Table of Contents

Online Documentation

Thanks for using and supporting Stream Deck Integration!

Full documentation is included in this PDF, though, the use of the online version is **highly recommended**.

This offline documentation is an automatic conversion of the online version, available at:

- https://docs.f10.dev/streamdeckintegration

The online documentation is always up-to-date, and includes useful features like search support.

If you produce a cool project using this integration, I would love to hear about it! Maybe even feature it if you would allow for it. You can send any cool stuff to **adam@f10.dev**.

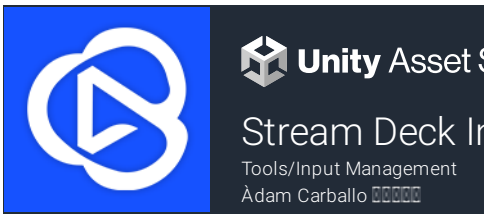
For any feedback, suggestions, issues or questions not covered by the current documentation, please send an email to **support@f10.dev**.

Stream Deck Integration Manual

The Stream Deck Integration package allows any official Stream Deck hardware running official software to interact and communicate with the Unity Editor and / or built Unity projects.

The package is available for purchase at the Unity Asset Store.

The Stream Deck plugin is available for free at the Elgato Marketplace. 
The plugin can also be manually installed from the included plugin file (`.streamDeckPlugin`) inside the package.



Requirements

Unity Version

The package has been fully tested in many Unity versions, but it's currently developed using **Unity 2019.4 LTS**. The package supports all versions from **Unity 5.6** to **Unity 6** (including yearly releases in between, e.g. 2018.4 LTS, and others).

Higher Unity versions should also work without any major issues. If any fixes are required for future versions, new updates will be released along feature complete Unity versions. Alpha and Beta Unity versions (**.a** & **.b**) are not continuously supported, and any issues found on them will only be addressed on full releases.

Older Unity versions (namely **Unity 5.5** or lower) are not supported. There is no reason to believe that moderately recent versions shouldn't work, but changes in Unity's API may break the package.

> **IMPORTANT**
>
> Some Unity versions may have some missing features. The package targets 2018.4 LTS as the base version, with lower versions being supported as "legacy".
>
> The documentation marks which features are not available.

System Requirements

Unity Editor

Any system that can run the Unity Editor supports all features of this package.

Keep in mind that there is no official support for Stream Deck hardware under Linux. There are multiple open-source projects that aim to support Stream Deck hardware under Linux, but none of them support Stream Deck Plugins, which is required for the package to communicate with a Stream Deck.

Built project

Similar to Editor support, only **Windows** and **MacOS** built projects are supported by default, as this is a limitation based on Stream Deck's supported platforms.

Network

The package uses the port **2245** to establish an internal communication between the Unity Editor or built project and the Stream Deck official software.

There is currently no support to change this port manually, so if any other process is occupying the port, the plugin won't be able to initialise.

Supported Hardware

Any officially released Stream Deck hardware (and software) will work with this package, as long as the official Stream Deck software with the required plugin are running in the same system as the Unity Editor or built project.

All the following units are supported:

- **Stream Deck** with 15 keys

- **Stream Deck MK.2** with 15 keys

- **Stream Deck +** with 8 keys, 4 dials and a touchscreen

- **Stream Deck Mini** with 6 keys

- **Stream Deck XL** with 32 keys

- **Stream Deck Mobile** (iOS & Android app)

- **Stream Deck Pedal** with 3 actuators

- **Corsair Voyager** with 10 macro "S-keys"

Unity Package Manual

Quickstart

Import the package using the Package Manager (or the Asset Store, depending on your Unity version). Make sure all files inside the `Scripts` folder are imported. The `Demo` folder is optional.

The package doesn't require any extra configuration inside the Unity Editor to function.

Assuming a Stream Deck is connected to the system, the official software is running and the plugin is installed, the Editor should connect automatically upon assembly refresh (after importing the package).

To expose your custom fields, properties, methods and / or menus to be invoked by a Stream Deck, refer to Exposing Members.

More information and a quickstart regarding the plugin can be found in the Plugin Manual.

