Table of Contents

Namespace Limosa

Classes

[BondedScope](#)
    Type for all paired Scope.

[BondedScope.ScopeState](#)
    Type of Scope's state

[ButtonState](#)
    Type of ButtonState

[ButtonStateChanged](#)
    Type for ButtonState change This contains the previous and current ButtonState instances after the state has changed.

[ConvertedData](#)
    Type of Scope converted data that is calculated from raw data see: RawData

[IntegerChanged](#)
    Type for Integer value change This contains the previous and current value after the value has changed.

[LimosaProtocol](#)
    Singleton class that serves as the base for Yo itself and Scope operations and data exchange

[RawData](#)
    Type of Scope Raw data

[ScopeAddress](#)
    Type of Scope address

[ScopeButton](#)

[ScopeInfo](#)
    Type of Scope static information.

[ScopeOffset](#)
    Type of Scope offset Elements are similar to elements of type ConvertedData. see: ConvertedData

[ServiceBindState](#)
    Type for state indicating whether the service in AAR is bound or not.

[Vector4f](#)

Type of Vector4f This is used for quaternion. see: RawData.quaternion

Class BondedScope

Namespace: [Limosa](#)

Assembly: Assembly-CSharp.dll

Type for all paired Scope.

```
[Serializable]
public class BondedScope
```

Inheritance

[object](#) ← BondedScope

Inherited Members

[object.Equals(object)](#) , [object.Equals(object, object)](#) , [object.GetHashCode()](#) , [object.GetType()](#) ,
[object.MemberwiseClone()](#) , [object.ReferenceEquals(object, object)](#)

Fields

scopes

List of Scope's state. see: ScopeState

```
public List<BondedScope.ScopeState> scopes
```

Field Value

[List](#)<[BondedScope](#).[ScopeState](#)>

Methods

ToString()

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

Returns

[string](#)

Class BondedScope.ScopeState

Namespace: [Limosa](#)

Assembly: Assembly-CSharp.dll

Type of Scope's state

```
[Serializable]
public class BondedScope.ScopeState
```

Inheritance

[object](#) ← BondedScope.ScopeState

Inherited Members

[object.Equals(object)](#) , [object.Equals(object, object)](#) , [object.GetHashCode()](#) , [object.GetType()](#) ,
[object.MemberwiseClone()](#) , [object.ReferenceEquals(object, object)](#)

Fields

address

String of Scope address

```
public string address
```

Field Value

[string](#)

state

Whether the service in AAR is bound or not. true: Bound false: Not bound

```
public bool state
```

Field Value

[bool ↗](#)

Methods

ToString()

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

Returns

[string ↗](#)

Class ButtonState

Namespace: Limosa

Assembly: Assembly-CSharp.dll

Type of ButtonState

```
[Serializable]
public class ButtonState
```

Inheritance

object ← ButtonState

Inherited Members
object.Equals(object) , [object.Equals(object, object)](object.Equals(object,object)) , object.GetHashCode() , object.GetType() ,
object.MemberwiseClone() , [object.ReferenceEquals(object, object)](object.ReferenceEquals(object,object))

Fields

down

Whether "Down" button is pressed or not. 1: Pressed 0: Not pressed

```
public int down
```

Field Value

[int](int)

left

Whether "Left" button is pressed or not. 1: Pressed 0: Not pressed

```
public int left
```

Field Value

[int ↗](#)

main

Whether "Main" button is pressed or not. 1: Pressed 0: Not pressed

```
public int main
```

Field Value

[int ↗](#)

menu

Whether "Menu" button is pressed or not. 1: Pressed 0: Not pressed

```
public int menu
```

Field Value

[int ↗](#)

power

Whether "Power" button is pressed or not. 1: Pressed 0: Not pressed

```
public int power
```

Field Value

[int ↗](#)

right

Whether "Right" button is pressed or not. 1: Pressed 0: Not pressed

```
public int right
```

Field Value

[int ↗](#)

trigger

Whether "Trigger" button is pressed or not. 1: Pressed 0: Not pressed

```
public int trigger
```

Field Value

[int ↗](#)

up

Whether "Up" button is pressed or not. 1: Pressed 0: Not pressed

```
public int up
```

