject	

RemoveReplayObjectFromPlaybackOperation(ReplayPlaybackOperation, GameObject)

Declaration

```
public static void RemoveReplayObjectFromPlaybackOperation(ReplayPlaybackOperation playbackOperation,
GameObject gameObject)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayPlaybackOperation	playbackOperation	
GameObject	gameObject	

RemoveReplayObjectFromPlaybackScenes(ReplayObject)

Declaration

```
public static void RemoveReplayObjectFromPlaybackScenes(ReplayObject playbackObject)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayObject	playbackObject	

RemoveReplayObjectFromPlaybackScenes(GameObject)

Declaration

```
public static void RemoveReplayObjectFromPlaybackScenes(GameObject gameObject)
```

Parameters

TYPE	NAME	DESCRIPTION
GameObject	gameObject	

RemoveReplayObjectFromRecordOperation(ReplayRecordOperation, ReplayObject)

Declaration

```
public static void RemoveReplayObjectFromRecordOperation(ReplayRecordOperation recordOperation, ReplayObject
replayObject)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayRecordOperation	recordOperation	
ReplayObject	replayObject	

RemoveReplayObjectFromRecordOperation(ReplayRecordOperation, GameObject)

Declaration

```
public static void RemoveReplayObjectFromRecordOperation(ReplayRecordOperation recordOperation, GameObject
gameObject)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayRecordOperation	recordOperation	
GameObject	gameObject	

RemoveReplayObjectFromRecordScenes(ReplayObject)

Declaration

```
public static void RemoveReplayObjectFromRecordScenes(ReplayObject replayObject)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayObject	replayObject	

RemoveReplayObjectFromRecordScenes(GameObject)

Declaration

```
public static void RemoveReplayObjectFromRecordScenes(GameObject gameObject)
```

Parameters

TYPE	NAME	DESCRIPTION
GameObject	gameObject	

ReplayTick(float, ReplayUpdateMode)

Update all running replay services using the specified delta time.

Declaration

```
public static void ReplayTick(float deltaTime, ReplayUpdateMode updateMode = ReplayUpdateMode.Manual)
```

Parameters

TYPE	NAME	DESCRIPTION
float	deltaTime	The amount of time in seconds that has passed since the last update. This value must be greater than 0
ReplayUpdateMode	updateMode	

StopPlaybackOperation(ReplayPlaybackOperation)

Declaration

```
public static void StopPlaybackOperation(ReplayPlaybackOperation playback)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayPlaybackOperation	playback	

Class ReplayMaterial

Inheritance

object
Object
Component
Behaviour
MonoBehaviour
ReplayBehaviour
ReplayRecordableBehaviour
ReplayMaterial

Implements

IReplaySerialize

Inherited Members

ReplayRecordableBehaviour.Formatter
ReplayRecordableBehaviour.OnDestroy()
ReplayBehaviour.ReplayIdentity
ReplayBehaviour.ReplayObject
ReplayBehaviour.HasPersistentData
ReplayBehaviour.ReplayPersistentData
ReplayBehaviour.Variables
ReplayBehaviour.HasVariables
ReplayBehaviour.IsRecording
ReplayBehaviour.IsRecordingPaused
ReplayBehaviour.IsRecordingOrPaused
ReplayBehaviour.IsReplaying
ReplayBehaviour.IsPlaybackPaused
ReplayBehaviour.IsReplayingOrPaused
ReplayBehaviour.PlaybackTime
ReplayBehaviour.PlaybackTimeNormalized
ReplayBehaviour.PlaybackTimeScale
ReplayBehaviour.PlaybackDirection
ReplayBehaviour.Awake()
ReplayBehaviour.OnEnable()
ReplayBehaviour.OnDisable()
ReplayBehaviour.OnReplayStart()
ReplayBehaviour.OnReplayEnd()
ReplayBehaviour.OnReplayPlayPause(bool)
ReplayBehaviour.OnReplayCapture()
ReplayBehaviour.OnReplayEvent(ushort, ReplayState)
ReplayBehaviour.OnReplaySpawned(Vector3, Quaternion)
ReplayBehaviour.ForceRegenerateIdentity()
ReplayBehaviour.RecordVariable(ReplayVariable)
ReplayBehaviour.RecordEvent(ushort, ReplayState)
ReplayBehaviour.RecordMethodCall(Action)
ReplayBehaviour.RecordMethodCall<T>(Action<T>, T)
ReplayBehaviour.RecordMethodCall<T0, T1>(Action<T0, T1>, T0, T1)
ReplayBehaviour.RecordMethodCall<T0, T1, T2>(Action<T0, T1, T2>, T0, T1, T2)
ReplayBehaviour.RecordMethodCall<T0, T1, T2, T3>(Action<T0, T1, T2, T3>, T0, T1, T2, T3)
MonoBehaviour.IsInvoking()
MonoBehaviour.CancelInvoke()

MonoBehaviour.Invoke(string, float)
MonoBehaviour.InvokeRepeating(string, float, float)
MonoBehaviour.CancelInvoke(string)
MonoBehaviour.IsInvoking(string)
MonoBehaviour.StartCoroutine(string)
MonoBehaviour.StartCoroutine(string, object)
MonoBehaviour.StartCoroutine(IEnumerator)
MonoBehaviour.StartCoroutine_Auto(IEnumerator)
MonoBehaviour.StopCoroutine(IEnumerator)
MonoBehaviour.StopCoroutine(Coroutine)
MonoBehaviour.StopCoroutine(string)
MonoBehaviour.StopAllCoroutines()
MonoBehaviour.print(object)
MonoBehaviour.useGUILayout
MonoBehaviour.runInEditMode
Behaviour.enabled
Behaviour.isActiveAndEnabled
Component.GetComponent(Type)
Component.GetComponent<T>()
Component.TryGetComponent(Type, out Component)
Component.TryGetComponent<T>(out T)
Component.GetComponent(string)
Component.GetComponentInChildren(Type, bool)
Component.GetComponentInChildren(Type)
Component.GetComponentInChildren<T>(bool)
Component.GetComponentInChildren<T>()
Component.GetComponentsInChildren(Type, bool)
Component.GetComponentsInChildren(Type)
Component.GetComponentsInChildren<T>(bool)
Component.GetComponentsInChildren<T>(bool, List<T>)
Component.GetComponentsInChildren<T>()
Component.GetComponentsInChildren<T>(List<T>)
Component.GetComponentInParent(Type, bool)
Component.GetComponentInParent(Type)
Component.GetComponentInParent<T>(bool)
Component.GetComponentInParent<T>()
Component.GetComponentsInParent(Type, bool)
Component.GetComponentsInParent(Type)
Component.GetComponentsInParent<T>(bool)
Component.GetComponentsInParent<T>(bool, List<T>)
Component.GetComponentsInParent<T>()
Component.GetComponents(Type)
Component.GetComponents(Type, List<Component>)
Component.GetComponents<T>(List<T>)
Component.GetComponents<T>()
Component.CompareTag(string)
Component.SendMessageUpwards(string, object, SendMessageOptions)
Component.SendMessageUpwards(string, object)
Component.SendMessageUpwards(string)
Component.SendMessageUpwards(string, SendMessageOptions)
Component.SendMessage(string, object)
Component.SendMessage(string)

Component.SendMessage(string, object, SendMessageOptions)
Component.SendMessage(string, SendMessageOptions)
Component.BroadcastMessage(string, object, SendMessageOptions)
Component.BroadcastMessage(string, object)
Component.BroadcastMessage(string)
Component.BroadcastMessage(string, SendMessageOptions)
Component.transform
Component.gameObject
Component.tag
Object.GetInstanceID()
Object.GetHashCode()
Object.Equals(object)
Object.Instantiate(Object, Vector3, Quaternion)
Object.Instantiate(Object, Vector3, Quaternion, Transform)
Object.Instantiate(Object)
Object.Instantiate(Object, Transform)
Object.Instantiate(Object, Transform, bool)
Object.Instantiate<T>(T)
Object.Instantiate<T>(T, Vector3, Quaternion)
Object.Instantiate<T>(T, Vector3, Quaternion, Transform)
Object.Instantiate<T>(T, Transform)
Object.Instantiate<T>(T, Transform, bool)
Object.Destroy(Object, float)
Object.Destroy(Object)
Object.DestroyImmediate(Object, bool)
Object.DestroyImmediate(Object)
Object.FindObjectsOfType(Type)
Object.FindObjectsOfType(Type, bool)
Object.DontDestroyOnLoad(Object)
Object.DestroyObject(Object, float)
Object.DestroyObject(Object)
Object.FindSceneObjectsOfType(Type)
Object.FindObjectsOfTypeIncludingAssets(Type)
Object.FindObjectsOfType<T>()
Object.FindObjectsOfType<T>(bool)
Object.FindObjectOfType<T>()
Object.FindObjectOfType<T>(bool)
Object.FindObjectsOfTypeAll(Type)
Object.FindObjectOfType(Type)
Object.FindObjectOfType(Type, bool)
Object.ToString()
Object.name
Object.hideFlags
object.Equals(object, object)
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
public class ReplayMaterial : ReplayRecordableBehaviour, IReplaySerialize
```

Fields

materialIndex

Declaration

```
[Tooltip("The index of the renderer material to record or '-1' if the main material should be used")]
public int materialIndex
```

Field Value

TYPE	DESCRIPTION
int	

observedRenderer

Declaration

```
public Renderer observedRenderer
```

Field Value

TYPE	DESCRIPTION
Renderer	

recordFlags

Declaration

```
[HideInInspector]
public ReplayMaterial.ReplayMaterialFlags recordFlags
```

Field Value

TYPE	DESCRIPTION
ReplayMaterial.ReplayMaterialFlags	

Methods

OnReplayDeserialize(ReplayState)

Called by the replay system when the recorder component should deserialize any necessary data during playback.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState containing the recorded data

Overrides

ReplayRecordableBehaviour.OnReplayDeserialize(ReplayState)

OnReplayReset()

Called by the replay system during playback when cached values should be reset to safe default to avoid glitches or inaccuracies

Called by the replay system during playback when cached values should be reset to safe default to avoid glitches or inaccuracies in the playback.

Declaration

```
protected override void OnReplayReset()
```

Overrides

ReplayBehaviour.OnReplayReset()

OnReplaySerialize(ReplayState)

Called by the replay system when the recorder component should serialize and necessary data during recording.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState used to store the serialized data

Overrides

ReplayRecordableBehaviour.OnReplaySerialize(ReplayState)

OnReplayUpdate(float)

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

Declaration

```
protected override void OnReplayUpdate(float t)
```

Parameters

TYPE	NAME	DESCRIPTION
float	t	A normalized value representing the progress between replay snapshots. Use this value for interpolation or similar smoothing passes

Overrides

ReplayBehaviour.OnReplayUpdate(float)

Reset()

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

Declaration

```
protected override void Reset()
```

Overrides

ReplayBehaviour.Reset()

Implements

IReplaySerialize

Enum ReplayMaterial.ReplayMaterialFlags

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
[Flags]
public enum ReplayMaterial.ReplayMaterialFlags
```

Fields

NAME	DESCRIPTION
Color	
DoubleSidedGlobalIllumination	
GlobalIlluminationFlags	
Interpolate	
MainTextureOffset	
MainTextureScale	
None	
SharedMaterial	

Class ReplayMaterialChange

Inheritance

object
Object
Component
Behaviour
MonoBehaviour
ReplayBehaviour
ReplayRecordableBehaviour
ReplayMaterialChange

Implements

IReplaySerialize

Inherited Members

ReplayRecordableBehaviour.Formatter
ReplayRecordableBehaviour.OnDestroy()
ReplayBehaviour.ReplayIdentity
ReplayBehaviour.ReplayObject
ReplayBehaviour.HasPersistentData
ReplayBehaviour.ReplayPersistentData
ReplayBehaviour.Variables
ReplayBehaviour.HasVariables
ReplayBehaviour.IsRecording
ReplayBehaviour.IsRecordingPaused
ReplayBehaviour.IsRecordingOrPaused
ReplayBehaviour.IsReplaying
ReplayBehaviour.IsPlaybackPaused
ReplayBehaviour.IsReplayingOrPaused
ReplayBehaviour.PlaybackTime
ReplayBehaviour.PlaybackTimeNormalized
ReplayBehaviour.PlaybackTimeScale
ReplayBehaviour.PlaybackDirection
ReplayBehaviour.Awake()
ReplayBehaviour.OnEnable()
ReplayBehaviour.OnDisable()
ReplayBehaviour.OnReplayStart()
ReplayBehaviour.OnReplayEnd()
ReplayBehaviour.OnReplayPlayPause(bool)
ReplayBehaviour.OnReplayReset()
ReplayBehaviour.OnReplayCapture()
ReplayBehaviour.OnReplayUpdate(float)
ReplayBehaviour.OnReplayEvent(ushort, ReplayState)
ReplayBehaviour.OnReplaySpawned(Vector3, Quaternion)
ReplayBehaviour.ForceRegenerateIdentity()
ReplayBehaviour.RecordVariable(ReplayVariable)
ReplayBehaviour.RecordEvent(ushort, ReplayState)
ReplayBehaviour.RecordMethodCall(Action)
ReplayBehaviour.RecordMethodCall<T>(Action<T>, T)
ReplayBehaviour.RecordMethodCall<T0, T1>(Action<T0, T1>, T0, T1)
ReplayBehaviour.RecordMethodCall<T0, T1, T2>(Action<T0, T1, T2>, T0, T1, T2)
ReplayBehaviour.RecordMethodCall<T0, T1, T2, T3>(Action<T0, T1, T2, T3>, T0, T1, T2, T3)

MonoBehaviour.IsInvoking()
MonoBehaviour.CancelInvoke()
MonoBehaviour.Invoke(string, float)
MonoBehaviour.InvokeRepeating(string, float, float)
MonoBehaviour.CancelInvoke(string)
MonoBehaviour.IsInvoking(string)
MonoBehaviour.StartCoroutine(string)
MonoBehaviour.StartCoroutine(string, object)
MonoBehaviour.StartCoroutine(IEnumerator)
MonoBehaviour.StartCoroutine_Auto(IEnumerator)
MonoBehaviour.StopCoroutine(IEnumerator)
MonoBehaviour.StopCoroutine(Coroutine)
MonoBehaviour.StopCoroutine(string)
MonoBehaviour.StopAllCoroutines()
MonoBehaviour.print(object)
MonoBehaviour.useGUILayout
MonoBehaviour.runInEditMode
Behaviour.enabled
Behaviour.isActiveAndEnabled
Component.GetComponent(Type)
Component.GetComponent<T>()
Component.TryGetComponent(Type, out Component)
Component.TryGetComponent<T>(out T)
Component.GetComponent(string)
Component.GetComponentInChildren(Type, bool)
Component.GetComponentInChildren(Type)
Component.GetComponentInChildren<T>(bool)
Component.GetComponentInChildren<T>()
Component.GetComponentsInChildren(Type, bool)
Component.GetComponentsInChildren(Type)
Component.GetComponentsInChildren<T>(bool)
Component.GetComponentsInChildren<T>(bool, List<T>)
Component.GetComponentsInChildren<T>()
Component.GetComponentsInChildren<T>(List<T>)
Component.GetComponentInParent(Type, bool)
Component.GetComponentInParent(Type)
Component.GetComponentInParent<T>(bool)
Component.GetComponentInParent<T>()
Component.GetComponentsInParent(Type, bool)
Component.GetComponentsInParent(Type)
Component.GetComponentsInParent<T>(bool)
Component.GetComponentsInParent<T>(bool, List<T>)
Component.GetComponentsInParent<T>()
Component.GetComponents(Type)
Component.GetComponents(Type, List<Component>)
Component.GetComponents<T>(List<T>)
Component.GetComponents<T>()
Component.CompareTag(string)
Component.SendMessageUpwards(string, object, SendMessageOptions)
Component.SendMessageUpwards(string, object)
Component.SendMessageUpwards(string)
Component.SendMessageUpwards(string, SendMessageOptions)

Component.SendMessage(string, object)
Component.SendMessage(string)
Component.SendMessage(string, object, SendMessageOptions)
Component.SendMessage(string, SendMessageOptions)
Component.BroadcastMessage(string, object, SendMessageOptions)
Component.BroadcastMessage(string, object)
Component.BroadcastMessage(string)
Component.BroadcastMessage(string, SendMessageOptions)
Component.transform
Component.gameObject
Component.tag
Object.GetInstanceID()
Object.GetHashCode()
Object.Equals(object)
Object.Instantiate(Object, Vector3, Quaternion)
Object.Instantiate(Object, Vector3, Quaternion, Transform)
Object.Instantiate(Object)
Object.Instantiate(Object, Transform)
Object.Instantiate(Object, Transform, bool)
Object.Instantiate<T>(T)
Object.Instantiate<T>(T, Vector3, Quaternion)
Object.Instantiate<T>(T, Vector3, Quaternion, Transform)
Object.Instantiate<T>(T, Transform)
Object.Instantiate<T>(T, Transform, bool)
Object.Destroy(Object, float)
Object.Destroy(Object)
Object.DestroyImmediate(Object, bool)
Object.DestroyImmediate(Object)
Object.FindObjectsOfType(Type)
Object.FindObjectsOfType(Type, bool)
Object.DontDestroyOnLoad(Object)
Object.DestroyObject(Object, float)
Object.DestroyObject(Object)
Object.FindSceneObjectsOfType(Type)
Object.FindObjectsOfTypeIncludingAssets(Type)
Object.FindObjectsOfType<T>()
Object.FindObjectsOfType<T>(bool)
Object.FindObjectOfType<T>()
Object.FindObjectOfType<T>(bool)
Object.FindObjectsOfTypeAll(Type)
Object.FindObjectOfType(Type)
Object.FindObjectOfType(Type, bool)
Object.ToString()
Object.name
Object.hideFlags
object.Equals(object, object)
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
public class ReplayMaterialChange : ReplayRecordableBehaviour, IReplaySerialize
```

Fields

availableMaterials

Declaration

```
public List<Material> availableMaterials
```

Field Value

TYPE	DESCRIPTION
List<Material>	

defaultMaterial

Declaration

```
public Material defaultMaterial
```

Field Value

TYPE	DESCRIPTION
Material	

observedRenderer

Declaration

```
public Renderer observedRenderer
```

Field Value

TYPE	DESCRIPTION
Renderer	

recordFlags

Declaration

```
[HideInInspector]
public ReplayMaterialChange.ReplayMaterialChangeFlags recordFlags
```

Field Value

TYPE	DESCRIPTION
ReplayMaterialChange.ReplayMaterialChangeFlags	

Methods

GetAssignedMaterialIndex(int)

Declaration

```
public int GetAssignedMaterialIndex(int slot = -1)
```

Parameters

TYPE	NAME	DESCRIPTION
int	slot	

Returns

TYPE	DESCRIPTION
int	

OnReplayDeserialize(ReplayState)

Called by the replay system when the recorder component should deserialize any necessary data during playback.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState containing the recorded data

Overrides

ReplayRecordableBehaviour.OnReplayDeserialize(ReplayState)

OnReplaySerialize(ReplayState)

Called by the replay system when the recorder component should serialize and necessary data during recording.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState used to store the serialized data

Overrides

ReplayRecordableBehaviour.OnReplaySerialize(ReplayState)

Reset()

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

Declaration

```
protected override void Reset()
```

Overrides

ReplayBehaviour.Reset()

Implements

IReplaySerialize

Enum ReplayMaterialChange.ReplayMaterialChangeFlags

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
[Flags]
public enum ReplayMaterialChange.ReplayMaterialChangeFlags
```

Fields

NAME	DESCRIPTION
AllMaterials	
None	
SharedMaterial	

Class ReplayMetadata

Stores all additional non-essential information about a replay. Can be useful to help display information about the replay such as when it was created, or which Unity scene is required for best playback accuracy. You can also derive from this class to add additional custom metadata fields that you would like to save. Note that only primitive types and arrays will be serialized and only reference types that are marked as SerializableAttribute will be saved (Serialization follows standard Unity practices but does not support reference types unless marked as serializable).

Inheritance

object

ReplayMetadata

Implements

IReplayStreamSerialize

IReplayTokenSerialize

Inherited Members

object.Equals(object)

object.Equals(object, object)

object.GetHashCode()

object.GetType()

object.MemberwiseClone()

object.ReferenceEquals(object, object)

object.ToString()

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
[Serializable]
public class ReplayMetadata : IReplayStreamSerialize, IReplayTokenSerialize
```

Constructors

ReplayMetadata()

Create a new instance.

Declaration

```
public ReplayMetadata()
```

ReplayMetadata(string)

Create a new instance with the specified metadata replay name.

Declaration

```
public ReplayMetadata(string replayName = null)
```

Parameters

TYPE	NAME	DESCRIPTION
string	replayName	The name of the replay

Properties

AppName

Get the name of the app that created this replay. By default this will use the value of UnityEngine.Application.productName.

Declaration

```
public string AppName { get; set; }
```

Property Value

TYPE	DESCRIPTION
string	

DeveloperName

Get the name of the app developer that created this replay. By default this will use the value of UnityEngine.Application.companyName.

Declaration

```
public string DeveloperName { get; set; }
```

Property Value

TYPE	DESCRIPTION
string	

ReplayName

A name for the replay to help identify it.

Declaration

```
public string ReplayName { get; set; }
```

Property Value

TYPE	DESCRIPTION
string	

SceneId

The id of the Unity scene that was active when the replay was recorded. Use UpdateSceneMetadata(Scene) to modify this value.

Declaration

```
public int SceneId { get; }
```

Property Value

TYPE	DESCRIPTION
int	

SceneName

The name of the Unity scene that was active when the replay was recorded. Use UpdateSceneMetadata(Scene) to modify this value.

Declaration

```
public string SceneName { get; }
```

Property Value

TYPE	DESCRIPTION
string	

ScenePath

The path of the Unity scene that was active when the replay was recorded. Use UpdateSceneMetadata(Scene) to modify this value.

Declaration

```
public string ScenePath { get; }
```

Property Value

TYPE	DESCRIPTION
string	

TypeName

Get serializable type name of this metadata type.

Declaration

```
public string TypeName { get; }
```

Property Value

TYPE	DESCRIPTION
string	

UserName

Get the name of the user that created this replay. By default this will use the value of UserName.

Declaration

```
public string UserName { get; set; }
```

Property Value

TYPE	DESCRIPTION
string	

Methods

CopyTo(ReplayMetadata)

Copy the current metadata to the specified metadata object.

Declaration

```
public bool CopyTo(ReplayMetadata destination)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayMetadata	destination	

Returns

TYPE	DESCRIPTION
bool	True if the copy was successful or false if not

Exceptions

TYPE	CONDITION
ArgumentNullException	

CreateFromType(string)

Create a ReplayMetadata instance from the specified type name. The type name must be a valid ReplayMetadata type or derived type.

Declaration

```
public static ReplayMetadata CreateFromType(string typeName)
```

Parameters

TYPE	NAME	DESCRIPTION
string	typeName	The type name

Returns

TYPE	DESCRIPTION
ReplayMetadata	

UpdateMetadata()

Update all metadata from default sources. Scene information will be updated from UnityEngine.SceneManagement.SceneManager.GetActiveScene() and company and product info will be updated based on Unity player settings.

Declaration

```
public void UpdateMetadata()
```

UpdateSceneMetadata(Scene)

Update all metadata related to scene info from the specified scene.

Declaration

```
public void UpdateSceneMetadata(Scene scene)
```

Parameters

TYPE	NAME	DESCRIPTION
Scene	scene	The Unity scene to store metadata for

Implements

IReplayStreamSerialize
IReplayTokenSerialize

Class ReplayMethodAttribute

Use this attribute to mark a method declared in a ReplayBehaviour script as recordable. The target method must not return a value and must only use primitive parameter types up to a limit of 4 arguments.

Inheritance

object
Attribute
ReplayMethodAttribute

Implements

_Attribute

Inherited Members

Attribute.Equals(object)
Attribute.GetCustomAttribute(Assembly, Type)
Attribute.GetCustomAttribute(Assembly, Type, bool)
Attribute.GetCustomAttribute(MemberInfo, Type)
Attribute.GetCustomAttribute(MemberInfo, Type, bool)
Attribute.GetCustomAttribute(Module, Type)
Attribute.GetCustomAttribute(Module, Type, bool)
Attribute.GetCustomAttribute(ParameterInfo, Type)
Attribute.GetCustomAttribute(ParameterInfo, Type, bool)
Attribute.GetCustomAttributes(Assembly)
Attribute.GetCustomAttributes(Assembly, bool)
Attribute.GetCustomAttributes(Assembly, Type)
Attribute.GetCustomAttributes(Assembly, Type, bool)
Attribute.GetCustomAttributes(MemberInfo)
Attribute.GetCustomAttributes(MemberInfo, bool)
Attribute.GetCustomAttributes(MemberInfo, Type)
Attribute.GetCustomAttributes(MemberInfo, Type, bool)
Attribute.GetCustomAttributes(Module)
Attribute.GetCustomAttributes(Module, bool)
Attribute.GetCustomAttributes(Module, Type)
Attribute.GetCustomAttributes(Module, Type, bool)
Attribute.GetCustomAttributes(ParameterInfo)
Attribute.GetCustomAttributes(ParameterInfo, bool)
Attribute.GetCustomAttributes(ParameterInfo, Type)
Attribute.GetCustomAttributes(ParameterInfo, Type, bool)
Attribute.GetHashCode()
Attribute.IsDefaultAttribute()
Attribute.IsDefined(Assembly, Type)
Attribute.IsDefined(Assembly, Type, bool)
Attribute.IsDefined(MemberInfo, Type)
Attribute.IsDefined(MemberInfo, Type, bool)
Attribute.IsDefined(Module, Type)
Attribute.IsDefined(Module, Type, bool)
Attribute.IsDefined(ParameterInfo, Type)
Attribute.IsDefined(ParameterInfo, Type, bool)
Attribute.Match(object)
Attribute.TypeId
object.Equals(object, object)
object.GetType()

object.ReferenceEquals(object, object)

object.ToString()

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
[AttributeUsage(AttributeTargets.Method, AllowMultiple = false, Inherited = false)]
public sealed class ReplayMethodAttribute : Attribute, _Attribute
```

Implements

_Attribute

Class ReplayObject

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

Inheritance

object
Object
Component
Behaviour
MonoBehaviour
ReplayObject

Implements

IReplaySerialize
ISerializationCallbackReceiver

Inherited Members

MonoBehaviour.IsInvoking()
MonoBehaviour.CancelInvoke()
MonoBehaviour.Invoke(string, float)
MonoBehaviour.InvokeRepeating(string, float, float)
MonoBehaviour.CancelInvoke(string)
MonoBehaviour.IsInvoking(string)
MonoBehaviour.StartCoroutine(string)
MonoBehaviour.StartCoroutine(string, object)
MonoBehaviour.StartCoroutine(IEnumerator)
MonoBehaviour.StartCoroutine_Auto(IEnumerator)
MonoBehaviour.StopCoroutine(IEnumerator)
MonoBehaviour.StopCoroutine(Coroutine)
MonoBehaviour.StopCoroutine(string)
MonoBehaviour.StopAllCoroutines()
MonoBehaviour.print(object)
MonoBehaviour.useGUILayout
MonoBehaviour.runInEditMode
Behaviour.enabled
Behaviour.isActiveAndEnabled
Component.GetComponent(Type)
Component.GetComponent<T>()
Component.TryGetComponent(Type, out Component)
Component.TryGetComponent<T>(out T)
Component.GetComponent(string)
Component.GetComponentInChildren(Type, bool)
Component.GetComponentInChildren(Type)
Component.GetComponentInChildren<T>(bool)
Component.GetComponentInChildren<T>()
Component.GetComponentsInChildren(Type, bool)
Component.GetComponentsInChildren(Type)
Component.GetComponentsInChildren<T>(bool)
Component.GetComponentsInChildren<T>(bool, List<T>)
Component.GetComponentsInChildren<T>()
Component.GetComponentsInChildren<T>(List<T>)
Component.GetComponentInParent(Type, bool)
Component.GetComponentInParent(Type)

Component.GetComponentInParent<T>(bool)
Component.GetComponentInParent<T>()
Component.GetComponentsInParent(Type, bool)
Component.GetComponentsInParent(Type)
Component.GetComponentsInParent<T>(bool)
Component.GetComponentsInParent<T>(bool, List<T>)
Component.GetComponentsInParent<T>()
Component.GetComponents(Type)
Component.GetComponents(Type, List<Component>)
Component.GetComponents<T>(List<T>)
Component.GetComponents<T>()
Component.CompareTag(string)
Component.SendMessageUpwards(string, object, SendMessageOptions)
Component.SendMessageUpwards(string, object)
Component.SendMessageUpwards(string)
Component.SendMessageUpwards(string, SendMessageOptions)
Component.SendMessage(string, object)
Component.SendMessage(string)
Component.SendMessage(string, object, SendMessageOptions)
Component.SendMessage(string, SendMessageOptions)
Component.BroadcastMessage(string, object, SendMessageOptions)
Component.BroadcastMessage(string, object)
Component.BroadcastMessage(string)
Component.BroadcastMessage(string, SendMessageOptions)
Component.transform
Component.gameObject
Component.tag
Object.GetInstanceID()
Object.GetHashCode()
Object.Equals(object)
Object.Instantiate(Object, Vector3, Quaternion)
Object.Instantiate(Object, Vector3, Quaternion, Transform)
Object.Instantiate(Object)
Object.Instantiate(Object, Transform)
Object.Instantiate(Object, Transform, bool)
Object.Instantiate<T>(T)
Object.Instantiate<T>(T, Vector3, Quaternion)
Object.Instantiate<T>(T, Vector3, Quaternion, Transform)
Object.Instantiate<T>(T, Transform)
Object.Instantiate<T>(T, Transform, bool)
Object.Destroy(Object, float)
Object.Destroy(Object)
Object.DestroyImmediate(Object, bool)
Object.DestroyImmediate(Object)
Object.FindObjectsOfType(Type)
Object.FindObjectsOfType(Type, bool)
Object.DontDestroyOnLoad(Object)
Object.DestroyObject(Object, float)
Object.DestroyObject(Object)
Object.FindSceneObjectsOfType(Type)
Object.FindObjectsOfTypeIncludingAssets(Type)
Object.FindObjectsOfType<T>()

Object.FindObjectsOfType<T>(bool)
Object.FindObjectOfType<T>()
Object.FindObjectOfType<T>(bool)
Object.FindObjectsOfTypeAll(Type)
Object.FindObjectOfType(Type)
Object.FindObjectOfType(Type, bool)
Object.ToString()
Object.name
Object.hideFlags
object.Equals(object, object)
object.GetType()
object.ReferenceEquals(object, object)

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
[ExecuteInEditMode]
[DisallowMultipleComponent]
[DefaultExecutionOrder(-100)]
public sealed class ReplayObject : MonoBehaviour, IReplaySerialize, ISerializationCallbackReceiver
```

Properties

AllReplayObjects

Get all registered replay objects that exist in all loaded scenes.

Declaration

```
public static HashSet<ReplayObject> AllReplayObjects { get; }
```

Property Value

TYPE	DESCRIPTION
HashSet<ReplayObject>	

Behaviours

Get all replay behaviours managed by this replay object.

Declaration

```
public IReadOnlyList<ReplayBehaviour> Behaviours { get; }
```

Property Value

TYPE	DESCRIPTION
IReadOnlyList<ReplayBehaviour>	

IsPlaybackPaused

Returns a value indicating whether this replay object is included in a playback operation that is currently paused.

Declaration

```
public bool IsPlaybackPaused { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

IsPrefab

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

Declaration

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

Property Value

TYPE	DESCRIPTION
bool	

IsRecording

Returns a value indicating whether this replay object is included in an active record operation. This value will be false if recording is paused. IsRecordingPaused to check if the recording has been paused, or IsReplayingOrPaused to get an inclusive value.

Declaration

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

Property Value

TYPE	DESCRIPTION
bool	

IsRecordingOrPaused

Returns a value indicating whether this replay object is included in an active or paused record operation.

Declaration

```
public bool IsRecordingOrPaused { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

IsRecordingPaused

Returns a value indicating whether this replay object is included in any record operation that is currently paused.

Declaration

```
public bool IsRecordingPaused { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

IsReplaying

Returns a value indicating whether this replay object is included in an active replay operation. This value will be false if the replay is paused. IsPlaybackPaused to check if the replay has been paused, or IsReplayingOrPaused to get an inclusive value.

Declaration

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

Property Value

TYPE	DESCRIPTION
bool	

IsReplayingOrPaused

Returns a value indicating whether this replay object is included in an active or paused replay operation.

Declaration

```
public bool IsReplayingOrPaused { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

LifecycleProvider

Get the ReplayObjectLifecycleProvider responsible for the creation and destruction of this replay object.

Declaration

```
public ReplayObjectLifecycleProvider LifecycleProvider { get; }
```

Property Value

TYPE	DESCRIPTION
ReplayObjectLifecycleProvider	

ObservedComponents

Get all replay components that are observed and managed by this replay object.

Declaration

```
public IReadOnlyList<ReplayBehaviour> ObservedComponents { get; }
```

Property Value

TYPE	DESCRIPTION
IReadOnlyList<ReplayBehaviour>	

TYPE	DESCRIPTION

PlaybackOperation

Get the current playback operation for this replay object if it is currently part of a replay. It is only possible for any given replay object to be associated with a single playback operation at any time, although an object can be recorded multiple times.

Declaration

```
public ReplayPlaybackOperation PlaybackOperation { get; }
```

Property Value

TYPE	DESCRIPTION
ReplayPlaybackOperation	

PrefabIdentity

Get the unique prefab ReplayIdentity for this ReplayObject which links to the associated replay prefab.

Declaration

```
public ReplayIdentity PrefabIdentity { get; }
```

Property Value

TYPE	DESCRIPTION
ReplayIdentity	

RecordOperations

Get all record operations that this replay object is currently associated with. It is possible for any given replay object to be recorded by multiple difference record operations simultaneously.

Declaration

```
public IReadOnlyList<ReplayRecordOperation> RecordOperations { get; }
```

Property Value

TYPE	DESCRIPTION
IReadOnlyList<ReplayRecordOperation>	

ReplayIdentity

Get the unique ReplayIdentity for this ReplayObject.

Declaration

```
public ReplayIdentity ReplayIdentity { get; set; }
```

Property Value

TYPE	DESCRIPTION
ReplayIdentity	

Methods

Call(ReplayIdentity, Action)

Declaration

```
public void Call(ReplayIdentity senderIdentity, Action method)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayIdentity	senderIdentity	
Action	method	

Call<T>(ReplayIdentity, Action<T>, T)

Declaration

```
public void Call<T>(ReplayIdentity senderIdentity, Action<T> method, T arg)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayIdentity	senderIdentity	
Action<T>	method	
T	arg	

Type Parameters

NAME	DESCRIPTION
T	

Call<T0, T1>(ReplayIdentity, Action<T0, T1>, T0, T1)

Declaration

```
public void Call<T0, T1>(ReplayIdentity senderIdentity, Action<T0, T1> method, T0 arg0, T1 arg1)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayIdentity	senderIdentity	
Action<T0, T1>	method	
T0	arg0	

TYPE	NAME	DESCRIPTION
T1	arg1	

Type Parameters

NAME	DESCRIPTION
T0	
T1	

Call<T0, T1, T2>(ReplayIdentity, Action<T0, T1, T2>, T0, T1, T2)

Declaration

```
public void Call<T0, T1, T2>(ReplayIdentity senderIdentity, Action<T0, T1, T2> method, T0 arg0, T1 arg1, T2
arg2)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayIdentity	senderIdentity	
Action<T0, T1, T2>	method	
T0	arg0	
T1	arg1	
T2	arg2	

Type Parameters

NAME	DESCRIPTION
T0	
T1	
T2	

Call<T0, T1, T2, T3>(ReplayIdentity, Action<T0, T1, T2, T3>, T0, T1, T2, T3)

Declaration

```
public void Call<T0, T1, T2, T3>(ReplayIdentity senderIdentity, Action<T0, T1, T2, T3> method, T0 arg0, T1
arg1, T2 arg2, T3 arg3)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayIdentity	senderIdentity	
Action<T0, T1, T2, T3>	method	

TYPE	NAME	DESCRIPTION
T0	arg0	
T1	arg1	
T2	arg2	
T3	arg3	

Type Parameters

NAME	DESCRIPTION
T0	
T1	
T2	
T3	

CheckComponentListIntegrity()

Returns a value indicating whether the observed component list is valid or needs o be rebuilt.

Declaration

```
public bool CheckComponentListIntegrity()
```

Returns

TYPE	DESCRIPTION
bool	True if the collection is valid or false if not

CloneReplayObjectIdentity(ReplayObject, ReplayObject)

Declaration

```
public static bool CloneReplayObjectIdentity(ReplayObject cloneFromObject, ReplayObject cloneToObject)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayObject	cloneFromObject	
ReplayObject	cloneToObject	

Returns

TYPE	DESCRIPTION
bool	

CloneReplayObjectIdentity(GameObject, GameObject)

Declaration

```
public static bool CloneReplayObjectIdentity(GameObject cloneFromObject, GameObject cloneToObject)
```

Parameters

TYPE	NAME	DESCRIPTION
GameObject	cloneFromObject	
GameObject	cloneToObject	

Returns

TYPE	DESCRIPTION
bool	

ForceRegenerateIdentity()

Force the ReplayIdentity to be regenerated with a unique value.

Declaration

```
public void ForceRegenerateIdentity()
```

ForceRegenerateIdentityWithObservedComponents()

Force the ReplayIdentity and all observed component id's to be regenerated with unique values.

Declaration

```
public void ForceRegenerateIdentityWithObservedComponents()
```

GetReplayBehaviour(ReplayIdentity)

Get the ReplayBehaviour observed by this ReplayObject with the specified ReplayIdentity.

Declaration

```
public ReplayBehaviour GetReplayBehaviour(ReplayIdentity replayIdentity)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayIdentity	replayIdentity	

Returns

TYPE	DESCRIPTION
ReplayBehaviour	

IsComponentObserved(ReplayBehaviour)

Returns a value indicating whether the specified recorder component is observed by this ReplayObject.

Declaration

```
public bool IsComponentObserved(ReplayBehaviour component)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayBehaviour	component	The recorder component to check

Returns

TYPE	DESCRIPTION
bool	True if the component is observed or false if not

OnReplayDeserialize(ReplayState)

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

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState to deserialize the data to

OnReplayDeserialize(ReplayState, bool)

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

Declaration

```
public void OnReplayDeserialize(ReplayState state, bool simulate)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState to deserialize the data from
bool	simulate	True if replay components should be simulated

OnReplaySerialize(ReplayState)

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

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState to serialize the data to

RebuildComponentList()

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.

Declaration

```
public void RebuildComponentList()
```

RecordReplayEvent(ReplayIdentity, ushort, ReplayState)

Declaration

```
public void RecordReplayEvent(ReplayIdentity senderIdentity, ushort eventID, ReplayState eventData = null)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayIdentity	senderIdentity	
ushort	eventID	
ReplayState	eventData	

RecordReplayVariable(ReplayIdentity, ReplayVariable)

Declaration

```
public void RecordReplayVariable(ReplayIdentity senderIdentity, ReplayVariable replayVariable)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayIdentity	senderIdentity	
ReplayVariable	replayVariable	

Reset()

Called by Unity editor. Can also be called by scripts to force update the component.

Declaration

```
public void Reset()
```

UpdateRuntimeComponents()

Declaration

```
public void UpdateRuntimeComponents()
```

Implements

IReplaySerialize

UnityEngine.ISerializationCallbackReceiver

Struct ReplayObject.ReplayObjectReference

Inherited Members

ValueType.Equals(object)
ValueType.GetHashCode()
ValueType.ToString()
object.Equals(object, object)
object.GetType()
object.ReferenceEquals(object, object)

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
[Serializable]
public struct ReplayObject.ReplayObjectReference
```

Constructors

ReplayObjectReference(ReplayObject)

Declaration

```
public ReplayObjectReference(ReplayObject obj)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayObject	obj	

Fields

reference

Declaration

```
public ReplayObject reference
```

Field Value

TYPE	DESCRIPTION
ReplayObject	

Class ReplayOperation

Represents a dedicated replay operation in progress. Provides access to API's common to both recording and playback operations.

Inheritance

object
ReplayOperation
ReplayPlaybackOperation
ReplayRecordOperation

Implements

IDisposable

Inherited Members

object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: UltimateReplay
Assembly: UltimateReplay.dll

Syntax

```
public abstract class ReplayOperation : IDisposable
```

Constructors

ReplayOperation(ReplayManager, ReplayScene, ReplayStorage)

Create a new replay operation.

Declaration

```
protected ReplayOperation(ReplayManager manager, ReplayScene scene, ReplayStorage storage)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayManager	manager	The replay manager instance that will perform updates for this operation
ReplayScene	scene	The replay scene associated with this replay operation
ReplayStorage	storage	The replay storage associated with this replay operation

Exceptions

TYPE	CONDITION

TYPE	CONDITION
ArgumentNullException	The specified replay manager is null

Fields

manager

The replay manager instance.

Declaration

```
protected ReplayManager manager
```

Field Value

TYPE	DESCRIPTION
ReplayManager	

scene

The replay scene associated with this replay operation.

Declaration

```
protected ReplayScene scene
```

Field Value

TYPE	DESCRIPTION
ReplayScene	

storage

The replay storage associated with this replay operation.