Built Project Support

If your project only requires the Stream Deck to work in the Unity Editor, no extra configuration is required.

To add support for a Stream Deck to interact with a built project, a GameObject must contain the `StreamDeckRuntime` component. The component will automatically start a connection on `Awake`, and disconnect when destroyed.

Example Scene

Inside the `Demo` folder you can find a scene (named `Demo` as well) containing two GameObjects (`StreamDeckRuntime` and `StreamDeckExample`).

For a quick understanding of all the API available, `StreamDeckExample.cs` and `StreamDeckStaticExample.cs` contain fully commented code. For a more visual documentation, check the **Scripting API** instead.

Exposing Members

By default, the Stream Deck plugin and package supports handling the Editor state (Play / Pause) and execute menu items.

To invoke methods, or set fields and properties directly, members must be exposed to the package logic using attributes in code.

Adding / Removing Instanced Objects

For non-static classes, each instance must add itself to be exposed and available to be called by a Stream Deck. It must also remove itself before being destroyed, or exceptions could occur if a Stream Deck tries to invoke a destroyed object.

The following example will keep all the exposed members of a `MonoBehaviour` available while the object is alive and instanced:

```
private void OnEnable() {
  // Registers this class as a StreamDeck enabled class
  StreamDeck.Add(this);
}

private void OnDisable() {
  // Removes this class as a StreamDeck enabled class
  StreamDeck.Remove(this);
}
```

Adding Static Objects

Static classes can add all the exposed members using a slightly different method. Since a static class will never be destroyed, once added static classed can't be removed.

The following example will add all the exposed members of a static class as soon as the object is constructed:

```
static MyStaticClass() {
  // Registers this class as a StreamDeck enabled class
  StreamDeck.AddStatic(typeof(MyStaticClass));
}
```

> **NOTE**
>
> Make sure that any class that contains attributes (and you want exposed to the Stream Deck) contains calls to `StreamDeck.Add()` and `StreamDeck.Remove()`.

Single Member (Button)

To add only one specific field, property or method to be exposed and available to be called by a Stream Deck action, use the `StreamDeckButtonAttribute`. If no `ID` is defined on the attribute, the member name will be used:

```
[StreamDeckButton]
public int _field = 0;

[StreamDeckButton]
private float Property { get; set; }

[StreamDeckButton("CustomMethodId")]
public void Method() {
    // [...]
}
```

Single Member (Dial)

Although dials (for example, found in Stream Deck +) can invoke some of the members defined with `StreamDeckButtonAttribute`, using `StreamDeckDialAttribute` adds support for `action` type, `state` and `value` parsing on Unity's side.

This attribute **can only be used in method declarations, and requires all the parameters mentioned above in the correct order to function**. The method can differentiate between dial actions (`press` or `rotation`), the action state (`on` or `off`) and the incremental amount of rotation in the current call (positive or negative value).

If no `ID` is defined on the attribute, the member name will be used:

```
[StreamDeckDial]
public void Method(StreamDeckDialAction action, bool state, double value) {
  // [...]
}

[StreamDeckDial("CustomMethodId")]
public void AnotherMethod(StreamDeckDialAction action, bool state, double value) {
  // [...]
}
```

Multiple Members (Group)

> **WARNING**
>
> Stream Deck Groups are part of the legacy API and may undergo breaking changes in the distant future. It is therefore advised to stick to single member declarations (`[StreamDeckButton]`) for now.

To add and expose all members inside a class (useful for debug-only classes) to be called by a Stream Deck, use the `StreamDeckGroupAttribute`. If no `ID` is defined on the attribute, the class name will be used:

```
[StreamDeckGroup]
private class MyClass {
  // [...]
}

[StreamDeckGroup("CustomGroupId")]
private class MyOtherClass {
  // [...]
}
```

> **NOTE**
>
> Classes, fields, properties and methods can be private, internal or public, as the package uses reflection to invoke them.

Action Settings

Stream Deck actions (also known as "buttons") can be customized directly from Unity, allowing manually pre-defined actions to have their title or icon / image changed from the editor, or even at runtime from a built project.

Currently, only the following features are supported. The listed elements here are mostly limited by the Stream Deck SDK own methods and features.