Field Value

[int ↗](#)

value

Bitmap for all buttons. The order of the following elements indicates whether or not each button is pressed, and this is stored as a bit. (If it is pressed, it is 1.)

```
public int value
```

Field Value

[int ↗](#)

view

Whether "View" button is pressed or not. 1: Pressed 0: Not pressed

```
public int view
```

Field Value

[int ↗](#)

Methods

ToString()

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

Returns

[string ↗](#)

Class ButtonStateChanged

Namespace: [Limosa](#)

Assembly: Assembly-CSharp.dll

Type for ButtonState change This contains the previous and current ButtonState instances after the state has changed.

```
[Serializable]
public class ButtonStateChanged
```

Inheritance

[object](#) ← ButtonStateChanged

Inherited Members

[object.Equals(object)](#) , [object.Equals(object, object)](#) , [object.GetHashCode()](#) , [object.GetType()](#) , [object.MemberwiseClone()](#) , [object.ReferenceEquals(object, object)](#)

Fields

address

String of Scope address

```
public string address
```

Field Value

[string](#)

current

Current ButtonState instances see: ButtonState

```
public ButtonState current
```

Field Value

[ButtonState](#)

previous

Previous ButtonState instances see: ButtonState

```
public ButtonState previous
```

Field Value

[ButtonState](#)

Methods

ToString()

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

Returns

[string](#)⤢

Class ConvertedData

Namespace: [Limosa](#)

Assembly: Assembly-CSharp.dll

Type of Scope converted data that is calculated from raw data see: RawData

```
[Serializable]
public class ConvertedData
```

Inheritance

[object](#) ← ConvertedData

Inherited Members

[object.Equals(object)](#) , [object.Equals(object, object)](#) , [object.GetHashCode()](#) , [object.GetType()](#) , [object.MemberwiseClone()](#) , [object.ReferenceEquals(object, object)](#)

Fields

address

String of Scope address. Also see: ScopeAddress

```
public string address
```

Field Value

[string](#)

distance

Destance from Scope distance sensor to obstacles Unit is mm

```
public float distance
```

Field Value

[float ↗](#)

pitch

Pitch degree of scope orientation

```
public float pitch
```

Field Value

[float ↗](#)

roll

Roll degree of scope orientation

```
public float roll
```

Field Value

[float ↗](#)

x

X-axis value of pointing position on the Yo screen 0.0~1080.0

```
public float x
```

Field Value

[float ↗](#)

y

Y-axis value of pointing position on the Yo screen 0.0~1920.0

```
public float y
```

Field Value

[float](#)⧉

yaw

Yaw degree of scope orientation

```
public float yaw
```

Field Value

[float](#)⧉

Methods

ToString()

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

Returns

[string](#)⧉

Class IntegerChanged

Namespace: [Limosa](#)

Assembly: Assembly-CSharp.dll

Type for Integer value change This contains the previous and current value after the value has changed.

```
[Serializable]
public class IntegerChanged
```

Inheritance

[object](#) ← IntegerChanged

Inherited Members

[object.Equals(object)](#) , [object.Equals(object, object)](#) , [object.GetHashCode()](#) , [object.GetType()](#) ,
[object.MemberwiseClone()](#) , [object.ReferenceEquals(object, object)](#)

Fields

address

String of Scope address

```
public string address
```

Field Value

[string](#)

current

Current int value

```
public int current
```

Field Value

[int](#)

previous

Previous int value

```
public int previous
```

Field Value

[int](#)

Methods

ToString()

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

Returns

[string](#)

Class LimosaProtocol

Namespace: Limosa

Assembly: Assembly-CSharp.dll

Singleton class that serves as the base for Yo itself and Scope operations and data exchange

```
public class LimosaProtocol : MonoBehaviour
```