Declaration

```
protected ReplayStorage storage
```

Field Value

TYPE	DESCRIPTION
ReplayStorage	

Properties

IsDisposed

Check if this replay operation has been disposed.

Declaration

```
public abstract bool IsDisposed { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

Scene

Get the replay scene associated with this replay operation. The replay scene contains information about all replay objects currently being recorded or replayed by this operation. Note that it is possible for multiple replay objects to appear in many different replay scenes.

Declaration

```
public ReplayScene Scene { get; }
```

Property Value

TYPE	DESCRIPTION
ReplayScene	

Storage

Get the replay storage associated with this replay operation.

Declaration

```
public ReplayStorage Storage { get; }
```

Property Value

TYPE	DESCRIPTION
ReplayStorage	

UpdateMode

Get the ReplayUpdateMode for this replay operation. This value determines at what stage in the Unity game loop the replay operation is updated.

Declaration

```
public abstract ReplayUpdateMode UpdateMode { get; }
```

Property Value

TYPE	DESCRIPTION
ReplayUpdateMode	

Methods

CheckDisposed()

Throw an exception if this replay operation has been disposed.

Declaration

```
protected abstract void CheckDisposed()
```

Exceptions

TYPE	CONDITION
ObjectDisposedException	The replay operation was disposed

Dispose()

Dispose this replay operation. This will cause the operation to be stopped and this operation should no longer be used.

Declaration

```
public abstract void Dispose()
```

ReplayTick(float)

Should be called with a delta time value to update the replay operation manually. Make sure that UpdateMode is set to Manual to take full control over the update cycle. Delta time should be the amount of time in seconds since the last ReplayTick(float) call was made. Can be called multiple times per frame, but note that replay objects in the scene may not have moved since the last tick in this case.

Declaration

```
public abstract void ReplayTick(float delta)
```

Parameters

TYPE	NAME	DESCRIPTION
float	delta	

ReplayTickFixedUpdate(float)

Should be called from Unity 'FixedUpdate' method to update the replay operation. Will not do anything if UpdateMode is not set to FixedUpdate. See also ReplayTick(float) to update the operation manually.

Declaration

```
public void ReplayTickFixedUpdate(float deltaTime)
```

Parameters

TYPE	NAME	DESCRIPTION
float	deltaTime	The amount of time in seconds that has passed since the last update. This value must be greater than 0

ReplayTickLateUpdate(float)

Should be called from Unity 'LateUpdate' method to update the replay operation. Will not do anything if UpdateMode is not set to LateUpdate. See also ReplayTick(float) to update the operation manually.

Declaration

```
public void ReplayTickLateUpdate(float deltaTime)
```

Parameters

TYPE	NAME	DESCRIPTION
float	deltaTime	The amount of time in seconds that has passed since the last update. This value must be greater than 0

ReplayTickUpdate(float)

Should be called from Unity 'Update' method to update the replay operation. Will not do anything if UpdateMode is not set to Update. See also ReplayTick(float) to update the operation manually.

Declaration

```
public void ReplayTickUpdate(float deltaTime)
```

Parameters

TYPE	NAME	DESCRIPTION
float	deltaTime	The amount of time in seconds that has passed since the last update. This value must be greater than 0

Implements

IDisposable

Class ReplayParentChange

Inheritance

object
Object
Component
Behaviour
MonoBehaviour
ReplayBehaviour
ReplayRecordableBehaviour
ReplayParentChange

Implements

IReplaySerialize

Inherited Members

ReplayRecordableBehaviour.OnDestroy()
ReplayBehaviour.ReplayIdentity
ReplayBehaviour.ReplayObject
ReplayBehaviour.HasPersistentData
ReplayBehaviour.ReplayPersistentData
ReplayBehaviour.Variables
ReplayBehaviour.HasVariables
ReplayBehaviour.IsRecording
ReplayBehaviour.IsRecordingPaused
ReplayBehaviour.IsRecordingOrPaused
ReplayBehaviour.IsReplaying
ReplayBehaviour.IsPlaybackPaused
ReplayBehaviour.IsReplayingOrPaused
ReplayBehaviour.PlaybackTime
ReplayBehaviour.PlaybackTimeNormalized
ReplayBehaviour.PlaybackTimeScale
ReplayBehaviour.PlaybackDirection
ReplayBehaviour.Reset()
ReplayBehaviour.OnEnable()
ReplayBehaviour.OnDisable()
ReplayBehaviour.OnReplayStart()
ReplayBehaviour.OnReplayEnd()
ReplayBehaviour.OnReplayPlayPause(bool)
ReplayBehaviour.OnReplayReset()
ReplayBehaviour.OnReplayCapture()
ReplayBehaviour.OnReplayUpdate(float)
ReplayBehaviour.OnReplayEvent(ushort, ReplayState)
ReplayBehaviour.OnReplaySpawned(Vector3, Quaternion)
ReplayBehaviour.ForceRegenerateIdentity()
ReplayBehaviour.RecordVariable(ReplayVariable)
ReplayBehaviour.RecordEvent(ushort, ReplayState)
ReplayBehaviour.RecordMethodCall(Action)
ReplayBehaviour.RecordMethodCall<T>(Action<T>, T)
ReplayBehaviour.RecordMethodCall<T0, T1>(Action<T0, T1>, T0, T1)
ReplayBehaviour.RecordMethodCall<T0, T1, T2>(Action<T0, T1, T2>, T0, T1, T2)
ReplayBehaviour.RecordMethodCall<T0, T1, T2, T3>(Action<T0, T1, T2, T3>, T0, T1, T2, T3)
MonoBehaviour.IsInvoking()

MonoBehaviour.CancelInvoke()
MonoBehaviour.Invoke(string, float)
MonoBehaviour.InvokeRepeating(string, float, float)
MonoBehaviour.CancelInvoke(string)
MonoBehaviour.IsInvoking(string)
MonoBehaviour.StartCoroutine(string)
MonoBehaviour.StartCoroutine(string, object)
MonoBehaviour.StartCoroutine(IEnumerator)
MonoBehaviour.StartCoroutine_Auto(IEnumerator)
MonoBehaviour.StopCoroutine(IEnumerator)
MonoBehaviour.StopCoroutine(Coroutine)
MonoBehaviour.StopCoroutine(string)
MonoBehaviour.StopAllCoroutines()
MonoBehaviour.print(object)
MonoBehaviour.useGUILayout
MonoBehaviour.runInEditMode
Behaviour.enabled
Behaviour.isActiveAndEnabled
Component.GetComponent(Type)
Component.GetComponent<T>()
Component.TryGetComponent(Type, out Component)
Component.TryGetComponent<T>(out T)
Component.GetComponent(string)
Component.GetComponentInChildren(Type, bool)
Component.GetComponentInChildren(Type)
Component.GetComponentInChildren<T>(bool)
Component.GetComponentInChildren<T>()
Component.GetComponentsInChildren(Type, bool)
Component.GetComponentsInChildren(Type)
Component.GetComponentsInChildren<T>(bool)
Component.GetComponentsInChildren<T>(bool, List<T>)
Component.GetComponentsInChildren<T>()
Component.GetComponentsInChildren<T>(List<T>)
Component.GetComponentInParent(Type, bool)
Component.GetComponentInParent(Type)
Component.GetComponentInParent<T>(bool)
Component.GetComponentInParent<T>()
Component.GetComponentsInParent(Type, bool)
Component.GetComponentsInParent(Type)
Component.GetComponentsInParent<T>(bool)
Component.GetComponentsInParent<T>(bool, List<T>)
Component.GetComponentsInParent<T>()
Component.GetComponents(Type)
Component.GetComponents(Type, List<Component>)
Component.GetComponents<T>(List<T>)
Component.GetComponents<T>()
Component.CompareTag(string)
Component.SendMessageUpwards(string, object, SendMessageOptions)
Component.SendMessageUpwards(string, object)
Component.SendMessageUpwards(string)
Component.SendMessageUpwards(string, SendMessageOptions)
Component.SendMessage(string, object)

Component.SendMessage(string)
Component.SendMessage(string, object, SendMessageOptions)
Component.SendMessage(string, SendMessageOptions)
Component.BroadcastMessage(string, object, SendMessageOptions)
Component.BroadcastMessage(string, object)
Component.BroadcastMessage(string)
Component.BroadcastMessage(string, SendMessageOptions)
Component.transform
Component.gameObject
Component.tag
Object.GetInstanceID()
Object.GetHashCode()
Object.Equals(object)
Object.Instantiate(Object, Vector3, Quaternion)
Object.Instantiate(Object, Vector3, Quaternion, Transform)
Object.Instantiate(Object)
Object.Instantiate(Object, Transform)
Object.Instantiate(Object, Transform, bool)
Object.Instantiate<T>(T)
Object.Instantiate<T>(T, Vector3, Quaternion)
Object.Instantiate<T>(T, Vector3, Quaternion, Transform)
Object.Instantiate<T>(T, Transform)
Object.Instantiate<T>(T, Transform, bool)
Object.Destroy(Object, float)
Object.Destroy(Object)
Object.DestroyImmediate(Object, bool)
Object.DestroyImmediate(Object)
Object.FindObjectsOfType(Type)
Object.FindObjectsOfType(Type, bool)
Object.DontDestroyOnLoad(Object)
Object.DestroyObject(Object, float)
Object.DestroyObject(Object)
Object.FindSceneObjectsOfType(Type)
Object.FindObjectsOfTypeIncludingAssets(Type)
Object.FindObjectsOfType<T>()
Object.FindObjectsOfType<T>(bool)
Object.FindObjectOfType<T>()
Object.FindObjectOfType<T>(bool)
Object.FindObjectsOfTypeAll(Type)
Object.FindObjectOfType(Type)
Object.FindObjectOfType(Type, bool)
Object.ToString()
Object.name
Object.hideFlags
object.Equals(object, object)
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
public class ReplayParentChange : ReplayRecordableBehaviour, IReplaySerialize
```

Properties

Formatter

An optional ReplayFormatter that is used to serialize a particular component. Providing a formatter via his property can greatly reduce the amount of data that a ReplayObject needs to store.

Declaration

```
public override ReplayFormatter Formatter { get; }
```

Property Value

TYPE	DESCRIPTION
ReplayFormatter	

Overrides

ReplayRecordableBehaviour.Formatter

Methods

Awake()

Called by Unity.

Declaration

```
protected override void Awake()
```

Overrides

ReplayBehaviour.Awake()

OnReplayDeserialize(ReplayState)

Called by the replay system when the recorder component should deserialize any necessary data during playback.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState containing the recorded data

Overrides

ReplayRecordableBehaviour.OnReplayDeserialize(ReplayState)

OnReplaySerialize(ReplayState)

Called by the replay system when the recorder component should serialize and necessary data during recording.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState used to store the serialized data

Overrides

ReplayRecordableBehaviour.OnReplaySerialize(ReplayState)

Implements

IReplaySerialize

Class ReplayParticleSystem

A replay component which can be used to record and replay the Unity ParticleSystem.

Inheritance

object

Object

Component

Behaviour

MonoBehaviour

ReplayBehaviour

ReplayRecordableBehaviour

ReplayParticleSystem

Implements

IReplaySerialize

Inherited Members

ReplayRecordableBehaviour.Formatter

ReplayRecordableBehaviour.OnDestroy()

ReplayBehaviour.ReplayIdentity

ReplayBehaviour.ReplayObject

ReplayBehaviour.HasPersistentData

ReplayBehaviour.ReplayPersistentData

ReplayBehaviour.Variables

ReplayBehaviour.HasVariables

ReplayBehaviour.IsRecording

ReplayBehaviour.IsRecordingPaused

ReplayBehaviour.IsRecordingOrPaused

ReplayBehaviour.IsReplaying

ReplayBehaviour.IsPlaybackPaused

ReplayBehaviour.IsReplayingOrPaused

ReplayBehaviour.PlaybackTime

ReplayBehaviour.PlaybackTimeNormalized

ReplayBehaviour.PlaybackTimeScale

ReplayBehaviour.PlaybackDirection

ReplayBehaviour.Awake()

ReplayBehaviour.OnEnable()

ReplayBehaviour.OnDisable()

ReplayBehaviour.OnReplayStart()

ReplayBehaviour.OnReplayEnd()

ReplayBehaviour.OnReplayPlayPause(bool)

ReplayBehaviour.OnReplayCapture()

ReplayBehaviour.OnReplayEvent(ushort, ReplayState)

ReplayBehaviour.OnReplaySpawned(Vector3, Quaternion)

ReplayBehaviour.ForceRegenerateIdentity()

ReplayBehaviour.RecordVariable(ReplayVariable)

ReplayBehaviour.RecordEvent(ushort, ReplayState)

ReplayBehaviour.RecordMethodCall(Action)

ReplayBehaviour.RecordMethodCall<T>(Action<T>, T)

ReplayBehaviour.RecordMethodCall<T0, T1>(Action<T0, T1>, T0, T1)

ReplayBehaviour.RecordMethodCall<T0, T1, T2>(Action<T0, T1, T2>, T0, T1, T2)

ReplayBehaviour.RecordMethodCall<T0, T1, T2, T3>(Action<T0, T1, T2, T3>, T0, T1, T2, T3)

MonoBehaviour.IsInvoking()
MonoBehaviour.CancelInvoke()
MonoBehaviour.Invoke(string, float)
MonoBehaviour.InvokeRepeating(string, float, float)
MonoBehaviour.CancelInvoke(string)
MonoBehaviour.IsInvoking(string)
MonoBehaviour.StartCoroutine(string)
MonoBehaviour.StartCoroutine(string, object)
MonoBehaviour.StartCoroutine(IEnumerator)
MonoBehaviour.StartCoroutine_Auto(IEnumerator)
MonoBehaviour.StopCoroutine(IEnumerator)
MonoBehaviour.StopCoroutine(Coroutine)
MonoBehaviour.StopCoroutine(string)
MonoBehaviour.StopAllCoroutines()
MonoBehaviour.print(object)
MonoBehaviour.useGUILayout
MonoBehaviour.runInEditMode
Behaviour.enabled
Behaviour.isActiveAndEnabled
Component.GetComponent(Type)
Component.GetComponent<T>()
Component.TryGetComponent(Type, out Component)
Component.TryGetComponent<T>(out T)
Component.GetComponent(string)
Component.GetComponentInChildren(Type, bool)
Component.GetComponentInChildren(Type)
Component.GetComponentInChildren<T>(bool)
Component.GetComponentInChildren<T>()
Component.GetComponentsInChildren(Type, bool)
Component.GetComponentsInChildren(Type)
Component.GetComponentsInChildren<T>(bool)
Component.GetComponentsInChildren<T>(bool, List<T>)
Component.GetComponentsInChildren<T>()
Component.GetComponentsInChildren<T>(List<T>)
Component.GetComponentInParent(Type, bool)
Component.GetComponentInParent(Type)
Component.GetComponentInParent<T>(bool)
Component.GetComponentInParent<T>()
Component.GetComponentsInParent(Type, bool)
Component.GetComponentsInParent(Type)
Component.GetComponentsInParent<T>(bool)
Component.GetComponentsInParent<T>(bool, List<T>)
Component.GetComponentsInParent<T>()
Component.GetComponents(Type)
Component.GetComponents(Type, List<Component>)
Component.GetComponents<T>(List<T>)
Component.GetComponents<T>()
Component.CompareTag(string)
Component.SendMessageUpwards(string, object, SendMessageOptions)
Component.SendMessageUpwards(string, object)
Component.SendMessageUpwards(string)
Component.SendMessageUpwards(string, SendMessageOptions)

Component.SendMessage(string, object)
Component.SendMessage(string)
Component.SendMessage(string, object, SendMessageOptions)
Component.SendMessage(string, SendMessageOptions)
Component.BroadcastMessage(string, object, SendMessageOptions)
Component.BroadcastMessage(string, object)
Component.BroadcastMessage(string)
Component.BroadcastMessage(string, SendMessageOptions)
Component.transform
Component.gameObject
Component.tag
Object.GetInstanceID()
Object.GetHashCode()
Object.Equals(object)
Object.Instantiate(Object, Vector3, Quaternion)
Object.Instantiate(Object, Vector3, Quaternion, Transform)
Object.Instantiate(Object)
Object.Instantiate(Object, Transform)
Object.Instantiate(Object, Transform, bool)
Object.Instantiate<T>(T)
Object.Instantiate<T>(T, Vector3, Quaternion)
Object.Instantiate<T>(T, Vector3, Quaternion, Transform)
Object.Instantiate<T>(T, Transform)
Object.Instantiate<T>(T, Transform, bool)
Object.Destroy(Object, float)
Object.Destroy(Object)
Object.DestroyImmediate(Object, bool)
Object.DestroyImmediate(Object)
Object.FindObjectsOfType(Type)
Object.FindObjectsOfType(Type, bool)
Object.DontDestroyOnLoad(Object)
Object.DestroyObject(Object, float)
Object.DestroyObject(Object)
Object.FindSceneObjectsOfType(Type)
Object.FindObjectsOfTypeIncludingAssets(Type)
Object.FindObjectsOfType<T>()
Object.FindObjectsOfType<T>(bool)
Object.FindObjectOfType<T>()
Object.FindObjectOfType<T>(bool)
Object.FindObjectsOfTypeAll(Type)
Object.FindObjectOfType(Type)
Object.FindObjectOfType(Type, bool)
Object.ToString()
Object.name
Object.hideFlags
object.Equals(object, object)
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
public class ReplayParticleSystem : ReplayRecordableBehaviour, IReplaySerialize
```

Fields

observedParticleSystem

The Unity particle system that will be recorded and also used for playback.

Declaration

```
public ParticleSystem observedParticleSystem
```

Field Value

TYPE	DESCRIPTION
ParticleSystem	

updateFlags

The ReplayParticleSystem.ReplayParticleSystemFlags to specify which features are enabled.

Declaration

```
[HideInInspector]
public ReplayParticleSystem.ReplayParticleSystemFlags updateFlags
```

Field Value

TYPE	DESCRIPTION
ReplayParticleSystem.ReplayParticleSystemFlags	

Methods

OnReplayDeserialize(ReplayState)

Called by the replay system when the component should deserialize previously recorded data.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The state object to read from

Overrides

ReplayRecordableBehaviour.OnReplayDeserialize(ReplayState)

OnReplayReset()

Called by the replay system when persistent data should be reset.

Declaration

```
protected override void OnReplayReset()
```

Overrides

ReplayBehaviour.OnReplayReset()

OnReplaySerialize(ReplayState)

Called by the replay system when the component should serialize its recorded data.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The state object to write to

Overrides

ReplayRecordableBehaviour.OnReplaySerialize(ReplayState)

OnReplayUpdate(float)

Called by the replay system during playback mode.

Declaration

```
protected override void OnReplayUpdate(float t)
```

Parameters

TYPE	NAME	DESCRIPTION
float	t	

Overrides

ReplayBehaviour.OnReplayUpdate(float)

Reset()

Called by Unity editor.

Declaration

```
protected override void Reset()
```

Overrides

ReplayBehaviour.Reset()

Start()

Called by Unity.

Declaration

```
public void Start()
```

Implements

IReplaySerialize

Enum ReplayParticleSystem.ReplayParticleSystemFlags

Replay flags used to determine which component features are enabled.

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
[Flags]
public enum ReplayParticleSystem.ReplayParticleSystemFlags
```

Fields

NAME	DESCRIPTION
Interpolate	Interpolate the supported particle system values such as time offset for smoother playback.
None	No features are enabled.

Class ReplayParticleSystemV2

Inheritance

object
Object
Component
Behaviour
MonoBehaviour
ReplayBehaviour
ReplayRecordableBehaviour
ReplayParticleSystemV2

Implements

IReplaySerialize

Inherited Members

ReplayRecordableBehaviour.Formatter
ReplayRecordableBehaviour.OnDestroy()
ReplayBehaviour.ReplayIdentity
ReplayBehaviour.ReplayObject
ReplayBehaviour.HasPersistentData
ReplayBehaviour.ReplayPersistentData
ReplayBehaviour.Variables
ReplayBehaviour.HasVariables
ReplayBehaviour.IsRecording
ReplayBehaviour.IsRecordingPaused
ReplayBehaviour.IsRecordingOrPaused
ReplayBehaviour.IsReplaying
ReplayBehaviour.IsPlaybackPaused
ReplayBehaviour.IsReplayingOrPaused
ReplayBehaviour.PlaybackTime
ReplayBehaviour.PlaybackTimeNormalized
ReplayBehaviour.PlaybackTimeScale
ReplayBehaviour.PlaybackDirection
ReplayBehaviour.Reset()
ReplayBehaviour.Awake()
ReplayBehaviour.OnEnable()
ReplayBehaviour.OnDisable()
ReplayBehaviour.OnReplayPlayPause(bool)
ReplayBehaviour.OnReplayReset()
ReplayBehaviour.OnReplayCapture()
ReplayBehaviour.OnReplayUpdate(float)
ReplayBehaviour.OnReplayEvent(ushort, ReplayState)
ReplayBehaviour.OnReplaySpawned(Vector3, Quaternion)
ReplayBehaviour.ForceRegenerateIdentity()
ReplayBehaviour.RecordVariable(ReplayVariable)
ReplayBehaviour.RecordEvent(ushort, ReplayState)
ReplayBehaviour.RecordMethodCall(Action)
ReplayBehaviour.RecordMethodCall<T>(Action<T>, T)
ReplayBehaviour.RecordMethodCall<T0, T1>(Action<T0, T1>, T0, T1)
ReplayBehaviour.RecordMethodCall<T0, T1, T2>(Action<T0, T1, T2>, T0, T1, T2)
ReplayBehaviour.RecordMethodCall<T0, T1, T2, T3>(Action<T0, T1, T2, T3>, T0, T1, T2, T3)
MonoBehaviour.IsInvoking()

MonoBehaviour.CancelInvoke()
MonoBehaviour.Invoke(string, float)
MonoBehaviour.InvokeRepeating(string, float, float)
MonoBehaviour.CancelInvoke(string)
MonoBehaviour.IsInvoking(string)
MonoBehaviour.StartCoroutine(string)
MonoBehaviour.StartCoroutine(string, object)
MonoBehaviour.StartCoroutine(IEnumerator)
MonoBehaviour.StartCoroutine_Auto(IEnumerator)
MonoBehaviour.StopCoroutine(IEnumerator)
MonoBehaviour.StopCoroutine(Coroutine)
MonoBehaviour.StopCoroutine(string)
MonoBehaviour.StopAllCoroutines()
MonoBehaviour.print(object)
MonoBehaviour.useGUILayout
MonoBehaviour.runInEditMode
Behaviour.enabled
Behaviour.isActiveAndEnabled
Component.GetComponent(Type)
Component.GetComponent<T>()
Component.TryGetComponent(Type, out Component)
Component.TryGetComponent<T>(out T)
Component.GetComponent(string)
Component.GetComponentInChildren(Type, bool)
Component.GetComponentInChildren(Type)
Component.GetComponentInChildren<T>(bool)
Component.GetComponentInChildren<T>()
Component.GetComponentsInChildren(Type, bool)
Component.GetComponentsInChildren(Type)
Component.GetComponentsInChildren<T>(bool)
Component.GetComponentsInChildren<T>(bool, List<T>)
Component.GetComponentsInChildren<T>()
Component.GetComponentsInChildren<T>(List<T>)
Component.GetComponentInParent(Type, bool)
Component.GetComponentInParent(Type)
Component.GetComponentInParent<T>(bool)
Component.GetComponentInParent<T>()
Component.GetComponentsInParent(Type, bool)
Component.GetComponentsInParent(Type)
Component.GetComponentsInParent<T>(bool)
Component.GetComponentsInParent<T>(bool, List<T>)
Component.GetComponentsInParent<T>()
Component.GetComponents(Type)
Component.GetComponents(Type, List<Component>)
Component.GetComponents<T>(List<T>)
Component.GetComponents<T>()
Component.CompareTag(string)
Component.SendMessageUpwards(string, object, SendMessageOptions)
Component.SendMessageUpwards(string, object)
Component.SendMessageUpwards(string)
Component.SendMessageUpwards(string, SendMessageOptions)
Component.SendMessage(string, object)

Component.SendMessage(string)
Component.SendMessage(string, object, SendMessageOptions)
Component.SendMessage(string, SendMessageOptions)
Component.BroadcastMessage(string, object, SendMessageOptions)
Component.BroadcastMessage(string, object)
Component.BroadcastMessage(string)
Component.BroadcastMessage(string, SendMessageOptions)
Component.transform
Component.gameObject
Component.tag
Object.GetInstanceID()
Object.GetHashCode()
Object.Equals(object)
Object.Instantiate(Object, Vector3, Quaternion)
Object.Instantiate(Object, Vector3, Quaternion, Transform)
Object.Instantiate(Object)
Object.Instantiate(Object, Transform)
Object.Instantiate(Object, Transform, bool)
Object.Instantiate<T>(T)
Object.Instantiate<T>(T, Vector3, Quaternion)
Object.Instantiate<T>(T, Vector3, Quaternion, Transform)
Object.Instantiate<T>(T, Transform)
Object.Instantiate<T>(T, Transform, bool)
Object.Destroy(Object, float)
Object.Destroy(Object)
Object.DestroyImmediate(Object, bool)
Object.DestroyImmediate(Object)
Object.FindObjectsOfType(Type)
Object.FindObjectsOfType(Type, bool)
Object.DontDestroyOnLoad(Object)
Object.DestroyObject(Object, float)
Object.DestroyObject(Object)
Object.FindSceneObjectsOfType(Type)
Object.FindObjectsOfTypeIncludingAssets(Type)
Object.FindObjectsOfType<T>()
Object.FindObjectsOfType<T>(bool)
Object.FindObjectOfType<T>()
Object.FindObjectOfType<T>(bool)
Object.FindObjectsOfTypeAll(Type)
Object.FindObjectOfType(Type)
Object.FindObjectOfType(Type, bool)
Object.ToString()
Object.name
Object.hideFlags
object.Equals(object, object)
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
public class ReplayParticleSystemV2 : ReplayRecordableBehaviour, IReplaySerialize
```

Fields

observedParticleSystem

Declaration

```
public ParticleSystem observedParticleSystem
```

Field Value

TYPE	DESCRIPTION
ParticleSystem	

Methods

OnReplayDeserialize(ReplayState)

Called by the replay system when the recorder component should deserialize any necessary data during playback.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState containing the recorded data

Overrides

ReplayRecordableBehaviour.OnReplayDeserialize(ReplayState)

OnReplayEnd()

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'

Declaration

```
protected override void OnReplayEnd()
```

Overrides

ReplayBehaviour.OnReplayEnd()

OnReplaySerialize(ReplayState)

Called by the replay system when the recorder component should serialize and necessary data during recording.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState used to store the serialized data

Overrides

ReplayRecordableBehaviour.OnReplaySerialize(ReplayState)

OnReplayStart()

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.

Declaration

```
protected override void OnReplayStart()
```

Overrides

ReplayBehaviour.OnReplayStart()

Start()

Declaration

```
public void Start()
```

Implements

IReplaySerialize

Class ReplayPlaybackOperation

Represents a dedicated playback operation in progress. Provides access to all playback replated API's for a specific playback operation.

Inheritance

object
ReplayOperation
ReplayPlaybackOperation

Implements

IDisposable

Inherited Members

ReplayOperation.Scene
ReplayOperation.Storage
ReplayOperation.ReplayTickUpdate(float)
ReplayOperation.ReplayTickLateUpdate(float)
ReplayOperation.ReplayTickFixedUpdate(float)
object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
public sealed class ReplayPlaybackOperation : ReplayOperation, IDisposable
```

Fields

OnPlaybackEnd

Declaration

```
public UnityEvent OnPlaybackEnd
```

Field Value

TYPE	DESCRIPTION
UnityEvent	

OnPlaybackLooped

Declaration

```
public UnityEvent OnPlaybackLooped
```

Field Value

TYPE	DESCRIPTION
UnityEvent	

defaultPlaybackRate

The default playback fps rate.

Declaration

```
public const float defaultPlaybackRate = 60
```

Field Value

TYPE	DESCRIPTION
float	

Properties

Duration

Get the duration of the replay.

Declaration

```
public float Duration { get; }
```

Property Value

TYPE	DESCRIPTION
float	

EndBehaviour

Get the current playback end behaviour which determines what will happen when the replay reaches the end. By default, playback will end and the associated replay scene will switch back to live mode so that gameplay can resume.

Declaration

```
public PlaybackEndBehaviour EndBehaviour { get; }
```

Property Value

TYPE	DESCRIPTION
PlaybackEndBehaviour	

IsDisposed

Check if this playback operation has been disposed. A playback operation becomes disposed when playback has been stopped, at which point the API becomes unusable.

Declaration

```
public override bool IsDisposed { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

Overrides

ReplayOperation.IsDisposed

IsPlaybackPaused

Returns a value indicating whether the playback is currently paused.

Declaration

```
public bool IsPlaybackPaused { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

IsReplaying

Returns a value indicating whether playback is in progress and the playback is not currently paused.

Declaration

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

Property Value

TYPE	DESCRIPTION
bool	

IsReplayingOrPaused

Returns a value indicating whether playback is in progress or if the playback is currently paused.

Declaration

```
public bool IsReplayingOrPaused { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

Options

Get the ReplayPlaybackOptions for this replay operation.

Declaration

```
public ReplayPlaybackOptions Options { get; }
```

Property Value

TYPE	DESCRIPTION
ReplayPlaybackOptions	

PlaybackDirection

The current playback direction. Use Backward to replay in reverse.

Declaration

```
public PlaybackDirection PlaybackDirection { get; set; }
```

Property Value

TYPE	DESCRIPTION
PlaybackDirection	

PlaybackRate

The target number of playback frames that will be simulated per second. Higher rates will allow for smooth and more accurate playback, but may have an additional performance hit. The replay system will not be able to playback faster than your current frame rate so there is no benefit in setting a value of '90' for example if you game will only run at 60 fps. Set this value to negative and the playback operation will run as fast as possible. Default value is 60fps.

Declaration

```
public float PlaybackRate { get; }
```

Property Value

TYPE	DESCRIPTION
float	

PlaybackTime

Get the current playback time of this operation in seconds. Playback time will always be between 0 and Duration. To change the current playback time use SeekPlayback(float, PlaybackOrigin, bool) or SeekPlaybackNormalized(float, bool).

Declaration

```
public float PlaybackTime { get; }
```

Property Value

TYPE	DESCRIPTION
float	

PlaybackTimeNormalized

Get the current normalized playback time of this operation. The normalized time will always be between 0 and 1, where 0 represents that start of the replay, 1 represents the end of the relay, and 0.5 represents the middle of the replay. Can be used to easily seek to common offsets such as (middle) without needing to calculate the time based on Duration.

Declaration

```
public float PlaybackTimeNormalized { get; }
```

Property Value

TYPE	DESCRIPTION
float	

PlaybackTimeScale

The current playback time scale which represents the speed at which playback will occur. The playback time scale is used as a multiplier so a value of 1 represents normal speed, 2 represents twice the speed, and 0.5 represents half the speed.

Declaration

```
public float PlaybackTimeScale { get; set; }
```

Property Value

TYPE	DESCRIPTION
float	

RestoreSceneMode

The current scene restore mode which determines what will happen to the associated replay objects when playback ends. KeepState means that replay objects will maintain their current state when the replay ends, meaning that gameplay can continue from the current playback positions. RestoreState means that the replay system will restore all replay objects to their original state immediately before playback began.

Declaration

```
public RestoreSceneMode RestoreSceneMode { get; set; }
```

Property Value

TYPE	DESCRIPTION
RestoreSceneMode	

SeekSnap

The current playback seek snap setting. Seek snap determines how seeking behavious in relation to the snapshots that are available from the recording. SnapToFrame means that the replay system will clamp to the nearest snapshot. This gives a snappy effect while drag seeking as the replay jumps to the nearest recorded snapshot. Smooth means that the replay system may interpolate between multiple snapshots if the time values falls between 2 snapshots. This gives a smooth replay while drag seeking.

Declaration

```
public PlaybackSeekSnap SeekSnap { get; set; }
```

Property Value

TYPE	DESCRIPTION
PlaybackSeekSnap	

UpdateMode

Get the ReplayUpdateMode for this replay operation. This value determines at what stage in the Unity game loop the playback operation is updated.

Declaration

```
public override ReplayUpdateMode UpdateMode { get; }
```

Property Value

TYPE	DESCRIPTION
ReplayUpdateMode	

Overrides

ReplayOperation.UpdateMode

Methods

CheckDisposed()

Throw an exception if this playback operation has been disposed.

Declaration

```
protected override void CheckDisposed()
```

Overrides

ReplayOperation.CheckDisposed()

Exceptions

TYPE	CONDITION
ObjectDisposedException	The replay operation was disposed

Dispose()

Dispose this replay operation. This will cause the playback to be stopped and this playback operation should no longer be used.

Declaration

```
public override void Dispose()
```

Overrides

ReplayOperation.Dispose()

PausePlayback()

Pause the current playback operation. Playback will not be updated but all associated replay objects will remain in playback mode.

Declaration

```
public void PausePlayback()
```

ReplayTick(float)

The main update call for this replay operation. Can be called manually if required, but if manually update is required then it is recommended to use ReplayTick(float, ReplayUpdateMode).

Declaration

```
public override void ReplayTick(float delta)
```

Parameters

TYPE	NAME	DESCRIPTION

TYPE	NAME	DESCRIPTION
float	delta	The amount of time in seconds that has passed since the last update

Overrides

ReplayOperation.ReplayTick(float)

ResumePlayback()

Resume the current playback operation. The replay will carry on from the point at which it was paused.

Declaration

```
public void ResumePlayback()
```

SeekPlayback(float, PlaybackOrigin, bool)

Jump to a new time stamp in the replay and update all replaying objects. The time stamp is specified in seconds and should usually be between 0 - Duration, although negative values are allowed when using relative seeking from the current time stamp. You can specify a relative time offset if you wanted to seek + or - 5 seconds for example using the PlaybackOrigin enum to specify the seek mode. Take a look at SeekPlaybackNormalized(float, bool) if you want to seek using a normalized value between 0-1. Seeking will be performed smoothly by default meaning that interpolation may occur between 2 snapshots since the input time stamp is unlikely to exactly match any given snapshot time stamp. This behaviour can be disabled if required so that seeking will snap to the nearest snapshot using SeekSnap. Seeking can mean that the replay will jump over many snapshots meaning that replay events and method may go uncalled during the seeking process which may or may not be desirable. You can force the replay system to trigger any such events or method calls that may have been missed using the `simulateMissedFrames` parameter. Note though that enabling this option can be extremely performance intensive so is only recommended for smaller replays with few replay objects.

Declaration

```
public void SeekPlayback(float time, PlaybackOrigin origin = PlaybackOrigin.Start, bool simulateMissedFrames =
false)
```

Parameters

TYPE	NAME	DESCRIPTION
float	time	The time in seconds to seek to, or to use as an offset depending upon the `origin` value>
PlaybackOrigin	origin	The origin where the seek should start from. Use Current if you want to seek + or - a few seconds for example
bool	simulateMissedFrames	Should the missed frames between seek positions be simulated. NOT RECOMMENDED FOR LARGER REPLAYS

SeekPlaybackNormalized(float, bool)

Declaration

```
public void SeekPlaybackNormalized(float timeNormalized, bool simulateMissedFrames = false)
```

Parameters

TYPE	NAME	DESCRIPTION
float	timeNormalized	
bool	simulateMissedFrames	

StopPlayback()

Stop this playback operation. Playback will stop and this operation will be disposed so should no longed be used after this call.

Declaration

```
public void StopPlayback()
```

StopPlaybackDelayed(float)

Stop this playback operation after the specified amount of second has passed. Playback will stop after the specified time and this operation will be disposed so should no longed be used after this call.

Declaration

```
public void StopPlaybackDelayed(float delay)
```

Parameters

TYPE	NAME	DESCRIPTION
float	delay	The amount of time in seconds to wait until the record operation is stopped

Implements

IDisposable

Class ReplayPlaybackOptions

A number of options used to control the playback behaviour.

Inheritance

object

ReplayPlaybackOptions

Implements

ISerializationCallbackReceiver

Inherited Members

object.Equals(object)

object.Equals(object, object)

object.GetHashCode()

object.GetType()

object.MemberwiseClone()

object.ReferenceEquals(object, object)

object.ToString()

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
[Serializable]
public class ReplayPlaybackOptions : ISerializationCallbackReceiver
```

Constructors

ReplayPlaybackOptions()

Create a new playback options instance with default settings.

Declaration

```
public ReplayPlaybackOptions()
```

ReplayPlaybackOptions(PlaybackEndBehaviour, int)

Create a new playback options instance with the specified end behaviour and frame rate.

Declaration

```
public ReplayPlaybackOptions(PlaybackEndBehaviour endBehaviour, int playbackFPS = -1)
```

Parameters

TYPE	NAME	DESCRIPTION
PlaybackEndBehaviour	endBehaviour	The end behaviour which indicates what should happen when the end of the replay is reached
int	playbackFPS	The target playback frate rate or '-1' for unlimited frame rate

Properties

IsPlaybackFPSUnlimited

Returns a value indicating whether the playback fps is unlimited. Ie: set to '-1'.

Declaration

```
public bool IsPlaybackFPSUnlimited { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

PlaybackEndBehaviour

When should happen when the replay reaches the end of its playback.

Declaration

```
public PlaybackEndBehaviour PlaybackEndBehaviour { get; set; }
```

Property Value

TYPE	DESCRIPTION
PlaybackEndBehaviour	

PlaybackFPS

The target playback frame rate. Use '-1' to set the playback fps to unlimited which will update every game tick. Playback updates can run more frequently than the record rate but interpolation can blend key frames to create smooth replays.

Declaration

```
public float PlaybackFPS { get; set; }
```

Property Value

TYPE	DESCRIPTION
float	

PlaybackUpdateMode

The update method used to update the playback operation. Used for compatibility with other systems that update objects in other update methods such as LateUpdate.

Declaration

```
public ReplayUpdateMode PlaybackUpdateMode { get; set; }
```

Property Value

TYPE	DESCRIPTION
ReplayUpdateMode	

Implements

UnityEngine.ISerializationCallbackReceiver

Class ReplayPreparerIgnoreAttribute

Inheritance

object
Attribute
ReplayPreparerIgnoreAttribute

Implements

_Attribute

Inherited Members

Attribute.Equals(object)
Attribute.GetCustomAttribute(Assembly, Type)
Attribute.GetCustomAttribute(Assembly, Type, bool)
Attribute.GetCustomAttribute(MemberInfo, Type)
Attribute.GetCustomAttribute(MemberInfo, Type, bool)
Attribute.GetCustomAttribute(Module, Type)
Attribute.GetCustomAttribute(Module, Type, bool)
Attribute.GetCustomAttribute(ParameterInfo, Type)
Attribute.GetCustomAttribute(ParameterInfo, Type, bool)
Attribute.GetCustomAttributes(Assembly)
Attribute.GetCustomAttributes(Assembly, bool)
Attribute.GetCustomAttributes(Assembly, Type)
Attribute.GetCustomAttributes(Assembly, Type, bool)
Attribute.GetCustomAttributes(MemberInfo)
Attribute.GetCustomAttributes(MemberInfo, bool)
Attribute.GetCustomAttributes(MemberInfo, Type)
Attribute.GetCustomAttributes(MemberInfo, Type, bool)
Attribute.GetCustomAttributes(Module)
Attribute.GetCustomAttributes(Module, bool)
Attribute.GetCustomAttributes(Module, Type)
Attribute.GetCustomAttributes(Module, Type, bool)
Attribute.GetCustomAttributes(ParameterInfo)
Attribute.GetCustomAttributes(ParameterInfo, bool)
Attribute.GetCustomAttributes(ParameterInfo, Type)
Attribute.GetCustomAttributes(ParameterInfo, Type, bool)
Attribute.GetHashCode()
Attribute.IsDefaultAttribute()
Attribute.IsDefined(Assembly, Type)
Attribute.IsDefined(Assembly, Type, bool)
Attribute.IsDefined(MemberInfo, Type)
Attribute.IsDefined(MemberInfo, Type, bool)
Attribute.IsDefined(Module, Type)
Attribute.IsDefined(Module, Type, bool)
Attribute.IsDefined(ParameterInfo, Type)
Attribute.IsDefined(ParameterInfo, Type, bool)
Attribute.Match(object)
Attribute.TypeId
object.Equals(object, object)
object.GetType()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
[AttributeUsage(AttributeTargets.Class)]
public sealed class ReplayPreparerIgnoreAttribute : Attribute, _Attribute
```

Implements

_Attribute

Class ReplayRecordOperation

Represents a dedicated record operation in progress. Provides access to all recording related API's for a specific record operation.

Inheritance

object
ReplayOperation
ReplayRecordOperation

Implements

IDisposable

Inherited Members

ReplayOperation.Scene
ReplayOperation.Storage
ReplayOperation.ReplayTickUpdate(float)
ReplayOperation.ReplayTickLateUpdate(float)
ReplayOperation.ReplayTickFixedUpdate(float)
object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
public sealed class ReplayRecordOperation : ReplayOperation, IDisposable
```

Fields

defaultRecordRate

The default record fps rate.

Declaration

```
public const float defaultRecordRate = 8
```

Field Value

TYPE	DESCRIPTION
float	

Properties

IsDisposed

Check if this record operation has been disposed. A record operation becomes disposed when recording has been stopped, at which point the API becomes unusable.

Declaration

```
public override bool IsDisposed { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

Overrides

ReplayOperation.IsDisposed

IsRecording

Returns a value indicating whether recording is in progress and the recording is not currently paused.