Inheritance

object ← Object ← Component ← Behaviour ← MonoBehaviour ← LimosaProtocol

Inherited Members

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

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.InstantiateAsync<T>(T) , Object.InstantiateAsync<T>(T, Transform) ,
Object.InstantiateAsync<T>(T, Vector3, Quaternion) ,
Object.InstantiateAsync<T>(T, Transform, Vector3, Quaternion) , Object.InstantiateAsync<T>(T, int) ,
Object.InstantiateAsync<T>(T, int, Transform) ,
Object.InstantiateAsync<T>(T, int, Vector3, Quaternion) ,
Object.InstantiateAsync<T>(T, int, ReadOnlySpan<Vector3>, ReadOnlySpan<Quaternion>) ,
Object.InstantiateAsync<T>(T, int, Transform, Vector3, Quaternion) ,
Object.InstantiateAsync<T>(T, int, Transform, Vector3, Quaternion, CancellationToken) ,
Object.InstantiateAsync<T>(T, int, Transform, ReadOnlySpan<Vector3>, ReadOnlySpan<Quaternion>) ,
Object.InstantiateAsync<T>(T, int, Transform, ReadOnlySpan<Vector3>, ReadOnlySpan<Quaternion>,
CancellationToken) ,
Object.InstantiateAsync<T>(T, InstantiateParameters, CancellationToken) ,
Object.InstantiateAsync<T>(T, int, InstantiateParameters, CancellationToken) ,
Object.InstantiateAsync<T>(T, Vector3, Quaternion, InstantiateParameters, CancellationToken) ,
Object.InstantiateAsync<T>(T, int, Vector3, Quaternion, InstantiateParameters, CancellationToken) ,
Object.InstantiateAsync<T>(T, int, ReadOnlySpan<Vector3>, ReadOnlySpan<Quaternion>,
InstantiateParameters, CancellationToken) ,
Object.Instantiate(Object, Vector3, Quaternion) ,
Object.Instantiate(Object, Vector3, Quaternion, Transform) , Object.Instantiate(Object) ,
Object.Instantiate(Object, Scene) , Object.Instantiate<T>(T, InstantiateParameters) ,
Object.Instantiate<T>(T, Vector3, Quaternion, InstantiateParameters) ,
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.FindObjectsByType(Type, FindObjectsSortMode) ,
Object.FindObjectsByType(Type, FindObjectsInactive, FindObjectsSortMode) ,
Object.DontDestroyOnLoad(Object) , Object.DestroyObject(Object, float) ,

Object.DestroyObject(Object) , [Object.FindSceneObjectsOfType(Type)](#) ,
[Object.FindObjectsOfTypeIncludingAssets(Type)](#) , Object.FindObjectsOfType<T>() ,
Object.FindObjectsByType<T>(FindObjectsSortMode) , [Object.FindObjectsOfType<T>(bool)](#) ,
Object.FindObjectsByType<T>(FindObjectsInactive, FindObjectsSortMode) ,
Object.FindObjectOfType<T>() , [Object.FindObjectOfType<T>(bool)](#) ,
Object.FindFirstObjectByType<T>() , Object.FindAnyObjectByType<T>() ,
Object.FindFirstObjectByType<T>(FindObjectsInactive) ,
Object.FindAnyObjectByType<T>(FindObjectsInactive) , [Object.FindObjectsOfTypeAll(Type)](#) ,
[Object.FindObjectOfType(Type)](#) , [Object.FindFirstObjectByType(Type)](#) ,
[Object.FindAnyObjectByType(Type)](#) , [Object.FindObjectOfType(Type, bool)](#) ,
[Object.FindFirstObjectByType(Type, FindObjectsInactive)](#) ,
[Object.FindAnyObjectByType(Type, FindObjectsInactive)](#) , Object.ToString() , Object.name ,
Object.hideFlags , [object.Equals(object, object)](#) , [object.GetType()](#) , [object.MemberwiseClone()](#) ,
[object.ReferenceEquals(object, object)](#)

Fields

OnConnectionStateChangedEvent

UnityEvent that is fired when scope connection states are changed.

```
public UnityEvent OnConnectionStateChangedEvent
```

Field Value

UnityEvent

Properties

Instance

Return the instance of LimosaProtocol

```
public static LimosaProtocol Instance { get; }
```

Property Value

IsServiceBound

Whether the service in AAR is bound or not. Before service is bound, calling the following methods will be ignore : Vibrate(string[], int) VibrateToAll(int) GetScopes() GetScopeInfo(string) GetBatteryLevel(string)