Declaration

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

Property Value

TYPE	DESCRIPTION
bool	

IsRecordingOrPaused

Returns a value indicating whether recording is in progress or if the recording is currently paused.

Declaration

```
public bool IsRecordingOrPaused { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

IsRecordingPaused

Returns a value indicating whether the recording is currently paused.

Declaration

```
public bool IsRecordingPaused { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

Options

Get the ReplayRecordOptions for this replay operation.

Declaration

```
public ReplayRecordOptions Options { get; }
```

Property Value

TYPE	DESCRIPTION
ReplayRecordOptions	

RecordRate

The target number of snapshot frames that will be recorded per second. Higher rates will allow for smooth and more accurate replay, but may have an additional performance hit. The replay system will not be able to record faster than your current frame rate so there is no benefit in setting a value of '90' for example if you game will only run at 60 fps. Set this value to negative and the record operation will run as fast as possible. When interpolation is used, record rates of '16' or much lower can be possible depending upon the particular game, which can save on storage space and performance.

Declaration

```
public float RecordRate { get; }
```

Property Value

TYPE	DESCRIPTION
float	

UpdateMode

Get the ReplayUpdateMode for this replay operation. This value determines at what stage in the Unity game loop the record operation is updated.

Declaration

```
public override ReplayUpdateMode UpdateMode { get; }
```

Property Value

TYPE	DESCRIPTION
ReplayUpdateMode	

Overrides

ReplayOperation.UpdateMode

Methods

CheckDisposed()

Throw an exception if this record operation has been disposed.

Declaration

```
protected override void CheckDisposed()
```

Overrides

ReplayOperation.CheckDisposed()

Exceptions

TYPE	CONDITION
ObjectDisposedException	The replay operation was disposed

Dispose()

Dispose this replay operation. This will cause the recording to be stopped and this record operation should no longer be used.

Declaration

```
public override void Dispose()
```

Overrides

ReplayOperation.Dispose()

PauseRecording()

Pause the current record operation. No replay snapshots will be captured while recording is paused.

Declaration

```
public void PauseRecording()
```

ReplayTick(float)

The main update call for this replay operation. Can be called manually if required, but if manually update is required then it is recommended to use ReplayTick(float, ReplayUpdateMode).

Declaration

```
public override void ReplayTick(float delta)
```

Parameters

TYPE	NAME	DESCRIPTION
float	delta	The amount of time in seconds that has passed since the last update

Overrides

ReplayOperation.ReplayTick(float)

ResumeRecording()

Resume the current record operation. The recording will carry on from the point at which it was paused.

Declaration

```
public void ResumeRecording()
```

StopRecording()

Stop this record operation. Recording will stop and this operation will be disposed so should no longed be used after this call.

Declaration

```
public void StopRecording()
```

StopRecordingDelayed(float)

Stop this record operation after the specified amount of second has passed. Recording will stop after the specified time and this operation will be disposed so should no longed be used after this call.

Declaration

```
public void StopRecordingDelayed(float delay)
```

Parameters

TYPE	NAME	DESCRIPTION
float	delay	The amount of time in seconds to wait until the record operation is stopped

Implements

IDisposable

Class ReplayRecordOptions

A number of options that can be used to control the record behaviour.

Inheritance

object

ReplayRecordOptions

Implements

ISerializationCallbackReceiver

Inherited Members

object.Equals(object)

object.Equals(object, object)

object.GetHashCode()

object.GetType()

object.ReferenceEquals(object, object)

object.ToString()

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
[Serializable]
public sealed class ReplayRecordOptions : ISerializationCallbackReceiver
```

Fields

DefaultRecordFPS

The default fps value for record operations.

Declaration

```
public const float DefaultRecordFPS = 12
```

Field Value

TYPE	DESCRIPTION
float	

MaxRecordFPS

The maximum allowable record frame rate.

Declaration

```
public const float MaxRecordFPS = 60
```

Field Value

TYPE	DESCRIPTION
float	

MinRecordFPS

The minimum allowable record frame rate.

Declaration

```
public const float MinRecordFPS = 1
```

Field Value

TYPE	DESCRIPTION
float	

Properties

RecordFPS

The target record frame rate. Higher frame rates will result in more storage consumption but better replay accuracy.

Declaration

```
public float RecordFPS { get; set; }
```

Property Value

TYPE	DESCRIPTION
float	

RecordUpdateMode

The update method used to update the record operation. Used for compatibility with other systems that update objects in other update methods such as LateUpdate.

Declaration

```
public ReplayUpdateMode RecordUpdateMode { get; set; }
```

Property Value

TYPE	DESCRIPTION
ReplayUpdateMode	

Implements

UnityEngine.ISerializationCallbackReceiver

Class ReplayRecordableBehaviour

Derive from this base class to create custom recorder components.

Inheritance

object
Object
Component
Behaviour
MonoBehaviour
ReplayBehaviour
ReplayRecordableBehaviour
ReplayAnimator
ReplayAudio
ReplayBlendShape
ReplayComponentEnabledState
ReplayEnabledState
ReplayLineRenderer
ReplayMaterial
ReplayMaterialChange
ReplayParentChange
ReplayParticleSystem
ReplayParticleSystemV2
ReplayRiggedGeneric
ReplayRiggedHumanoid
ReplayTrailRenderer
ReplayTransform

Implements

IReplaySerialize

Inherited Members

ReplayBehaviour.ReplayIdentity
ReplayBehaviour.ReplayObject
ReplayBehaviour.HasPersistentData
ReplayBehaviour.ReplayPersistentData
ReplayBehaviour.Variables
ReplayBehaviour.HasVariables
ReplayBehaviour.IsRecording
ReplayBehaviour.IsRecordingPaused
ReplayBehaviour.IsRecordingOrPaused
ReplayBehaviour.IsReplaying
ReplayBehaviour.IsPlaybackPaused
ReplayBehaviour.IsReplayingOrPaused
ReplayBehaviour.PlaybackTime
ReplayBehaviour.PlaybackTimeNormalized
ReplayBehaviour.PlaybackTimeScale
ReplayBehaviour.PlaybackDirection
ReplayBehaviour.Reset()
ReplayBehaviour.Awake()
ReplayBehaviour.OnEnable()
ReplayBehaviour.OnDisable()
ReplayBehaviour.OnReplayStart()

ReplayBehaviour.OnReplayEnd()
ReplayBehaviour.OnReplayPlayPause(bool)
ReplayBehaviour.OnReplayReset()
ReplayBehaviour.OnReplayCapture()
ReplayBehaviour.OnReplayUpdate(float)
ReplayBehaviour.OnReplayEvent(ushort, ReplayState)
ReplayBehaviour.OnReplaySpawned(Vector3, Quaternion)
ReplayBehaviour.ForceRegenerateIdentity()
ReplayBehaviour.RecordVariable(ReplayVariable)
ReplayBehaviour.RecordEvent(ushort, ReplayState)
ReplayBehaviour.RecordMethodCall(Action)
ReplayBehaviour.RecordMethodCall<T>(Action<T>, T)
ReplayBehaviour.RecordMethodCall<T0, T1>(Action<T0, T1>, T0, T1)
ReplayBehaviour.RecordMethodCall<T0, T1, T2>(Action<T0, T1, T2>, T0, T1, T2)
ReplayBehaviour.RecordMethodCall<T0, T1, T2, T3>(Action<T0, T1, T2, T3>, T0, T1, T2, T3)
MonoBehaviour.IsInvoking()
MonoBehaviour.CancelInvoke()
MonoBehaviour.Invoke(string, float)
MonoBehaviour.InvokeRepeating(string, float, float)
MonoBehaviour.CancelInvoke(string)
MonoBehaviour.IsInvoking(string)
MonoBehaviour.StartCoroutine(string)
MonoBehaviour.StartCoroutine(string, object)
MonoBehaviour.StartCoroutine(IEnumerator)
MonoBehaviour.StartCoroutine_Auto(IEnumerator)
MonoBehaviour.StopCoroutine(IEnumerator)
MonoBehaviour.StopCoroutine(Coroutine)
MonoBehaviour.StopCoroutine(string)
MonoBehaviour.StopAllCoroutines()
MonoBehaviour.print(object)
MonoBehaviour.useGUILayout
MonoBehaviour.runInEditMode
Behaviour.enabled
Behaviour.isActiveAndEnabled
Component.GetComponent(Type)
Component.GetComponent<T>()
Component.TryGetComponent(Type, out Component)
Component.TryGetComponent<T>(out T)
Component.GetComponent(string)
Component.GetComponentInChildren(Type, bool)
Component.GetComponentInChildren(Type)
Component.GetComponentInChildren<T>(bool)
Component.GetComponentInChildren<T>()
Component.GetComponentsInChildren(Type, bool)
Component.GetComponentsInChildren(Type)
Component.GetComponentsInChildren<T>(bool)
Component.GetComponentsInChildren<T>(bool, List<T>)
Component.GetComponentsInChildren<T>()
Component.GetComponentsInChildren<T>(List<T>)
Component.GetComponentInParent(Type, bool)
Component.GetComponentInParent(Type)
Component.GetComponentInParent<T>(bool)

Component.GetComponentInParent<T>()
Component.GetComponentsInParent(Type, bool)
Component.GetComponentsInParent(Type)
Component.GetComponentsInParent<T>(bool)
Component.GetComponentsInParent<T>(bool, List<T>)
Component.GetComponentsInParent<T>()
Component.GetComponents(Type)
Component.GetComponents(Type, List<Component>)
Component.GetComponents<T>(List<T>)
Component.GetComponents<T>()
Component.CompareTag(string)
Component.SendMessageUpwards(string, object, SendMessageOptions)
Component.SendMessageUpwards(string, object)
Component.SendMessageUpwards(string)
Component.SendMessageUpwards(string, SendMessageOptions)
Component.SendMessage(string, object)
Component.SendMessage(string)
Component.SendMessage(string, object, SendMessageOptions)
Component.SendMessage(string, SendMessageOptions)
Component.BroadcastMessage(string, object, SendMessageOptions)
Component.BroadcastMessage(string, object)
Component.BroadcastMessage(string)
Component.BroadcastMessage(string, SendMessageOptions)
Component.transform
Component.gameObject
Component.tag
Object.GetInstanceID()
Object.GetHashCode()
Object.Equals(object)
Object.Instantiate(Object, Vector3, Quaternion)
Object.Instantiate(Object, Vector3, Quaternion, Transform)
Object.Instantiate(Object)
Object.Instantiate(Object, Transform)
Object.Instantiate(Object, Transform, bool)
Object.Instantiate<T>(T)
Object.Instantiate<T>(T, Vector3, Quaternion)
Object.Instantiate<T>(T, Vector3, Quaternion, Transform)
Object.Instantiate<T>(T, Transform)
Object.Instantiate<T>(T, Transform, bool)
Object.Destroy(Object, float)
Object.Destroy(Object)
Object.DestroyImmediate(Object, bool)
Object.DestroyImmediate(Object)
Object.FindObjectsOfType(Type)
Object.FindObjectsOfType(Type, bool)
Object.DontDestroyOnLoad(Object)
Object.DestroyObject(Object, float)
Object.DestroyObject(Object)
Object.FindSceneObjectsOfType(Type)
Object.FindObjectsOfTypeIncludingAssets(Type)
Object.FindObjectsOfType<T>()
Object.FindObjectsOfType<T>(bool)

Object.FindObjectOfType<T>()
Object.FindObjectOfType<T>(bool)
Object.FindObjectsOfTypeAll(Type)
Object.FindObjectOfType(Type)
Object.FindObjectOfType(Type, bool)
Object.ToString()
Object.name
Object.hideFlags
object.Equals(object, object)
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
public abstract class ReplayRecordableBehaviour : ReplayBehaviour, IReplaySerialize
```

Properties

Formatter

An optional ReplayFormatter that is used to serialize a particular component. Providing a formatter via his property can greatly reduce the amount of data that a ReplayObject needs to store.

Declaration

```
public virtual ReplayFormatter Formatter { get; }
```

Property Value

TYPE	DESCRIPTION
ReplayFormatter	

Methods

OnDestroy()

Called by Unity.

Declaration

```
protected override void OnDestroy()
```

Overrides

ReplayBehaviour.OnDestroy()

OnReplayDeserialize(ReplayState)

Called by the replay system when the recorder component should deserialize any necessary data during playback.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState containing the recorded data

OnReplaySerialize(ReplayState)

Called by the replay system when the recorder component should serialize and necessary data during recording.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState used to store the serialized data

Implements

IReplaySerialize

Class ReplayRiggedGeneric

Inheritance

object

Object

Component

Behaviour

MonoBehaviour

ReplayBehaviour

ReplayRecordableBehaviour

ReplayRiggedGeneric

Implements

IReplaySerialize

Inherited Members

ReplayBehaviour.ReplayIdentity

ReplayBehaviour.ReplayObject

ReplayBehaviour.HasPersistentData

ReplayBehaviour.ReplayPersistentData

ReplayBehaviour.Variables

ReplayBehaviour.HasVariables

ReplayBehaviour.IsRecording

ReplayBehaviour.IsRecordingPaused

ReplayBehaviour.IsRecordingOrPaused

ReplayBehaviour.IsReplaying

ReplayBehaviour.IsPlaybackPaused

ReplayBehaviour.IsReplayingOrPaused

ReplayBehaviour.PlaybackTime

ReplayBehaviour.PlaybackTimeNormalized

ReplayBehaviour.PlaybackTimeScale

ReplayBehaviour.PlaybackDirection

ReplayBehaviour.ForceRegenerateIdentity()

ReplayBehaviour.RecordVariable(ReplayVariable)

ReplayBehaviour.RecordEvent(ushort, ReplayState)

ReplayBehaviour.RecordMethodCall(Action)

ReplayBehaviour.RecordMethodCall<T>(Action<T>, T)

ReplayBehaviour.RecordMethodCall<T0, T1>(Action<T0, T1>, T0, T1)

ReplayBehaviour.RecordMethodCall<T0, T1, T2>(Action<T0, T1, T2>, T0, T1, T2)

ReplayBehaviour.RecordMethodCall<T0, T1, T2, T3>(Action<T0, T1, T2, T3>, T0, T1, T2, T3)

MonoBehaviour.IsInvoking()

MonoBehaviour.CancelInvoke()

MonoBehaviour.Invoke(string, float)

MonoBehaviour.InvokeRepeating(string, float, float)

MonoBehaviour.CancelInvoke(string)

MonoBehaviour.IsInvoking(string)

MonoBehaviour.StartCoroutine(string)

MonoBehaviour.StartCoroutine(string, object)

MonoBehaviour.StartCoroutine(IEnumerator)

MonoBehaviour.StartCoroutine_Auto(IEnumerator)

MonoBehaviour.StopCoroutine(IEnumerator)

MonoBehaviour.StopCoroutine(Coroutine)

MonoBehaviour.StopCoroutine(string)

MonoBehaviour.StopAllCoroutines()
MonoBehaviour.print(object)
MonoBehaviour.useGUILayout
MonoBehaviour.runInEditMode
Behaviour.enabled
Behaviour.isActiveAndEnabled
Component.GetComponent(Type)
Component.GetComponent<T>()
Component.TryGetComponent(Type, out Component)
Component.TryGetComponent<T>(out T)
Component.GetComponent(string)
Component.GetComponentInChildren(Type, bool)
Component.GetComponentInChildren(Type)
Component.GetComponentInChildren<T>(bool)
Component.GetComponentInChildren<T>()
Component.GetComponentsInChildren(Type, bool)
Component.GetComponentsInChildren(Type)
Component.GetComponentsInChildren<T>(bool)
Component.GetComponentsInChildren<T>(bool, List<T>)
Component.GetComponentsInChildren<T>()
Component.GetComponentsInChildren<T>(List<T>)
Component.GetComponentInParent(Type, bool)
Component.GetComponentInParent(Type)
Component.GetComponentInParent<T>(bool)
Component.GetComponentInParent<T>()
Component.GetComponentsInParent(Type, bool)
Component.GetComponentsInParent(Type)
Component.GetComponentsInParent<T>(bool)
Component.GetComponentsInParent<T>(bool, List<T>)
Component.GetComponentsInParent<T>()
Component.GetComponents(Type)
Component.GetComponents(Type, List<Component>)
Component.GetComponents<T>(List<T>)
Component.GetComponents<T>()
Component.CompareTag(string)
Component.SendMessageUpwards(string, object, SendMessageOptions)
Component.SendMessageUpwards(string, object)
Component.SendMessageUpwards(string)
Component.SendMessageUpwards(string, SendMessageOptions)
Component.SendMessage(string, object)
Component.SendMessage(string)
Component.SendMessage(string, object, SendMessageOptions)
Component.SendMessage(string, SendMessageOptions)
Component.BroadcastMessage(string, object, SendMessageOptions)
Component.BroadcastMessage(string, object)
Component.BroadcastMessage(string)
Component.BroadcastMessage(string, SendMessageOptions)
Component.transform
Component.gameObject
Component.tag
Object.GetInstanceID()
Object.GetHashCode()

Object.Equals(object)
Object.Instantiate(Object, Vector3, Quaternion)
Object.Instantiate(Object, Vector3, Quaternion, Transform)
Object.Instantiate(Object)
Object.Instantiate(Object, Transform)
Object.Instantiate(Object, Transform, bool)
Object.Instantiate<T>(T)
Object.Instantiate<T>(T, Vector3, Quaternion)
Object.Instantiate<T>(T, Vector3, Quaternion, Transform)
Object.Instantiate<T>(T, Transform)
Object.Instantiate<T>(T, Transform, bool)
Object.Destroy(Object, float)
Object.Destroy(Object)
Object.DestroyImmediate(Object, bool)
Object.DestroyImmediate(Object)
Object.FindObjectsOfType(Type)
Object.FindObjectsOfType(Type, bool)
Object.DontDestroyOnLoad(Object)
Object.DestroyObject(Object, float)
Object.DestroyObject(Object)
Object.FindSceneObjectsOfType(Type)
Object.FindObjectsOfTypeIncludingAssets(Type)
Object.FindObjectsOfType<T>()
Object.FindObjectsOfType<T>(bool)
Object.FindObjectOfType<T>()
Object.FindObjectOfType<T>(bool)
Object.FindObjectsOfTypeAll(Type)
Object.FindObjectOfType(Type)
Object.FindObjectOfType(Type, bool)
Object.ToString()
Object.name
Object.hideFlags
object.Equals(object, object)
object.GetType()
object.ReferenceEquals(object, object)

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
[DisallowMultipleComponent]
public sealed class ReplayRiggedGeneric : ReplayRecordableBehaviour, IReplaySerialize
```

Fields

observedBones

Declaration

```
public Transform[] observedBones
```

Field Value

TYPE	DESCRIPTION
Transform[]	

observedRootBone

Declaration

```
public Transform observedRootBone
```

Field Value

TYPE	DESCRIPTION
Transform	

Properties

BonePositionPrecision

Declaration

```
public RecordPrecision BonePositionPrecision { get; set; }
```

Property Value

TYPE	DESCRIPTION
RecordPrecision	

BoneRotationPrecision

Declaration

```
public RecordPrecision BoneRotationPrecision { get; set; }
```

Property Value

TYPE	DESCRIPTION
RecordPrecision	

BoneScalePrecision

Declaration

```
public RecordPrecision BoneScalePrecision { get; set; }
```

Property Value

TYPE	DESCRIPTION
RecordPrecision	

Formatter

An optional ReplayFormatter that is used to serialize a particular component. Providing a formatter via his property can greatly reduce the amount of data that a ReplayObject needs to store.

Declaration

```
public override ReplayFormatter Formatter { get; }
```

Property Value

TYPE	DESCRIPTION
ReplayFormatter	

Overrides

ReplayRecordableBehaviour.Formatter

ReplayBonePosition

Declaration

```
public RecordAxisFlags ReplayBonePosition { get; set; }
```

Property Value

TYPE	DESCRIPTION
RecordAxisFlags	

ReplayBoneRotation

Declaration

```
public RecordAxisFlags ReplayBoneRotation { get; set; }
```

Property Value

TYPE	DESCRIPTION
RecordAxisFlags	

ReplayBoneScale

Declaration

```
public RecordAxisFlags ReplayBoneScale { get; set; }
```

Property Value

TYPE	DESCRIPTION
RecordAxisFlags	

Methods

AutoDetectRigBones()

Declaration

```
public void AutoDetectRigBones()
```

Awake()

Called by Unity.

Declaration

```
protected override void Awake()
```

Overrides

ReplayBehaviour.Awake()

OnReplayDeserialize(ReplayState)

Called by the replay system when the recorder component should deserialize any necessary data during playback.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState containing the recorded data

Overrides

ReplayRecordableBehaviour.OnReplayDeserialize(ReplayState)

OnReplayReset()

Called by the replay system during playback when cached values should be reset to safe default to avoid glitches or inaccuracies in the playback.

Declaration

```
protected override void OnReplayReset()
```

Overrides

ReplayBehaviour.OnReplayReset()

OnReplaySerialize(ReplayState)

Called by the replay system when the recorder component should serialize and necessary data during recording.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState used to store the serialized data

Overrides

ReplayRecordableBehaviour.OnReplaySerialize(ReplayState)

OnReplayStart()

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.

Declaration

```
protected override void OnReplayStart()
```

Overrides

ReplayBehaviour.OnReplayStart()

OnReplayUpdate(float)

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

Declaration

```
protected override void OnReplayUpdate(float t)
```

Parameters

TYPE	NAME	DESCRIPTION
float	t	A normalized value representing the progress between replay snapshots. Use this value for interpolation or similar smoothing passes

Overrides

ReplayBehaviour.OnReplayUpdate(float)

Reset()

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

Declaration

```
protected override void Reset()
```

Overrides

ReplayBehaviour.Reset()

Implements

IReplaySerialize

Class ReplayRiggedHumanoid

Inheritance

object
Object
Component
Behaviour
MonoBehaviour
ReplayBehaviour
ReplayRecordableBehaviour
ReplayRiggedHumanoid

Implements

IReplaySerialize

Inherited Members

ReplayBehaviour.ReplayIdentity
ReplayBehaviour.ReplayObject
ReplayBehaviour.HasPersistentData
ReplayBehaviour.ReplayPersistentData
ReplayBehaviour.Variables
ReplayBehaviour.HasVariables
ReplayBehaviour.IsRecording
ReplayBehaviour.IsRecordingPaused
ReplayBehaviour.IsRecordingOrPaused
ReplayBehaviour.IsReplaying
ReplayBehaviour.IsPlaybackPaused
ReplayBehaviour.IsReplayingOrPaused
ReplayBehaviour.PlaybackTime
ReplayBehaviour.PlaybackTimeNormalized
ReplayBehaviour.PlaybackTimeScale
ReplayBehaviour.PlaybackDirection
ReplayBehaviour.ForceRegenerateIdentity()
ReplayBehaviour.RecordVariable(ReplayVariable)
ReplayBehaviour.RecordEvent(ushort, ReplayState)
ReplayBehaviour.RecordMethodCall(Action)
ReplayBehaviour.RecordMethodCall<T>(Action<T>, T)
ReplayBehaviour.RecordMethodCall<T0, T1>(Action<T0, T1>, T0, T1)
ReplayBehaviour.RecordMethodCall<T0, T1, T2>(Action<T0, T1, T2>, T0, T1, T2)
ReplayBehaviour.RecordMethodCall<T0, T1, T2, T3>(Action<T0, T1, T2, T3>, T0, T1, T2, T3)
MonoBehaviour.IsInvoking()
MonoBehaviour.CancelInvoke()
MonoBehaviour.Invoke(string, float)
MonoBehaviour.InvokeRepeating(string, float, float)
MonoBehaviour.CancelInvoke(string)
MonoBehaviour.IsInvoking(string)
MonoBehaviour.StartCoroutine(string)
MonoBehaviour.StartCoroutine(string, object)
MonoBehaviour.StartCoroutine(IEnumerator)
MonoBehaviour.StartCoroutine_Auto(IEnumerator)
MonoBehaviour.StopCoroutine(IEnumerator)
MonoBehaviour.StopCoroutine(Coroutine)
MonoBehaviour.StopCoroutine(string)

MonoBehaviour.StopAllCoroutines()

MonoBehaviour.print(object)

MonoBehaviour.useGUILayout

MonoBehaviour.runInEditMode

Behaviour.enabled

Behaviour.isActiveAndEnabled

Component.GetComponent(Type)

Component.GetComponent<T>()

Component.TryGetComponent(Type, out Component)

Component.TryGetComponent<T>(out T)

Component.GetComponent(string)

Component.GetComponentInChildren(Type, bool)

Component.GetComponentInChildren(Type)

Component.GetComponentInChildren<T>(bool)

Component.GetComponentInChildren<T>()

Component.GetComponentsInChildren(Type, bool)

Component.GetComponentsInChildren(Type)

Component.GetComponentsInChildren<T>(bool)

Component.GetComponentsInChildren<T>(bool, List<T>)

Component.GetComponentsInChildren<T>()

Component.GetComponentsInChildren<T>(List<T>)

Component.GetComponentInParent(Type, bool)

Component.GetComponentInParent(Type)

Component.GetComponentInParent<T>(bool)

Component.GetComponentInParent<T>()

Component.GetComponentsInParent(Type, bool)

Component.GetComponentsInParent(Type)

Component.GetComponentsInParent<T>(bool)

Component.GetComponentsInParent<T>(bool, List<T>)

Component.GetComponentsInParent<T>()

Component.GetComponents(Type)

Component.GetComponents(Type, List<Component>)

Component.GetComponents<T>(List<T>)

Component.GetComponents<T>()

Component.CompareTag(string)

Component.SendMessageUpwards(string, object, SendMessageOptions)

Component.SendMessageUpwards(string, object)

Component.SendMessageUpwards(string)

Component.SendMessageUpwards(string, SendMessageOptions)

Component.SendMessage(string, object)

Component.SendMessage(string)

Component.SendMessage(string, object, SendMessageOptions)

Component.SendMessage(string, SendMessageOptions)

Component.BroadcastMessage(string, object, SendMessageOptions)

Component.BroadcastMessage(string, object)

Component.BroadcastMessage(string)

Component.BroadcastMessage(string, SendMessageOptions)

Component.transform

Component.gameObject

Component.tag

Object.GetInstanceID()

Object.GetHashCode()

Object.Equals(object)
Object.Instantiate(Object, Vector3, Quaternion)
Object.Instantiate(Object, Vector3, Quaternion, Transform)
Object.Instantiate(Object)
Object.Instantiate(Object, Transform)
Object.Instantiate(Object, Transform, bool)
Object.Instantiate<T>(T)
Object.Instantiate<T>(T, Vector3, Quaternion)
Object.Instantiate<T>(T, Vector3, Quaternion, Transform)
Object.Instantiate<T>(T, Transform)
Object.Instantiate<T>(T, Transform, bool)
Object.Destroy(Object, float)
Object.Destroy(Object)
Object.DestroyImmediate(Object, bool)
Object.DestroyImmediate(Object)
Object.FindObjectsOfType(Type)
Object.FindObjectsOfType(Type, bool)
Object.DontDestroyOnLoad(Object)
Object.DestroyObject(Object, float)
Object.DestroyObject(Object)
Object.FindSceneObjectsOfType(Type)
Object.FindObjectsOfTypeIncludingAssets(Type)
Object.FindObjectsOfType<T>()
Object.FindObjectsOfType<T>(bool)
Object.FindObjectOfType<T>()
Object.FindObjectOfType<T>(bool)
Object.FindObjectsOfTypeAll(Type)
Object.FindObjectOfType(Type)
Object.FindObjectOfType(Type, bool)
Object.ToString()
Object.name
Object.hideFlags
object.Equals(object, object)
object.GetType()
object.ReferenceEquals(object, object)

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
public sealed class ReplayRiggedHumanoid : ReplayRecordableBehaviour, IReplaySerialize
```

Fields

observedAnimator

Declaration

```
public Animator observedAnimator
```

Field Value

TYPE	DESCRIPTION
Animator	

observedRoot

Declaration

```
public Transform observedRoot
```

Field Value

TYPE	DESCRIPTION
Transform	

Properties

BodyPositionPrecision

Declaration

```
public RecordPrecision BodyPositionPrecision { get; set; }
```

Property Value

TYPE	DESCRIPTION
RecordPrecision	

BodyRotationPrecision

Declaration

```
public RecordPrecision BodyRotationPrecision { get; set; }
```

Property Value

TYPE	DESCRIPTION
RecordPrecision	

Formatter

An optional ReplayFormatter that is used to serialize a particular component. Providing a formatter via his property can greatly reduce the amount of data that a ReplayObject needs to store.

Declaration

```
public override ReplayFormatter Formatter { get; }
```

Property Value

TYPE	DESCRIPTION
ReplayFormatter	

Overrides

ReplayRecordableBehaviour.Formatter

MuslceValuesPrecision

Declaration

```
public RecordPrecision MuslceValuesPrecision { get; set; }
```

Property Value

TYPE	DESCRIPTION
RecordPrecision	

ReplayBodyPosition

Declaration

```
public RecordFullAxisFlags ReplayBodyPosition { get; set; }
```

Property Value

TYPE	DESCRIPTION
RecordFullAxisFlags	

ReplayBodyRotation

Declaration

```
public RecordFullAxisFlags ReplayBodyRotation { get; set; }
```

Property Value

TYPE	DESCRIPTION
RecordFullAxisFlags	

Methods

Awake()

Called by Unity.

Declaration

```
protected override void Awake()
```

Overrides

ReplayBehaviour.Awake()

OnReplayDeserialize(ReplayState)

Called by the replay system when the recorder component should deserialize any necessary data during playback.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState containing the recorded data

Overrides

ReplayRecordableBehaviour.OnReplayDeserialize(ReplayState)

OnReplayReset()

Called by the replay system during playback when cached values should be reset to safe default to avoid glitches or inaccuracies in the playback.

Declaration

```
protected override void OnReplayReset()
```

Overrides

ReplayBehaviour.OnReplayReset()

OnReplaySerialize(ReplayState)

Called by the replay system when the recorder component should serialize and necessary data during recording.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState used to store the serialized data

Overrides

ReplayRecordableBehaviour.OnReplaySerialize(ReplayState)

OnReplayStart()

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.

Declaration

```
protected override void OnReplayStart()
```

Overrides

ReplayBehaviour.OnReplayStart()

OnReplayUpdate(float)

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

Declaration

```
protected override void OnReplayUpdate(float t)
```

Parameters

TYPE	NAME	DESCRIPTION
float	t	A normalized value representing the progress between replay snapshots. Use this value for interpolation or similar smoothing passes

Overrides

ReplayBehaviour.OnReplayUpdate(float)

Reset()

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

Declaration

```
protected override void Reset()
```

Overrides

ReplayBehaviour.Reset()

Implements

IReplaySerialize

Class ReplayScene

A ReplayScene contains information about all active replay objects.

Inheritance

object
ReplayScene

Inherited Members

object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
public sealed class ReplayScene
```

Constructors

ReplayScene(IEnumerable<ReplayObject>, IReplayPreparer)

Create a new replay scene from the specified collection or replay objects.

Declaration

```
public ReplayScene(IEnumerable<ReplayObject> replayObjects, IReplayPreparer replayPreparer = null)
```

Parameters

TYPE	NAME	DESCRIPTION
IEnumerable<ReplayObject>	replayObjects	A collection of ReplayObject that will be added to the scene
IReplayPreparer	replayPreparer	A IReplayPreparer implementation used to prepare scene objects when switching between playback and live scene modes

ReplayScene(ReplayObject, IReplayPreparer)

Create a new replay scene and add the specified replay object.

Declaration

```
public ReplayScene(ReplayObject replayObject, IReplayPreparer replayPreparer = null)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayObject	replayObject	The single ReplayObject to add to the scene

TYPE	NAME	DESCRIPTION
IReplayPreparer	replayPreparer	A IReplayPreparer implementation used to prepare scene objects when switching between playback and live scene modes

ReplayScene(IReplayPreparer)

Create a new replay scene with no ReplayObject added.

Declaration

```
public ReplayScene(IReplayPreparer replayPreparer = null)
```

Parameters

TYPE	NAME	DESCRIPTION
IReplayPreparer	replayPreparer	A IReplayPreparer implementation used to prepare scene objects when switching between playback and live scene modes

Fields

restorePreviousSceneState

A value indicating whether the replay objects stored in this scene instance should be reverted to their initial state when playback ends.

Declaration

```
public bool restorePreviousSceneState
```

Field Value

TYPE	DESCRIPTION
bool	

Properties

ActiveReplayBehaviours

Get a collection of all ReplayBehaviour components that are registered in this ReplayScene.

Declaration

```
public HashSet<ReplayBehaviour> ActiveReplayBehaviours { get; }
```

Property Value

TYPE	DESCRIPTION
HashSet<ReplayBehaviour>	

ActiveReplayObjects

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

Declaration

```
public HashSet<ReplayObject> ActiveReplayObjects { get; }
```

Property Value

TYPE	DESCRIPTION
HashSet<ReplayObject>	

IsEmpty

Returns a value indicating whether the ReplayScene contains any ReplayObject.

Declaration

```
public bool IsEmpty { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

ReplayEnabled

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.

Declaration

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

Property Value

TYPE	DESCRIPTION
bool	

Methods

AddReplayObject(ReplayObject)

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.

Declaration

```
public void AddReplayObject(ReplayObject replayObject)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayObject	replayObject	The ReplayObject to register

Exceptions

TYPE	CONDITION
ArgumentNullException	The specified game object is null

AddReplayObject(GameObject)

Add the specified game object to the replay scene. Only game objects with a ReplayObject attached will be accepted. Replay objects must be added to a replay scene in order to be recorded or replayed by the replay system.

Declaration

```
public void AddReplayObject(GameObject gameObject)
```

Parameters

TYPE	NAME	DESCRIPTION
GameObject	gameObject	The target game object to add to the replay scene

Exceptions

TYPE	CONDITION
ArgumentNullException	The specified game object is null
InvalidOperationException	The specified game object does not have a ReplayObject attached

CaptureSnapshot(float, int, ReplayPersistentData)

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

Declaration

```
public ReplaySnapshot CaptureSnapshot(float timeStamp, int sequenceID, ReplayPersistentData persistentData)
```

Parameters

TYPE	NAME	DESCRIPTION
float	timeStamp	The timestamp for the frame indicating its position in the playback sequence
int	sequenceID	
ReplayPersistentData	persistentData	

Returns

TYPE	DESCRIPTION
ReplaySnapshot	A new snapshot of the current replay scene

CheckIntegrity(bool)

Check if any registered ReplayObject have been invalidated or destroyed since they were added to the scene.

Declaration

```
public bool CheckIntegrity(bool throwOnError)
```

Parameters

TYPE	NAME	DESCRIPTION
bool	throwOnError	True if an exception should be thrown if there are integrity issues

Returns

TYPE	DESCRIPTION
bool	True if this scene is valid or false if one or more registered ReplayObject have been destroyed but not unregistered

Exceptions

TYPE	CONDITION
Exception	The replay scene contains one or more destroyed objects

Clear()

Remove all replay objects form this replay scene.

Declaration

```
public void Clear()
```

FromCurrentScene(IReplayPreparer)

Create a new replay scene from the active Unity scene. All ReplayObject in the active scene will be added to the ReplayScene result. The active scene is equivalent of UnityEngine.SceneManagement.SceneManager.GetActiveScene();

Declaration

```
public static ReplayScene FromCurrentScene(IReplayPreparer preparer = null)
```

Parameters

TYPE	NAME	DESCRIPTION
IReplayPreparer	preparer	

Returns

TYPE	DESCRIPTION
ReplayScene	A new ReplayScene instance

FromScene(Scene, IReplayPreparer)

Create a new replay scene from the specified Unity scene. All ReplayScene in the specified scene will be added to the ReplayScene result.

Declaration

```
public static ReplayScene FromScene(Scene scene, IReplayPreparer preparer = null)
```

Parameters

TYPE	NAME	DESCRIPTION
Scene	scene	The Unity scene used to create the ReplayScene
IReplayPreparer	preparer	

Returns

TYPE	DESCRIPTION
ReplayScene	A new ReplayScene instance

GetReplayObject(ReplayIdentity)

Attempt to find a ReplayObject with the specified ReplayIdentity

Declaration

```
public ReplayObject GetReplayObject(ReplayIdentity replayIdentity)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayIdentity	replayIdentity	The identity of the object to find

Returns

TYPE	DESCRIPTION
ReplayObject	A ReplayObject with the specified ID or null if the object was not found

HasReplayObject(ReplayIdentity)

Check if the replay scene has a ReplayObject registered with the specified ReplayIdentity.

Declaration

```
public bool HasReplayObject(ReplayIdentity replayIdentity)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayIdentity	replayIdentity	The id of the to search for

Returns

TYPE	DESCRIPTION
bool	True if an object with the specified id is added to this ReplayScene

RemoveReplayObject(ReplayObject)

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.

Declaration

```
public void RemoveReplayObject(ReplayObject replayObject)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayObject	replayObject	

RemoveReplayObject(GameObject)

Unregisters a replay object from this replay scene.

Declaration

```
public void RemoveReplayObject(GameObject gameObject)
```

Parameters

TYPE	NAME	DESCRIPTION
GameObject	gameObject	

RestoreSnapshot(ReplaySnapshot, ReplayStorage, bool)

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

Declaration

```
public void RestoreSnapshot(ReplaySnapshot snapshot, ReplayStorage storage, bool simulate = false)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplaySnapshot	snapshot	The snapshot to restore
ReplayStorage	storage	The ReplayStorage used to restore dynamic object information from

TYPE	NAME	DESCRIPTION
bool	simulate	

SetReplaySceneMode(ReplaySceneMode, ReplayStorage, RestoreSceneMode)

Set the current replay scene mode. Use this method to switch the scene between playback and live modes. Playback modes will run the replayPreparer on all scene objects to disable or re-enable elements that could affect playback.

Declaration

```
public void SetReplaySceneMode(ReplaySceneMode mode, ReplayStorage storage, RestoreSceneMode restoreMode =
RestoreSceneMode.RestoreState)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplaySceneMode	mode	The scene mode to switch to
ReplayStorage	storage	
RestoreSceneMode	restoreMode	

Events

OnReplayObjectAdded

Called when a replay object was added to this ReplayScene.

Declaration

```
public event Action<ReplayObject> OnReplayObjectAdded
```

Event Type

TYPE	DESCRIPTION
Action<ReplayObject>	

OnReplayObjectRemoved

Called when a replay object was removed from this ReplayScene.

Declaration

```
public event Action<ReplayObject> OnReplayObjectRemoved
```

Event Type

TYPE	DESCRIPTION
Action<ReplayObject>	

Enum ReplaySceneMode

The scene state value used to determine which mode a particular scene instance is in.

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
public enum ReplaySceneMode
```

Fields

NAME	DESCRIPTION
Live	The scene and all child objects are in live mode meaning gameplay can continue as normal.
Playback	The scene and all child objects are in playback mode. Objects in the scene should not be interfered with and will be updated frequently.
Record	The scene and all child objects are in record mode. Gameplay can continue but objects will be sampled frequently.

Class ReplaySettings

Stores global settings used by the replay system.

Inheritance

object
Object
ScriptableObject
ReplaySettings

Inherited Members

ScriptableObject.SetDirty()
ScriptableObject.CreateInstance(string)
ScriptableObject.CreateInstance(Type)
ScriptableObject.CreateInstance<T>()
Object.GetInstanceID()
Object.GetHashCode()
Object.Equals(object)
Object.Instantiate(Object, Vector3, Quaternion)
Object.Instantiate(Object, Vector3, Quaternion, Transform)
Object.Instantiate(Object)
Object.Instantiate(Object, Transform)
Object.Instantiate(Object, Transform, bool)
Object.Instantiate<T>(T)
Object.Instantiate<T>(T, Vector3, Quaternion)
Object.Instantiate<T>(T, Vector3, Quaternion, Transform)
Object.Instantiate<T>(T, Transform)
Object.Instantiate<T>(T, Transform, bool)
Object.Destroy(Object, float)
Object.Destroy(Object)
Object.DestroyImmediate(Object, bool)
Object.DestroyImmediate(Object)
Object.FindObjectsOfType(Type)
Object.FindObjectsOfType(Type, bool)
Object.DontDestroyOnLoad(Object)
Object.DestroyObject(Object, float)
Object.DestroyObject(Object)
Object.FindSceneObjectsOfType(Type)
Object.FindObjectsOfTypeIncludingAssets(Type)
Object.FindObjectsOfType<T>()
Object.FindObjectsOfType<T>(bool)
Object.FindObjectOfType<T>()
Object.FindObjectOfType<T>(bool)
Object.FindObjectsOfTypeAll(Type)
Object.FindObjectOfType(Type)
Object.FindObjectOfType(Type, bool)
Object.ToString()
Object.name
Object.hideFlags
object.Equals(object, object)
object.GetType()
object.ReferenceEquals(object, object)

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
[Serializable]
public sealed class ReplaySettings : ScriptableObject
```

Properties

DefaultReplayPreparer

Get the DefaultReplayPreparer that will be used to prepare replay objects by default.

Declaration

```
public DefaultReplayPreparer DefaultReplayPreparer { get; }
```

Property Value

TYPE	DESCRIPTION
DefaultReplayPreparer	

PlaybackOptions

Get the default ReplayPlaybackOptions that will be used if no options are provided by code.

Declaration

```
public ReplayPlaybackOptions PlaybackOptions { get; }
```

Property Value

TYPE	DESCRIPTION
ReplayPlaybackOptions	

PrefabProviders

Get all ReplayObjectLifecycleProvider that have been setup by the user.

Declaration

```
public IReadOnlyList<ReplayObjectLifecycleProvider> PrefabProviders { get; }
```

Property Value

TYPE	DESCRIPTION
IReadOnlyList<ReplayObjectLifecycleProvider>	

RecordOptions

Get the default ReplayRecordOptions that will be used if no options are provided by code.

Declaration

```
public ReplayRecordOptions RecordOptions { get; }
```

Property Value

TYPE	DESCRIPTION
ReplayRecordOptions	

Methods

AddPrefabProvider(ReplayObjectLifecycleProvider)

Declaration

```
public void AddPrefabProvider(ReplayObjectLifecycleProvider provider)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayObjectLifecycleProvider	provider	

GetPrefabProvider(ReplayIdentity)

Get the ReplayObjectLifecycleProvider for the replay prefab with the specified prefab id.

Declaration

```
public ReplayObjectLifecycleProvider GetPrefabProvider(ReplayIdentity prefabId)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayIdentity	prefabId	The replay id for the replay prefab

Returns

TYPE	DESCRIPTION
ReplayObjectLifecycleProvider	The associated ReplayObjectLifecycleProvider or null if the prefab id could not be found

HasPrefabProvider(ReplayIdentity)

Returns a value indicating whether the specified replay id is a valid prefab id and has a ReplayObjectLifecycleProvider associated with it.

Declaration

```
public bool HasPrefabProvider(ReplayIdentity prefabId)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayIdentity	prefabId	The replay id for a given replay prefab

Returns

TYPE	DESCRIPTION
bool	True if a provider is registered or false if not

InstantiatePrefabProvider(ReplayIdentity, Vector3, Quaternion)

Attempt to instantiate a replay prefab instance for the specified prefab id.

Declaration

```
public ReplayObject InstantiatePrefabProvider(ReplayIdentity prefabId, Vector3 position, Quaternion rotation)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayIdentity	prefabId	The replay prefab id for the target replay object prefab
Vector3	position	The position where the replay object should be instantiated
Quaternion	rotation	The initial rotation of the replay object

Returns

TYPE	DESCRIPTION
ReplayObject	An instantiated ReplayObject or null if the specified prefab id could not be found

RemovePrefabProvider(ReplayObjectLifecycleProvider)

Declaration

```
public void RemovePrefabProvider(ReplayObjectLifecycleProvider provider)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayObjectLifecycleProvider	provider	

Class ReplayState

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

Inheritance

object

ReplayState

Implements

IDisposable

IReplayReusable

IReplaySerialize

IReplaySnapshotStorable

IReplayStreamSerialize

IReplayTokenSerialize

Inherited Members

object.Equals(object)

object.Equals(object, object)

object.GetHashCode()

object.GetType()

object.ReferenceEquals(object, object)

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
public sealed class ReplayState : IDisposable, IReplayReusable, IReplaySerialize, IReplaySnapshotStorable,
IReplayStreamSerialize, IReplayTokenSerialize
```

Fields

pool

Declaration

```
public static readonly ReplayInstancePool<ReplayState> pool
```

Field Value

TYPE	DESCRIPTION
ReplayInstancePool<ReplayState>	

Properties

AsHexString

Declaration

```
[ReplayTokenSerialize("Raw Data")]
public string AsHexString { get; set; }
```

Property Value

TYPE	DESCRIPTION
string	

CanRead

Returns true if the state contains any more data.

Declaration

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

Property Value

TYPE	DESCRIPTION
bool	

DataHash

Declaration

```
public long DataHash { get; }
```

Property Value

TYPE	DESCRIPTION
long	

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.

Declaration

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

Property Value

TYPE	DESCRIPTION
bool	

Size

Returns the size of the object state in bytes.

Declaration

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

Property Value

TYPE	DESCRIPTION
int	

Methods

Append(ReplayState)

Declaration

```
public void Append(ReplayState data)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	data	

Clear()

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

Declaration

```
public void Clear()
```

CopyTo(ReplayState)

Copy all data to the target ReplayState. All state information such as dataHash and readPointer will be maintained. This ReplayState must not be empty (Must contain data) otherwise this method will return false. The destination ReplayState must be empty otherwise this method will return false.

Declaration

```
public bool CopyTo(ReplayState destination)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	destination	

Returns

TYPE	DESCRIPTION
bool	True if the copy was successful or false if not

Exceptions

TYPE	CONDITION
ArgumentNullException	Destination state is null
ObjectDisposedException	This ReplayState or destination ReplayState is disposed

Dispose()

Declaration

```
public void Dispose()
```

EnsureCapacity(int)

Declaration

```
public void EnsureCapacity(int size)
```

Parameters

TYPE	NAME	DESCRIPTION
int	size	

FromByteArray(byte[])

Declaration

```
public static ReplayState FromByteArray(byte[] rawStateData)
```

Parameters

TYPE	NAME	DESCRIPTION
byte[]	rawStateData	

Returns

TYPE	DESCRIPTION
ReplayState	

GetDeserializeMethod(Type)

Declaration

```
public static MethodInfo GetDeserializeMethod(Type type)
```

Parameters

TYPE	NAME	DESCRIPTION
Type	type	

Returns

TYPE	DESCRIPTION
MethodInfo	

GetDeserializeMethod<T>()

Declaration

```
public static MethodInfo GetDeserializeMethod<T>()
```

Returns

TYPE	DESCRIPTION
MethodInfo	

Type Parameters

NAME	DESCRIPTION
T	

GetSerializeMethod(Type)

Declaration

```
public static MethodInfo GetSerializeMethod(Type type)
```

Parameters

TYPE	NAME	DESCRIPTION
Type	type	

Returns

TYPE	DESCRIPTION
MethodInfo	

GetSerializeMethod<T>()

Declaration

```
public static MethodInfo GetSerializeMethod<T>()
```

Returns

TYPE	DESCRIPTION
MethodInfo	

Type Parameters

NAME	DESCRIPTION
T	

InitializeFromData(byte[])

Declaration

```
public void InitializeFromData(byte[] stateData)
```

Parameters

TYPE	NAME	DESCRIPTION
byte[]	stateData	

IsDataEqual(ReplayState)

Declaration

```
public bool IsDataEqual(ReplayState other)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	other	

Returns

TYPE	DESCRIPTION
bool	

IsTypeSerializable(Type)

Declaration

```
public static bool IsTypeSerializable(Type type)
```

Parameters

TYPE	NAME	DESCRIPTION
Type	type	

Returns

TYPE	DESCRIPTION
bool	

IsTypeSerializable<T>()

Declaration

```
public static bool IsTypeSerializable<T>()
```

Returns

TYPE	DESCRIPTION
bool	

Type Parameters

NAME	DESCRIPTION
T	

PrepareForRead()

Prepares the state for read operations by seeking the read pointer back to the start.

Declaration

```
public void PrepareForRead()
```

ReadBool()

Read a bool from the state.

Declaration

```
public bool ReadBool()
```

Returns

TYPE	DESCRIPTION
bool	Bool value

ReadByte()

Read a byte from the state.

Declaration

```
public byte ReadByte()
```

Returns

TYPE	DESCRIPTION
byte	Byte value

ReadBytes(byte[], int, int)

Fill a byte array with data from the state.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
byte[]	buffer	The byte array to store data in
int	offset	The index offset to start filling the buffer at
int	amount	The number of bytes to read

ReadBytes(int)

Read a byte array from the state.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
int	amount	The number of bytes to read

Returns

TYPE	DESCRIPTION
byte[]	Byte array value

ReadColor()

Read a color from the state.

Declaration

```
public Color ReadColor()
```

Returns

TYPE	DESCRIPTION
Color	Color value

ReadColor32()

Read a color32 from the state.

Declaration

```
public Color32 ReadColor32()
```

Returns

TYPE	DESCRIPTION
Color32	Color32 value

ReadDouble()

Declaration

```
public double ReadDouble()
```

Returns

TYPE	DESCRIPTION
double	

ReadHalf()

Attempts to read a low precision float. You should only use this method when the value is relatively small (less than 65000) and accuracy is not essential. When read, a half value will almost certainly be within +-0.015f tolerance of the original value.

Declaration

```
public float ReadHalf()
```

Returns

TYPE	DESCRIPTION
float	float value

ReadInt16()

Read a short from the state.

Declaration

```
public short ReadInt16()
```

Returns

TYPE	DESCRIPTION
short	Short value

ReadInt32()

Read an int from the state.

Declaration

```
public int ReadInt32()
```

Returns

TYPE	DESCRIPTION
int	Int value

ReadInt64()

Declaration

```
public long ReadInt64()
```

Returns

TYPE	DESCRIPTION
long	

ReadQuaternion()

Read a quaternion from the state.

Declaration

```
public Quaternion ReadQuaternion()
```

Returns

TYPE	DESCRIPTION
Quaternion	Quaternion value

ReadQuaternionHalf()

Attempts to read a low precision quaternion. When read, a half value will almost certainly be within +-0.015f tolerance of the original value.

Declaration

```
public Quaternion ReadQuaternionHalf()
```

Returns

TYPE	DESCRIPTION
Quaternion	quaternion value

ReadSByte()

Declaration

```
public sbyte ReadSByte()
```

Returns

TYPE	DESCRIPTION
sbyte	

ReadSerializable(IReplaySerialize)

Declaration

```
public bool ReadSerializable(IReplaySerialize replaySerializable)
```

Parameters

TYPE	NAME	DESCRIPTION
IReplaySerialize	replaySerializable	

Returns

TYPE	DESCRIPTION
bool	

ReadSerializable<T>()

Declaration

```
public T ReadSerializable<T>() where T : IReplaySerialize, new()
```

Returns

TYPE	DESCRIPTION
T	

Type Parameters

NAME	DESCRIPTION
T	

ReadSerializable<T>(ref T)

Declaration

```
public bool ReadSerializable<T>(ref T replaySerializable) where T : IReplaySerialize
```

Parameters

TYPE	NAME	DESCRIPTION
T	replaySerializable	

Returns

TYPE	DESCRIPTION
bool	

Type Parameters

NAME	DESCRIPTION
T	

ReadSingle()

Read a float from the state.

Declaration

```
public float ReadSingle()
```

Returns

TYPE	DESCRIPTION
float	Float value

ReadState()

Declaration

```
public ReplayState ReadState()
```

Returns

TYPE	DESCRIPTION
ReplayState	

ReadString()

Read a string from the state

Declaration

```
public string ReadString()
```

Returns

TYPE	DESCRIPTION
string	string value

ReadUInt16()

Declaration

```
public ushort ReadUInt16()
```

Returns

TYPE	DESCRIPTION
ushort	

ReadUInt32()

Declaration

```
public uint ReadUInt32()
```

Returns

TYPE	DESCRIPTION
uint	

ReadUInt64()

Declaration

```
public ulong ReadUInt64()
```

Returns

TYPE	DESCRIPTION
ulong	

ReadVector2()

Read a vector2 from the state.

Declaration

```
public Vector2 ReadVector2()
```

Returns

TYPE	DESCRIPTION
Vector2	Vector2 value

ReadVector2Half()

Attempts to read a low precision vector2. When read, a half value will almost certainly be within +-0.015f tolerance of the original value.

Declaration

```
public Vector2 ReadVector2Half()
```

Returns

TYPE	DESCRIPTION
Vector2	vector2 value

ReadVector3()

Read a vector3 from the state.

Declaration

```
public Vector3 ReadVector3()
```

Returns

TYPE	DESCRIPTION
Vector3	Vector3 value

ReadVector3Half()

Attempts to read a low precision vector3. When read, a half value will almost certainly be within +-0.015f tolerance of the original value.

Declaration

```
public Vector3 ReadVector3Half()
```

Returns

TYPE	DESCRIPTION
Vector3	vector3 value

ReadVector4()

Read a vector4 from the state.

Declaration

```
public Vector4 ReadVector4()
```

Returns

TYPE	DESCRIPTION
Vector4	Vector4 value

ReadVector4Half()

Attempts to read a low precision vector4. When read, a half value will almost certainly be within +-0.015f tolerance of the original value.

Declaration

```
public Vector4 ReadVector4Half()
```

Returns

TYPE	DESCRIPTION
Vector4	vector4 value

ToArray()

Get the ReplayState data as a byte array.

Declaration

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

Returns

TYPE	DESCRIPTION
byte[]	A byte array of data

ToString()

Declaration

```
public override string ToString()
```

Returns

TYPE	DESCRIPTION
string	

Overrides

object.ToString()

Write(bool)

Write a bool to the state

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
bool	value	bool value

Write(byte)

Write a byte to the state.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
byte	value	Byte value

Write(byte[])

Write a byte array to the state.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
byte[]	bytes	Byte array value

Write(byte[], int, int)

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

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
byte[]	bytes	Byte array value
int	offset	The start index to read data from the array

TYPE	NAME	DESCRIPTION
int	length	The amount of data to read

Write(double)

Declaration

```
public void Write(double value)
```

Parameters

TYPE	NAME	DESCRIPTION
double	value	

Write(short)

Write a short to the state.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
short	value	Short value

Write(int)

Write an int to the state.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
int	value	Int value

Write(long)

Declaration

```
public void Write(long value)
```

Parameters

TYPE	NAME	DESCRIPTION
long	value	

Write(sbyte)

Declaration

```
public void Write(sbyte value)
```

Parameters

TYPE	NAME	DESCRIPTION
sbyte	value	

Write(float)

Write a float to the state.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
float	value	Float value

Write(string)

Write a string to the state.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
string	value	string value

Write(ushort)

Declaration

```
public void Write(ushort value)
```

Parameters

TYPE	NAME	DESCRIPTION
ushort	value	

Write(uint)

Declaration

```
public void Write(uint value)
```

Parameters

TYPE	NAME	DESCRIPTION
uint	value	

Write(ulong)

Declaration

```
public void Write(ulong value)
```

Parameters

TYPE	NAME	DESCRIPTION
ulong	value	

Write(IReplaySerialize)

Declaration

```
public void Write(IReplaySerialize replaySerializable)
```

Parameters

TYPE	NAME	DESCRIPTION
IReplaySerialize	replaySerializable	

Write(in Color32)

Write a color32 value to the state.

Declaration

```
public void Write(in Color32 value)
```

Parameters

TYPE	NAME	DESCRIPTION
Color32	value	Color32 value

Write(in Color)

Write a color to the state.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Color	value	Color value

Write(in Quaternion)

Write a quaternion to the state.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Quaternion	value	Quaternion value

Write(in Vector2)

Write a vector2 to the state.

Declaration

```
public void Write(in Vector2 value)
```

Parameters

TYPE	NAME	DESCRIPTION
Vector2	value	Vector2 value

Write(in Vector3)

Write a vector3 to the state.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Vector3	value	Vector3 value

Write(in Vector4)

Write a vector4 to the state.

Declaration

```
public void Write(in Vector4 value)
```

Parameters

TYPE	NAME	DESCRIPTION
Vector4	value	Vector4 value

WriteHalf(float)

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. When read, a half value will almost certainly be within +-0.015f tolerance of the original value.

Declaration

```
public void WriteHalf(float value)
```

Parameters

TYPE	NAME	DESCRIPTION
float	value	float value

WriteHalf(in Quaternion)

Write a quaternion to the state using half precision packing. Accuracy may be lost by storing low precision values. When read, a half value will almost certainly be within +-0.015f tolerance of the original value.

Declaration

```
public void WriteHalf(in Quaternion value)
```

Parameters

TYPE	NAME	DESCRIPTION
Quaternion	value	quaternion value

WriteHalf(in Vector2)

Write a vector2 to the state using half precision packing. Accuracy may be lost by storing low precision values. When read, a half value will almost certainly be within +-0.015f tolerance of the original value.

Declaration

```
public void WriteHalf(in Vector2 value)
```

Parameters

TYPE	NAME	DESCRIPTION
Vector2	value	vector2 value

WriteHalf(in Vector3)

Write a vector3 to the state using half precision packing. Accuracy may be lost by storing low precision values. When read, a half value will almost certainly be within +-0.015f tolerance of the original value.

Declaration

```
public void WriteHalf(in Vector3 value)
```

Parameters

TYPE	NAME	DESCRIPTION
Vector3	value	vector3 value

WriteHalf(in Vector4)

Write a vector4 to the state using half precision packing. Accuracy may be lost by storing low precision values. When read, a half value will almost certainly be within +-0.015f tolerance of the original value.

Declaration

```
public void WriteHalf(in Vector4 value)
```

Parameters

TYPE	NAME	DESCRIPTION
Vector4	value	vector4 value

Implements

IDisposable
IReplayReusable
IReplaySerialize
IReplaySnapshotStorable
IReplayStreamSerialize
IReplayTokenSerialize

Class ReplayTokenSerializeAttribute

Attribute used to mark members as serializable using a text format. The serialized name can be specified via the attribute or the member name will be used if no name is provided.

Inheritance

object
Attribute
ReplayTokenSerializeAttribute

Implements

_Attribute

Inherited Members

Attribute.Equals(object)
Attribute.GetCustomAttribute(Assembly, Type)
Attribute.GetCustomAttribute(Assembly, Type, bool)
Attribute.GetCustomAttribute(MemberInfo, Type)
Attribute.GetCustomAttribute(MemberInfo, Type, bool)
Attribute.GetCustomAttribute(Module, Type)
Attribute.GetCustomAttribute(Module, Type, bool)
Attribute.GetCustomAttribute(ParameterInfo, Type)
Attribute.GetCustomAttribute(ParameterInfo, Type, bool)
Attribute.GetCustomAttributes(Assembly)
Attribute.GetCustomAttributes(Assembly, bool)
Attribute.GetCustomAttributes(Assembly, Type)
Attribute.GetCustomAttributes(Assembly, Type, bool)
Attribute.GetCustomAttributes(MemberInfo)
Attribute.GetCustomAttributes(MemberInfo, bool)
Attribute.GetCustomAttributes(MemberInfo, Type)
Attribute.GetCustomAttributes(MemberInfo, Type, bool)
Attribute.GetCustomAttributes(Module)
Attribute.GetCustomAttributes(Module, bool)
Attribute.GetCustomAttributes(Module, Type)
Attribute.GetCustomAttributes(Module, Type, bool)
Attribute.GetCustomAttributes(ParameterInfo)
Attribute.GetCustomAttributes(ParameterInfo, bool)
Attribute.GetCustomAttributes(ParameterInfo, Type)
Attribute.GetCustomAttributes(ParameterInfo, Type, bool)
Attribute.GetHashCode()
Attribute.IsDefaultAttribute()
Attribute.IsDefined(Assembly, Type)
Attribute.IsDefined(Assembly, Type, bool)
Attribute.IsDefined(MemberInfo, Type)
Attribute.IsDefined(MemberInfo, Type, bool)
Attribute.IsDefined(Module, Type)
Attribute.IsDefined(Module, Type, bool)
Attribute.IsDefined(ParameterInfo, Type)
Attribute.IsDefined(ParameterInfo, Type, bool)
Attribute.Match(object)
Attribute.TypeId
object.Equals(object, object)
object.GetType()

object.ReferenceEquals(object, object)
object.ToString()

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
[AttributeUsage(AttributeTargets.Property|AttributeTargets.Field, AllowMultiple = false, Inherited = false)]
public sealed class ReplayTokenSerializeAttribute : Attribute, _Attribute
```

Constructors

ReplayTokenSerializeAttribute(string)

Declaration

```
public ReplayTokenSerializeAttribute(string overrideName = null)
```

Parameters

TYPE	NAME	DESCRIPTION
string	overrideName	

Properties

OverrideName

Declaration

```
public string OverrideName { get; }
```

Property Value

TYPE	DESCRIPTION
string	

Methods

GetSerializeName(string)

Declaration

```
public string GetSerializeName(string fallback)
```

Parameters

TYPE	NAME	DESCRIPTION
string	fallback	

Returns

TYPE	DESCRIPTION
string	

Implements

_Attribute

Class ReplayTrailRenderer

Inheritance

object
Object
Component
Behaviour
MonoBehaviour
ReplayBehaviour
ReplayRecordableBehaviour
ReplayTrailRenderer

Implements

IReplaySerialize

Inherited Members

ReplayRecordableBehaviour.Formatter
ReplayRecordableBehaviour.OnDestroy()
ReplayBehaviour.ReplayIdentity
ReplayBehaviour.ReplayObject
ReplayBehaviour.HasPersistentData
ReplayBehaviour.ReplayPersistentData
ReplayBehaviour.Variables
ReplayBehaviour.HasVariables
ReplayBehaviour.IsRecording
ReplayBehaviour.IsRecordingPaused
ReplayBehaviour.IsRecordingOrPaused
ReplayBehaviour.IsReplaying
ReplayBehaviour.IsPlaybackPaused
ReplayBehaviour.IsReplayingOrPaused
ReplayBehaviour.PlaybackTime
ReplayBehaviour.PlaybackTimeNormalized
ReplayBehaviour.PlaybackTimeScale
ReplayBehaviour.PlaybackDirection
ReplayBehaviour.Reset()
ReplayBehaviour.Awake()
ReplayBehaviour.OnEnable()
ReplayBehaviour.OnDisable()
ReplayBehaviour.OnReplayStart()
ReplayBehaviour.OnReplayPlayPause(bool)
ReplayBehaviour.OnReplayCapture()
ReplayBehaviour.OnReplayEvent(ushort, ReplayState)
ReplayBehaviour.OnReplaySpawned(Vector3, Quaternion)
ReplayBehaviour.ForceRegenerateIdentity()
ReplayBehaviour.RecordVariable(ReplayVariable)
ReplayBehaviour.RecordEvent(ushort, ReplayState)
ReplayBehaviour.RecordMethodCall(Action)
ReplayBehaviour.RecordMethodCall<T>(Action<T>, T)
ReplayBehaviour.RecordMethodCall<T0, T1>(Action<T0, T1>, T0, T1)
ReplayBehaviour.RecordMethodCall<T0, T1, T2>(Action<T0, T1, T2>, T0, T1, T2)
ReplayBehaviour.RecordMethodCall<T0, T1, T2, T3>(Action<T0, T1, T2, T3>, T0, T1, T2, T3)
MonoBehaviour.IsInvoking()
MonoBehaviour.CancelInvoke()

MonoBehaviour.Invoke(string, float)
MonoBehaviour.InvokeRepeating(string, float, float)
MonoBehaviour.CancelInvoke(string)
MonoBehaviour.IsInvoking(string)
MonoBehaviour.StartCoroutine(string)
MonoBehaviour.StartCoroutine(string, object)
MonoBehaviour.StartCoroutine(IEnumerator)
MonoBehaviour.StartCoroutine_Auto(IEnumerator)
MonoBehaviour.StopCoroutine(IEnumerator)
MonoBehaviour.StopCoroutine(Coroutine)
MonoBehaviour.StopCoroutine(string)
MonoBehaviour.StopAllCoroutines()
MonoBehaviour.print(object)
MonoBehaviour.useGUILayout
MonoBehaviour.runInEditMode
Behaviour.enabled
Behaviour.isActiveAndEnabled
Component.GetComponent(Type)
Component.GetComponent<T>()
Component.TryGetComponent(Type, out Component)
Component.TryGetComponent<T>(out T)
Component.GetComponent(string)
Component.GetComponentInChildren(Type, bool)
Component.GetComponentInChildren(Type)
Component.GetComponentInChildren<T>(bool)
Component.GetComponentInChildren<T>()
Component.GetComponentsInChildren(Type, bool)
Component.GetComponentsInChildren(Type)
Component.GetComponentsInChildren<T>(bool)
Component.GetComponentsInChildren<T>(bool, List<T>)
Component.GetComponentsInChildren<T>()
Component.GetComponentsInChildren<T>(List<T>)
Component.GetComponentInParent(Type, bool)
Component.GetComponentInParent(Type)
Component.GetComponentInParent<T>(bool)
Component.GetComponentInParent<T>()
Component.GetComponentsInParent(Type, bool)
Component.GetComponentsInParent(Type)
Component.GetComponentsInParent<T>(bool)
Component.GetComponentsInParent<T>(bool, List<T>)
Component.GetComponentsInParent<T>()
Component.GetComponents(Type)
Component.GetComponents(Type, List<Component>)
Component.GetComponents<T>(List<T>)
Component.GetComponents<T>()
Component.CompareTag(string)
Component.SendMessageUpwards(string, object, SendMessageOptions)
Component.SendMessageUpwards(string, object)
Component.SendMessageUpwards(string)
Component.SendMessageUpwards(string, SendMessageOptions)
Component.SendMessage(string, object)
Component.SendMessage(string)

Component.SendMessage(string, object, SendMessageOptions)
Component.SendMessage(string, SendMessageOptions)
Component.BroadcastMessage(string, object, SendMessageOptions)
Component.BroadcastMessage(string, object)
Component.BroadcastMessage(string)
Component.BroadcastMessage(string, SendMessageOptions)
Component.transform
Component.gameObject
Component.tag
Object.GetInstanceID()
Object.GetHashCode()
Object.Equals(object)
Object.Instantiate(Object, Vector3, Quaternion)
Object.Instantiate(Object, Vector3, Quaternion, Transform)
Object.Instantiate(Object)
Object.Instantiate(Object, Transform)
Object.Instantiate(Object, Transform, bool)
Object.Instantiate<T>(T)
Object.Instantiate<T>(T, Vector3, Quaternion)
Object.Instantiate<T>(T, Vector3, Quaternion, Transform)
Object.Instantiate<T>(T, Transform)
Object.Instantiate<T>(T, Transform, bool)
Object.Destroy(Object, float)
Object.Destroy(Object)
Object.DestroyImmediate(Object, bool)
Object.DestroyImmediate(Object)
Object.FindObjectsOfType(Type)
Object.FindObjectsOfType(Type, bool)
Object.DontDestroyOnLoad(Object)
Object.DestroyObject(Object, float)
Object.DestroyObject(Object)
Object.FindSceneObjectsOfType(Type)
Object.FindObjectsOfTypeIncludingAssets(Type)
Object.FindObjectsOfType<T>()
Object.FindObjectsOfType<T>(bool)
Object.FindObjectOfType<T>()
Object.FindObjectOfType<T>(bool)
Object.FindObjectsOfTypeAll(Type)
Object.FindObjectOfType(Type)
Object.FindObjectOfType(Type, bool)
Object.ToString()
Object.name
Object.hideFlags
object.Equals(object, object)
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
public class ReplayTrailRenderer : ReplayRecordableBehaviour, IReplaySerialize
```

Fields

clearOnReplayEnd

Declaration

```
public bool clearOnReplayEnd
```

Field Value

TYPE	DESCRIPTION
bool	

observedTrailRenderer

Declaration

```
public TrailRenderer observedTrailRenderer
```

Field Value

TYPE	DESCRIPTION
TrailRenderer	

updateFlags

Declaration

```
[HideInInspector]
public ReplayTrailRenderer.ReplayTrailRendererFlags updateFlags
```

Field Value

TYPE	DESCRIPTION
ReplayTrailRenderer.ReplayTrailRendererFlags	

Methods

OnReplayDeserialize(ReplayState)

Called by the replay system when the recorder component should deserialize any necessary data during playback.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState containing the recorded data

Overrides

ReplayRecordableBehaviour.OnReplayDeserialize(ReplayState)

OnReplayEnd()

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'

Declaration

```
protected override void OnReplayEnd()
```

Overrides

ReplayBehaviour.OnReplayEnd()

OnReplayReset()

Called by the replay system during playback when cached values should be reset to safe default to avoid glitches or inaccuracies in the playback.

Declaration

```
protected override void OnReplayReset()
```

Overrides

ReplayBehaviour.OnReplayReset()

OnReplaySerialize(ReplayState)

Called by the replay system when the recorder component should serialize and necessary data during recording.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState used to store the serialized data

Overrides

ReplayRecordableBehaviour.OnReplaySerialize(ReplayState)

OnReplayUpdate(float)

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

Declaration

```
protected override void OnReplayUpdate(float t)
```

Parameters

TYPE	NAME	DESCRIPTION
float	t	A normalized value representing the progress between replay snapshots. Use this value for interpolation or similar smoothing passes

Overrides

ReplayBehaviour.OnReplayUpdate(float)

Start()

Declaration

```
public void Start()
```

Implements

IReplaySerialize

Enum ReplayTrailRenderer.ReplayTrailRendererFlags

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
[Flags]
public enum ReplayTrailRenderer.ReplayTrailRendererFlags
```

Fields

NAME	DESCRIPTION
Interpolate	
None	

Class ReplayTransform

Inheritance

object
Object
Component
Behaviour
MonoBehaviour
ReplayBehaviour
ReplayRecordableBehaviour
ReplayTransform

Implements

IReplaySerialize

Inherited Members

ReplayRecordableBehaviour.OnDestroy()
ReplayBehaviour.ReplayIdentity
ReplayBehaviour.ReplayObject
ReplayBehaviour.HasPersistentData
ReplayBehaviour.ReplayPersistentData
ReplayBehaviour.Variables
ReplayBehaviour.HasVariables
ReplayBehaviour.IsRecording
ReplayBehaviour.IsRecordingPaused
ReplayBehaviour.IsRecordingOrPaused
ReplayBehaviour.IsReplaying
ReplayBehaviour.IsPlaybackPaused
ReplayBehaviour.IsReplayingOrPaused
ReplayBehaviour.PlaybackTime
ReplayBehaviour.PlaybackTimeNormalized
ReplayBehaviour.PlaybackTimeScale
ReplayBehaviour.PlaybackDirection
ReplayBehaviour.OnEnable()
ReplayBehaviour.OnDisable()
ReplayBehaviour.OnReplayStart()
ReplayBehaviour.OnReplayEnd()
ReplayBehaviour.OnReplayPlayPause(bool)
ReplayBehaviour.OnReplayCapture()
ReplayBehaviour.OnReplayEvent(ushort, ReplayState)
ReplayBehaviour.ForceRegenerateIdentity()
ReplayBehaviour.RecordVariable(ReplayVariable)
ReplayBehaviour.RecordEvent(ushort, ReplayState)
ReplayBehaviour.RecordMethodCall(Action)
ReplayBehaviour.RecordMethodCall<T>(Action<T>, T)
ReplayBehaviour.RecordMethodCall<T0, T1>(Action<T0, T1>, T0, T1)
ReplayBehaviour.RecordMethodCall<T0, T1, T2>(Action<T0, T1, T2>, T0, T1, T2)
ReplayBehaviour.RecordMethodCall<T0, T1, T2, T3>(Action<T0, T1, T2, T3>, T0, T1, T2, T3)
MonoBehaviour.IsInvoking()
MonoBehaviour.CancelInvoke()
MonoBehaviour.Invoke(string, float)
MonoBehaviour.InvokeRepeating(string, float, float)
MonoBehaviour.CancelInvoke(string)

MonoBehaviour.IsInvoking(string)
MonoBehaviour.StartCoroutine(string)
MonoBehaviour.StartCoroutine(string, object)
MonoBehaviour.StartCoroutine(IEnumerator)
MonoBehaviour.StartCoroutine_Auto(IEnumerator)
MonoBehaviour.StopCoroutine(IEnumerator)
MonoBehaviour.StopCoroutine(Coroutine)
MonoBehaviour.StopCoroutine(string)
MonoBehaviour.StopAllCoroutines()
MonoBehaviour.print(object)
MonoBehaviour.useGUILayout
MonoBehaviour.runInEditMode
Behaviour.enabled
Behaviour.isActiveAndEnabled
Component.GetComponent(Type)
Component.GetComponent<T>()
Component.TryGetComponent(Type, out Component)
Component.TryGetComponent<T>(out T)
Component.GetComponent(string)
Component.GetComponentInChildren(Type, bool)
Component.GetComponentInChildren(Type)
Component.GetComponentInChildren<T>(bool)
Component.GetComponentInChildren<T>()
Component.GetComponentsInChildren(Type, bool)
Component.GetComponentsInChildren(Type)
Component.GetComponentsInChildren<T>(bool)
Component.GetComponentsInChildren<T>(bool, List<T>)
Component.GetComponentsInChildren<T>()
Component.GetComponentsInChildren<T>(List<T>)
Component.GetComponentInParent(Type, bool)
Component.GetComponentInParent(Type)
Component.GetComponentInParent<T>(bool)
Component.GetComponentInParent<T>()
Component.GetComponentsInParent(Type, bool)
Component.GetComponentsInParent(Type)
Component.GetComponentsInParent<T>(bool)
Component.GetComponentsInParent<T>(bool, List<T>)
Component.GetComponentsInParent<T>()
Component.GetComponents(Type)
Component.GetComponents(Type, List<Component>)
Component.GetComponents<T>(List<T>)
Component.GetComponents<T>()
Component.CompareTag(string)
Component.SendMessageUpwards(string, object, SendMessageOptions)
Component.SendMessageUpwards(string, object)
Component.SendMessageUpwards(string)
Component.SendMessageUpwards(string, SendMessageOptions)
Component.SendMessage(string, object)
Component.SendMessage(string)
Component.SendMessage(string, object, SendMessageOptions)
Component.SendMessage(string, SendMessageOptions)
Component.BroadcastMessage(string, object, SendMessageOptions)

Component.BroadcastMessage(string, object)
Component.BroadcastMessage(string)
Component.BroadcastMessage(string, SendMessageOptions)
Component.transform
Component.gameObject
Component.tag
Object.GetInstanceID()
Object.GetHashCode()
Object.Equals(object)
Object.Instantiate(Object, Vector3, Quaternion)
Object.Instantiate(Object, Vector3, Quaternion, Transform)
Object.Instantiate(Object)
Object.Instantiate(Object, Transform)
Object.Instantiate(Object, Transform, bool)
Object.Instantiate<T>(T)
Object.Instantiate<T>(T, Vector3, Quaternion)
Object.Instantiate<T>(T, Vector3, Quaternion, Transform)
Object.Instantiate<T>(T, Transform)
Object.Instantiate<T>(T, Transform, bool)
Object.Destroy(Object, float)
Object.Destroy(Object)
Object.DestroyImmediate(Object, bool)
Object.DestroyImmediate(Object)
Object.FindObjectsOfType(Type)
Object.FindObjectsOfType(Type, bool)
Object.DontDestroyOnLoad(Object)
Object.DestroyObject(Object, float)
Object.DestroyObject(Object)
Object.FindSceneObjectsOfType(Type)
Object.FindObjectsOfTypeIncludingAssets(Type)
Object.FindObjectsOfType<T>()
Object.FindObjectsOfType<T>(bool)
Object.FindObjectOfType<T>()
Object.FindObjectOfType<T>(bool)
Object.FindObjectsOfTypeAll(Type)
Object.FindObjectOfType(Type)
Object.FindObjectOfType(Type, bool)
Object.ToString()
Object.name
Object.hideFlags
object.Equals(object, object)
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
[DisallowMultipleComponent]
public class ReplayTransform : ReplayRecordableBehaviour, IReplaySerialize
```

Properties

Formatter

An optional ReplayFormatter that is used to serialize a particular component. Providing a formatter via his property can greatly reduce the amount of data that a ReplayObject needs to store.

Declaration

```
public override ReplayFormatter Formatter { get; }
```

Property Value

TYPE	DESCRIPTION
ReplayFormatter	

Overrides

ReplayRecordableBehaviour.Formatter

PositionPrecision

Declaration

```
public RecordPrecision PositionPrecision { get; set; }
```

Property Value

TYPE	DESCRIPTION
RecordPrecision	

PositionSpace

Declaration

```
public RecordSpace PositionSpace { get; set; }
```

Property Value

TYPE	DESCRIPTION
RecordSpace	

ReplayPosition

Declaration

```
public RecordAxisFlags ReplayPosition { get; set; }
```

Property Value

TYPE	DESCRIPTION
RecordAxisFlags	

ReplayRotation

Declaration

```
public RecordAxisFlags ReplayRotation { get; set; }
```

Property Value

TYPE	DESCRIPTION
RecordAxisFlags	

ReplayScale

Declaration

```
public RecordAxisFlags ReplayScale { get; set; }
```

Property Value

TYPE	DESCRIPTION
RecordAxisFlags	

RotationPrecision

Declaration

```
public RecordPrecision RotationPrecision { get; set; }
```

Property Value

TYPE	DESCRIPTION
RecordPrecision	

RotationSpace

Declaration

```
public RecordSpace RotationSpace { get; set; }
```

Property Value

TYPE	DESCRIPTION
RecordSpace	

ScalePrecision

Declaration

```
public RecordPrecision ScalePrecision { get; set; }
```

Property Value

TYPE	DESCRIPTION
RecordPrecision	

Methods

Awake()

Called by Unity.

Declaration

```
protected override void Awake()
```

Overrides

ReplayBehaviour.Awake()

OnReplayDeserialize(ReplayState)

Called by the replay system when the recorder component should deserialize any necessary data during playback.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState containing the recorded data

Overrides

ReplayRecordableBehaviour.OnReplayDeserialize(ReplayState)

OnReplayReset()

Called by the replay system during playback when cached values should be reset to safe default to avoid glitches or inaccuracies in the playback.

Declaration

```
protected override void OnReplayReset()
```

Overrides

ReplayBehaviour.OnReplayReset()

OnReplaySerialize(ReplayState)

Called by the replay system when the recorder component should serialize and necessary data during recording.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState used to store the serialized data

Overrides

ReplayRecordableBehaviour.OnReplaySerialize(ReplayState)

OnReplaySpawned(Vector3, Quaternion)

Called by the replay system when the object has been spawned from a prefab instance during playback.

Declaration

```
protected override void OnReplaySpawned(Vector3 position, Quaternion rotation)
```

Parameters

TYPE	NAME	DESCRIPTION
Vector3	position	
Quaternion	rotation	

Overrides

ReplayBehaviour.OnReplaySpawned(Vector3, Quaternion)

OnReplayUpdate(float)

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

Declaration

```
protected override void OnReplayUpdate(float t)
```

Parameters

TYPE	NAME	DESCRIPTION
float	t	A normalized value representing the progress between replay snapshots. Use this value for interpolation or similar smoothing passes

Overrides

ReplayBehaviour.OnReplayUpdate(float)

Reset()

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

Declaration

```
protected override void Reset()
```

Overrides

ReplayBehaviour.Reset()

Implements

IReplaySerialize

Enum ReplayUpdateMode

The update method used by the replay manager for all recording and replaying samples.

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
public enum ReplayUpdateMode
```

Fields

NAME	DESCRIPTION
FixedUpdate	Use the fixed update method.
LateUpdate	Use the late update method.
Manual	The user must manually update the replay operation.
Update	Use the Update method.

Class ReplayVarAttribute

Use this attribute on a field to mark it for recording. The type the field is defined in must inheit 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

object
Attribute
ReplayVarAttribute

Implements

_Attribute

Inherited Members

Attribute.Equals(object)
Attribute.GetCustomAttribute(Assembly, Type)
Attribute.GetCustomAttribute(Assembly, Type, bool)
Attribute.GetCustomAttribute(MemberInfo, Type)
Attribute.GetCustomAttribute(MemberInfo, Type, bool)
Attribute.GetCustomAttribute(Module, Type)
Attribute.GetCustomAttribute(Module, Type, bool)
Attribute.GetCustomAttribute(ParameterInfo, Type)
Attribute.GetCustomAttribute(ParameterInfo, Type, bool)
Attribute.GetCustomAttributes(Assembly)
Attribute.GetCustomAttributes(Assembly, bool)
Attribute.GetCustomAttributes(Assembly, Type)
Attribute.GetCustomAttributes(Assembly, Type, bool)
Attribute.GetCustomAttributes(MemberInfo)
Attribute.GetCustomAttributes(MemberInfo, bool)
Attribute.GetCustomAttributes(MemberInfo, Type)
Attribute.GetCustomAttributes(MemberInfo, Type, bool)
Attribute.GetCustomAttributes(Module)
Attribute.GetCustomAttributes(Module, bool)
Attribute.GetCustomAttributes(Module, Type)
Attribute.GetCustomAttributes(Module, Type, bool)
Attribute.GetCustomAttributes(ParameterInfo)
Attribute.GetCustomAttributes(ParameterInfo, bool)
Attribute.GetCustomAttributes(ParameterInfo, Type)
Attribute.GetCustomAttributes(ParameterInfo, Type, bool)
Attribute.GetHashCode()
Attribute.IsDefaultAttribute()
Attribute.IsDefined(Assembly, Type)
Attribute.IsDefined(Assembly, Type, bool)
Attribute.IsDefined(MemberInfo, Type)
Attribute.IsDefined(MemberInfo, Type, bool)
Attribute.IsDefined(Module, Type)
Attribute.IsDefined(Module, Type, bool)
Attribute.IsDefined(ParameterInfo, Type)
Attribute.IsDefined(ParameterInfo, Type, bool)
Attribute.Match(object)
Attribute.TypeId
object.Equals(object, object)
object.GetType()

object.ReferenceEquals(object, object)
object.ToString()

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
[AttributeUsage(AttributeTargets.Field)]
public sealed class ReplayVarAttribute : Attribute, _Attribute
```

Constructors

ReplayVarAttribute(bool)

Create a new ReplayVarAttribute for a field.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
bool	interpolated	Should the field value be interpolated between frames

Fields

interpolate

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 float. Basic Unity types such as UnityEngine.Vector2 and UnityEngine.Color also support interpolation.