Also see: OnServiceBindStateChanged

```
public bool IsServiceBound { get; }
```

Property Value

[bool](#)⊠

ScopeConnectionState

Dictionary of scope connection state at that point in time. Key: Scope address Value: Connection state (true: connected, false: not connected)

```
public IReadOnlyDictionary<string, bool> ScopeConnectionState { get; }
```

Property Value

[IReadOnlyDictionary](#)⊠ <[string](#)⊠, [bool](#)⊠ >

ScopeConvertedData

Dictionary of scope converted data at that point in time. Key: Scope address Value: ConvertedData

```
public IReadOnlyDictionary<string, ConvertedData> ScopeConvertedData { get; }
```

Property Value

[IReadOnlyDictionary](#)⊠ <[string](#)⊠, [ConvertedData](#)>

ScopeOffset

Dictionary of scope converted offset converted data at that point in time. Key: Scope address Value: Converted offset data

```
public IReadOnlyDictionary<string, ConvertedData> ScopeOffset { get; }
```

Property Value

[IReadOnlyDictionary](#)⟨ <[string](#)⟨, [ConvertedData](#)>

ScopeRawData

Dictionary of scope raw data at that point in time. Key: Scope address Value: Raw data

```
public IReadOnlyDictionary<string, RawData> ScopeRawData { get; }
```

Property Value

[IReadOnlyDictionary](#)⟨ <[string](#)⟨, [RawData](#)>

ScopeRawOffset

Dictionary of scope offset raw data at that point in time. Key: Scope address Value: Raw offset data

```
public IReadOnlyDictionary<string, RawData> ScopeRawOffset { get; }
```

Property Value

[IReadOnlyDictionary](#)⟨ <[string](#)⟨, [RawData](#)>

Methods

GetBatteryLevel(string)

Get Scope's current battery level Also see: OnServiceBindStateChanged

```
public virtual int GetBatteryLevel(string targetAddress)
```

Parameters

targetAddress [string](https://...)⧉

    Target Scope's address

Returns

[int](https://...)⧉

    Battery leve value. The value is the ratio of the remaining battery charge to the full charge and is a percentage value from 0 to 100.

See Also

[OnBatteryLevelChanged(string)](https://...)

GetScopeInfo(string)

Get Scope's static information

```
public virtual ScopeInfo GetScopeInfo(string targetAddress)
```

Parameters

targetAddress [string](https://...)⧉

Returns

[ScopeInfo](https://...)

    ScopeInfo instance that contains Scope's static information.

GetScopes()

Get all paired Scopes.

```
public virtual BondedScope GetScopes()
```

Returns

[BondedScope](#)

   BondedScope instance that contains all paired Scopes information.

GoBackToLimosa()

Go back to Yo OS.

```
public virtual void GoBackToLimosa()
```

Remarks

This method performs just going back to Yo OS. Please handle the termination of the application yourself.

OnBatteryLevelChanged(string)

Callback method called when scope battery level is changed. The value is the ratio of the remaining battery charge to the full charge and is a percentage value from 0 to 100.

```
protected virtual void OnBatteryLevelChanged(string data)
```

Parameters

data [string](#)⬀

   Json text serialized with IntegerChanged type information

OnButtonsStateChanged(string)

Callback method called when scope button state is changed (pressed or released).

```
protected virtual void OnButtonsStateChanged(string data)
```

Parameters

data string⧉

   Json text serialized with ButtonStateChanged type information

OnConnected(string)

Callback method called when Scope is connected

```
protected virtual void OnConnected(string data)
```

Parameters

data string⧉

   Json text serialized with ScopeAddress type information

OnDataConverted(string)

```
protected virtual void OnDataConverted(string data)
```

Parameters

data string⧉

OnDisconnected(string)

Callback method called when Scope is diconnected

```
protected virtual void OnDisconnected(string data)
```

Parameters

data [string](#)

    Json text serialized with ScopeAddress type information

OnEncoderValueChanged(string)

Callback method called when scope zoom ring value is changed. Clockwise rotation increases value.

```
protected virtual void OnEncoderValueChanged(string data)
```