Declaration

```
public bool interpolate
```

Field Value

TYPE	DESCRIPTION
bool	

Implements

_Attribute

Enum RestoreSceneMode

Namespace: UltimateReplay

Assembly: UltimateReplay.dll

Syntax

```
public enum RestoreSceneMode
```

Fields

NAME	DESCRIPTION
KeepState	Do not restore the scene state and keep replay objects in their current state at the time the replay ends. Use this option for rewind time effect for example to keep playing the game from a certain point in the replay.
RestoreState	Restore the scene state to just before the replay started.

Namespace UltimateReplay.ComponentData

Classes

ReplayVariable

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

Structs

ReplayComponentData

Contains all serialized data relating to a specific recorder component.

ReplayEventData

Contains data about a recorded replay event.

ReplayMethodData

Contains data about a serialized method call.

ReplayVariableData

Contains all necessary data to serialize a replay variable with its value.

Struct ReplayComponentData

Contains all serialized data relating to a specific recorder component.

Implements

IReplaySerialize
IReplayTokenSerialize
IDisposable

Inherited Members

ValueType.Equals(object)
ValueType.GetHashCode()
ValueType.ToString()
object.Equals(object, object)
object.GetType()
object.ReferenceEquals(object, object)

Namespace: UltimateReplay.ComponentData

Assembly: UltimateReplay.dll

Syntax

```
public struct ReplayComponentData : IReplaySerialize, IReplayTokenSerialize, IDisposable
```

Constructors

ReplayComponentData(ReplayIdentity, int, ReplayState)

Create a new instance.

Declaration

```
public ReplayComponentData(ReplayIdentity behaviourIdentity, int componentSerializerID, ReplayState
componentStateData)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayIdentity	behaviourIdentity	The identity of the behaviour component
int	componentSerializerID	The id of the component serializer
ReplayState	componentStateData	The data associated with the component

Properties

BehaviourIdentity

The ReplayIdentity of the behaviour script that the data belongs to.

Declaration

```
public ReplayIdentity BehaviourIdentity { get; }
```

Property Value

TYPE	DESCRIPTION
ReplayIdentity	

ComponentSerializerID

An id value used to identify the corrosponding serializer or '-1' if a serializer id could not be generated.

Declaration

```
public int ComponentSerializerID { get; }
```

Property Value

TYPE	DESCRIPTION
int	

ComponentStateData

The ReplayState containing all data that was serialized by the component.

Declaration

```
public ReplayState ComponentStateData { get; }
```

Property Value

TYPE	DESCRIPTION
ReplayState	

Methods

CreateFormatter()

Declaration

```
public ReplayFormatter CreateFormatter()
```

Returns

TYPE	DESCRIPTION
ReplayFormatter	

CreateFormatter<T>()

Declaration

```
public T CreateFormatter<T>() where T : ReplayFormatter
```

Returns

TYPE	DESCRIPTION
T	

Type Parameters

NAME	DESCRIPTION
T	

DeserializeComponent(IReplaySerialize)

Deserialize the component data onto the specified component serializer instance. The specified serialize must be the correct type or have the correct serializer id.

Declaration

```
public bool DeserializeComponent(IReplaySerialize componentSerializer)
```

Parameters

TYPE	NAME	DESCRIPTION
IReplaySerialize	componentSerializer	An IReplaySerialize implementation that should be a correct typed serializer

Returns

TYPE	DESCRIPTION
bool	True if the deserialize was successful or false if not

Dispose()

Release the component data.

Declaration

```
public void Dispose()
```

GetFormatter()

Declaration

```
public ReplayFormatter GetFormatter()
```

Returns

TYPE	DESCRIPTION
ReplayFormatter	

GetFormatter<T>()

Declaration

```
public T GetFormatter<T>() where T : ReplayFormatter
```

Returns

TYPE	DESCRIPTION
T	

Type Parameters

NAME	DESCRIPTION
T	

OnReplayDeserialize(ReplayState)

Deserialize the component data from the specified ReplayState.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The object state to read from

OnReplaySerialize(ReplayState)

Serialize the component data to the specified ReplayState.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The object state to write to

ResolveFormatterType()

Try to resolve the type of the corresponding formatter type.

Declaration

```
public Type ResolveFormatterType()
```

Returns

TYPE	DESCRIPTION
Type	The type of the matching serialize or null if the type could not be resolved

Implements

IReplaySerialize
IReplayTokenSerialize
IDisposable

Struct ReplayEventData

Contains data about a recorded replay event.

Implements

IReplaySerialize

Inherited Members

ValueType.Equals(object)
ValueType.GetHashCode()
ValueType.ToString()
object.Equals(object, object)
object.GetType()
object.ReferenceEquals(object, object)

Namespace: UltimateReplay.ComponentData

Assembly: UltimateReplay.dll

Syntax

```
public struct ReplayEventData : IReplaySerialize
```

Constructors

ReplayEventData(ReplayIdentity, ushort)

Create a new instance.

Declaration

```
public ReplayEventData(ReplayIdentity behaviourIdentity, ushort eventID)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayIdentity	behaviourIdentity	The identity of the behaviour component that recorded the event
ushort	eventID	The unique id used to identify the event

ReplayEventData(ReplayIdentity, ushort, ReplayState)

Create a new instance.

Declaration

```
public ReplayEventData(ReplayIdentity behaviourIdentity, ushort eventID, ReplayState eventState)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayIdentity	behaviourIdentity	The identity of the behaviour component that recorded the event
ushort	eventID	The unique id used to identify the event

TYPE	NAME	DESCRIPTION
ReplayState	eventState	The optional event state data

Properties

BehaviourIdentity

The identity of the behaviour component that recorded the event.

Declaration

```
public ReplayIdentity BehaviourIdentity { get; }
```

Property Value

TYPE	DESCRIPTION
ReplayIdentity	

EventID

The unique event id as generated by the user.

Declaration

```
public ushort EventID { get; }
```

Property Value

TYPE	DESCRIPTION
ushort	

EventState

The optional event state data containing data for the event.

Declaration

```
public ReplayState EventState { get; }
```

Property Value

TYPE	DESCRIPTION
ReplayState	

HasEventState

Returns a value indicating whether the EventState has any data or not.

Declaration

```
public bool HasEventState { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

Methods

OnReplayDeserialize(ReplayState)

Deserialize the event information from the specified ReplayState.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The object state to read from

OnReplaySerialize(ReplayState)

Serialize the event information to the specified ReplayState.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The object state to write to

Implements

IReplaySerialize

Struct ReplayMethodData

Contains data about a serialized method call.

Implements

IReplaySerialize

Inherited Members

ValueType.Equals(object)
ValueType.GetHashCode()
ValueType.ToString()
object.Equals(object, object)
object.GetType()
object.ReferenceEquals(object, object)

Namespace: UltimateReplay.ComponentData

Assembly: UltimateReplay.dll

Syntax

```
public struct ReplayMethodData : IReplaySerialize
```

Constructors

ReplayMethodData(ReplayIdentity, MethodInfo, params object[])

Create a new instance.

Declaration

```
public ReplayMethodData(ReplayIdentity behaviourIdentity, MethodInfo targetMethod, params object[]
methodArguments)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayIdentity	behaviourIdentity	The identity of the behaviour component that recorded the method call
MethodInfo	targetMethod	The target method information
object[]	methodArguments	The argument list for the target method

Properties

BehaviourIdentity

The ReplayIdentity of the replay component that recorded the method call.

Declaration

```
public ReplayIdentity BehaviourIdentity { get; }
```

Property Value

TYPE	DESCRIPTION
ReplayIdentity	

MethodArguments

The method argument values that were passed to the method. Method arguments can only be primitive types such as int.

Declaration

```
public object[] MethodArguments { get; }
```

Property Value

TYPE	DESCRIPTION
object[]	

TargetMethod

The method info for the target recorded method.

Declaration

```
public MethodInfo TargetMethod { get; }
```

Property Value

TYPE	DESCRIPTION
MethodInfo	

Methods

OnReplayDeserialize(ReplayState)

Deserialize the method data from the specified ReplayState.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The object state to read from

OnReplaySerialize(ReplayState)

Serialize the method data to the specified ReplayState.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The object state to write to

Implements

IReplaySerialize

Class ReplayVariable

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

Inheritance

object
ReplayVariable

Implements

IReplaySerialize

Inherited Members

object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: UltimateReplay.ComponentData

Assembly: UltimateReplay.dll

Syntax

```
public sealed class ReplayVariable : IReplaySerialize
```

Constructors

ReplayVariable(ReplayBehaviour, FieldInfo, ReplayVarAttribute)

Create a new ReplayVariable.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayBehaviour	owner	The ReplayBehaviour that this ReplayVariable is defined in
FieldInfo	field	The field info for the variable field
ReplayVarAttribute	attribute	The ReplayVarAttribute for the field

Properties

Attribute

Get the ReplayVarAttribute associated with this ReplayVariable.

Declaration

```
public ReplayVarAttribute Attribute { get; }
```

Property Value

TYPE	DESCRIPTION
ReplayVarAttribute	

Behaviour

Get the ReplayBehaviour that this variable belongs to.

Declaration

```
public ReplayBehaviour Behaviour { get; }
```

Property Value

TYPE	DESCRIPTION
ReplayBehaviour	

FieldOffset

Get the managed field offset value to uniquely identify the variable.

Declaration

```
public int FieldOffset { get; }
```

Property Value

TYPE	DESCRIPTION
int	

IsInterpolated

Returns true if this ReplayVariable should be interpolated between frames.

Declaration

```
public bool IsInterpolated { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

IsInterpolationSupported

Returns true if this ReplayVariable supports interpolation. Interpolation can only be supported if the variable type has a registered interpolator.

Declaration

```
public bool IsInterpolationSupported { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

Name

Get the name of this ReplayVariable.

Declaration

```
public string Name { get; }
```

Property Value

TYPE	DESCRIPTION
string	

Value

The current value for this ReplayVariable.

Declaration

```
public object Value { get; set; }
```

Property Value

TYPE	DESCRIPTION
object	

gameObject

Get the game object that this ReplayVariable is attached to.

Declaration

```
public GameObject gameObject { get; }
```

Property Value

TYPE	DESCRIPTION
GameObject	

Methods

CanInterpolate(Type)

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

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
Type	type	The system type to check for interpolation support

Returns

TYPE	DESCRIPTION
bool	True if interpolation is supported or faluse if it is not

Interpolate(float)

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

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
float	delta	The normalized delta representing the progression from the last frame to the next frame

InterpolateByte(object, object, float)

Default interpolator for byte.

Declaration

```
public static object InterpolateByte(object last, object next, float delta)
```

Parameters

TYPE	NAME	DESCRIPTION
object	last	Last value
object	next	Next value
float	delta	Interpolation delta

Returns

TYPE	DESCRIPTION
object	The interpolated byte value

InterpolateColor(object, object, float)

Default interpolator for Color.

Declaration

```
public static object InterpolateColor(object last, object next, float delta)
```

Parameters

TYPE	NAME	DESCRIPTION
object	last	Last value
object	next	Next value
float	delta	Interpolation delta

Returns

TYPE	DESCRIPTION
object	The interpolated Color value

InterpolateColor32(object, object, float)

Default interpolator for Color32.

Declaration

```
public static object InterpolateColor32(object last, object next, float delta)
```

Parameters

TYPE	NAME	DESCRIPTION
object	last	Last value
object	next	Next value
float	delta	Interpolation delta

Returns

TYPE	DESCRIPTION
object	The interpolated Color32 value

InterpolateDouble(object, object, float)

Default interpolator for double.

Declaration

```
public static object InterpolateDouble(object last, object next, float delta)
```

Parameters

TYPE	NAME	DESCRIPTION
object	last	Last value
object	next	Next value
float	delta	Interpolation delta

Returns

TYPE	DESCRIPTION
object	The interpolated double value

InterpolateFloat(object, object, float)

Default interpolator for float.

Declaration

```
public static object InterpolateFloat(object last, object next, float delta)
```

Parameters

TYPE	NAME	DESCRIPTION
object	last	Last value
object	next	Next value
float	delta	Interpolation delta

Returns

TYPE	DESCRIPTION
object	The interpolated float value

InterpolateInt(object, object, float)

Default interpolator for int.

Declaration

```
public static object InterpolateInt(object last, object next, float delta)
```

Parameters

TYPE	NAME	DESCRIPTION
object	last	Last value
object	next	Next value
float	delta	Interpolation delta

Returns

TYPE	DESCRIPTION
object	The interpolated int value

InterpolateLong(object, object, float)

Default interpolator for long.

Declaration

```
public static object InterpolateLong(object last, object next, float delta)
```

Parameters

TYPE	NAME	DESCRIPTION
object	last	Last value
object	next	Next value
float	delta	Interpolation delta

Returns

TYPE	DESCRIPTION
object	The interpolated long value

InterpolateQuat(object, object, float)

Default interpolator for Quaternion.

Declaration

```
public static object InterpolateQuat(object last, object next, float delta)
```

Parameters

TYPE	NAME	DESCRIPTION
object	last	Last value
object	next	Next value
float	delta	Interpolation delta

Returns

TYPE	DESCRIPTION
object	The interpolated Quaternion value

InterpolateShort(object, object, float)

Default interpolator for short.

Declaration

```
public static object InterpolateShort(object last, object next, float delta)
```

Parameters

TYPE	NAME	DESCRIPTION
object	last	Last value
object	next	Next value
float	delta	Interpolation delta

Returns

TYPE	DESCRIPTION
object	The interpolated short value

InterpolateValue(object, object, float)

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.

Declaration

```
public static object InterpolateValue(object last, object next, float delta)
```

Parameters

TYPE	NAME	DESCRIPTION
object	last	The value of the variable in the last frame
object	next	The value of the variable in the next frame
float	delta	The normalized delta representing the progression from the last frame to the next frame

Returns

TYPE	DESCRIPTION
object	The interpolated value result or null if interpolation is not supported for the type

InterpolateVec2(object, object, float)

Default interpolator for Vector2.

Declaration

```
public static object InterpolateVec2(object last, object next, float delta)
```

Parameters

TYPE	NAME	DESCRIPTION
object	last	Last value
object	next	Next value
float	delta	Interpolation delta

Returns

TYPE	DESCRIPTION
object	The interpolated Vector2 value

InterpolateVec3(object, object, float)

Default interpolator for Vector3.

Declaration

```
public static object InterpolateVec3(object last, object next, float delta)
```

Parameters

TYPE	NAME	DESCRIPTION
object	last	Last value
object	next	Next value
float	delta	Interpolation delta

Returns

TYPE	DESCRIPTION
object	The interpolated Vector3 value

InterpolateVec4(object, object, float)

Default interpolator for Vector4.

Declaration

```
public static object InterpolateVec4(object last, object next, float delta)
```

Parameters

TYPE	NAME	DESCRIPTION
object	last	Last value
object	next	Next value
float	delta	Interpolation delta

Returns

TYPE	DESCRIPTION
object	The interpolated Vector4 value

OnReplayDeserialize(ReplayState)

Called by the replay system when the variable should be deserialized.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState to deserialize the data from

OnReplaySerialize(ReplayState)

Called by the replay system when the variable should be serialized.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState to serialize the data into

RegisterCustomInterpolator<T>(Func<object, object, float, object>)

Allows a custom interpolation method to be registered so that unsupported variable types can be interpolated automatically.

Declaration

```
public static void RegisterCustomInterpolator<T>(Func<object, object, float, object> interpolatorFunc)
```

Parameters

TYPE	NAME	DESCRIPTION
Func<object, object, float, object>	interpolatorFunc	The interpolation method to invoke when interpolation of the custom type is required

Type Parameters

NAME	DESCRIPTION
T	The type of varaible that the custom interpolation should be used for

UpdateValueRange(object, object)

Sets the current interpolation range for the ReplayVariable value.

Declaration

```
public void UpdateValueRange(object last, object next)
```

Parameters

TYPE	NAME	DESCRIPTION
object	last	The value of the variable in the last frame
object	next	The value of the variable in the next frame

Implements

IReplaySerialize

Struct ReplayVariableData

Contains all necessary data to serialize a replay variable with its value.

Implements

IReplaySerialize

Inherited Members

ValueType.Equals(object)
ValueType.GetHashCode()
ValueType.ToString()
object.Equals(object, object)
object.GetType()
object.ReferenceEquals(object, object)

Namespace: UltimateReplay.ComponentData

Assembly: UltimateReplay.dll

Syntax

```
public struct ReplayVariableData : IReplaySerialize
```

Constructors

ReplayVariableData(ReplayIdentity, ReplayVariable)

Create a new variable data instance.

Declaration

```
public ReplayVariableData(ReplayIdentity behaviourIdentity, ReplayVariable variable)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayIdentity	behaviourIdentity	The ReplayIdentity of the owning behaviour
ReplayVariable	variable	The ReplayVariable instance

Properties

BehaviourIdentity

The ReplayIdentity of the ReplayBehaviour that the variable belongs to.

Declaration

```
public ReplayIdentity BehaviourIdentity { get; }
```

Property Value

TYPE	DESCRIPTION
ReplayIdentity	

VariableFieldOffset

The field offset used to uniqluey identify the variable.

Declaration

```
public int VariableFieldOffset { get; }
```

Property Value

TYPE	DESCRIPTION
int	

VariableStateData

The ReplayState containing the variable value.

Declaration

```
public ReplayState VariableStateData { get; }
```

Property Value

TYPE	DESCRIPTION
ReplayState	

Methods

IsMatchedToVariable(ReplayVariable)

Returns a value indicating whther the specified ReplayVariable corrosponds to this variable data.

Declaration

```
public bool IsMatchedToVariable(ReplayVariable variable)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayVariable	variable	The ReplayVariable instance to check

Returns

TYPE	DESCRIPTION
bool	True if the variable data targets the specified variable instance or false if not

OnReplayDeserialize(ReplayState)

Deserialize the variable data from the specified ReplayState.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The object state to read from

OnReplaySerialize(ReplayState)

Serialize the variable data to the specified ReplayState.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The object state to write to

ResolveAndDeserializeVariable(ReplayObject)

Try to resolve and deserialize the variable data for the specified ReplayObject. This will attempt to find the target variable on one of the observed components and will deserialize and update that variable if found.

Declaration

```
public bool ResolveAndDeserializeVariable(ReplayObject tagretObject)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayObject	tagretObject	The ReplayObject to try and resolve

Returns

TYPE	DESCRIPTION
bool	True if the variable was found and updated or false if not

Implements

IReplaySerialize

Namespace UltimateReplay.Formatters

Classes

ReplayAnimatorFormatter

ReplayAudioFormatter

ReplayBlendShapeFormatter

ReplayEnabledStateFormatter

A dedicated formatter used to serialize and deserialize data for the ReplayEnabledState component.

ReplayFormatter

ReplayObjectFormatter

ReplayParentChangeFormatter

ReplayRiggedGenericFormatter

ReplayRiggedHumanoidFormatter

ReplayTransformFormatter

Structs

ReplayAnimatorFormatter.ReplayAnimatorIKTarget

Contains data about a specific animator IK limb.

ReplayAnimatorFormatter.ReplayAnimatorParameter

Contains data about a specific animator parameter.

ReplayAnimatorFormatter.ReplayAnimatorState

Contains data about a specific animator state.

Enums

ReplayAnimatorFormatter.ReplayAnimatorSerializeFlags

Serialize flags used to indicate which data elements are stored.

Class ReplayAnimatorFormatter

Inheritance

object
ReplayFormatter
ReplayAnimatorFormatter

Implements

IReplaySerialize
IReplayTokenSerialize

Inherited Members

ReplayFormatter.FormatterId
ReplayFormatter.CreateFormatter(byte)
ReplayFormatter.GetFormatter(byte)
ReplayFormatter.CreateFormatter<T>(byte)
ReplayFormatter.GetFormatter<T>(byte)
ReplayFormatter.GetFormatterType(byte)
ReplayFormatter.GetFormatterOfType<T>()
object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: UltimateReplay.Formatters

Assembly: UltimateReplay.dll

Syntax

```
public sealed class ReplayAnimatorFormatter : ReplayFormatter, IReplaySerialize, IReplayTokenSerialize
```

Fields

IKLimbCount

Declaration

```
public const int IKLimbCount = 4
```

Field Value

TYPE	DESCRIPTION
int	

Properties

IKTargets

Get all ReplayAnimatorFormatter.ReplayAnimatorIKTarget that will be serialized.

Declaration

```
public ReplayAnimatorFormatter.ReplayAnimatorIKTarget[] IKTargets { get; set; }
```

Property Value

TYPE	DESCRIPTION
ReplayAnimatorIKTarget[]	

LowPrecision

Declaration

```
public bool LowPrecision { get; set; }
```

Property Value

TYPE	DESCRIPTION
bool	

MainState

Get the ReplayAnimatorFormatter.ReplayAnimatorState information for the main state.

Declaration

```
public ReplayAnimatorFormatter.ReplayAnimatorState MainState { get; set; }
```

Property Value

TYPE	DESCRIPTION
ReplayAnimatorFormatter.ReplayAnimatorState	

Parameters

Get all ReplayAnimatorFormatter.ReplayAnimatorParameter that will be serialized.

Declaration

```
public ReplayAnimatorFormatter.ReplayAnimatorParameter[] Parameters { get; set; }
```

Property Value

TYPE	DESCRIPTION
ReplayAnimatorParameter[]	

ReplayParameters

Declaration

```
public bool ReplayParameters { get; set; }
```

Property Value

TYPE	DESCRIPTION
bool	

States

Get all ReplayAnimatorFormatter.ReplayAnimatorState information for all sub states.

Declaration

```
public ReplayAnimatorFormatter.ReplayAnimatorState[] States { get; set; }
```

Property Value

TYPE	DESCRIPTION
ReplayAnimatorState[]	

Methods

GetIKTargetInfo(AvatarIKGoal)

Declaration

```
public ReplayAnimatorFormatter.ReplayAnimatorIKTarget GetIKTargetInfo(AvatarIKGoal goal)
```

Parameters

TYPE	NAME	DESCRIPTION
AvatarIKGoal	goal	

Returns

TYPE	DESCRIPTION
ReplayAnimatorFormatter.ReplayAnimatorIKTarget	

OnReplayDeserialize(ReplayState)

Invoke this method to deserialize the animator data from the specified ReplayState.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The state object which should contain valid animator data

Overrides

ReplayFormatter.OnReplayDeserialize(ReplayState)

OnReplaySerialize(ReplayState)

Invoke this method to serialize the animator data to the specified ReplayState.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The state object used to store the data

Overrides

ReplayFormatter.OnReplaySerialize(ReplayState)

SetIKTargetInfo(AvatarIKGoal, in ReplayAnimatorIKTarget)

Declaration

```
public void SetIKTargetInfo(AvatarIKGoal goal, in ReplayAnimatorFormatter.ReplayAnimatorIKTarget target)
```

Parameters

TYPE	NAME	DESCRIPTION
AvatarIKGoal	goal	
ReplayAnimatorFormatter.ReplayAnimatorIKTarget	target	

Implements

IReplaySerialize
IReplayTokenSerialize

Struct ReplayAnimatorFormatter.ReplayAnimatorIKTarget

Contains data about a specific animator IK limb.

Inherited Members

ValueType.Equals(object)
ValueType.GetHashCode()
ValueType.ToString()
object.Equals(object, object)
object.GetType()
object.ReferenceEquals(object, object)

Namespace: UltimateReplay.Formatters

Assembly: UltimateReplay.dll

Syntax

```
public struct ReplayAnimatorFormatter.ReplayAnimatorIKTarget
```

Fields

positionWeight

The position weight for the IK limb.

Declaration

```
public float positionWeight
```

Field Value

TYPE	DESCRIPTION
float	

rotationWeight

The rotation weight for the IK limb.

Declaration

```
public float rotationWeight
```

Field Value

TYPE	DESCRIPTION
float	

target

The target IK limb.

Declaration

```
public AvatarIKGoal target
```

Field Value

TYPE	DESCRIPTION
AvatarIKGoal	

targetPosition

The target position for the IK limb.

Declaration

```
public Vector3 targetPosition
```

Field Value

TYPE	DESCRIPTION
Vector3	

targetRotation

The target rotation for the IK limb.

Declaration

```
public Quaternion targetRotation
```

Field Value

TYPE	DESCRIPTION
Quaternion	

Struct ReplayAnimatorFormatter.ReplayAnimatorParameter

Contains data about a specific animator parameter.

Inherited Members

ValueType.Equals(object)
ValueType.GetHashCode()
ValueType.ToString()
object.Equals(object, object)
object.GetType()
object.ReferenceEquals(object, object)

Namespace: UltimateReplay.Formatters

Assembly: UltimateReplay.dll

Syntax

```
public struct ReplayAnimatorFormatter.ReplayAnimatorParameter
```

Fields

boolValue

The bool value of the parameter.

Declaration

```
public bool boolValue
```

Field Value

TYPE	DESCRIPTION
bool	

floatValue

The float value of the parameter.

Declaration

```
public float floatValue
```

Field Value

TYPE	DESCRIPTION
float	

intValue

The integer value of the parameter.

Declaration

```
public int intValue
```

Field Value

TYPE	DESCRIPTION
int	

nameHash

The name hash of the parameter.

Declaration

```
public int nameHash
```

Field Value

TYPE	DESCRIPTION
int	

parameterType

The UnityEngine.AnimatorControllerParameterType which describes the type of parameter.

Declaration

```
public AnimatorControllerParameterType parameterType
```

Field Value

TYPE	DESCRIPTION
AnimatorControllerParameterType	

Enum ReplayAnimatorFormatter.ReplayAnimatorSerializeFlags

Serialize flags used to indicate which data elements are stored.

Namespace: UltimateReplay.Formatters

Assembly: UltimateReplay.dll

Syntax

```
[Flags]
public enum ReplayAnimatorFormatter.ReplayAnimatorSerializeFlags : byte
```

Fields

NAME	DESCRIPTION
IKPosition	
IKRotation	
IKWeights	
LowPrecision	Supported data elements will be serialized using low precision mode.
MainState	The main state layer data will be serialized.
Parameters	Parameter values will be serialized.
SubStates	Sub state layers will be serialized.

Struct ReplayAnimatorFormatter.ReplayAnimatorState

Contains data about a specific animator state.

Inherited Members

ValueType.Equals(object)
ValueType.GetHashCode()
ValueType.ToString()
object.Equals(object, object)
object.GetType()
object.ReferenceEquals(object, object)

Namespace: UltimateReplay.Formatters

Assembly: UltimateReplay.dll

Syntax

```
public struct ReplayAnimatorFormatter.ReplayAnimatorState
```

Fields

normalizedTime

The normalized playback time of the current animation.

Declaration

```
public float normalizedTime
```

Field Value

TYPE	DESCRIPTION
float	

speed

The current speed of the animation.

Declaration

```
public float speed
```

Field Value

TYPE	DESCRIPTION
float	

speedMultiplier

The current speed multiplier value.

Declaration

```
public float speedMultiplier
```

Field Value

TYPE	DESCRIPTION
float	

stateHash

The hash of the current animator state.

Declaration

```
public int stateHash
```

Field Value

TYPE	DESCRIPTION
int	

Class ReplayAudioFormatter

Inheritance

object
ReplayFormatter
ReplayAudioFormatter

Implements

IReplaySerialize
IReplayTokenSerialize

Inherited Members

ReplayFormatter.FormatterId
ReplayFormatter.CreateFormatter(byte)
ReplayFormatter.GetFormatter(byte)
ReplayFormatter.CreateFormatter<T>(byte)
ReplayFormatter.GetFormatter<T>(byte)
ReplayFormatter.GetFormatterType(byte)
ReplayFormatter.GetFormatterOfType<T>()
object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: UltimateReplay.Formatters

Assembly: UltimateReplay.dll

Syntax

```
public sealed class ReplayAudioFormatter : ReplayFormatter, IReplaySerialize, IReplayTokenSerialize
```

Properties

IsPlaying

Declaration

```
public bool IsPlaying { get; set; }
```

Property Value

TYPE	DESCRIPTION
bool	

Pitch

Declaration

```
public float Pitch { get; }
```

Property Value

TYPE	DESCRIPTION
float	

ReverbZoneMix

Declaration

```
public float ReverbZoneMix { get; }
```

Property Value

TYPE	DESCRIPTION
float	

SpatialBlend

Declaration

```
public float SpatialBlend { get; }
```

Property Value

TYPE	DESCRIPTION
float	

StereoPan

Declaration

```
public float StereoPan { get; }
```

Property Value

TYPE	DESCRIPTION
float	

TimeSample

Declaration

```
public int TimeSample { get; }
```

Property Value

TYPE	DESCRIPTION
int	

Volume

Declaration

```
public float Volume { get; }
```

Property Value

TYPE	DESCRIPTION
float	

Methods

OnReplayDeserialize(ReplayState)

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

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState to read the data from

Overrides

ReplayFormatter.OnReplayDeserialize(ReplayState)

OnReplaySerialize(ReplayState)

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

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState to write the data to

Overrides

ReplayFormatter.OnReplaySerialize(ReplayState)

Implements

IReplaySerialize
IReplayTokenSerialize

Class ReplayBlendShapeFormatter

Inheritance

object
ReplayBlendShapeFormatter

Inherited Members

object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: UltimateReplay.Formatters

Assembly: UltimateReplay.dll

Syntax

```
public sealed class ReplayBlendShapeFormatter
```

Properties

BlendWeights

Declaration

```
public IList<float> BlendWeights { get; }
```

Property Value

TYPE	DESCRIPTION
IList<float>	

Methods

OnReplayDeserialize(ReplayState)

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	

OnReplaySerialize(ReplayState)

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	

SyncSkinnedRenderer(SkinnedMeshRenderer)

Declaration

```
public void SyncSkinnedRenderer(SkinnedMeshRenderer sync)
```

Parameters

TYPE	NAME	DESCRIPTION
SkinnedMeshRenderer	sync	

UpdateFromSkinnedRenderer(SkinnedMeshRenderer)

Declaration

```
public void UpdateFromSkinnedRenderer(SkinnedMeshRenderer from)
```

Parameters

TYPE	NAME	DESCRIPTION
SkinnedMeshRenderer	from	

Class ReplayEnabledStateFormatter

A dedicated formatter used to serialize and deserialize data for the ReplayEnabledState component.

Inheritance

object
ReplayFormatter
ReplayEnabledStateFormatter

Implements

IReplaySerialize
IReplayTokenSerialize

Inherited Members

ReplayFormatter.FormatterId
ReplayFormatter.CreateFormatter(byte)
ReplayFormatter.GetFormatter(byte)
ReplayFormatter.CreateFormatter<T>(byte)
ReplayFormatter.GetFormatter<T>(byte)
ReplayFormatter.GetFormatterType(byte)
ReplayFormatter.GetFormatterOfType<T>()
object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: UltimateReplay.Formatters

Assembly: UltimateReplay.dll

Syntax

```
public sealed class ReplayEnabledStateFormatter : ReplayFormatter, IReplaySerialize, IReplayTokenSerialize
```

Properties

Enabled

The enabled state of the object.

Declaration

```
public bool Enabled { get; set; }
```

Property Value

TYPE	DESCRIPTION
bool	

Methods

OnReplayDeserialize(ReplayState)

Invoke this method to deserialize the enabled state from the specified ReplayState.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The state object to read from

Overrides

ReplayFormatter.OnReplayDeserialize(ReplayState)

OnReplaySerialize(ReplayState)

Invoke this method to serialize the enabled state data to the specified ReplayState.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The state object to write to

Overrides

ReplayFormatter.OnReplaySerialize(ReplayState)

SyncBehaviour(Behaviour)

Declaration

```
public void SyncBehaviour(Behaviour sync)
```

Parameters

TYPE	NAME	DESCRIPTION
Behaviour	sync	

SyncGameObject(GameObject)

Declaration

```
public void SyncGameObject(GameObject sync)
```

Parameters

TYPE	NAME	DESCRIPTION
GameObject	sync	

UpdateFromBehaviour(Behaviour)

Declaration

```
public void UpdateFromBehaviour(Behaviour from)
```

Parameters

TYPE	NAME	DESCRIPTION
Behaviour	from	

UpdateFromGameObject(GameObject)

Declaration

```
public void UpdateFromGameObject(GameObject from)
```

Parameters

TYPE	NAME	DESCRIPTION
GameObject	from	

Implements

IReplaySerialize
IReplayTokenSerialize

Class ReplayFormatter

Inheritance

object
ReplayFormatter
ReplayAnimatorFormatter
ReplayAudioFormatter
ReplayEnabledStateFormatter
ReplayObjectFormatter
ReplayParentChangeFormatter
ReplayRiggedGenericFormatter
ReplayRiggedHumanoidFormatter
ReplayTransformFormatter

Implements

IReplaySerialize
IReplayTokenSerialize

Inherited Members

object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: UltimateReplay.Formatters

Assembly: UltimateReplay.dll

Syntax

```
public abstract class ReplayFormatter : IReplaySerialize, IReplayTokenSerialize
```

Constructors

ReplayFormatter()

Declaration

```
protected ReplayFormatter()
```

Properties

FormatterId

Declaration

```
public byte FormatterId { get; }
```

Property Value

TYPE	DESCRIPTION
byte	

Methods

CreateFormatter(byte)

Declaration

```
public static ReplayFormatter CreateFormatter(byte formatterId)
```

Parameters

TYPE	NAME	DESCRIPTION
byte	formatterId	

Returns

TYPE	DESCRIPTION
ReplayFormatter	

CreateFormatter<T>(byte)

Declaration

```
public static T CreateFormatter<T>(byte formatterId) where T : ReplayFormatter
```

Parameters

TYPE	NAME	DESCRIPTION
byte	formatterId	

Returns

TYPE	DESCRIPTION
T	

Type Parameters

NAME	DESCRIPTION
T	

GetFormatter(byte)

Declaration

```
public static ReplayFormatter GetFormatter(byte formatterId)
```

Parameters

TYPE	NAME	DESCRIPTION
byte	formatterId	

Returns

TYPE	DESCRIPTION
ReplayFormatter	

GetFormatterOfType<T>()

Declaration

```
public static T GetFormatterOfType<T>() where T : ReplayFormatter
```

Returns

TYPE	DESCRIPTION
T	

Type Parameters

NAME	DESCRIPTION
T	

GetFormatterType(byte)

Declaration

```
public static Type GetFormatterType(byte formatterId)
```

Parameters

TYPE	NAME	DESCRIPTION
byte	formatterId	

Returns

TYPE	DESCRIPTION
Type	

GetFormatter<T>(byte)

Declaration

```
public static T GetFormatter<T>(byte formatterId) where T : ReplayFormatter
```

Parameters

TYPE	NAME	DESCRIPTION
byte	formatterId	

Returns

TYPE	DESCRIPTION
T	

Type Parameters

NAME	DESCRIPTION
T	

OnReplayDeserialize(ReplayState)

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

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState to read the data from

OnReplaySerialize(ReplayState)

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

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState to write the data to

Implements

IReplaySerialize
IReplayTokenSerialize

Class ReplayObjectFormatter

Inheritance

object
ReplayFormatter
ReplayObjectFormatter

Implements

IReplaySerialize
IReplayTokenSerialize

Inherited Members

ReplayFormatter.FormatterId
ReplayFormatter.CreateFormatter(byte)
ReplayFormatter.GetFormatter(byte)
ReplayFormatter.CreateFormatter<T>(byte)
ReplayFormatter.GetFormatter<T>(byte)
ReplayFormatter.GetFormatterType(byte)
ReplayFormatter.GetFormatterOfType<T>()
object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: UltimateReplay.Formatters

Assembly: UltimateReplay.dll

Syntax

```
public sealed class ReplayObjectFormatter : ReplayFormatter, IReplaySerialize, IReplayTokenSerialize
```

Properties

ComponentStates

A collection of ReplayComponentData containing all the necessary persistent data for all observed components.

Declaration

```
public IList<ReplayComponentData> ComponentStates { get; }
```

Property Value

TYPE	DESCRIPTION
IList<ReplayComponentData>	

EventStates

A collection of ReplayEventData containing all the necessary persistent data for all recorded events.

Declaration

```
public IList<ReplayEventData> EventStates { get; }
```

Property Value

TYPE	DESCRIPTION
IList<ReplayEventData>	

MethodStates

A collection of ReplayMethodData containing all the necessary persistent data for all recorded methods.

Declaration

```
public IList<ReplayMethodData> MethodStates { get; }
```

Property Value

TYPE	DESCRIPTION
IList<ReplayMethodData>	

PrefabIdentity

The ReplayIdentity of the parent prefab if applicable.

Declaration

```
public ReplayIdentity PrefabIdentity { get; set; }
```

Property Value

TYPE	DESCRIPTION
ReplayIdentity	

VariableStates

A collection of ReplayVariableData containing all the necessary persistent data for all recorded variables.

Declaration

```
public IList<ReplayVariableData> VariableStates { get; }
```

Property Value

TYPE	DESCRIPTION
IList<ReplayVariableData>	

Methods

OnReplayDeserialize(ReplayState)

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

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState to read the data from

Overrides

ReplayFormatter.OnReplayDeserialize(ReplayState)

OnReplayDeserialize(ReplayState, bool)

Declaration

```
public void OnReplayDeserialize(ReplayState state, bool simulate)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	
bool	simulate	

OnReplaySerialize(ReplayState)

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

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState to write the data to

Overrides

ReplayFormatter.OnReplaySerialize(ReplayState)

Implements

IReplaySerialize
IReplayTokenSerialize

Class ReplayParentChangeFormatter

Inheritance

object
ReplayFormatter
ReplayParentChangeFormatter

Implements

IReplaySerialize
IReplayTokenSerialize

Inherited Members

ReplayFormatter.FormatterId
ReplayFormatter.CreateFormatter(byte)
ReplayFormatter.GetFormatter(byte)
ReplayFormatter.CreateFormatter<T>(byte)
ReplayFormatter.GetFormatter<T>(byte)
ReplayFormatter.GetFormatterType(byte)
ReplayFormatter.GetFormatterOfType<T>()
object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: UltimateReplay.Formatters

Assembly: UltimateReplay.dll

Syntax

```
public sealed class ReplayParentChangeFormatter : ReplayFormatter, IReplaySerialize, IReplayTokenSerialize
```

Properties

HasParent

Declaration

```
public bool HasParent { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

ParentIdentity

Declaration

```
public ReplayIdentity ParentIdentity { get; set; }
```

Property Value

TYPE	DESCRIPTION
ReplayIdentity	

Methods

OnReplayDeserialize(ReplayState)

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

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState to read the data from

Overrides

ReplayFormatter.OnReplayDeserialize(ReplayState)

OnReplaySerialize(ReplayState)

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

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState to write the data to

Overrides

ReplayFormatter.OnReplaySerialize(ReplayState)

SyncTransform(Transform, ReplayScene)

Declaration

```
public void SyncTransform(Transform sync, ReplayScene scene)
```

Parameters

TYPE	NAME	DESCRIPTION
Transform	sync	
ReplayScene	scene	

UpdateFromTransform(Transform)

Declaration

```
public void UpdateFromTransform(Transform from)
```

Parameters

TYPE	NAME	DESCRIPTION
Transform	from	

Implements

IReplaySerialize
IReplayTokenSerialize

Class ReplayRiggedGenericFormatter

Inheritance

object
ReplayFormatter
ReplayRiggedGenericFormatter

Implements

IReplaySerialize
IReplayTokenSerialize

Inherited Members

ReplayFormatter.FormatterId
ReplayFormatter.CreateFormatter(byte)
ReplayFormatter.GetFormatter(byte)
ReplayFormatter.CreateFormatter<T>(byte)
ReplayFormatter.GetFormatter<T>(byte)
ReplayFormatter.GetFormatterType(byte)
ReplayFormatter.GetFormatterOfType<T>()
object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: UltimateReplay.Formatters

Assembly: UltimateReplay.dll

Syntax

```
public sealed class ReplayRiggedGenericFormatter : ReplayFormatter, IReplaySerialize, IReplayTokenSerialize
```

Properties

BoneCount

Declaration

```
public int BoneCount { get; }
```

Property Value

TYPE	DESCRIPTION
int	

BonePositionAxis

Declaration

```
public RecordAxisFlags BonePositionAxis { get; set; }
```

Property Value

TYPE	DESCRIPTION
RecordAxisFlags	

BonePositionPrecision

Declaration

```
public RecordPrecision BonePositionPrecision { get; set; }
```

Property Value

TYPE	DESCRIPTION
RecordPrecision	

BoneRotationAxis

Declaration

```
public RecordAxisFlags BoneRotationAxis { get; set; }
```

Property Value

TYPE	DESCRIPTION
RecordAxisFlags	

BoneRotationPrecision

Declaration

```
public RecordPrecision BoneRotationPrecision { get; set; }
```

Property Value

TYPE	DESCRIPTION
RecordPrecision	

BoneScaleAxis

Declaration

```
public RecordAxisFlags BoneScaleAxis { get; set; }
```

Property Value

TYPE	DESCRIPTION
RecordAxisFlags	

BoneScalePrecision

Declaration

```
public RecordPrecision BoneScalePrecision { get; set; }
```

Property Value

TYPE	DESCRIPTION
RecordPrecision	

RootPosition

Declaration

```
public Vector3 RootPosition { get; }
```

Property Value

TYPE	DESCRIPTION
Vector3	

RootPositionAxis

Declaration

```
public RecordAxisFlags RootPositionAxis { get; set; }
```

Property Value

TYPE	DESCRIPTION
RecordAxisFlags	

RootPositionPrecision

Declaration

```
public RecordPrecision RootPositionPrecision { get; set; }
```

Property Value

TYPE	DESCRIPTION
RecordPrecision	

RootRotation

Declaration

```
public Quaternion RootRotation { get; }
```

Property Value

TYPE	DESCRIPTION
Quaternion	

RootRotationAxis

Declaration

```
public RecordAxisFlags RootRotationAxis { get; set; }
```

Property Value

TYPE	DESCRIPTION
RecordAxisFlags	

RootRotationPrecision

Declaration

```
public RecordPrecision RootRotationPrecision { get; set; }
```

Property Value

TYPE	DESCRIPTION
RecordPrecision	

RootScale

Declaration

```
public Vector3 RootScale { get; }
```

Property Value

TYPE	DESCRIPTION
Vector3	

RootScaleAxis

Declaration

```
public RecordAxisFlags RootScaleAxis { get; set; }
```

Property Value

TYPE	DESCRIPTION
RecordAxisFlags	

Methods

GetBonePosition(int)

Declaration

```
public Vector3 GetBonePosition(int index)
```

Parameters

TYPE	NAME	DESCRIPTION
int	index	

Returns

TYPE	DESCRIPTION
Vector3	

GetBoneRotation(int)

Declaration

```
public Quaternion GetBoneRotation(int index)
```

Parameters

TYPE	NAME	DESCRIPTION
int	index	

Returns

TYPE	DESCRIPTION
Quaternion	

GetBoneScale(int)

Declaration

```
public Vector3 GetBoneScale(int index)
```

Parameters

TYPE	NAME	DESCRIPTION
int	index	

Returns

TYPE	DESCRIPTION
Vector3	

GetBoneTransform(int, out Vector3, out Quaternion, out Vector3)

Declaration

```
public void GetBoneTransform(int index, out Vector3 position, out Quaternion rotation, out Vector3 scale)
```

Parameters

TYPE	NAME	DESCRIPTION
int	index	
Vector3	position	
Quaternion	rotation	
Vector3	scale	

OnReplayDeserialize(ReplayState)

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

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState to read the data from

Overrides

ReplayFormatter.OnReplayDeserialize(ReplayState)

OnReplaySerialize(ReplayState)

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

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState to write the data to

Overrides

ReplayFormatter.OnReplaySerialize(ReplayState)

Implements

IReplaySerialize
IReplayTokenSerialize

Class ReplayRiggedHumanoidFormatter

Inheritance

object
ReplayFormatter
ReplayRiggedHumanoidFormatter

Implements

IReplaySerialize
IReplayTokenSerialize

Inherited Members

ReplayFormatter.FormatterId
ReplayFormatter.CreateFormatter(byte)
ReplayFormatter.GetFormatter(byte)
ReplayFormatter.CreateFormatter<T>(byte)
ReplayFormatter.GetFormatter<T>(byte)
ReplayFormatter.GetFormatterType(byte)
ReplayFormatter.GetFormatterOfType<T>()
object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: UltimateReplay.Formatters

Assembly: UltimateReplay.dll

Syntax

```
public sealed class ReplayRiggedHumanoidFormatter : ReplayFormatter, IReplaySerialize, IReplayTokenSerialize
```

Properties

BodyPosition

Declaration

```
public Vector3 BodyPosition { get; }
```

Property Value

TYPE	DESCRIPTION
Vector3	

BodyRotation

Declaration

```
public Quaternion BodyRotation { get; }
```

Property Value

TYPE	DESCRIPTION
Quaternion	

MuscleValues

Declaration

```
public IReadOnlyList<float> MuscleValues { get; }
```

Property Value

TYPE	DESCRIPTION
IReadOnlyList<float>	

Methods

OnReplayDeserialize(ReplayState)

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

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState to read the data from

Overrides

ReplayFormatter.OnReplayDeserialize(ReplayState)

OnReplaySerialize(ReplayState)

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

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState to write the data to

Overrides

ReplayFormatter.OnReplaySerialize(ReplayState)

Implements

IReplaySerialize
IReplayTokenSerialize

Class ReplayTransformFormatter

Inheritance

object
ReplayFormatter
ReplayTransformFormatter

Implements

IReplaySerialize
IReplayTokenSerialize

Inherited Members

ReplayFormatter.FormatterId
ReplayFormatter.CreateFormatter(byte)
ReplayFormatter.GetFormatter(byte)
ReplayFormatter.CreateFormatter<T>(byte)
ReplayFormatter.GetFormatter<T>(byte)
ReplayFormatter.GetFormatterType(byte)
ReplayFormatter.GetFormatterOfType<T>()
object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: UltimateReplay.Formatters

Assembly: UltimateReplay.dll

Syntax

```
public sealed class ReplayTransformFormatter : ReplayFormatter, IReplaySerialize, IReplayTokenSerialize
```

Properties

Position

Declaration

```
public Vector3 Position { get; }
```

Property Value

TYPE	DESCRIPTION
Vector3	

PositionAxis

Declaration

```
public RecordAxisFlags PositionAxis { get; set; }
```

Property Value

TYPE	DESCRIPTION
RecordAxisFlags	

PositionPrecision

Declaration

```
public RecordPrecision PositionPrecision { get; set; }
```

Property Value

TYPE	DESCRIPTION
RecordPrecision	

PositionSpace

Declaration

```
public RecordSpace PositionSpace { get; set; }
```

Property Value

TYPE	DESCRIPTION
RecordSpace	

Rotation

Declaration

```
public Quaternion Rotation { get; }
```

Property Value

TYPE	DESCRIPTION
Quaternion	

RotationAxis

Declaration

```
public RecordAxisFlags RotationAxis { get; set; }
```

Property Value

TYPE	DESCRIPTION
RecordAxisFlags	

RotationPrecision

Declaration

```
public RecordPrecision RotationPrecision { get; set; }
```

Property Value

TYPE	DESCRIPTION
RecordPrecision	

RotationSpace

Declaration

```
public RecordSpace RotationSpace { get; set; }
```

Property Value

TYPE	DESCRIPTION
RecordSpace	

Scale

Declaration

```
public Vector3 Scale { get; }
```

Property Value

TYPE	DESCRIPTION
Vector3	

ScaleAxis

Declaration

```
public RecordAxisFlags ScaleAxis { get; set; }
```

Property Value

TYPE	DESCRIPTION
RecordAxisFlags	

ScalePrecision

Declaration

```
public RecordPrecision ScalePrecision { get; set; }
```

Property Value

TYPE	DESCRIPTION
RecordPrecision	

Methods

OnReplayDeserialize(ReplayState)

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

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState to read the data from

Overrides

ReplayFormatter.OnReplayDeserialize(ReplayState)

OnReplaySerialize(ReplayState)

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

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState to write the data to

Overrides

ReplayFormatter.OnReplaySerialize(ReplayState)

SyncTransform(Transform)

Declaration

```
public void SyncTransform(Transform sync)
```

Parameters

TYPE	NAME	DESCRIPTION
Transform	sync	

SyncTransformPosition(Transform)

Declaration

```
public void SyncTransformPosition(Transform sync)
```

Parameters

TYPE	NAME	DESCRIPTION
Transform	sync	

SyncTransformRotation(Transform)

Declaration

```
public void SyncTransformRotation(Transform sync)
```

Parameters

TYPE	NAME	DESCRIPTION
Transform	sync	

SyncTransformScale(Transform)

Declaration

```
public void SyncTransformScale(Transform sync)
```

Parameters

TYPE	NAME	DESCRIPTION
Transform	sync	

UpdateFromTransform(Transform, bool)

Declaration

```
public void UpdateFromTransform(Transform from, bool includeScale = false)
```

Parameters

TYPE	NAME	DESCRIPTION
Transform	from	
bool	includeScale	

UpdateFromTransform(Transform, RecordAxisFlags, RecordAxisFlags, RecordAxisFlags, RecordSpace, RecordSpace, RecordPrecision, RecordPrecision, RecordPrecision)

Declaration

```
public void UpdateFromTransform(Transform from, RecordAxisFlags position, RecordAxisFlags rotation,
RecordAxisFlags scale = RecordAxisFlags.None, RecordSpace positionSpace = RecordSpace.World, RecordSpace
rotationSpace = RecordSpace.World, RecordPrecision positionPrecision = RecordPrecision.FullPrecision32Bit,
RecordPrecision rotationPrecision = RecordPrecision.FullPrecision32Bit, RecordPrecision scalePrecision =
RecordPrecision.FullPrecision32Bit)
```

Parameters

TYPE	NAME	DESCRIPTION
Transform	from	
RecordAxisFlags	position	
RecordAxisFlags	rotation	
RecordAxisFlags	scale	
RecordSpace	positionSpace	
RecordSpace	rotationSpace	
RecordPrecision	positionPrecision	

TYPE	NAME	DESCRIPTION
RecordPrecision	rotationPrecision	
RecordPrecision	scalePrecision	

Implements

IReplaySerialize
IReplayTokenSerialize

Namespace UltimateReplay.Lifecycle

Classes

ReplayInstancePool<T>

An instance pool used to recycle managed non-Unity objects.

ReplayObjectCustomLifecycleProvider

ReplayObjectDefaultLifecycleProvider

ReplayObjectLifecycleProvider

ReplayObjectResourcesLifecycleProvider

Interfaces

IReplayReusable

Used to initialize existing and reused instances in conjunction with the ReplayInstancePool<T>.

Interface IReplayReusable

Used to initialize existing and reused instances in conjunction with the ReplayInstancePool<T>.

Namespace: UltimateReplay.Lifecycle

Assembly: UltimateReplay.dll

Syntax

```
public interface IReplayReusable
```

Methods

Initialize()

Called when an existing instance is about to be returned from the pool. This method should reset any field members to default or safe values.

Declaration

```
void Initialize()
```

Class ReplayInstancePool<T>

An instance pool used to recycle managed non-Unity objects.

Inheritance

object
ReplayInstancePool<T>

Inherited Members

object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: UltimateReplay.Lifecycle

Assembly: UltimateReplay.dll

Syntax

```
public sealed class ReplayInstancePool<T>
```

Type Parameters

NAME	DESCRIPTION
T	The type of object to manage

Methods

GetReusable()

Get an existing recycled instance or create a new instance if required. Instances which implement the IReplayReusable interface will have the Initialize() method called if a recycled instance is used.

Declaration

```
public T GetReusable()
```

Returns

TYPE	DESCRIPTION
T	An instance of T

PushReusable(T)

Return an existing instance to the pool which is no longer required.

Declaration

```
public void PushReusable(T reusableInstance)
```

Parameters

TYPE	NAME	DESCRIPTION
T	reusableInstance	The T instance to return to the pool

Class ReplayObjectCustomLifecycleProvider

Inheritance

object
Object
ScriptableObject
ReplayObjectLifecycleProvider
ReplayObjectCustomLifecycleProvider

Inherited Members

ReplayObjectLifecycleProvider.DestroyReplayObject(ReplayObject)
ScriptableObject.SetDirty()
ScriptableObject.CreateInstance(string)
ScriptableObject.CreateInstance(Type)
ScriptableObject.CreateInstance<T>()
Object.GetInstanceID()
Object.GetHashCode()
Object.Equals(object)
Object.Instantiate(Object, Vector3, Quaternion)
Object.Instantiate(Object, Vector3, Quaternion, Transform)
Object.Instantiate(Object)
Object.Instantiate(Object, Transform)
Object.Instantiate(Object, Transform, bool)
Object.Instantiate<T>(T)
Object.Instantiate<T>(T, Vector3, Quaternion)
Object.Instantiate<T>(T, Vector3, Quaternion, Transform)
Object.Instantiate<T>(T, Transform)
Object.Instantiate<T>(T, Transform, bool)
Object.Destroy(Object, float)
Object.Destroy(Object)
Object.DestroyImmediate(Object, bool)
Object.DestroyImmediate(Object)
Object.FindObjectsOfType(Type)
Object.FindObjectsOfType(Type, bool)
Object.DontDestroyOnLoad(Object)
Object.DestroyObject(Object, float)
Object.DestroyObject(Object)
Object.FindSceneObjectsOfType(Type)
Object.FindObjectsOfTypeIncludingAssets(Type)
Object.FindObjectsOfType<T>()
Object.FindObjectsOfType<T>(bool)
Object.FindObjectOfType<T>()
Object.FindObjectOfType<T>(bool)
Object.FindObjectsOfTypeAll(Type)
Object.FindObjectOfType(Type)
Object.FindObjectOfType(Type, bool)
Object.ToString()
Object.name
Object.hideFlags
object.Equals(object, object)
object.GetType()
object.ReferenceEquals(object, object)

Namespace: UltimateReplay.Lifecycle

Assembly: UltimateReplay.dll

Syntax

```
public sealed class ReplayObjectCustomLifecycleProvider : ReplayObjectLifecycleProvider
```

Fields

customProvider

Declaration

```
public ReplayObjectLifecycleProvider customProvider
```

Field Value

TYPE	DESCRIPTION
ReplayObjectLifecycleProvider	

Properties

IsAssigned

Declaration

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

Property Value

TYPE	DESCRIPTION
bool	

Overrides

ReplayObjectLifecycleProvider.IsAssigned

ItemName

Declaration

```
public override string ItemName { get; }
```

Property Value

TYPE	DESCRIPTION
string	

Overrides

ReplayObjectLifecycleProvider.ItemName

ItemPrefabIdentity

Declaration

```
public override ReplayIdentity ItemPrefabIdentity { get; }
```

Property Value

TYPE	DESCRIPTION
ReplayIdentity	

Overrides

ReplayObjectLifecycleProvider.ItemPrefabIdentity

Methods

DestroyReplayInstance(ReplayObject)

Declaration

```
public override void DestroyReplayInstance(ReplayObject replayInstance)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayObject	replayInstance	

Overrides

ReplayObjectLifecycleProvider.DestroyReplayInstance(ReplayObject)

InstantiateReplayInstance(Vector3, Quaternion)

Declaration

```
public override ReplayObject InstantiateReplayInstance(Vector3 position, Quaternion rotation)
```

Parameters

TYPE	NAME	DESCRIPTION
Vector3	position	
Quaternion	rotation	

Returns

TYPE	DESCRIPTION
ReplayObject	

Overrides

ReplayObjectLifecycleProvider.InstantiateReplayInstance(Vector3, Quaternion)

Class ReplayObjectDefaultLifecycleProvider

Inheritance

object

Object

ScriptableObject

ReplayObjectLifecycleProvider

ReplayObjectDefaultLifecycleProvider

Inherited Members

ReplayObjectLifecycleProvider.DestroyReplayObject(ReplayObject)

ScriptableObject.SetDirty()

ScriptableObject.CreateInstance(string)

ScriptableObject.CreateInstance(Type)

ScriptableObject.CreateInstance<T>()

Object.GetInstanceID()

Object.GetHashCode()

Object.Equals(object)

Object.Instantiate(Object, Vector3, Quaternion)

Object.Instantiate(Object, Vector3, Quaternion, Transform)

Object.Instantiate(Object)

Object.Instantiate(Object, Transform)

Object.Instantiate(Object, Transform, bool)

Object.Instantiate<T>(T)

Object.Instantiate<T>(T, Vector3, Quaternion)

Object.Instantiate<T>(T, Vector3, Quaternion, Transform)

Object.Instantiate<T>(T, Transform)

Object.Instantiate<T>(T, Transform, bool)

Object.Destroy(Object, float)

Object.Destroy(Object)

Object.DestroyImmediate(Object, bool)

Object.DestroyImmediate(Object)

Object.FindObjectsOfType(Type)

Object.FindObjectsOfType(Type, bool)

Object.DontDestroyOnLoad(Object)

Object.DestroyObject(Object, float)

Object.DestroyObject(Object)

Object.FindSceneObjectsOfType(Type)

Object.FindObjectsOfTypeIncludingAssets(Type)

Object.FindObjectsOfType<T>()

Object.FindObjectsOfType<T>(bool)

Object.FindObjectOfType<T>()

Object.FindObjectOfType<T>(bool)

Object.FindObjectsOfTypeAll(Type)

Object.FindObjectOfType(Type)

Object.FindObjectOfType(Type, bool)

Object.ToString()

Object.name

Object.hideFlags

object.Equals(object, object)

object.GetType()

object.MemberwiseClone()

object.ReferenceEquals(object, object)

Namespace: UltimateReplay.Lifecycle

Assembly: UltimateReplay.dll

Syntax

```
[Serializable]
public class ReplayObjectDefaultLifecycleProvider : ReplayObjectLifecycleProvider
```

Fields

allowPooling

Declaration

```
public bool allowPooling
```

Field Value

TYPE	DESCRIPTION
bool	

replayPrefab

Declaration

```
public ReplayObject replayPrefab
```

Field Value

TYPE	DESCRIPTION
ReplayObject	

Properties

IsAssigned

Declaration

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

Property Value

TYPE	DESCRIPTION
bool	

Overrides

ReplayObjectLifecycleProvider.IsAssigned

ItemName

Declaration

```
public override string ItemName { get; }
```

Property Value

TYPE	DESCRIPTION
string	

Overrides

ReplayObjectLifecycleProvider.ItemName

ItemPrefabIdentity

Declaration

```
public override ReplayIdentity ItemPrefabIdentity { get; }
```

Property Value

TYPE	DESCRIPTION
ReplayIdentity	

Overrides

ReplayObjectLifecycleProvider.ItemPrefabIdentity

Methods

DestroyReplayInstance(ReplayObject)

Declaration

```
public override void DestroyReplayInstance(ReplayObject replayInstance)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayObject	replayInstance	

Overrides

ReplayObjectLifecycleProvider.DestroyReplayInstance(ReplayObject)

InstantiateReplayInstance()

Declaration

```
public ReplayObject InstantiateReplayInstance()
```

Returns

TYPE	DESCRIPTION
ReplayObject	

InstantiateReplayInstance(Vector3, Quaternion)

Declaration

```
public override ReplayObject InstantiateReplayInstance(Vector3 position, Quaternion rotation)
```

Parameters

TYPE	NAME	DESCRIPTION
Vector3	position	
Quaternion	rotation	

Returns

TYPE	DESCRIPTION
ReplayObject	

Overrides

ReplayObjectLifecycleProvider.InstantiateReplayInstance(Vector3, Quaternion)

Class ReplayObjectLifecycleProvider

Inheritance

object
Object
ScriptableObject
ReplayObjectLifecycleProvider
ReplayObjectCustomLifecycleProvider
ReplayObjectDefaultLifecycleProvider
ReplayObjectResourcesLifecycleProvider

Inherited Members

ScriptableObject.SetDirty()
ScriptableObject.CreateInstance(string)
ScriptableObject.CreateInstance(Type)
ScriptableObject.CreateInstance<T>()
Object.GetInstanceID()
Object.GetHashCode()
Object.Equals(object)
Object.Instantiate(Object, Vector3, Quaternion)
Object.Instantiate(Object, Vector3, Quaternion, Transform)
Object.Instantiate(Object)
Object.Instantiate(Object, Transform)
Object.Instantiate(Object, Transform, bool)
Object.Instantiate<T>(T)
Object.Instantiate<T>(T, Vector3, Quaternion)
Object.Instantiate<T>(T, Vector3, Quaternion, Transform)
Object.Instantiate<T>(T, Transform)
Object.Instantiate<T>(T, Transform, bool)
Object.Destroy(Object, float)
Object.Destroy(Object)
Object.DestroyImmediate(Object, bool)
Object.DestroyImmediate(Object)
Object.FindObjectsOfType(Type)
Object.FindObjectsOfType(Type, bool)
Object.DontDestroyOnLoad(Object)
Object.DestroyObject(Object, float)
Object.DestroyObject(Object)
Object.FindSceneObjectsOfType(Type)
Object.FindObjectsOfTypeIncludingAssets(Type)
Object.FindObjectsOfType<T>()
Object.FindObjectsOfType<T>(bool)
Object.FindObjectOfType<T>()
Object.FindObjectOfType<T>(bool)
Object.FindObjectsOfTypeAll(Type)
Object.FindObjectOfType(Type)
Object.FindObjectOfType(Type, bool)
Object.ToString()
Object.name
Object.hideFlags
object.Equals(object, object)
object.GetType()

object.MemberwiseClone()

object.ReferenceEquals(object, object)

Namespace: UltimateReplay.Lifecycle

Assembly: UltimateReplay.dll

Syntax

```
[Serializable]
public abstract class ReplayObjectLifecycleProvider : ScriptableObject
```

Properties

IsAssigned

Declaration

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

Property Value

TYPE	DESCRIPTION
bool	

ItemName

Declaration

```
public abstract string ItemName { get; }
```

Property Value

TYPE	DESCRIPTION
string	

ItemPrefabIdentity

Declaration

```
public abstract ReplayIdentity ItemPrefabIdentity { get; }
```

Property Value

TYPE	DESCRIPTION
ReplayIdentity	

Methods

DestroyReplayInstance(ReplayObject)

Declaration

```
public abstract void DestroyReplayInstance(ReplayObject replayInstance)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayObject	replayInstance	

DestroyReplayObject(ReplayObject)

Declaration

```
public static void DestroyReplayObject(ReplayObject obj)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayObject	obj	

InstantiateReplayInstance(Vector3, Quaternion)

Declaration

```
public abstract ReplayObject InstantiateReplayInstance(Vector3 position, Quaternion rotation)
```

Parameters

TYPE	NAME	DESCRIPTION
Vector3	position	
Quaternion	rotation	

Returns

TYPE	DESCRIPTION
ReplayObject	

Class ReplayObjectResourcesLifecycleProvider

Inheritance

object
Object
ScriptableObject
ReplayObjectLifecycleProvider
ReplayObjectResourcesLifecycleProvider

Inherited Members

ReplayObjectLifecycleProvider.DestroyReplayObject(ReplayObject)
ScriptableObject.SetDirty()
ScriptableObject.CreateInstance(string)
ScriptableObject.CreateInstance(Type)
ScriptableObject.CreateInstance<T>()
Object.GetInstanceID()
Object.GetHashCode()
Object.Equals(object)
Object.Instantiate(Object, Vector3, Quaternion)
Object.Instantiate(Object, Vector3, Quaternion, Transform)
Object.Instantiate(Object)
Object.Instantiate(Object, Transform)
Object.Instantiate(Object, Transform, bool)
Object.Instantiate<T>(T)
Object.Instantiate<T>(T, Vector3, Quaternion)
Object.Instantiate<T>(T, Vector3, Quaternion, Transform)
Object.Instantiate<T>(T, Transform)
Object.Instantiate<T>(T, Transform, bool)
Object.Destroy(Object, float)
Object.Destroy(Object)
Object.DestroyImmediate(Object, bool)
Object.DestroyImmediate(Object)
Object.FindObjectsOfType(Type)
Object.FindObjectsOfType(Type, bool)
Object.DontDestroyOnLoad(Object)
Object.DestroyObject(Object, float)
Object.DestroyObject(Object)
Object.FindSceneObjectsOfType(Type)
Object.FindObjectsOfTypeIncludingAssets(Type)
Object.FindObjectsOfType<T>()
Object.FindObjectsOfType<T>(bool)
Object.FindObjectOfType<T>()
Object.FindObjectOfType<T>(bool)
Object.FindObjectsOfTypeAll(Type)
Object.FindObjectOfType(Type)
Object.FindObjectOfType(Type, bool)
Object.ToString()
Object.name
Object.hideFlags
object.Equals(object, object)
object.GetType()
object.MemberwiseClone()

object.ReferenceEquals(object, object)

Namespace: UltimateReplay.Lifecycle

Assembly: UltimateReplay.dll

Syntax

```
[Serializable]
public class ReplayObjectResourcesLifecycleProvider : ReplayObjectLifecycleProvider
```

Fields

allowPooling

Declaration

```
public bool allowPooling
```

Field Value

TYPE	DESCRIPTION
bool	

asyncLoadOnStartup

Declaration

```
public bool asyncLoadOnStartup
```

Field Value

TYPE	DESCRIPTION
bool	

resourcesPath

Declaration

```
public string resourcesPath
```

Field Value

TYPE	DESCRIPTION
string	

Properties

IsAssigned

Declaration

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

Property Value

TYPE	DESCRIPTION
bool	

Overrides

ReplayObjectLifecycleProvider.IsAssigned

ItemName

Declaration

```
public override string ItemName { get; }
```

Property Value

TYPE	DESCRIPTION
string	

Overrides

ReplayObjectLifecycleProvider.ItemName

ItemPrefabIdentity

Declaration

```
public override ReplayIdentity ItemPrefabIdentity { get; }
```

Property Value

TYPE	DESCRIPTION
ReplayIdentity	

Overrides

ReplayObjectLifecycleProvider.ItemPrefabIdentity

Methods

DestroyReplayInstance(ReplayObject)

Declaration

```
public override void DestroyReplayInstance(ReplayObject replayInstance)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayObject	replayInstance	

Overrides

ReplayObjectLifecycleProvider.DestroyReplayInstance(ReplayObject)

InstantiateReplayInstance()

Declaration

```
public ReplayObject InstantiateReplayInstance()
```

Returns

TYPE	DESCRIPTION
ReplayObject	

InstantiateReplayInstance(Vector3, Quaternion)

Declaration

```
public override ReplayObject InstantiateReplayInstance(Vector3 position, Quaternion rotation)
```

Parameters

TYPE	NAME	DESCRIPTION
Vector3	position	
Quaternion	rotation	

Returns

TYPE	DESCRIPTION
ReplayObject	

Overrides

ReplayObjectLifecycleProvider.InstantiateReplayInstance(Vector3, Quaternion)

Namespace UltimateReplay.Serializers

Classes

ReplayMaterialChangeSerializer

ReplayMaterialSerializer

ReplayParticleSystemSerializer

ReplayPointRendererSerializer

Enums

ReplayMaterialChangeSerializer.ReplayMaterialChangeSerializeFlags

ReplayMaterialSerializer.ReplayMaterialSerializeFlags

ReplayParticleSystemSerializer.ReplayParticleSystemSerializeFlags

ReplayPointRendererSerializer.ReplayPointRendererSerializeFlags

Class ReplayMaterialChangeSerializer

Inheritance

object
ReplayMaterialChangeSerializer

Implements

IReplaySerialize

Inherited Members

object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: UltimateReplay.Serializers

Assembly: UltimateReplay.dll

Syntax

```
public sealed class ReplayMaterialChangeSerializer : IReplaySerialize
```

Properties

MaterialIndex

Declaration

```
public int MaterialIndex { get; }
```

Property Value

TYPE	DESCRIPTION
int	

MaterialIndexes

Declaration

```
public int[] MaterialIndexes { get; }
```

Property Value

TYPE	DESCRIPTION
int[]	

SerializeFlags

Declaration

```
public ReplayMaterialChangeSerializer.ReplayMaterialChangeSerializeFlags SerializeFlags { get; set; }
```

Property Value

TYPE	DESCRIPTION
ReplayMaterialChangeSerializer.ReplayMaterialChangeSerializeFlags	

Methods

GetActiveMaterial(IList<Material>)

Declaration

```
public Material GetActiveMaterial(IList<Material> possibleMaterials)
```

Parameters

TYPE	NAME	DESCRIPTION
IList<Material>	possibleMaterials	

Returns

TYPE	DESCRIPTION
Material	

GetActiveMaterials(IList<Material>, Material[])

Declaration

```
public int GetActiveMaterials(IList<Material> possibleMaterials, Material[] results)
```

Parameters

TYPE	NAME	DESCRIPTION
IList<Material>	possibleMaterials	
Material[]	results	

Returns

TYPE	DESCRIPTION
int	

OnReplayDeserialize(ReplayState)

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

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState to read the data from

OnReplaySerialize(ReplayState)

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

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState to write the data to

SetActiveMaterial(IList<Material>, Material)

Declaration

```
public void SetActiveMaterial(IList<Material> possibleMaterials, Material activeMaterial)
```

Parameters

TYPE	NAME	DESCRIPTION
IList<Material>	possibleMaterials	
Material	activeMaterial	

SetActiveMaterials(IList<Material>, Material[])

Declaration

```
public void SetActiveMaterials(IList<Material> possibleMaterials, Material[] activeMaterials)
```

Parameters

TYPE	NAME	DESCRIPTION
IList<Material>	possibleMaterials	
Material[]	activeMaterials	

Implements

IReplaySerialize

Enum ReplayMaterialChangeSerializer.ReplayMaterialChangeSerializeFlags

Namespace: UltimateReplay.Serializers

Assembly: UltimateReplay.dll

Syntax

```
[Flags]
public enum ReplayMaterialChangeSerializer.ReplayMaterialChangeSerializeFlags : ushort
```

Fields

NAME	DESCRIPTION
AllMaterials	
SharedMaterial	
_16BitIndex	
_32BitIndex	
_8BitIndex	

Class ReplayMaterialSerializer

Inheritance

object

ReplayMaterialSerializer

Implements

IReplaySerialize

Inherited Members

object.Equals(object)

object.Equals(object, object)

object.GetHashCode()

object.GetType()

object.MemberwiseClone()

object.ReferenceEquals(object, object)

object.ToString()

Namespace: UltimateReplay.Serializers

Assembly: UltimateReplay.dll

Syntax

```
public class ReplayMaterialSerializer : IReplaySerialize
```

Properties

Color

Declaration

```
public Color Color { get; set; }
```

Property Value

TYPE	DESCRIPTION
Color	

DoubleSidedGlobalIllumination

Declaration

```
public bool DoubleSidedGlobalIllumination { get; set; }
```

Property Value

TYPE	DESCRIPTION
bool	

GlobalIlluminationFlags

Declaration

```
public MaterialGlobalIlluminationFlags GlobalIlluminationFlags { get; set; }
```

Property Value

TYPE	DESCRIPTION
MaterialGlobalIlluminationFlags	

MainTextureOffset

Declaration

```
public Vector2 MainTextureOffset { get; set; }
```

Property Value

TYPE	DESCRIPTION
Vector2	

MainTextureScale

Declaration

```
public Vector2 MainTextureScale { get; set; }
```

Property Value

TYPE	DESCRIPTION
Vector2	

SerializeFlags

Declaration

```
public ReplayMaterialSerializer.ReplayMaterialSerializeFlags SerializeFlags { get; set; }
```

Property Value

TYPE	DESCRIPTION
ReplayMaterialSerializer.ReplayMaterialSerializeFlags	

Methods

OnReplayDeserialize(ReplayState)

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

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState to read the data from

OnReplaySerialize(ReplayState)

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

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState to write the data to

Reset()

Declaration

```
public void Reset()
```

Implements

IReplaySerialize

Enum ReplayMaterialSerializer.ReplayMaterialSerializeFlags

Namespace: UltimateReplay.Serializers

Assembly: UltimateReplay.dll

Syntax

```
[Flags]
public enum ReplayMaterialSerializer.ReplayMaterialSerializeFlags : byte
```

Fields

NAME	DESCRIPTION
Color	
DoubleSidedGlobalIllumination	
GlobalIlluminationFlags	
MainTextureOffset	
MainTextureScale	
None	
SharedMaterial	

Class ReplayParticleSystemSerializer

Inheritance

object

ReplayParticleSystemSerializer

Implements

IReplaySerialize

Inherited Members

object.Equals(object)

object.Equals(object, object)

object.GetHashCode()

object.GetType()

object.ReferenceEquals(object, object)

object.ToString()

Namespace: UltimateReplay.Serializers

Assembly: UltimateReplay.dll

Syntax

```
public sealed class ReplayParticleSystemSerializer : IReplaySerialize
```

Properties

RandomSeed

Declaration

```
public uint RandomSeed { get; set; }
```

Property Value

TYPE	DESCRIPTION
uint	

SerializeFlags

Declaration

```
public ReplayParticleSystemSerializer.ReplayParticleSystemSerializeFlags SerializeFlags { get; set; }
```

Property Value

TYPE	DESCRIPTION
ReplayParticleSystemSerializer.ReplayParticleSystemSerializeFlags	

SimulationTime

Declaration

```
public float SimulationTime { get; set; }
```

Property Value

TYPE	DESCRIPTION
float	

Methods

OnReplayDeserialize(ReplayState)

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

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState to read the data from

OnReplaySerialize(ReplayState)

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

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState to write the data to

Implements

IReplaySerialize

Enum ReplayParticleSystemSerializer.ReplayParticleSystemSerializeFlags

Namespace: UltimateReplay.Serializers

Assembly: UltimateReplay.dll

Syntax

```
[Flags]
public enum ReplayParticleSystemSerializer.ReplayParticleSystemSerializeFlags : ushort
```

Fields

NAME	DESCRIPTION
LowPrecision	
None	

Class ReplayPointRendererSerializer

Inheritance

object

ReplayPointRendererSerializer

Implements

IReplaySerialize

Inherited Members

object.Equals(object)

object.Equals(object, object)

object.GetHashCode()

object.GetType()

object.ReferenceEquals(object, object)

object.ToString()

Namespace: UltimateReplay.Serializers

Assembly: UltimateReplay.dll

Syntax

```
public sealed class ReplayPointRendererSerializer : IReplaySerialize
```

Properties

Points

Declaration

```
public IList<Vector3> Points { get; }
```

Property Value

TYPE	DESCRIPTION
IList<Vector3>	

SerializeFlags

Declaration

```
public ReplayPointRendererSerializer.ReplayPointRendererSerializeFlags SerializeFlags { get; set; }
```

Property Value

TYPE	DESCRIPTION
ReplayPointRendererSerializer.ReplayPointRendererSerializeFlags	

Methods

OnReplayDeserialize(ReplayState)

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

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState to read the data from

OnReplaySerialize(ReplayState)

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

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState to write the data to

Reset()

Declaration

```
public void Reset()
```

Implements

IReplaySerialize

Enum ReplayPointRendererSerializer.ReplayPointRendererSerializeFlags

Namespace: UltimateReplay.Serializers

Assembly: UltimateReplay.dll

Syntax

```
[Flags]
public enum ReplayPointRendererSerializer.ReplayPointRendererSerializeFlags : byte
```

Fields

NAME	DESCRIPTION
HalfPrecisionCount	
LowPrecision	
None	

Namespace UltimateReplay.StatePreparation

Classes

ComponentPreparer

ComponentPreparer<T>

DefaultReplayPreparer

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

DefaultReplayPreparer.ComponentPreparerSettings

SerializableType

Interfaces

IReplayPreparer

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 scritps 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.

Class ComponentPreparer

Inheritance

object
ComponentPreparer
ComponentPreparer<T>

Inherited Members

object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: UltimateReplay.StatePreparation

Assembly: UltimateReplay.dll

Syntax

```
public abstract class ComponentPreparer
```

Fields

enabled

Declaration

```
public bool enabled
```

Field Value

TYPE	DESCRIPTION
bool	

Properties

Attribute

Declaration

```
protected ReplayComponentPreparerAttribute Attribute { get; set; }
```

Property Value

TYPE	DESCRIPTION
ReplayComponentPreparerAttribute	

Preparers

Declaration

```
public static IReadOnlyList<ComponentPreparer> Preparers { get; }
```

Property Value

TYPE	DESCRIPTION
IReadOnlyList<ComponentPreparer>	

Methods

FindPreparer(Type)

Declaration

```
public static ComponentPreparer FindPreparer(Type componentType)
```

Parameters

TYPE	NAME	DESCRIPTION
Type	componentType	

Returns

TYPE	DESCRIPTION
ComponentPreparer	

InitializePreparers()

Declaration

```
public static void InitializePreparers()
```

Class ComponentPreparer<T>

Inheritance

object
ComponentPreparer
ComponentPreparer<T>

Inherited Members

ComponentPreparer.enabled
ComponentPreparer.Preparers
ComponentPreparer.Attribute
ComponentPreparer.InitializePreparers()
ComponentPreparer.FindPreparer(Type)
object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: UltimateReplay.StatePreparation

Assembly: UltimateReplay.dll

Syntax

```
public abstract class ComponentPreparer<T> : ComponentPreparer where T : Component
```

Type Parameters

NAME	DESCRIPTION
T	

Methods

PrepareForGameplay(T, ReplayState)

Declaration

```
public abstract void PrepareForGameplay(T component, ReplayState additionalData)
```

Parameters

TYPE	NAME	DESCRIPTION
T	component	
ReplayState	additionalData	

PrepareForPlayback(T, ReplayState)

Declaration

```
public abstract void PrepareForPlayback(T component, ReplayState additionalData)
```

Parameters

TYPE	NAME	DESCRIPTION
T	component	
ReplayState	additionalData	

Class DefaultReplayPreparer

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

Inheritance

object
DefaultReplayPreparer

Implements

IReplayPreparer
ISerializationCallbackReceiver

Inherited Members

object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: UltimateReplay.StatePreparation

Assembly: UltimateReplay.dll

Syntax

```
[Serializable]
public class DefaultReplayPreparer : IReplayPreparer, ISerializationCallbackReceiver
```

Properties

PreparerSettings

Declaration

```
public IList<DefaultReplayPreparer.ComponentPreparerSettings> PreparerSettings { get; }
```

Property Value

TYPE	DESCRIPTION
IList<DefaultReplayPreparer.ComponentPreparerSettings>	

SkipTypes

Declaration

```
public IList<SerializableType> SkipTypes { get; }
```

Property Value

TYPE	DESCRIPTION
IList<SerializableType>	

Methods

CreateInstance()

Declaration

```
public DefaultReplayPreparer CreateInstance()
```

Returns

TYPE	DESCRIPTION
DefaultReplayPreparer	

HasSkipType(Type)

Declaration

```
public bool HasSkipType(Type systemType)
```

Parameters

TYPE	NAME	DESCRIPTION
Type	systemType	

Returns

TYPE	DESCRIPTION
bool	

OnAfterDeserialize()

Implement this method to receive a callback after Unity deserializes your object.

Declaration

```
public void OnAfterDeserialize()
```

OnBeforeSerialize()

Implement this method to receive a callback before Unity serializes your object.

Declaration

```
public void OnBeforeSerialize()
```

PrepareForGameplay(ReplayObject)

Prepare the specified replay object for gameplay mode.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayObject	replayObject	The replay object to prepare

PrepareForPlayback(ReplayObject)

Prepare the specified replay object for playback mode.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayObject	replayObject	The replay object to prepare

Implements

IReplayPreparer
UnityEngine.ISerializationCallbackReceiver

Class DefaultReplayPreparer.ComponentPreparerSettings

Inheritance

object
DefaultReplayPreparer.ComponentPreparerSettings

Inherited Members

object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: UltimateReplay.StatePreparation

Assembly: UltimateReplay.dll

Syntax

```
[Serializable]
public class DefaultReplayPreparer.ComponentPreparerSettings
```

Fields

componentPreparerType

Declaration

```
public SerializableType componentPreparerType
```

Field Value

TYPE	DESCRIPTION
SerializableType	

enabled

Declaration

```
public bool enabled
```

Field Value

TYPE	DESCRIPTION
bool	

Interface IReplayPreparer

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 scritps 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.StatePreparation

Assembly: UltimateReplay.dll

Syntax

```
public interface IReplayPreparer
```

Methods

PrepareForGameplay(ReplayObject)

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.

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayObject	replayObject	The replay object to prepare

PrepareForPlayback(ReplayObject)

Prepares the specified replay object for playback. The implementing type should ensure that all game systems likley 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.

Declaration

```
void PrepareForPlayback(ReplayObject replayObject)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayObject	replayObject	The replay object that should be prepared

Class SerializableType

Inheritance

object

SerializableType

Inherited Members

object.Equals(object)

object.Equals(object, object)

object.GetHashCode()

object.GetType()

object.MemberwiseClone()

object.ReferenceEquals(object, object)

object.ToString()

Namespace: UltimateReplay.StatePreparation

Assembly: UltimateReplay.dll

Syntax

```
[Serializable]
public class SerializableType
```

Constructors

SerializableType()

Declaration

```
public SerializableType()
```

SerializableType(Type)

Declaration

```
public SerializableType(Type systemType)
```

Parameters

TYPE	NAME	DESCRIPTION
Type	systemType	

Properties

SystemType

Declaration

```
public Type SystemType { get; }
```

Property Value

TYPE	DESCRIPTION
Type	

Methods

ResolveType()

Declaration

```
public bool ResolveType()
```

Returns

TYPE	DESCRIPTION
bool	

Operators

implicit operator SerializableType(Type)

Declaration

```
public static implicit operator SerializableType(Type systemType)
```

Parameters

TYPE	NAME	DESCRIPTION
Type	systemType	

Returns

TYPE	DESCRIPTION
SerializableType	

Namespace UltimateReplay.Statistics

Classes

ReplayRecordableStatistics

ReplayStatisticsUtil

ReplayStorageTargetStatistics

Structs

ReplayRecordableStatistics.ReplayObjectStatistics

Class ReplayRecordableStatistics

Inheritance

object

ReplayRecordableStatistics

Inherited Members

object.Equals(object)

object.Equals(object, object)

object.GetHashCode()

object.GetType()

object.MemberwiseClone()

object.ReferenceEquals(object, object)

object.ToString()

Namespace: UltimateReplay.Statistics

Assembly: UltimateReplay.dll

Syntax

```
public static class ReplayRecordableStatistics
```

Methods

CalculateReplayRecordStorageUsage(ReplayObject)

Declaration

```
public static ReplayRecordableStatistics.ReplayObjectStatistics CalculateReplayRecordStorageUsage(ReplayObject
replayObject)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayObject	replayObject	

Returns

TYPE	DESCRIPTION
ReplayRecordableStatistics.ReplayObjectStatistics	

CalculateReplayRecordStorageUsage(params ReplayObject[])

Declaration

```
public static ReplayRecordableStatistics.ReplayObjectStatistics CalculateReplayRecordStorageUsage(params
ReplayObject[] replayObjects)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayObject[]	replayObjects	

Returns

TYPE	DESCRIPTION
ReplayRecordableStatistics.ReplayObjectStatistics	

CalculateReplayRecordStorageUsage(ReplayRecordableBehaviour)

Declaration

```
public static int CalculateReplayRecordStorageUsage(ReplayRecordableBehaviour replayBehaviour)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayRecordableBehaviour	replayBehaviour	

Returns

TYPE	DESCRIPTION
int	

CalculateReplayRecordStorageUsage(params ReplayRecordableBehaviour[])

Declaration

```
public static int CalculateReplayRecordStorageUsage(params ReplayRecordableBehaviour[] replayBehaviours)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayRecordableBehaviour[]	replayBehaviours	

Returns

TYPE	DESCRIPTION
int	

SupressStatisticsDuringEditMode()

Declaration

```
public static void SupressStatisticsDuringEditMode()
```

Struct ReplayRecordableStatistics.ReplayObjectStatistics

Inherited Members

ValueType.Equals(object)
ValueType.GetHashCode()
ValueType.ToString()
object.Equals(object, object)
object.GetType()
object.ReferenceEquals(object, object)

Namespace: UltimateReplay.Statistics

Assembly: UltimateReplay.dll

Syntax

```
public struct ReplayRecordableStatistics.ReplayObjectStatistics
```

Fields

byteSize

Declaration

```
public int byteSize
```

Field Value

TYPE	DESCRIPTION
int	

didSupressComponents

Declaration

```
public bool didSupressComponents
```

Field Value

TYPE	DESCRIPTION
bool	

evaluatedComponents

Declaration

```
public int evaluatedComponents
```

Field Value

TYPE	DESCRIPTION
int	

supressedComponents

Declaration

```
public int supressedComponents
```

Field Value

TYPE	DESCRIPTION
int	

Class ReplayStatisticsUtil

Inheritance

object

ReplayStatisticsUtil

Inherited Members

object.Equals(object)

object.Equals(object, object)

object.GetHashCode()

object.GetType()

object.MemberwiseClone()

object.ReferenceEquals(object, object)

object.ToString()

Namespace: UltimateReplay.Statistics

Assembly: UltimateReplay.dll

Syntax

```
public static class ReplayStatisticsUtil
```

Methods

GetByteSizeString(int)

Declaration

```
public static string GetByteSizeString(int bytes)
```

Parameters

TYPE	NAME	DESCRIPTION
int	bytes	

Returns

TYPE	DESCRIPTION
string	

GetMemorySizeSmallestUnit(int)

Declaration

```
public static decimal GetMemorySizeSmallestUnit(int amount)
```

Parameters

TYPE	NAME	DESCRIPTION
int	amount	

Returns

TYPE	DESCRIPTION
decimal	

GetMemoryUnitString(int)

Declaration

```
public static string GetMemoryUnitString(int amount)
```

Parameters

TYPE	NAME	DESCRIPTION
int	amount	

Returns

TYPE	DESCRIPTION
string	

Class ReplayStorageTargetStatistics

Inheritance

object

ReplayStorageTargetStatistics

Inherited Members

object.Equals(object)

object.Equals(object, object)

object.GetHashCode()

object.GetType()

object.MemberwiseClone()

object.ReferenceEquals(object, object)

object.ToString()

Namespace: UltimateReplay.Statistics

Assembly: UltimateReplay.dll

Syntax

```
public static class ReplayStorageTargetStatistics
```

Methods

CalculateReplayMemoryUsage()

Declaration

```
public static int CalculateReplayMemoryUsage()
```

Returns

TYPE	DESCRIPTION
int	

Namespace UltimateReplay.Storage

Classes

ReplayFileStorage

ReplayHighlightReelStorage

A special storage target that can combine multiple other storage sources into a single replay to create a highlight reel/montage. Useful for showing action replays in sequence or similar.