Parameters

data [string](#)

    Json text serialized with IntegerChanged type information

OnInitialized(string)

Callback method called when Scope is connected. This is called once after the scope is connected.

```
protected virtual void OnInitialized(string data)
```

Parameters

data [string](#)

    Json text serialized with ScopeAddress type information

OnOffsetReset(string)

Callback method called when scope pointing position is reset.

```
protected virtual void OnOffsetReset(string data)
```

Parameters

data [string](#)

Json text serialized with ScopeOffset type information

OnRawData(string)

Callback method called when scope send new data.

```
protected virtual void OnRawData(string data)
```

Parameters

data string↗

Json text serialized with RawData type information

OnServiceBindStateChanged(string)

Callback method called when the service in AAR is bound or unbound. Before service is bound, calling the following methods will be ignore : Vibrate(string[], int) VibrateToAll(int) GetScopes() GetScopeInfo(string) GetBatteryLevel(string)

Also see: OnServiceBindStateChanged

```
protected virtual void OnServiceBindStateChanged(string data)
```

Parameters

data string↗

Json text serialized with ServiceBindState type information

OnSignalTargetSet(string)

Callback method called when the service in AAR set signal target.

```
protected virtual void OnSignalTargetSet(string data)
```

Parameters

data string↗

    Signal Target Name

Vibrate(string[], int)

Vibrate Scope with the selected pattern.

```
public virtual void Vibrate(string[] targetAddress, int pattern)
```

Parameters

targetAddress string↗[]

    Target Scope's addresses

pattern int↗

    Vibration pattern.1 ~ 123

VibrateToAll(int)

```
public virtual void VibrateToAll(int pattern)
```

Parameters

pattern int↗

Class RawData

Namespace: [Limosa](#)

Assembly: Assembly-CSharp.dll

Type of Scope Raw data

```
[Serializable]
public class RawData
```

Inheritance

[object](#) ← RawData

Inherited Members

[object.Equals(object)](#) , [object.Equals(object, object)](#) , [object.GetHashCode()](#) , [object.GetType()](#) ,
[object.MemberwiseClone()](#) , [object.ReferenceEquals(object, object)](#)

Fields

address

String of Scope address. Also see: ScopeAddress

```
public string address
```

Field Value

[string](#)

button

Button state of Scope. Also see: LimosaProtocol.OnButtonsStateChanged

```
public ButtonState button
```

Field Value

[ButtonState](#)

See Also
[OnButtonsStateChanged](#)([string ⧉](#))

quaternion

Quaternion for scope orientation

```
public Vector4f quaternion
```

Field Value

[Vector4f](#)

zoom

Value of Scope Zoom ring. Clockwise rotation increases value. Also see:
LimosaProtocol.OnEncoderValueChanged

```
public int zoom
```

Field Value

[int ⧉](#)

See Also
[OnEncoderValueChanged](#)([string ⧉](#))

Methods

ToString()

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

Returns

[string](#)

Class ScopeAddress

Namespace: [Limosa](#)

Assembly: Assembly-CSharp.dll

Type of Scope address

```
[Serializable]
public class ScopeAddress
```

Inheritance

[object](#) ← ScopeAddress

Inherited Members

[object.Equals(object)](#) , [object.Equals(object, object)](#) , [object.GetHashCode()](#) , [object.GetType()](#) ,
[object.MemberwiseClone()](#) , [object.ReferenceEquals(object, object)](#)

Fields

address

String of Scope address

```
public string address
```

Field Value

[string](#)

Methods

ToString()

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

Returns

[string](#)

Class ScopeButton

Namespace: [Limosa](#)

Assembly: Assembly-CSharp.dll

```
public class ScopeButton : OnScreenControl
```

Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← OnScreenControl ← ScopeButton

Inherited Members

OnScreenControl.SendValueToControl<TValue>(TValue) , OnScreenControl.SentDefaultValueToControl() ,
OnScreenControl.OnEnable() , OnScreenControl.OnDisable() , OnScreenControl.controlPath ,
OnScreenControl.control , 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.destroyCancellationToken ,
MonoBehaviour.useGUILayout , MonoBehaviour.didStart , MonoBehaviour.didAwake ,
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.GetComponentIndex() , [Component.CompareTag(string)](#) ,