ReplayMemoryStorage

ReplayPersistentData

ReplaySegment

ReplaySnapshot

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.

ReplayStorage

Represents and abstract storage device capable of holding recorded state data for playback at a later date. Depending upon implementation, a ReplayStorage may be volatile or non-volatile.

ReplayStreamSource

Represents a data stream source that a replay stream can work with.

ReplayStreamStorage

ReplayStreamStorage.ReplaySegmentTable

ReplayStreamStorage.ReplayStreamHeader

ReplayStreamUtility

Structs

ReplaySnapshot.ReplayObjectCreatedData

ReplayStreamStorage.ReplaySegmentEntry

ReplayToken

Interfaces

IReplaySnapshotStorable

Represents a replay data stream that could be recorded data or a pointer to recorded data. Used for lossless compression to reduce storage size by combining snapshots frames with identical data.

IReplayStreamSerialize

Interface that should be implemented by any types that can be serialized to a steam object.

IReplayTokenSerialize

IReplayTokenSerializeProvider

Enums

ReplayFileType

ReplaySnapshot.ReplayObjectCreatedData.ReplaySerializeFlags

Represents initial data that may be stored by an object.

ReplaySnapshotStorableType

ReplayStorageAction

Represents a task that can be issued to a ReplayStorage.

ReplayStreamType

Interface IReplaySnapshotStorable

Represents a replay data stream that could be recorded data or a pointer to recorded data. Used for lossless compression to reduce storage size by combining snapshots frames with identical data.

Inherited Members

IReplayStreamSerialize.OnReplayStreamSerialize(BinaryWriter)
IReplayStreamSerialize.OnReplayStreamDeserialize(BinaryReader)

Namespace: UltimateReplay.Storage

Assembly: UltimateReplay.dll

Syntax

```
public interface IReplaySnapshotStorable : IReplayStreamSerialize
```

Properties

StorageType

Get the ReplaySnapshotStorableType of this replay data stream.

Declaration

```
ReplaySnapshotStorableType StorageType { get; }
```

Property Value

TYPE	DESCRIPTION
ReplaySnapshotStorableType	

Interface IReplayStreamSerialize

Interface that should be implemented by any types that can be serialized to a steam object.

Namespace: UltimateReplay.Storage

Assembly: UltimateReplay.dll

Syntax

```
public interface IReplayStreamSerialize
```

Methods

OnReplayStreamDeserialize(BinaryReader)

Called by the replay system when the object should deserialize its replay data from a binary source.

Declaration

```
void OnReplayStreamDeserialize(BinaryReader reader)
```

Parameters

TYPE	NAME	DESCRIPTION
BinaryReader	reader	The reader where the data is stored

OnReplayStreamSerialize(BinaryWriter)

Called by the replay system when the object should serialize its replay data into a binary target.

Declaration

```
void OnReplayStreamSerialize(BinaryWriter writer)
```

Parameters

TYPE	NAME	DESCRIPTION
BinaryWriter	writer	The writer object used to store data

Interface IReplayTokenSerialize

Namespace: UltimateReplay.Storage

Assembly: UltimateReplay.dll

Syntax

```
public interface IReplayTokenSerialize
```

Properties

SerializeTokens

Declaration

```
IEnumerable<ReplayToken> SerializeTokens { get; }
```

Property Value

TYPE	DESCRIPTION
IEnumerable<ReplayToken>	

Interface IReplayTokenSerializeProvider

Namespace: UltimateReplay.Storage

Assembly: UltimateReplay.dll

Syntax

```
public interface IReplayTokenSerializeProvider
```

Properties

SerializeTarget

Declaration

```
IReplayTokenSerialize SerializeTarget { get; set; }
```

Property Value

TYPE	DESCRIPTION
IReplayTokenSerialize	

Class ReplayFileStorage

Inheritance

object
ReplayStorage
ReplayFileStorage

Implements

IDisposable

Inherited Members

ReplayStorage.metadata
ReplayStorage.persistentData
ReplayStorage.IsLocked
ReplayStorage.IsDisposed
ReplayStorage.CheckDisposed()
ReplayStorage.Dispose()
ReplayStorage.Lock(ReplayOperation)
ReplayStorage.Unlock(ReplayOperation)
ReplayStorage.CopyTo(ReplayStorage)
ReplayStorage.CopyToAsync(ReplayStorage)
object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: UltimateReplay.Storage
Assembly: UltimateReplay.dll

Syntax

```
public abstract class ReplayFileStorage : ReplayStorage, IDisposable
```

Constructors

ReplayFileStorage(string, ReplayStreamStorage)

Declaration

```
protected ReplayFileStorage(string filePath, ReplayStreamStorage stream)
```

Parameters

TYPE	NAME	DESCRIPTION
string	filePath	
ReplayStreamStorage	stream	

Properties

CanRead

Get a value indicating whether this storage target is readable. Value will be true if the specified file exists.

Declaration

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

Property Value

TYPE	DESCRIPTION
bool	

Overrides

ReplayStorage.CanRead

CanWrite

Get a value indicating whether this storage target is writable. Value will be true if the file path is valid and accessible.

Declaration

```
public override bool CanWrite { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

Overrides

ReplayStorage.CanWrite

Duration

The amount of time in seconds that this recording lasts.

Declaration

```
public override float Duration { get; }
```

Property Value

TYPE	DESCRIPTION
float	

Overrides

ReplayStorage.Duration

FilePath

Declaration

```
public string FilePath { get; }
```

Property Value

TYPE	DESCRIPTION
string	

IdentitySize

Get the size in bytes required to serialize a ReplayIdentity.

Declaration

```
public override int IdentitySize { get; }
```

Property Value

TYPE	DESCRIPTION
int	

Overrides

ReplayStorage.IdentitySize

IsBuffering

Declaration

```
public bool IsBuffering { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

MemorySize

Get the total amount of bytes that this replay uses.

Declaration

```
public override int MemorySize { get; }
```

Property Value

TYPE	DESCRIPTION
int	

Overrides

ReplayStorage.MemorySize

Metadata

The user metadata associated with this storage target. Derive from ReplayMetadata and declare additional serialized fields in order to store custom metadata in a replay.

Declaration

```
public override ReplayMetadata Metadata { get; set; }
```

Property Value

TYPE	DESCRIPTION
ReplayMetadata	

Overrides

ReplayStorage.Metadata

PersistentData

The persistent data associated with this storage target. Typically used to store single shot data or object instantate data such as initial position, parent, etc.

Declaration

```
public override ReplayPersistentData PersistentData { get; set; }
```

Property Value

TYPE	DESCRIPTION
ReplayPersistentData	

Overrides

ReplayStorage.PersistentData

SnapshotSize

Get the total number of snapshots included in this replay.

Declaration

```
public override int SnapshotSize { get; }
```

Property Value

TYPE	DESCRIPTION
int	

Overrides

ReplayStorage.SnapshotSize

Methods

FetchSnapshot(int)

Recall a snapshot by its unique sequence id value. The sequence ID value indicates the snapshots 0-based index value for the recording sequence.

Declaration

```
public override ReplaySnapshot FetchSnapshot(int sequenceID)
```

Parameters

TYPE	NAME	DESCRIPTION
int	sequenceID	The sequence ID to fetch the snapshot for

Returns

TYPE	DESCRIPTION
ReplaySnapshot	The replay snapshot at the specified sequence id

Overrides

ReplayStorage.FetchSnapshot(int)

FetchSnapshot(float)

Recall a snapshot from the replay target based on the specified replay offset.

Declaration

```
public override ReplaySnapshot FetchSnapshot(float timeStamp)
```

Parameters

TYPE	NAME	DESCRIPTION
float	timeStamp	The time offset from the start of the recording pointing to the individual snapshot to recall

Returns

TYPE	DESCRIPTION
ReplaySnapshot	The replay snapshot at the specified offset

Overrides

ReplayStorage.FetchSnapshot(float)

FromFile(string, ReplayFileType, bool, CompressionLevel)

Declaration

```
public static ReplayFileStorage FromFile(string filePath, ReplayFileType fileType =
ReplayFileType.FromExtension, bool useSegmentCompression = true, CompressionLevel blockCompressionLevel =
CompressionLevel.Optimal)
```

Parameters

TYPE	NAME	DESCRIPTION
string	filePath	
ReplayFileType	fileType	
bool	useSegmentCompression	
CompressionLevel	blockCompressionLevel	

Returns

TYPE	DESCRIPTION
ReplayFileStorage	

FromFileBinary(string, bool, CompressionLevel)

Declaration

```
public static ReplayFileStorage FromFileBinary(string filePath, bool useSegmentCompression = true,
CompressionLevel blockCompressionLevel = CompressionLevel.Optimal)
```

Parameters

TYPE	NAME	DESCRIPTION
string	filePath	
bool	useSegmentCompression	
CompressionLevel	blockCompressionLevel	

Returns

TYPE	DESCRIPTION
ReplayFileStorage	

FromFileBson(string)

Declaration

```
public static ReplayFileStorage FromFileBson(string filePath)
```

Parameters

TYPE	NAME	DESCRIPTION
string	filePath	

Returns

TYPE	DESCRIPTION
ReplayFileStorage	

FromFileJson(string)

Declaration

```
public static ReplayFileStorage FromFileJson(string filePath)
```

Parameters

TYPE	NAME	DESCRIPTION
string	filePath	

Returns

TYPE	DESCRIPTION
ReplayFileStorage	

IsReplayFile(string)

Declaration

```
public static bool IsReplayFile(string filePath)
```

Parameters

TYPE	NAME	DESCRIPTION
string	filePath	

Returns

TYPE	DESCRIPTION
bool	

LoadFileCompletely()

Declaration

```
public void LoadFileCompletely()
```

LoadFileCompletelyAsync()

Declaration

```
public ReplayAsyncOperation LoadFileCompletelyAsync()
```

Returns

TYPE	DESCRIPTION
ReplayAsyncOperation	

OnDispose()

Called when the storage target should be disposed to cleanup.

Declaration

```
protected override void OnDispose()
```

Overrides

ReplayStorage.OnDispose()

Prepare(ReplayStorageAction)

Called by the recording system to notify the active ReplayStorage of an upcoming event.

Declaration

```
public override void Prepare(ReplayStorageAction mode)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayStorageAction	mode	The ReplayStorageAction that the target should prepare for

Overrides

ReplayStorage.Prepare(ReplayStorageAction)

ReadFileCompletely(string, ReplayFileType)

Load an existing replay file completely into memory in contrast to the default streaming on demand behaviour. Subsequent read requests such as FetchSnapshot(float) will be near instant since all data will be cached in memory. Note that this method will block the calling thread until the file has been completely loaded into memory. See ReadFileCompletelyAsync(string, ReplayFileType) for a non-blocking alternative. Note that this method is only recommended for relatively small replay files depending upon target device in order to avoid out of memory scenarios.

Declaration

```
public static ReplayFileStorage ReadFileCompletely(string filePath, ReplayFileType fileType =
ReplayFileType.FromExtension)
```

Parameters

TYPE	NAME	DESCRIPTION
string	filePath	The path to the replay file to load
ReplayFileType	fileType	The optional type of replay file to load which will be determined from the file extension by default

Returns

TYPE	DESCRIPTION
ReplayFileStorage	

Exceptions

TYPE	CONDITION
ArgumentException	The specified file path is null or empty
FileNotFoundException	The specified file path does not exist

ReadFileCompletelyAsync(string, ReplayFileType)

Declaration

```
public static ReplayAsyncOperation<ReplayFileStorage> ReadFileCompletelyAsync(string filePath, ReplayFileType
fileType = ReplayFileType.FromExtension)
```

Parameters

TYPE	NAME	DESCRIPTION
string	filePath	
ReplayFileType	fileType	

Returns

TYPE	DESCRIPTION
ReplayAsyncOperation<ReplayFileStorage>	

ReadMetadataOnly(string)

Declaration

```
public static ReplayMetadata ReadMetadataOnly(string filePath)
```

Parameters

TYPE	NAME	DESCRIPTION
string	filePath	

Returns

TYPE	DESCRIPTION
ReplayMetadata	

StoreSnapshot(ReplaySnapshot)

Store a replay snapshot in the replay target.

Declaration

```
public override void StoreSnapshot(ReplaySnapshot state)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplaySnapshot	state	The snapshot to store

Overrides

ReplayStorage.StoreSnapshot(ReplaySnapshot)

Implements

IDisposable

Enum ReplayFileType

Namespace: UltimateReplay.Storage

Assembly: UltimateReplay.dll

Syntax

```
public enum ReplayFileType
```

Fields

NAME	DESCRIPTION
Binary	The replay system will use a high performance binary file format for best performance and storage requirements.
Bson	The replay system will use the bson file format.
FromExtension	The replay system will select a file format based on file extension.
Json	The replay system will use a human readable json file format for the replay. Useful for working with replay files in other applications.

Class ReplayHighlightReelStorage

A special storage target that can combine multiple other storage sources into a single replay to create a highlight reel/montage. Useful for showing action replays in sequence or similar.

Inheritance

object
ReplayStorage
ReplayHighlightReelStorage

Implements

IDisposable

Inherited Members

ReplayStorage.IsLocked
ReplayStorage.Metadata
ReplayStorage.PersistentData
ReplayStorage.IsDisposed
ReplayStorage.Dispose()
ReplayStorage.CopyTo(ReplayStorage)
ReplayStorage.CopyToAsync(ReplayStorage)
object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: UltimateReplay.Storage
Assembly: UltimateReplay.dll

Syntax

```
public sealed class ReplayHighlightReelStorage : ReplayStorage, IDisposable
```

Constructors

ReplayHighlightReelStorage(IEnumerable<ReplayStorage>, bool)

Create a new instance with the specified storage inputs to combine into a highlights reel.

Declaration

```
public ReplayHighlightReelStorage(IEnumerable<ReplayStorage> highlights, bool disposeHighlights = true)
```

Parameters

TYPE	NAME	DESCRIPTION
IEnumerable<ReplayStorage>	highlights	A number of storage targets used to form a montage in the order specified
bool	disposeHighlights	True if all provided storage targets should also be disposed when this ReplayHighlightReelStorage is disposed

Exceptions

TYPE	CONDITION
ArgumentNullException	One or more storage targets in the specified IEnumerable<T> are null

Properties

CanRead

Does the storage target support read operations for playback mode.

Declaration

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

Property Value

TYPE	DESCRIPTION
bool	

Overrides

ReplayStorage.CanRead

CanWrite

Does the storage target support write operations for record mode.

Declaration

```
public override bool CanWrite { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

Overrides

ReplayStorage.CanWrite

Duration

Get the duration in seconds that the stored recording lasts.

Declaration

```
public override float Duration { get; }
```

Property Value

TYPE	DESCRIPTION
float	

Overrides

ReplayStorage.Duration

IdentitySize

Get the size in bytes of all ReplayIdentity stored in this recording. The byte size of ReplayIdentity may be changed for better storage size vs max number of possible replay objects.

Declaration

```
public override int IdentitySize { get; }
```

Property Value

TYPE	DESCRIPTION
int	

Overrides

ReplayStorage.IdentitySize

MemorySize

Get the amount of bytes that have been stored for the current recording. The number of bytes represents only the data recorded by the replay system and not actual memory usage.

Declaration

```
public override int MemorySize { get; }
```

Property Value

TYPE	DESCRIPTION
int	

Overrides

ReplayStorage.MemorySize

SnapshotSize

Get the total number of ReplaySnapshot stored in the current recording.

Declaration

```
public override int SnapshotSize { get; }
```

Property Value

TYPE	DESCRIPTION
int	

Overrides

ReplayStorage.SnapshotSize

Methods

FetchSnapshot(int)

Recall a snapshot by its unique sequence id value. The sequence ID value indicates the snapshots 0-based index value for the recording sequence.

Declaration

```
public override ReplaySnapshot FetchSnapshot(int sequenceID)
```

Parameters

TYPE	NAME	DESCRIPTION
int	sequenceID	The sequence ID to fetch the snapshot for

Returns

TYPE	DESCRIPTION
ReplaySnapshot	The replay snapshot at the specified sequence id

Overrides

ReplayStorage.FetchSnapshot(int)

FetchSnapshot(float)

Recall a snapshot from the replay target based on the specified replay offset.

Declaration

```
public override ReplaySnapshot FetchSnapshot(float timeStamp)
```

Parameters

TYPE	NAME	DESCRIPTION
float	timeStamp	The time offset from the start of the recording pointing to the individual snapshot to recall

Returns

TYPE	DESCRIPTION
ReplaySnapshot	The replay snapshot at the specified offset

Overrides

ReplayStorage.FetchSnapshot(float)

Lock(ReplayOperation)

Called by the replay system when a lock should be created on this storage target, typically when a record or playback operation is started. Used to prevent other replay operations from accessing the same storage target.

Declaration

```
protected override void Lock(ReplayOperation operation)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayOperation	operation	The ReplayOperation that claimed the storage target

Overrides

ReplayStorage.Lock(ReplayOperation)

OnDispose()

Called when the storage target should be disposed to cleanup.

Declaration

```
protected override void OnDispose()
```

Overrides

ReplayStorage.OnDispose()

Prepare(ReplayStorageAction)

Called by the recording system to notify the active ReplayStorage of an upcoming event.

Declaration

```
public override void Prepare(ReplayStorageAction mode)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayStorageAction	mode	The ReplayStorageAction that the target should prepare for

Overrides

ReplayStorage.Prepare(ReplayStorageAction)

StoreSnapshot(ReplaySnapshot)

Store a replay snapshot in the replay target.

Declaration

```
public override void StoreSnapshot(ReplaySnapshot state)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplaySnapshot	state	The snapshot to store

Overrides

ReplayStorage.StoreSnapshot(ReplaySnapshot)

Unlock(ReplayOperation)

Called by the replay system when a lock should be released on this storage target, typically when a record or playback operation is ended.

Declaration

```
protected override void Unlock(ReplayOperation operation)
```

Parameters

TYPE	NAME	DESCRIPTION

TYPE	NAME	DESCRIPTION
ReplayOperation	operation	The ReplayOperation that created the lock

Overrides

ReplayStorage.Unlock(ReplayOperation)

Implements

IDisposable

Class ReplayMemoryStorage

Inheritance

object
ReplayStorage
ReplayMemoryStorage

Implements

IDisposable

Inherited Members

ReplayStorage.IsLocked
ReplayStorage.Metadata
ReplayStorage.PersistentData
ReplayStorage.IsDisposed
ReplayStorage.Dispose()
ReplayStorage.CopyTo(ReplayStorage)
ReplayStorage.CopyToAsync(ReplayStorage)
object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: UltimateReplay.Storage

Assembly: UltimateReplay.dll

Syntax

```
public sealed class ReplayMemoryStorage : ReplayStorage, IDisposable
```

Constructors

ReplayMemoryStorage(string, float)

Declaration

```
public ReplayMemoryStorage(string replayName = null, float rollingBufferDuration = -1)
```

Parameters

TYPE	NAME	DESCRIPTION
string	replayName	
float	rollingBufferDuration	

Properties

CanRead

Get a value indicating whether this storage target is readable.

Declaration

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

Property Value

TYPE	DESCRIPTION
bool	

Overrides

ReplayStorage.CanRead

CanWrite

Get a value indicating whether this storage target is writable.

Declaration

```
public override bool CanWrite { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

Overrides

ReplayStorage.CanWrite

Duration

The amount of time in seconds that this recording lasts.

Declaration

```
public override float Duration { get; }
```

Property Value

TYPE	DESCRIPTION
float	

Overrides

ReplayStorage.Duration

IdentitySize

Get the size in bytes required to serialize a ReplayIdentity.

Declaration

```
public override int IdentitySize { get; }
```

Property Value

TYPE	DESCRIPTION
int	

Overrides

ReplayStorage.IdentitySize

MemorySize

Get the total amount of bytes that this replay uses.

Declaration

```
public override int MemorySize { get; }
```

Property Value

TYPE	DESCRIPTION
int	

Overrides

ReplayStorage.MemorySize

RollingBufferDuration

Declaration

```
public float RollingBufferDuration { get; }
```

Property Value

TYPE	DESCRIPTION
float	

SnapshotSize

Get the total number of snapshots included in this replay.

Declaration

```
public override int SnapshotSize { get; }
```

Property Value

TYPE	DESCRIPTION
int	

Overrides

ReplayStorage.SnapshotSize

Methods

FetchSnapshot(int)

Recall a snapshot by its unique sequence id value. The sequence ID value indicates the snapshots 0-based index value for the recording sequence.

Declaration

```
public override ReplaySnapshot FetchSnapshot(int sequenceID)
```

Parameters

TYPE	NAME	DESCRIPTION
int	sequenceID	The sequence ID to fetch the snapshot for

Returns

TYPE	DESCRIPTION
ReplaySnapshot	The replay snapshot at the specified sequence id

Overrides

ReplayStorage.FetchSnapshot(int)

FetchSnapshot(float)

Recall a snapshot from the replay target based on the specified replay offset.

Declaration

```
public override ReplaySnapshot FetchSnapshot(float timeStamp)
```

Parameters

TYPE	NAME	DESCRIPTION
float	timeStamp	The time offset from the start of the recording pointing to the individual snapshot to recall

Returns

TYPE	DESCRIPTION
ReplaySnapshot	The replay snapshot at the specified offset

Overrides

ReplayStorage.FetchSnapshot(float)

FromBytes(byte[])

Declaration

```
public bool FromBytes(byte[] bytes)
```

Parameters

TYPE	NAME	DESCRIPTION
byte[]	bytes	

Returns

TYPE	DESCRIPTION
bool	

LoadFromFile(string)

Declaration

```
public bool LoadFromFile(string replayFile)
```

Parameters

TYPE	NAME	DESCRIPTION
string	replayFile	

Returns

TYPE	DESCRIPTION
bool	

LoadFromFileAsync(string)

Declaration

```
public ReplayAsyncOperation LoadFromFileAsync(string replayFile)
```

Parameters

TYPE	NAME	DESCRIPTION
string	replayFile	

Returns

TYPE	DESCRIPTION
ReplayAsyncOperation	

OnDispose()

Called when the storage target should be disposed to cleanup.

Declaration

```
protected override void OnDispose()
```

Overrides

ReplayStorage.OnDispose()

Prepare(ReplayStorageAction)

Called by the recording system to notify the active ReplayStorage of an upcoming event.

Declaration

```
public override void Prepare(ReplayStorageAction mode)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayStorageAction	mode	The ReplayStorageAction that the target should prepare for

Overrides

ReplayStorage.Prepare(ReplayStorageAction)

SaveToFile(string)

Declaration

```
public bool SaveToFile(string replayFile)
```

Parameters

TYPE	NAME	DESCRIPTION
string	replayFile	

Returns

TYPE	DESCRIPTION
bool	

SaveToFileAsync(string)

Declaration

```
public ReplayAsyncOperation SaveToFileAsync(string replayFile)
```

Parameters

TYPE	NAME	DESCRIPTION
string	replayFile	

Returns

TYPE	DESCRIPTION
ReplayAsyncOperation	

StoreSnapshot(ReplaySnapshot)

Store a replay snapshot in the replay target.

Declaration

```
public override void StoreSnapshot(ReplaySnapshot state)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplaySnapshot	state	The snapshot to store

Overrides

ReplayStorage.StoreSnapshot(ReplaySnapshot)

ToBytes()

Declaration

```
public byte[] ToBytes()
```

Returns

TYPE	DESCRIPTION
byte[]	

Implements

IDisposable

Class ReplayPersistentData

Inheritance

object

ReplayPersistentData

Implements

IReplayStreamSerialize

IReplayTokenSerialize

Inherited Members

object.Equals(object)

object.Equals(object, object)

object.GetHashCode()

object.GetType()

object.ReferenceEquals(object, object)

object.ToString()

Namespace: UltimateReplay.Storage

Assembly: UltimateReplay.dll

Syntax

```
public sealed class ReplayPersistentData : IReplayStreamSerialize, IReplayTokenSerialize
```

Properties

PersistentIdentities

Declaration

```
public IEnumerable<ReplayIdentity> PersistentIdentities { get; }
```

Property Value

TYPE	DESCRIPTION
IEnumerable<ReplayIdentity>	

PersistentIdentitiesByTimestamp

Declaration

```
public IEnumerable<ReplayIdentity> PersistentIdentitiesByTimestamp { get; }
```

Property Value

TYPE	DESCRIPTION
IEnumerable<ReplayIdentity>	

Methods

CopyTo(ReplayPersistentData)

Declaration

```
public bool CopyTo(ReplayPersistentData destination)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayPersistentData	destination	

Returns

TYPE	DESCRIPTION
bool	

FetchPersistentData(ReplayIdentity)

Declaration

```
public ReplayState FetchPersistentData(ReplayIdentity id)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayIdentity	id	

Returns

TYPE	DESCRIPTION
ReplayState	

FetchPersistentDataByTimestamp(ReplayIdentity, float)

Declaration

```
public ReplayState FetchPersistentDataByTimestamp(ReplayIdentity id, float timestamp)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayIdentity	id	
float	timestamp	

Returns

TYPE	DESCRIPTION
ReplayState	

HasPersistentData(ReplayIdentity)

Declaration

```
public bool HasPersistentData(ReplayIdentity id)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayIdentity	id	

Returns

TYPE	DESCRIPTION
bool	

HasPersistentDataByTimestamp(ReplayIdentity)

Declaration

```
public bool HasPersistentDataByTimestamp(ReplayIdentity id)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayIdentity	id	

Returns

TYPE	DESCRIPTION
bool	

StorePersistentData(ReplayIdentity, ReplayState)

Declaration

```
public void StorePersistentData(ReplayIdentity id, ReplayState state)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayIdentity	id	
ReplayState	state	

StorePersistentDataByTimestamp(ReplayIdentity, float, ReplayState)

Declaration

```
public void StorePersistentDataByTimestamp(ReplayIdentity id, float timestamp, ReplayState state)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayIdentity	id	
float	timestamp	
ReplayState	state	

Implements

IReplayStreamSerialize
IReplayTokenSerialize

Class ReplaySegment

Inheritance

object

ReplaySegment

Implements

IDisposable

IReplayReusable

IReplayStreamSerialize

IReplayTokenSerialize

Inherited Members

object.Equals(object)

object.Equals(object, object)

object.GetHashCode()

object.GetType()

object.ReferenceEquals(object, object)

object.ToString()

Namespace: UltimateReplay.Storage

Assembly: UltimateReplay.dll

Syntax

```
public sealed class ReplaySegment : IDisposable, IReplayReusable, IReplayStreamSerialize,
IReplayTokenSerialize
```

Constructors

ReplaySegment()

Declaration

```
public ReplaySegment()
```

ReplaySegment(int, int)

Declaration

```
public ReplaySegment(int segmentID, int snapshotCount)
```

Parameters

TYPE	NAME	DESCRIPTION
int	segmentID	
int	snapshotCount	

Fields

pool

Declaration

```
public static readonly ReplayInstancePool<ReplaySegment> pool
```

Field Value

TYPE	DESCRIPTION
ReplayInstancePool<ReplaySegment>	

Properties

EndSequenceID

Declaration

```
public int EndSequenceID { get; }
```

Property Value

TYPE	DESCRIPTION
int	

EndSnapshot

Declaration

```
public ReplaySnapshot EndSnapshot { get; }
```

Property Value

TYPE	DESCRIPTION
ReplaySnapshot	

EndTimeStamp

Declaration

```
public float EndTimeStamp { get; }
```

Property Value

TYPE	DESCRIPTION
float	

IsCompressed

Declaration

```
public bool IsCompressed { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

IsEmpty

Declaration

```
public bool IsEmpty { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

IsFull

Declaration

```
public bool IsFull { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

SegmentID

Declaration

```
public int SegmentID { get; }
```

Property Value

TYPE	DESCRIPTION
int	

SnapshotCapacity

Declaration

```
public int SnapshotCapacity { get; }
```

Property Value

TYPE	DESCRIPTION
int	

SnapshotCount

Declaration

```
public int SnapshotCount { get; }
```

Property Value

TYPE	DESCRIPTION
int	

Snapshots

Declaration

```
public IEnumerable<ReplaySnapshot> Snapshots { get; }
```

Property Value

TYPE	DESCRIPTION
IEnumerable<ReplaySnapshot>	

StartSequenceID

Declaration

```
public int StartSequenceID { get; }
```

Property Value

TYPE	DESCRIPTION
int	

StartSnapshot

Declaration

```
public ReplaySnapshot StartSnapshot { get; }
```

Property Value

TYPE	DESCRIPTION
ReplaySnapshot	

StartTimeStamp

Declaration

```
public float StartTimeStamp { get; }
```

Property Value

TYPE	DESCRIPTION
float	

Methods

AddSnapshot(ReplaySnapshot)

Declaration

```
public void AddSnapshot(ReplaySnapshot snapshot)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplaySnapshot	snapshot	

CompressSegment()

Segment compression algorithm. Lossless algorithm which works by replacing replay state data that uses a hash seen in previous snapshots in this segment with a pointer object that links to the former snapshot data via index. This means that is snapshot data is unchanged for a few snapshots, it is possible to eliminate many replay state containers since they contain duplicate data.

Declaration

```
public void CompressSegment()
```

DecompressSegment()

Declaration

```
public void DecompressSegment()
```

Dispose()

Declaration

```
public void Dispose()
```

FetchSnapshot(int)

Declaration

```
public ReplaySnapshot FetchSnapshot(int sequenceId)
```

Parameters

TYPE	NAME	DESCRIPTION
int	sequenceId	

Returns

TYPE	DESCRIPTION
ReplaySnapshot	

FetchSnapshot(float)

Declaration

```
public ReplaySnapshot FetchSnapshot(float timeStamp)
```

Parameters

TYPE	NAME	DESCRIPTION
float	timeStamp	

Returns

TYPE	DESCRIPTION
ReplaySnapshot	

Implements

IDisposable
IReplayReusable
IReplayStreamSerialize
IReplayTokenSerialize

Class ReplaySnapshot

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

object
ReplaySnapshot

Implements

IDisposable
IReplayReusable
IReplayStreamSerialize
IReplayTokenSerialize

Inherited Members

object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.ReferenceEquals(object, object)

Namespace: UltimateReplay.Storage

Assembly: UltimateReplay.dll

Syntax

```
[Serializable]
public sealed class ReplaySnapshot : IDisposable, IReplayReusable, IReplayStreamSerialize,
IReplayTokenSerialize
```

Constructors

ReplaySnapshot(float, int)

Create a new snapshot with the specified time stamp.

Declaration

```
public ReplaySnapshot(float timeStamp, int sequenceID)
```

Parameters

TYPE	NAME	DESCRIPTION
float	timeStamp	The time stamp to give to this snapshot
int	sequenceID	

Fields

pool

Declaration

```
public static readonly ReplayInstancePool<ReplaySnapshot> pool
```

Field Value

TYPE	DESCRIPTION
ReplayInstancePool<ReplaySnapshot>	

startSequenceID

Declaration

```
public const int startSequenceID = 1
```

Field Value

TYPE	DESCRIPTION
int	

Properties

Identities

Declaration

```
public HashSet<ReplayIdentity> Identities { get; }
```

Property Value

TYPE	DESCRIPTION
HashSet<ReplayIdentity>	

SequenceID

The unique sequence id value for this snapshot. A sequence id is an ordered value starting from startSequenceID and counting upwards. You can get the previous snapshot in the replay using SequenceID -1, or the next snapshot using SequenceID +1.

Declaration

```
public int SequenceID { get; }
```

Property Value

TYPE	DESCRIPTION
int	

Size

Get the size in bytes of the snapshot data.

Declaration

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

Property Value

TYPE	DESCRIPTION
int	

TimeStamp

The time stamp for this snapshot. The time stamp is used to identify the snapshot location in the sequence.

Declaration

```
public float TimeStamp { get; }
```

Property Value

TYPE	DESCRIPTION
float	

Methods

Dispose()

Declaration

```
public void Dispose()
```

OnReplayStreamDeserialize(BinaryReader)

Called by the replay system when this ReplaySnapshot should be deserialized from binary.

Declaration

```
public void OnReplayStreamDeserialize(BinaryReader reader)
```

Parameters

TYPE	NAME	DESCRIPTION
BinaryReader	reader	The binary stream to read the data from

OnReplayStreamSerialize(BinaryWriter)

Called by the replay system when this ReplaySnapshot should be serialized to binary.

Declaration

```
public void OnReplayStreamSerialize(BinaryWriter writer)
```

Parameters

TYPE	NAME	DESCRIPTION
BinaryWriter	writer	The binary stream to write te data to

RecordSnapshot(ReplayIdentity, ReplayState)

Registers the specified replay state with this snapshot. The specified identity is used during playback to ensure that the replay objects receives the correct state to deserialize.

Declaration

```
public void RecordSnapshot(ReplayIdentity identity, ReplayState state)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayIdentity	identity	The identity of the object that was serialized
ReplayState	state	The state data for the object

Reset()

Clears all state information from the snapshot but keeps the time stamp.

Declaration

```
public void Reset()
```

RestoreReplayObjects(ReplayScene, ReplayPersistentData)

Attempts to restore any replay objects that were spawned or despawned during this snapshot.

Declaration

```
public void RestoreReplayObjects(ReplayScene scene, ReplayPersistentData persistentData)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayScene	scene	
ReplayPersistentData	persistentData	

RestoreSnapshot(ReplayIdentity)

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.

Declaration

```
public ReplayState RestoreSnapshot(ReplayIdentity identity)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayIdentity	identity	The identity of the object to deserialize

Returns

TYPE	DESCRIPTION
ReplayState	The state information for the specified identity or null if the identity does not exist

ToString()

Declaration

```
public override string ToString()
```

Returns

TYPE	DESCRIPTION
string	

Overrides

object.ToString()

Implements

IDisposable
IReplayReusable
IReplayStreamSerialize
IReplayTokenSerialize

Struct ReplaySnapshot.ReplayObjectCreatedData

Implements

IReplaySerialize

Inherited Members

ValueType.Equals(object)
ValueType.GetHashCode()
ValueType.ToString()
object.Equals(object, object)
object.GetType()
object.ReferenceEquals(object, object)

Namespace: UltimateReplay.Storage

Assembly: UltimateReplay.dll

Syntax

```
public struct ReplaySnapshot.ReplayObjectCreatedData : IReplaySerialize
```

Fields

flags

Declaration

```
[ReplayTokenSerialize("Serialize Flags")]
public ReplaySnapshot.ReplayObjectCreatedData.ReplaySerializeFlags flags
```

Field Value

TYPE	DESCRIPTION
ReplaySnapshot.ReplayObjectCreatedData.ReplaySerializeFlags	

objectIdentity

Initial replay object identity.

Declaration

```
[ReplayTokenSerialize("Object Identity")]
public ReplayIdentity objectIdentity
```

Field Value

TYPE	DESCRIPTION
ReplayIdentity	

observedComponentIdentities

The replay ids for all observed components ordered by array index.

Declaration

```
[ReplayTokenSerialize("Observed Component Identities")]
public ReplayIdentity[] observedComponentIdentities
```

Field Value

TYPE	DESCRIPTION
ReplayIdentity[]	

parentIdentity

Initial parent data.

Declaration

```
[ReplayTokenSerialize("Parent Identity")]
public ReplayIdentity parentIdentity
```

Field Value

TYPE	DESCRIPTION
ReplayIdentity	

position

Initial position data.

Declaration

```
[ReplayTokenSerialize("Position")]
public Vector3 position
```

Field Value

TYPE	DESCRIPTION
Vector3	

rotation

Initial rotation data.

Declaration

```
[ReplayTokenSerialize("Rotation")]
public Quaternion rotation
```

Field Value

TYPE	DESCRIPTION
Quaternion	

scale

Initial scale data.

Declaration

```
[ReplayTokenSerialize("Scale")]
public Vector3 scale
```

Field Value

TYPE	DESCRIPTION
Vector3	

timestamp

The timestamp when this object was instantiated.

Declaration

```
[ReplayTokenSerialize("Time Stamp")]
public float timestamp
```

Field Value

TYPE	DESCRIPTION
float	

Properties

InitialFlags

Declaration

```
public ReplaySnapshot.ReplayObjectCreatedData.ReplaySerializeFlags InitialFlags { get; }
```

Property Value

TYPE	DESCRIPTION
ReplaySnapshot.ReplayObjectCreatedData.ReplaySerializeFlags	

Methods

FromReplayObject(float, ReplayObject)

Declaration

```
public static ReplaySnapshot.ReplayObjectCreatedData FromReplayObject(float timeStamp, ReplayObject obj)
```

Parameters

TYPE	NAME	DESCRIPTION
float	timeStamp	
ReplayObject	obj	

Returns

TYPE	DESCRIPTION
ReplaySnapshot.ReplayObjectCreatedData	

GenerateDataFlags()

Declaration

```
public void GenerateDataFlags()
```

OnReplayDeserialize(ReplayState)

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

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState to read the data from

OnReplaySerialize(ReplayState)

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

Declaration

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

Parameters

TYPE	NAME	DESCRIPTION
ReplayState	state	The ReplayState to write the data to

Implements

IReplaySerialize

Enum ReplaySnapshot.ReplayObjectCreatedData.ReplaySerializeFlags

Represents initial data that may be stored by an object.

Namespace: UltimateReplay.Storage

Assembly: UltimateReplay.dll

Syntax

```
[Flags]
public enum ReplaySnapshot.ReplayObjectCreatedData.ReplaySerializeFlags : byte
```

Fields

NAME	DESCRIPTION
None	No initial data is stored.
Parent	Initial parent is recorded.
Position	Initial position is recorded.
Rotation	Initial rotation is recorded.
Scale	Initial scale is recorded.

Enum ReplaySnapshotStorableType

Namespace: UltimateReplay.Storage

Assembly: UltimateReplay.dll

Syntax

```
public enum ReplaySnapshotStorableType
```

Fields

NAME	DESCRIPTION
StatePointer	The storage element points to a replay data segment.
StateStorage	The storage element contains replay data.

Class ReplayStorage

Represents and abstract storage device capable of holding recorded state data for playback at a later date. Depending upon implementation, a ReplayStorage may be volatile or non-volatile.

Inheritance

object
ReplayStorage
ReplayFileStorage
ReplayHighlightReelStorage
ReplayMemoryStorage
ReplayStreamStorage

Implements

IDisposable

Inherited Members

object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: UltimateReplay.Storage

Assembly: UltimateReplay.dll

Syntax

```
[Serializable]
public abstract class ReplayStorage : IDisposable
```

Constructors

ReplayStorage(string)

Create a new instance.

Declaration

```
protected ReplayStorage(string replayName = null)
```

Parameters

TYPE	NAME	DESCRIPTION
string	replayName	Optional name for the replay

Fields

metadata

Declaration

```
protected ReplayMetadata metadata
```

Field Value

TYPE	DESCRIPTION
ReplayMetadata	

persistentData

Declaration

```
protected ReplayPersistentData persistentData
```

Field Value

TYPE	DESCRIPTION
ReplayPersistentData	

Properties

CanRead

Get a value indicating whether this storage target is readable.

Declaration

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

Property Value

TYPE	DESCRIPTION
bool	

CanWrite

Get a value indicating whether this storage target is writable.

Declaration

```
public abstract bool CanWrite { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

Duration

The amount of time in seconds that this recording lasts.