Component.CompareTag(TagHandle) ,
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.InstantiateAsync<T>(T) , Object.InstantiateAsync<T>(T, Transform) ,
Object.InstantiateAsync<T>(T, Vector3, Quaternion) ,
Object.InstantiateAsync<T>(T, Transform, Vector3, Quaternion) , Object.InstantiateAsync<T>(T, int) ,
Object.InstantiateAsync<T>(T, int, Transform) ,
Object.InstantiateAsync<T>(T, int, Vector3, Quaternion) ,
Object.InstantiateAsync<T>(T, int, ReadOnlySpan<Vector3>, ReadOnlySpan<Quaternion>) ,
Object.InstantiateAsync<T>(T, int, Transform, Vector3, Quaternion) ,
Object.InstantiateAsync<T>(T, int, Transform, Vector3, Quaternion, CancellationToken) ,
Object.InstantiateAsync<T>(T, int, Transform, ReadOnlySpan<Vector3>, ReadOnlySpan<Quaternion>) ,
Object.InstantiateAsync<T>(T, int, Transform, ReadOnlySpan<Vector3>, ReadOnlySpan<Quaternion>,
CancellationToken) ,
Object.InstantiateAsync<T>(T, InstantiateParameters, CancellationToken) ,
Object.InstantiateAsync<T>(T, int, InstantiateParameters, CancellationToken) ,
Object.InstantiateAsync<T>(T, Vector3, Quaternion, InstantiateParameters, CancellationToken) ,
Object.InstantiateAsync<T>(T, int, Vector3, Quaternion, InstantiateParameters, CancellationToken) ,
Object.InstantiateAsync<T>(T, int, ReadOnlySpan<Vector3>, ReadOnlySpan<Quaternion>,
InstantiateParameters, CancellationToken) ,
Object.Instantiate(Object, Vector3, Quaternion) ,
Object.Instantiate(Object, Vector3, Quaternion, Transform) , Object.Instantiate(Object) ,
Object.Instantiate(Object, Scene) , Object.Instantiate<T>(T, InstantiateParameters) ,
Object.Instantiate<T>(T, Vector3, Quaternion, InstantiateParameters) ,
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.FindObjectsByType(Type, FindObjectsSortMode) ,
Object.FindObjectsByType(Type, FindObjectsInactive, FindObjectsSortMode) ,

Object.DontDestroyOnLoad(Object) , Object.DestroyObject(Object, float) ,
Object.DestroyObject(Object) , Object.FindSceneObjectsOfType(Type) ,
Object.FindObjectsOfTypeIncludingAssets(Type) , Object.FindObjectsOfType<T>() ,
Object.FindObjectsByType<T>(FindObjectsSortMode) , Object.FindObjectsOfType<T>(bool) ,
Object.FindObjectsByType<T>(FindObjectsInactive, FindObjectsSortMode) ,
Object.FindObjectOfType<T>() , Object.FindObjectOfType<T>(bool) ,
Object.FindFirstObjectByType<T>() , Object.FindAnyObjectByType<T>() ,
Object.FindFirstObjectByType<T>(FindObjectsInactive) ,
Object.FindAnyObjectByType<T>(FindObjectsInactive) , Object.FindObjectsOfTypeAll(Type) ,
Object.FindObjectOfType(Type) , Object.FindFirstObjectByType(Type) ,
Object.FindAnyObjectByType(Type) , Object.FindObjectOfType(Type, bool) ,
Object.FindFirstObjectByType(Type, FindObjectsInactive) ,
Object.FindAnyObjectByType(Type, FindObjectsInactive) , Object.ToString() , Object.name ,
Object.hideFlags , object.Equals(object, object) , object.GetType() , object.MemberwiseClone() ,
object.ReferenceEquals(object, object)

Properties

controlPathInternal

```
protected override string controlPathInternal { get; set; }
```

Property Value

string

Methods

OnPressed()

```
public void OnPressed()
```

OnReleased()

```
public void OnReleased()
```

Class ScopeInfo

Namespace: [Limosa](#)

Assembly: Assembly-CSharp.dll

Type of Scope static information.

```
[Serializable]
public class ScopeInfo
```

Inheritance

[object](#) ← ScopeInfo

Inherited Members

[object.Equals(object)](#) , [object.Equals(object, object)](#) , [object.GetHashCode()](#) , [object.GetType()](#) ,
[object.MemberwiseClone()](#) , [object.ReferenceEquals(object, object)](#)

Fields

firmwareRev

String of firmware revision in software revision

```
public string firmwareRev
```

Field Value

[string](#)

hardwareRev

String of hardware revision

```
public string hardwareRev
```

Field Value

[string ↗](#)

manufacturer

String of manufacturer

```
public string manufacturer
```

Field Value

[string ↗](#)

modelNumber

String of model number

```
public string modelNumber
```

Field Value

[string ↗](#)

pnpId

String of PNP ID（Plug and Play ID）

```
public string pnpId
```

Field Value

[string ↗](#)

serialNumber

String of serial number

```
public string serialNumber
```

Field Value

[string](#)

softwareRev

String of application revision in software revision

```
public string softwareRev
```

Field Value

[string](#)

Methods

ToString()

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

Returns

[string](#)

Class ScopeOffset

Namespace: [Limosa](#)

Assembly: Assembly-CSharp.dll

Type of Scope offset Elements are similar to elements of type ConvertedData. see: ConvertedData

```
[Serializable]
public class ScopeOffset
```

Inheritance

[object](#) ← ScopeOffset

Inherited Members

[object.Equals(object)](#) , [object.Equals(object, object)](#) , [object.GetHashCode()](#) , [object.GetType()](#) , [object.MemberwiseClone()](#) , [object.ReferenceEquals(object, object)](#)

Fields

address

String of Scope address. Also see: ScopeAddress

```
public string address
```

Field Value

[string](#)

byUser

Whether reset is performed by user or not 0.0~1080.0

```
public bool byUser
```

Field Value

[bool](#)

pitch

Pitch degree of scope orientation

```
public float pitch
```

Field Value

[float](#)

roll

Roll degree of scope orientation

```
public float roll
```

Field Value

[float](#)

x

X-axis value of pointing position on the Yo screen 0.0~1080.0

```
public float x
```

Field Value

[float](#)

y

Y-axis value of pointing position on the Yo screen 0.0~1920.0

```
  public float y
```

Field Value

[float](#)⤤

yaw

Yaw degree of scope orientation

```
  public float yaw
```

Field Value

[float](#)⤤

Methods

ToString()

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

Returns

[string](#)⤤

Class ServiceBindState

Namespace: [Limosa](#)

Assembly: Assembly-CSharp.dll

Type for state indicating whether the service in AAR is bound or not.

```
[Serializable]
public class ServiceBindState
```

Inheritance

[object](#) ← ServiceBindState

Inherited Members

[object.Equals(object)](#) , [object.Equals(object, object)](#) , [object.GetHashCode()](#) , [object.GetType()](#) ,
[object.MemberwiseClone()](#) , [object.ReferenceEquals(object, object)](#)

Fields

state

Whether the service in AAR is bound or not. true: Bound false: Not bound

```
public bool state
```

Field Value

[bool](#)

Methods

ToString()

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

Returns

string

Class Vector4f

Namespace: [Limosa](#)

Assembly: Assembly-CSharp.dll

Type of Vector4f This is used for quaternion. see: RawData.quaternion

```
[Serializable]
public class Vector4f
```

Inheritance

[object](#) ← Vector4f

Inherited Members

[object.Equals(object)](#) , [object.Equals(object, object)](#) , [object.GetHashCode()](#) , [object.GetType()](#) , [object.MemberwiseClone()](#) , [object.ReferenceEquals(object, object)](#)

Fields

W

```
public float w
```

Field Value

[float](#)

X

```
public float x
```

Field Value

[float](#)

y

```
public float y
```

Field Value

[float↗](#)

z

```
public float z
```

Field Value

[float↗](#)

Methods

ToString()

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

Returns

[string↗](#)