Declaration

```
public abstract float Duration { get; }
```

Property Value

TYPE	DESCRIPTION
float	

IdentitySize

Get the size in bytes required to serialize a ReplayIdentity.

Declaration

```
public abstract int IdentitySize { get; }
```

Property Value

TYPE	DESCRIPTION
int	

IsDisposed

Return a value indicating whether this storage is currently disposed. A disposed storage target should no longer be used at all.

Declaration

```
public bool IsDisposed { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

IsLocked

Return a value indicating whether this storage is currently locked to a replay operation. A locked storage target cannot be used by another replay operation.

Declaration

```
public bool IsLocked { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

MemorySize

Get the total amount of bytes that this replay uses.

Declaration

```
public abstract int MemorySize { get; }
```

Property Value

TYPE	DESCRIPTION
int	

Metadata

The user metadata associated with this storage target. Derive from ReplayMetadata and declare additional serialized fields in order to store custom metadata in a replay.

Declaration

```
public virtual ReplayMetadata Metadata { get; set; }
```

Property Value

TYPE	DESCRIPTION
ReplayMetadata	

PersistentData

The persistent data associated with this storage target. Typically used to store single shot data or object instantate data such as initial position, parent, etc.

Declaration

```
public virtual ReplayPersistentData PersistentData { get; set; }
```

Property Value

TYPE	DESCRIPTION
ReplayPersistentData	

SnapshotSize

Get the total number of snapshots included in this replay.

Declaration

```
public abstract int SnapshotSize { get; }
```

Property Value

TYPE	DESCRIPTION
int	

Methods

CheckDisposed()

Throws an exception if the current storage is disposed.

Declaration

```
protected void CheckDisposed()
```

Exceptions

TYPE	CONDITION
ObjectDisposedException	Storage is disposed

CopyTo(ReplayStorage)

Copy the saved replay to the specified storage target. Duration must be greater than zero (Must contain data) otherwise this method will return false. Destination Duration must be zero (Must NOT contain data) otherwise this method will return false. Note that this operation can take some time to complete depending upon the size of the replay, and will block the calling thread until completed.

Declaration

```
public bool CopyTo(ReplayStorage destination)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayStorage	destination	The target ReplayStorage where data should be copied

Returns

TYPE	DESCRIPTION
bool	True if the copy was successful or false if not

Exceptions

TYPE	CONDITION
ArgumentNullException	Destination storage is null
ObjectDisposedException	This ReplayStorage or destination ReplayStorage is disposed

See Also

CopyToAsync(ReplayStorage)

CopyToAsync(ReplayStorage)

Copy the saved replay to the specified storage target without blocking the main thread. Duration must be greater than zero (Must contain data) otherwise this method will return failed operation. Destination Duration must be zero (Must NOT contain data) otherwise this method will return failed operation. Note that this operation can take some time to complete depending upon the size of the replay, but the main thread will not be blocked. The resulting ReplayAsyncOperation can be awaited in a coroutine if you need to wait for completion without blocking (Loading screen for example).

Declaration

```
public ReplayAsyncOperation CopyToAsync(ReplayStorage destination)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayStorage	destination	The target ReplayStorage where data should be copied

Returns

TYPE	DESCRIPTION

TYPE	DESCRIPTION
ReplayAsyncOperation	A ReplayAsyncOperation that contains information about the current state of the copy operation and can be awaited in a coroutine using `yield return`

Exceptions

TYPE	CONDITION
ArgumentNullException	Destination storage is null
ObjectDisposedException	This ReplayStorage or destination ReplayStorage is disposed

Dispose()

Release the storage target. Should always be called when you have finished using a storage target so that memory can be recycled and file/stream handles can be released. NOTE: The replay system will not call Dispose in normal circumstances and it must be called manually.

Declaration

```
public void Dispose()
```

FetchSnapshot(int)

Recall a snapshot by its unique sequence id value. The sequence ID value indicates the snapshots 0-based index value for the recording sequence.

Declaration

```
public abstract ReplaySnapshot FetchSnapshot(int sequenceID)
```

Parameters

TYPE	NAME	DESCRIPTION
int	sequenceID	The sequence ID to fetch the snapshot for

Returns

TYPE	DESCRIPTION
ReplaySnapshot	The replay snapshot at the specified sequence id

FetchSnapshot(float)

Recall a snapshot from the replay target based on the specified replay offset.

Declaration

```
public abstract ReplaySnapshot FetchSnapshot(float timeStamp)
```

Parameters

TYPE	NAME	DESCRIPTION
float	timeStamp	The time offset from the start of the recording pointing to the individual snapshot to recall

Returns

TYPE	DESCRIPTION
ReplaySnapshot	The replay snapshot at the specified offset

Lock(ReplayOperation)

Called by the replay system when a lock should be created on this storage target, typically when a record or playback operation is started. Used to prevent other replay operations from accessing the same storage target.

Declaration

```
protected virtual void Lock(ReplayOperation operation)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayOperation	operation	The ReplayOperation that claimed the storage target

OnDispose()

Called when the storage target should be disposed to cleanup.

Declaration

```
protected abstract void OnDispose()
```

Prepare(ReplayStorageAction)

Called by the recording system to notify the active ReplayStorage of an upcoming event.

Declaration

```
public abstract void Prepare(ReplayStorageAction mode)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayStorageAction	mode	The ReplayStorageAction that the target should prepare for

StoreSnapshot(ReplaySnapshot)

Store a replay snapshot in the replay target.

Declaration

```
public abstract void StoreSnapshot(ReplaySnapshot state)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplaySnapshot	state	The snapshot to store

Unlock(ReplayOperation)

Called by the replay system when a lock should be released on this storage target, typically when a record or playback operation is ended.

Declaration

```
protected virtual void Unlock(ReplayOperation operation)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayOperation	operation	The ReplayOperation that created the lock

Implements

IDisposable

Enum ReplayStorageAction

Represents a task that can be issued to a ReplayStorage.

Namespace: UltimateReplay.Storage

Assembly: UltimateReplay.dll

Syntax

```
public enum ReplayStorageAction
```

Fields

NAME	DESCRIPTION
Commit	The replay target should commit all data currently in memory to its end destination. Similar to a flush method.
Discard	The replay target should discard any recorded data.
Read	The replay target should prepare for subsequent read requests.
Write	The replay target should prepare for subsequent write requests.

Class ReplayStreamSource

Represents a data stream source that a replay stream can work with.

Inheritance

object
ReplayStreamSource

Implements

IDisposable

Inherited Members

object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: UltimateReplay.Storage

Assembly: UltimateReplay.dll

Syntax

```
public abstract class ReplayStreamSource : IDisposable
```

Properties

CanRead

Return a value indicating whether the current stream source can be read from.

Declaration

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

Property Value

TYPE	DESCRIPTION
bool	

CanWrite

Return a value indicating whether the current stream source can be written to.

Declaration

```
public abstract bool CanWrite { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

Methods

Dispose()

Declaration

```
public void Dispose()
```

DisposeStream(Stream)

In derived types, should dispose, close or finialie the spcified stream as it will no longer be used by the owning ReplayStreamStorage. The specified input stream will be a stream object created by either OpenForReading() or OpenForWriting(). The defualt behaivour will simply call Dispose().

Declaration

```
protected virtual void DisposeStream(Stream input)
```

Parameters

TYPE	NAME	DESCRIPTION
Stream	input	The target stream object to dispose

FromData(byte[])

Create a ReplayStreamSource from the specified byte array data. The specified array data must contain a valid replay data stream or playback will fail.

Declaration

```
public static ReplayStreamSource FromData(byte[] data)
```

Parameters

TYPE	NAME	DESCRIPTION
byte[]	data	Data source array

Returns

TYPE	DESCRIPTION
ReplayStreamSource	A ReplayStreamSource created from the specified input data

FromFile(string)

Declaration

```
public static ReplayStreamSource FromFile(string filePath)
```

Parameters

TYPE	NAME	DESCRIPTION
string	filePath	

Returns

TYPE	DESCRIPTION
ReplayStreamSource	

FromStream(Stream)

Declaration

```
public static ReplayStreamSource FromStream(Stream inputStream)
```

Parameters

TYPE	NAME	DESCRIPTION
Stream	inputStream	

Returns

TYPE	DESCRIPTION
ReplayStreamSource	

OpenForReading()

In derived types, should return an opened stream ready to receive read operations, or null if the stream could not be initialized. The resulting stream should support seeking operations.

Declaration

```
protected abstract Stream OpenForReading()
```

Returns

TYPE	DESCRIPTION
Stream	A valid stream object containing replay data ready for reading

OpenForWriting()

In derived types, should return an opened stream ready to receive write operations, or null if the stream could not be initialized. The resulting stream should support seeking, Position, and Length operations.

Declaration

```
protected abstract Stream OpenForWriting()
```

Returns

TYPE	DESCRIPTION
Stream	A valid stream object containing replay data ready for writing

OpenRead()

Declaration

```
public Stream OpenRead()
```

Returns

TYPE	DESCRIPTION
Stream	

OpenWrite()

Declaration

```
public Stream OpenWrite()
```

Returns

TYPE	DESCRIPTION
Stream	

Implements

IDisposable

Class ReplayStreamStorage

Inheritance

object
ReplayStorage
ReplayStreamStorage

Implements

IDisposable

Inherited Members

ReplayStorage.metadata
ReplayStorage.persistentData
ReplayStorage.IsLocked
ReplayStorage.Metadata
ReplayStorage.PersistentData
ReplayStorage.IsDisposed
ReplayStorage.CheckDisposed()
ReplayStorage.Dispose()
ReplayStorage.Lock(ReplayOperation)
ReplayStorage.Unlock(ReplayOperation)
ReplayStorage.CopyTo(ReplayStorage)
ReplayStorage.CopyToAsync(ReplayStorage)
object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: UltimateReplay.Storage

Assembly: UltimateReplay.dll

Syntax

```
public abstract class ReplayStreamStorage : ReplayStorage, IDisposable
```

Constructors

ReplayStreamStorage(string, bool)

Declaration

```
protected ReplayStreamStorage(string replayName = null, bool useSegmentCompression = false)
```

Parameters

TYPE	NAME	DESCRIPTION
string	replayName	
bool	useSegmentCompression	

Fields

readStream

Declaration

```
protected Stream readStream
```

Field Value

TYPE	DESCRIPTION
Stream	

snapshotsPerSegment

Declaration

```
protected int snapshotsPerSegment
```

Field Value

TYPE	DESCRIPTION
int	

useSegmentCompression

Declaration

```
protected bool useSegmentCompression
```

Field Value

TYPE	DESCRIPTION
bool	

writeStream

Declaration

```
protected Stream writeStream
```

Field Value

TYPE	DESCRIPTION
Stream	

Properties

CanRead

Get a value indicating whether this storage target is readable.

Declaration

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

Property Value

TYPE	DESCRIPTION
bool	

Overrides

ReplayStorage.CanRead

CanWrite

Get a value indicating whether this storage target is writable.

Declaration

```
public override bool CanWrite { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

Overrides

ReplayStorage.CanWrite

Duration

The amount of time in seconds that this recording lasts.

Declaration

```
public override float Duration { get; }
```

Property Value

TYPE	DESCRIPTION
float	

Overrides

ReplayStorage.Duration

IdentitySize

Get the size in bytes required to serialize a ReplayIdentity.

Declaration

```
public override int IdentitySize { get; }
```

Property Value

TYPE	DESCRIPTION
int	

Overrides

ReplayStorage.IdentitySize

IsBuffering

Declaration

```
public bool IsBuffering { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

MemorySize

Get the total amount of bytes that this replay uses.

Declaration

```
public override int MemorySize { get; }
```

Property Value

TYPE	DESCRIPTION
int	

Overrides

ReplayStorage.MemorySize

SnapshotSize

Get the total number of snapshots included in this replay.

Declaration

```
public override int SnapshotSize { get; }
```

Property Value

TYPE	DESCRIPTION
int	

Overrides

ReplayStorage.SnapshotSize

StreamSource

Declaration

```
protected abstract ReplayStreamSource StreamSource { get; }
```

Property Value

TYPE	DESCRIPTION
ReplayStreamSource	

Methods

FetchSnapshot(int)

Recall a snapshot by its unique sequence id value. The sequence ID value indicates the snapshots 0-based index value for the recording sequence.

Declaration

```
public override ReplaySnapshot FetchSnapshot(int sequenceID)
```

Parameters

TYPE	NAME	DESCRIPTION
int	sequenceID	The sequence ID to fetch the snapshot for

Returns

TYPE	DESCRIPTION
ReplaySnapshot	The replay snapshot at the specified sequence id

Overrides

ReplayStorage.FetchSnapshot(int)

FetchSnapshot(float)

Recall a snapshot from the replay target based on the specified replay offset.

Declaration

```
public override ReplaySnapshot FetchSnapshot(float timeStamp)
```

Parameters

TYPE	NAME	DESCRIPTION
float	timeStamp	The time offset from the start of the recording pointing to the individual snapshot to recall

Returns

TYPE	DESCRIPTION
ReplaySnapshot	The replay snapshot at the specified offset

Overrides

ReplayStorage.FetchSnapshot(float)

FromBytes(byte[])

Declaration

```
public static ReplayStreamStorage FromBytes(byte[] bytes)
```

Parameters

TYPE	NAME	DESCRIPTION
byte[]	bytes	

Returns

TYPE	DESCRIPTION
ReplayStreamStorage	

FromBytes(byte[], int, int)

Declaration

```
public static ReplayStreamStorage FromBytes(byte[] bytes, int index, int count)
```

Parameters

TYPE	NAME	DESCRIPTION
byte[]	bytes	
int	index	
int	count	

Returns

TYPE	DESCRIPTION
ReplayStreamStorage	

FromJsonString(string, Encoding)

Declaration

```
public static ReplayStreamStorage FromJsonString(string json, Encoding encoding = null)
```

Parameters

TYPE	NAME	DESCRIPTION
string	json	
Encoding	encoding	

Returns

TYPE	DESCRIPTION
ReplayStreamStorage	

FromStream(Stream, string, ReplayStreamType, bool, CompressionLevel)

Declaration

```
public static ReplayStreamStorage FromStream(Stream stream, string replayName = null, ReplayStreamType
streamType = ReplayStreamType.Default, bool useSegmentCompression = true, CompressionLevel
blockCompressionLevel = CompressionLevel.Optimal)
```

Parameters

TYPE	NAME	DESCRIPTION
Stream	stream	
string	replayName	
ReplayStreamType	streamType	

TYPE	NAME	DESCRIPTION
bool	useSegmentCompression	
CompressionLevel	blockCompressionLevel	

Returns

TYPE	DESCRIPTION
ReplayStreamStorage	

FromStreamBinary(Stream, string, bool, CompressionLevel)

Declaration

```
public static ReplayStreamStorage FromStreamBinary(Stream stream, string replayName = null, bool
useSegmentCompression = true, CompressionLevel blockCompressionLevel = CompressionLevel.Optimal)
```

Parameters

TYPE	NAME	DESCRIPTION
Stream	stream	
string	replayName	
bool	useSegmentCompression	
CompressionLevel	blockCompressionLevel	

Returns

TYPE	DESCRIPTION
ReplayStreamStorage	

FromStreamBson(Stream, string)

Declaration

```
public static ReplayStreamStorage FromStreamBson(Stream stream, string replayName = null)
```

Parameters

TYPE	NAME	DESCRIPTION
Stream	stream	
string	replayName	

Returns

TYPE	DESCRIPTION
ReplayStreamStorage	

FromStreamJson(Stream, string)

Declaration

```
public static ReplayStreamStorage FromStreamJson(Stream stream, string replayName = null)
```

Parameters

TYPE	NAME	DESCRIPTION
Stream	stream	
string	replayName	

Returns

TYPE	DESCRIPTION
ReplayStreamStorage	

LoadStreamCompletely()

Declaration

```
public void LoadStreamCompletely()
```

LoadStreamCompletelyAsync()

Declaration

```
public ReplayAsyncOperation LoadStreamCompletelyAsync()
```

Returns

TYPE	DESCRIPTION
ReplayAsyncOperation	

OnDispose()

Called when the storage target should be disposed to cleanup.

Declaration

```
protected override void OnDispose()
```

Overrides

ReplayStorage.OnDispose()

OnStreamCommit(Stream)

Declaration

```
protected virtual void OnStreamCommit(Stream writeStream)
```

Parameters

TYPE	NAME	DESCRIPTION
Stream	writeStream	

OnStreamOpenRead(Stream)

Declaration

```
protected virtual void OnStreamOpenRead(Stream readStream)
```

Parameters

TYPE	NAME	DESCRIPTION
Stream	readStream	

OnStreamOpenWrite(Stream)

Declaration

```
protected virtual void OnStreamOpenWrite(Stream writeStream)
```

Parameters

TYPE	NAME	DESCRIPTION
Stream	writeStream	

OnStreamSeek(Stream, long)

Declaration

```
protected virtual void OnStreamSeek(Stream stream, long offset)
```

Parameters

TYPE	NAME	DESCRIPTION
Stream	stream	
long	offset	

Prepare(ReplayStorageAction)

Called by the recording system to notify the active ReplayStorage of an upcoming event.

Declaration

```
public override void Prepare(ReplayStorageAction mode)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayStorageAction	mode	The ReplayStorageAction that the target should prepare for

Overrides

ReplayStorage.Prepare(ReplayStorageAction)

ReadBytesCompletely(byte[])

Declaration

```
public static ReplayStreamStorage ReadBytesCompletely(byte[] bytes)
```

Parameters

TYPE	NAME	DESCRIPTION
byte[]	bytes	

Returns

TYPE	DESCRIPTION
ReplayStreamStorage	

ReadBytesCompletely(byte[], int, int)

Declaration

```
public static ReplayStreamStorage ReadBytesCompletely(byte[] bytes, int index, int count)
```

Parameters

TYPE	NAME	DESCRIPTION
byte[]	bytes	
int	index	
int	count	

Returns

TYPE	DESCRIPTION
ReplayStreamStorage	

ReadBytesCompletelyAsync(byte[])

Declaration

```
public static ReplayAsyncOperation<ReplayStreamStorage> ReadBytesCompletelyAsync(byte[] bytes)
```

Parameters

TYPE	NAME	DESCRIPTION
byte[]	bytes	

Returns

TYPE	DESCRIPTION
ReplayAsyncOperation<ReplayStreamStorage>	

ReadBytesCompletelyAsync(byte[], int, int)

Declaration

```
public static ReplayAsyncOperation<ReplayStreamStorage> ReadBytesCompletelyAsync(byte[] bytes, int index, int
count)
```

Parameters

TYPE	NAME	DESCRIPTION
byte[]	bytes	
int	index	
int	count	

Returns

TYPE	DESCRIPTION
ReplayAsyncOperation<ReplayStreamStorage>	

ReadMetadataOnly(Stream, ReplayStreamType)

Declaration

```
public static ReplayMetadata ReadMetadataOnly(Stream stream, ReplayStreamType streamType =
ReplayStreamType.Default)
```

Parameters

TYPE	NAME	DESCRIPTION
Stream	stream	
ReplayStreamType	streamType	

Returns

TYPE	DESCRIPTION
ReplayMetadata	

ReadMetadataOnlyAsync(Stream, ReplayStreamType)

Declaration

```
public static ReplayAsyncOperation<ReplayMetadata> ReadMetadataOnlyAsync(Stream stream, ReplayStreamType type
= ReplayStreamType.Default)
```

Parameters

TYPE	NAME	DESCRIPTION
Stream	stream	
ReplayStreamType	type	

Returns

TYPE	DESCRIPTION
ReplayAsyncOperation<ReplayMetadata>	

ReadMetadataOnlyAsync<T>(Stream, ReplayStreamType)

Declaration

```
public static ReplayAsyncOperation<T> ReadMetadataOnlyAsync<T>(Stream stream, ReplayStreamType streamType =
ReplayStreamType.Default) where T : ReplayMetadata
```

Parameters

TYPE	NAME	DESCRIPTION
Stream	stream	
ReplayStreamType	streamType	

Returns

TYPE	DESCRIPTION
ReplayAsyncOperation<T>	

Type Parameters

NAME	DESCRIPTION
T	

ReadMetadataOnly<T>(Stream, ReplayStreamType)

Declaration

```
public static T ReadMetadataOnly<T>(Stream stream, ReplayStreamType streamType = ReplayStreamType.Default)
where T : ReplayMetadata
```

Parameters

TYPE	NAME	DESCRIPTION
Stream	stream	
ReplayStreamType	streamType	

Returns

TYPE	DESCRIPTION
T	

Type Parameters

NAME	DESCRIPTION
T	

ReadStreamCompletely(Stream, ReplayStreamType)

Declaration

```
public static ReplayStreamStorage ReadStreamCompletely(Stream stream, ReplayStreamType streamType =
ReplayStreamType.Default)
```

Parameters

TYPE	NAME	DESCRIPTION
Stream	stream	
ReplayStreamType	streamType	

Returns

TYPE	DESCRIPTION
ReplayStreamStorage	

ReadStreamCompletelyAsync(Stream, ReplayStreamType)

Declaration

```
public static ReplayAsyncOperation<ReplayStreamStorage> ReadStreamCompletelyAsync(Stream stream,
ReplayStreamType streamType = ReplayStreamType.Default)
```

Parameters

TYPE	NAME	DESCRIPTION
Stream	stream	
ReplayStreamType	streamType	

Returns

TYPE	DESCRIPTION
ReplayAsyncOperation<ReplayStreamStorage>	

StoreSnapshot(ReplaySnapshot)

Store a replay snapshot in the replay target.

Declaration

```
public override void StoreSnapshot(ReplaySnapshot state)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplaySnapshot	state	The snapshot to store

Overrides

ReplayStorage.StoreSnapshot(ReplaySnapshot)

ThreadReadReplayHeader(ref ReplayStreamHeader)

Declaration

```
protected abstract void ThreadReadReplayHeader(ref ReplayStreamStorage.ReplayStreamHeader header)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayStreamStorage.ReplayStreamHeader	header	

ThreadReadReplayMetadata(Type, ref ReplayMetadata)

Declaration

```
protected abstract void ThreadReadReplayMetadata(Type metadataType, ref ReplayMetadata metadata)
```

Parameters

TYPE	NAME	DESCRIPTION
Type	metadataType	
ReplayMetadata	metadata	

ThreadReadReplayPersistentData(ref ReplayPersistentData)

Declaration

```
protected abstract void ThreadReadReplayPersistentData(ref ReplayPersistentData data)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayPersistentData	data	

ThreadReadReplaySegment(ref ReplaySegment, int)

Declaration

```
protected abstract void ThreadReadReplaySegment(ref ReplaySegment segment, int segmentID)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplaySegment	segment	
int	segmentID	

ThreadReadReplaySegmentTable(ref ReplaySegmentTable)

Declaration

```
protected abstract void ThreadReadReplaySegmentTable(ref ReplayStreamStorage.ReplaySegmentTable table)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayStreamStorage.ReplaySegmentTable	table	

ThreadWriteReplayHeader(ReplayStreamHeader)

Declaration

```
protected abstract void ThreadWriteReplayHeader(ReplayStreamStorage.ReplayStreamHeader header)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayStreamStorage.ReplayStreamHeader	header	

ThreadWriteReplayMetadata(ReplayMetadata)

Declaration

```
protected abstract void ThreadWriteReplayMetadata(ReplayMetadata metadata)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayMetadata	metadata	

ThreadWriteReplayPersistentData(ReplayPersistentData)

Declaration

```
protected abstract void ThreadWriteReplayPersistentData(ReplayPersistentData data)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayPersistentData	data	

ThreadWriteReplaySegment(ReplaySegment)

Declaration

```
protected abstract void ThreadWriteReplaySegment(ReplaySegment segment)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplaySegment	segment	

ThreadWriteReplaySegmentTable(ReplaySegmentTable)

Declaration

```
protected abstract void ThreadWriteReplaySegmentTable(ReplayStreamStorage.ReplaySegmentTable table)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayStreamStorage.ReplaySegmentTable	table	

ToBytes()

Declaration

```
public byte[] ToBytes()
```

Returns

TYPE	DESCRIPTION
byte[]	

ToJsonString(Encoding)

Declaration

```
public string ToJsonString(Encoding encoding = null)
```

Parameters

TYPE	NAME	DESCRIPTION
Encoding	encoding	

Returns

TYPE	DESCRIPTION
string	

Implements

IDisposable

Struct ReplayStreamStorage.ReplaySegmentEntry

Implements

IReplayTokenSerialize

Inherited Members

ValueType.Equals(object)
ValueType.GetHashCode()
ValueType.ToString()
object.Equals(object, object)
object.GetType()
object.ReferenceEquals(object, object)

Namespace: UltimateReplay.Storage

Assembly: UltimateReplay.dll

Syntax

```
protected struct ReplayStreamStorage.ReplaySegmentEntry : IReplayTokenSerialize
```

Fields

endSequenceId

The sequence id of the replay snapshot that is the last entry of this segment.

Declaration

```
[ReplayTokenSerialize("End Sequence ID")]
public int endSequenceId
```

Field Value

TYPE	DESCRIPTION
int	

endTimeStamp

The timestamp of the replay snapshot that is the last entry of this segment.

Declaration

```
[ReplayTokenSerialize("End Time Stamp")]
public float endTimeStamp
```

Field Value

TYPE	DESCRIPTION
float	

segmentId

The unique id of this replay segment.

Declaration

```
[ReplayTokenSerialize("Segment ID")]
public int segmentId
```

Field Value

TYPE	DESCRIPTION
int	

startSequenceId

The sequence id of the replay snapshot that is the first entry of this segment.

Declaration

```
[ReplayTokenSerialize("Start Sequence ID")]
public int startSequenceId
```

Field Value

TYPE	DESCRIPTION
int	

startTimeStamp

The timestamp of the replay snapshot that is the first entry of this segment.

Declaration

```
[ReplayTokenSerialize("Start Time Stamp")]
public float startTimeStamp
```

Field Value

TYPE	DESCRIPTION
float	

streamOffset

The offset from the start of the stream data where the replay segment is located.

Declaration

```
[ReplayTokenSerialize("Stream Offset")]
public int streamOffset
```

Field Value

TYPE	DESCRIPTION
int	

Implements

IReplayTokenSerialize

Class ReplayStreamStorage.ReplaySegmentTable

Inheritance

object
ReplayStreamStorage.ReplaySegmentTable

Implements

IReplayStreamSerialize
IReplayTokenSerialize

Inherited Members

object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: UltimateReplay.Storage

Assembly: UltimateReplay.dll

Syntax

```
protected class ReplayStreamStorage.ReplaySegmentTable : IReplayStreamSerialize, IReplayTokenSerialize
```

Methods

AddSegment(ReplaySegmentEntry)

Declaration

```
public void AddSegment(ReplayStreamStorage.ReplaySegmentEntry segment)
```

Parameters

TYPE	NAME	DESCRIPTION
ReplayStreamStorage.ReplaySegmentEntry	segment	

GetSegmentDataOffset(int)

Declaration

```
public int GetSegmentDataOffset(int segmentId)
```

Parameters

TYPE	NAME	DESCRIPTION
int	segmentId	

Returns

TYPE	DESCRIPTION
int	

GetSegmentId(int)

Declaration

```
public int GetSegmentId(int sequenceId)
```

Parameters

TYPE	NAME	DESCRIPTION
int	sequenceId	

Returns

TYPE	DESCRIPTION
int	

GetSegmentId(float, float)

Declaration

```
public int GetSegmentId(float timestamp, float duration)
```

Parameters

TYPE	NAME	DESCRIPTION
float	timestamp	
float	duration	

Returns

TYPE	DESCRIPTION
int	

Implements

IReplayStreamSerialize
IReplayTokenSerialize

Class ReplayStreamStorage.ReplayStreamHeader

Inheritance

object
ReplayStreamStorage.ReplayStreamHeader

Implements

IReplayStreamSerialize
IReplayTokenSerialize

Inherited Members

object.Equals(object)
object.Equals(object, object)
object.GetHashCode()
object.GetType()
object.MemberwiseClone()
object.ReferenceEquals(object, object)
object.ToString()

Namespace: UltimateReplay.Storage

Assembly: UltimateReplay.dll

Syntax

```
protected class ReplayStreamStorage.ReplayStreamHeader : IReplayStreamSerialize, IReplayTokenSerialize
```

Fields

duration

The duration of the replay in seconds.

Declaration

```
public float duration
```

Field Value

TYPE	DESCRIPTION
float	

fileIdentifier

Unique id so that we know we are working with UR3.0 files.

Declaration

```
[ReplayTokenSerialize("File Identifier")]
public int fileIdentifier
```

Field Value

TYPE	DESCRIPTION
int	

identityByteSize

The size in bytes required to store a replay identity value. Can be 2 or 4 bytes.

Declaration

```
[ReplayTokenSerialize("Identity Byte Size")]
public ushort identityByteSize
```

Field Value

TYPE	DESCRIPTION
ushort	

memorySize

The amount of size in uncompressed bytes that the replay takes up.

Declaration

```
public int memorySize
```

Field Value

TYPE	DESCRIPTION
int	

metadataOffset

The stream offset to the metadata.

Declaration

```
public int metadataOffset
```

Field Value

TYPE	DESCRIPTION
int	

persistentDataOffset

The stream offset to the persistent data.

Declaration

```
public int persistentDataOffset
```

Field Value

TYPE	DESCRIPTION
int	

replayIdentifier

Declaration

```
public const int replayIdentifier = 808669781
```

Field Value

TYPE	DESCRIPTION
int	

replayVersion

Declaration

```
public const int replayVersion = 100
```

Field Value

TYPE	DESCRIPTION
int	

segmentTableOffset

The stream offset to the segment table. Segment table can map timestamps and sequence Ids to file offsets for replay segments.

Declaration

```
public int segmentTableOffset
```

Field Value

TYPE	DESCRIPTION
int	

snapshotCount

The number of snapshots in the replay.

Declaration

```
public int snapshotCount
```

Field Value

TYPE	DESCRIPTION
int	

version

The current version of this replay file format

Declaration

```
[ReplayTokenSerialize("Version")]
public int version
```

Field Value

TYPE	DESCRIPTION
int	

Properties

DurationFixedLengthString

Declaration

```
[ReplayTokenSerialize("Duration")]
public string DurationFixedLengthString { get; set; }
```

Property Value

TYPE	DESCRIPTION
string	

MemorySizeFixedLengthString

Declaration

```
[ReplayTokenSerialize("Memory Size")]
public string MemorySizeFixedLengthString { get; set; }
```

Property Value

TYPE	DESCRIPTION
string	

MetadataOffsetFixedLengthString

Declaration

```
[ReplayTokenSerialize("Metadata Offset")]
public string MetadataOffsetFixedLengthString { get; set; }
```

Property Value

TYPE	DESCRIPTION
string	

PersistentDataOffsetString

Declaration

```
[ReplayTokenSerialize("Persistent Data Offset")]
public string PersistentDataOffsetString { get; set; }
```

Property Value

TYPE	DESCRIPTION
string	

SegmentTableOffsetFixedLengthString

Declaration

```
[ReplayTokenSerialize("Segment Table Offset")]
public string SegmentTableOffsetFixedLengthString { get; set; }
```

Property Value

TYPE	DESCRIPTION
string	

SnapshotCountFixedLengthString

Declaration

```
[ReplayTokenSerialize("Snapshot Count")]
public string SnapshotCountFixedLengthString { get; set; }
```

Property Value

TYPE	DESCRIPTION
string	

Implements

IReplayStreamSerialize
IReplayTokenSerialize

Enum ReplayStreamType

Namespace: UltimateReplay.Storage

Assembly: UltimateReplay.dll

Syntax

```
public enum ReplayStreamType
```

Fields

NAME	DESCRIPTION
Binary	The replay system will use a high performance binary stream format for best performance and storage requirements.
Bson	The replay system with use bson file format.
Default	The result system will use the default replay format when writing or reading the stream (Binary format by default).
Json	The replay system will use a human readable json stream format for the replay. Useful for working with replay data in other applications when using a TextWriter for example.

Class ReplayStreamUtility

Inheritance

object

ReplayStreamUtility

Inherited Members

object.Equals(object)

object.Equals(object, object)

object.GetHashCode()

object.GetType()

object.MemberwiseClone()

object.ReferenceEquals(object, object)

object.ToString()

Namespace: UltimateReplay.Storage

Assembly: UltimateReplay.dll

Syntax

```
public static class ReplayStreamUtility
```

Struct ReplayToken

Inherited Members

ValueType.Equals(object)

ValueType.GetHashCode()

ValueType.ToString()

object.Equals(object, object)

object.GetType()

object.ReferenceEquals(object, object)

Namespace: UltimateReplay.Storage

Assembly: UltimateReplay.dll

Syntax

```
public struct ReplayToken
```

Fields

invalid

Declaration

```
public static readonly ReplayToken invalid
```

Field Value

TYPE	DESCRIPTION
ReplayToken	

Properties

Identifier

Declaration

```
public string Identifier { get; }
```

Property Value

TYPE	DESCRIPTION
string	

IsValid

Declaration

```
public bool IsValid { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

ValueType

Declaration

```
public Type ValueType { get; }
```

Property Value

TYPE	DESCRIPTION
Type	

Methods

Create(FieldInfo)

Declaration

```
public static ReplayToken Create(FieldInfo field)
```

Parameters

TYPE	NAME	DESCRIPTION
FieldInfo	field	

Returns

TYPE	DESCRIPTION
ReplayToken	

Create(PropertyInfo)

Declaration

```
public static ReplayToken Create(PropertyInfo property)
```

Parameters

TYPE	NAME	DESCRIPTION
PropertyInfo	property	

Returns

TYPE	DESCRIPTION
ReplayToken	

Create(string, Type)

Declaration

```
public static ReplayToken Create(string fieldOrPropertyName, Type declaringType)
```

Parameters

TYPE	NAME	DESCRIPTION
string	fieldOrPropertyName	
Type	declaringType	

Returns

TYPE	DESCRIPTION
ReplayToken	

FetchValue(object)

Declaration

```
public object FetchValue(object instance)
```

Parameters

TYPE	NAME	DESCRIPTION
object	instance	

Returns

TYPE	DESCRIPTION
object	

StoreValue(object, object)

Declaration

```
public void StoreValue(object instance, object value)
```

Parameters

TYPE	NAME	DESCRIPTION
object	instance	
object	value	

Tokenize(object)

Declaration

```
public static IEnumerable<ReplayToken> Tokenize(object instance)
```

Parameters

TYPE	NAME	DESCRIPTION
object	instance	

Returns

TYPE	DESCRIPTION
IEnumerable<ReplayToken>	

Tokenize(Type)

Declaration

```
public static IEnumerable<ReplayToken> Tokenize(Type type)
```

Parameters

TYPE	NAME	DESCRIPTION
Type	type	

Returns

TYPE	DESCRIPTION
IEnumerable<ReplayToken>	

Tokenize<T>()

Declaration

```
public static IEnumerable<ReplayToken> Tokenize<T>()
```

Returns

TYPE	DESCRIPTION
IEnumerable<ReplayToken>	

Type Parameters

NAME	DESCRIPTION
T	

Namespace UltimateReplay.Util

Classes

BitConverterNonAlloc

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.

Class BitConverterNonAlloc

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

object

BitConverterNonAlloc

Inherited Members

object.Equals(object)

object.Equals(object, object)

object.GetHashCode()

object.GetType()

object.MemberwiseClone()

object.ReferenceEquals(object, object)

object.ToString()

Namespace: UltimateReplay.Util

Assembly: UltimateReplay.dll

Syntax

```
public static class BitConverterNonAlloc
```

Methods

GetBool(byte[], int)

Retrieve a 8-bit bool from the specified byte array.

Declaration

```
public static bool GetBool(byte[] buffer, int offset)
```

Parameters

TYPE	NAME	DESCRIPTION
byte[]	buffer	The buffer to retrieve the bool from which must have a size of 1 or greater
int	offset	

Returns

TYPE	DESCRIPTION
bool	The unpacked bool value

GetBytes(byte[], int, bool)

Store an 8-bit bool into the specified byte array.

Declaration

```
public static void GetBytes(byte[] buffer, int offset, bool value)
```

Parameters

TYPE	NAME	DESCRIPTION
byte[]	buffer	The buffer to store the bool which must have a size of 1 or greater
int	offset	
bool	value	The bool value to store

GetBytes(byte[], int, double)

Store a 64-bit decimal value into the specified byte array.

Declaration

```
public static void GetBytes(byte[] buffer, int offset, double value)
```

Parameters

TYPE	NAME	DESCRIPTION
byte[]	buffer	The buffer to store the value which must have a size of 8 or greater
int	offset	
double	value	The value to store

GetBytes(byte[], int, short)

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

Declaration

```
public static void GetBytes(byte[] buffer, int offset, short value)
```

Parameters

TYPE	NAME	DESCRIPTION
byte[]	buffer	
int	offset	
short	value	

GetBytes(byte[], int, int)

Store a 32-bit int into the specified byte array.

Declaration

```
public static void GetBytes(byte[] buffer, int offset, int value)
```

Parameters

TYPE	NAME	DESCRIPTION
byte[]	buffer	The buffer to store the int which must have a size of 4 or greater
int	offset	
int	value	The int value to store

GetBytes(byte[], int, long)

Store a 64-bit int into the specified byte array.

Declaration

```
public static void GetBytes(byte[] buffer, int offset, long value)
```

Parameters

TYPE	NAME	DESCRIPTION
byte[]	buffer	The buffer to store the int which must have a size of 8 or greater
int	offset	
long	value	The int value to store

GetBytes(byte[], int, float)

Store a 32-bit float into the specified byte array.

Declaration

```
public static void GetBytes(byte[] buffer, int offset, float value)
```

Parameters

TYPE	NAME	DESCRIPTION
byte[]	buffer	The buffer to store the float which must have a size of 4 or greated
int	offset	
float	value	The float value to store

GetDouble(byte[], int)

Get a 64-bit decimal value from the specified byte array.

Declaration

```
public static double GetDouble(byte[] buffer, int offset)
```

Parameters

TYPE	NAME	DESCRIPTION
byte[]	buffer	The buffer to retrieve the data from which must have a size of 8 or greater
int	offset	

Returns

TYPE	DESCRIPTION
double	The unpacked double value

GetFloat(byte[], int)

Retrieve a 32-bit float from the specified byte array.

Declaration

```
public static float GetFloat(byte[] buffer, int offset)
```

Parameters

TYPE	NAME	DESCRIPTION
byte[]	buffer	The buffer to retrieve the float from which must have a size of 4 or greater
int	offset	

Returns

TYPE	DESCRIPTION
float	The unpacked float value

GetInt16(byte[], int)

Retrieve a 16-bit int from the specified byte array.

Declaration

```
public static short GetInt16(byte[] buffer, int offset)
```

Parameters

TYPE	NAME	DESCRIPTION
byte[]	buffer	The buffer to retrieve the short from which must have a size of 2 or greater

TYPE	NAME	DESCRIPTION
int	offset	

Returns

TYPE	DESCRIPTION
short	The unpacked short value

GetInt32(byte[], int)

Retrieve a 32-bit int from the specified byte array.

Declaration

```
public static int GetInt32(byte[] buffer, int offset)
```

Parameters

TYPE	NAME	DESCRIPTION
byte[]	buffer	The buffer to retrieve the int from which must have a size of 4 or greater
int	offset	

Returns

TYPE	DESCRIPTION
int	The unpacked int value

GetInt64(byte[], int)

Retrieve a 64-bit int from the specified byte array.

Declaration

```
public static long GetInt64(byte[] buffer, int offset)
```

Parameters

TYPE	NAME	DESCRIPTION
byte[]	buffer	The buffer to retrieve the int from which must have a size of 8 or greater
int	offset	

Returns

TYPE	DESCRIPTION

TYPE	DESCRIPTION
long	The unpacked long int value