Set Action Title

> **NOTE**
>
> The targeted action must not have any custom title in the Stream Deck software, as manually set titles have priority over SDK-driven ones. Make sure the action has no title, and the `Show Title` toggle is enabled.

The following example will change the title of the action that has `TitleTest` as it's ID:

```
[StreamDeckButton("TitleTest")]
private void TitleTest() {
  // Unreleated logic
}

public void ChangeTitle() {
  var title = "New Title!";
  StreamDeckSettings.SetButtonTitle(title, "TitleTest");
}
```

For actions defined as part of a group, make sure to include the group ID along with the member ID:

```
[StreamDeckGroup]
public class TestClass {
  private void TitleTest() {
    // Unreleated logic
  }
}

public void ChangeTitle() {
  var title = "New Title!";
  StreamDeckSettings.SetButtonTitle(title, "TitleTest", "TestClass");
}
```

Set Action Image / Icon

> **NOTE**
>
> The targeted action must not have any custom icon in the Stream Deck software, as manually set assets have priority over SDK-driven ones. Make sure the action has no icon. `Reset to Default` option can be used to remove a custom asset.

> **NOTE**
>
> Assets will be sent in `base64` format, as PNG images. Any `Texture2D` (not compressed, and with read-write support enabled) should be supported.

> **NOTE**
>
> Stream Deck's use low resolution screens, and won't benefit from large images. Assets with a resolution higher than `128x128px` won't have visible improvements. This may not be accurate for the mobile app version.

The following example will change the icon of the action that has `ImageTest` as it's ID:

```
[StreamDeckButton("ImageTest")]
private void ImageTest() {
  // Unreleated logic
}

public void ChangImage(Texture2D image) {
  StreamDeckSettings.SetButtonImage(image, "ImageTest");
}
```

For actions defined as part of a group, make sure to include the group ID along with the member ID:

```
[StreamDeckGroup]
public class TestClass {
  private void ImageTest() {
    // Unreleated logic
  }
}

public void ChangeImage(Texture2D image) {
  StreamDeckSettings.SetButtonImage(image, "ImageTest", "TestClass");
}
```

Set Action Value

> **NOTE**
>
> This will only affect actions that support / have value fields.

The following example will change the parameter value of the action that has `ValueExample` as it's ID:

```
[StreamDeckButton]
private void ValueExample(float value) {
  // Unreleated logic
}

public void ChangeValue(float newValue) {
  StreamDeckSettings.SetButtonValue(newValue, "ValueExample");
}
```

For actions defined as part of a group, make sure to include the group ID along with the member ID:

```
[StreamDeckGroup]
public class TestClass {
  private void ValueExample(float value) {
    // Unreleated logic
  }
}

public void ChangeValue(float newValue) {
  StreamDeckSettings.SetButtonValue(newValue, "ValueExample", "TestClass");
}
```

Configure Scripted Action

Scripted actions are fully managed by the integration on Unity's side (they don't expose any settings in the Stream Deck software), and must be configured by code.

The following example will configure an action that has `ScriptedExample` as it's ID:

```
public void Configure(Action callback) {
  StreamDeckSettings.SetScriptedButton("ScriptedExample", callback);
}
```

This method also supports setting the action's title and image directly without having to call any other methods:

```
public void Configure(string title, Texture2D icon, Action callback) {
  StreamDeckSettings.SetScriptedButton("ScriptedExample", callback, title, icon);
}
```

Configrued scripted actions can also be removed at any point using the following method:

```
StreamDeckSettings.RemoveScriptedButton("ScriptedExample");
```

Immediate Mode

All features support immediate or queued sending rates. By default, all actions sent by Unity to Stream Deck (or vice-versa) are automatically queued. For normal operation, the queue delay is almost invisible, and prevents race conditions.

If required, all methods can be called with the parameter `immediate` set to `true`, which will skip any current queued events, and instead communicate the setting change on the exact same frame. This can help if multiple settings should be changed on the same frame:

```
public void DoItNow() {
  var title = "Immediate Title";
  StreamDeckSettings.SetButtonTitle(title, "MemberId", "GroupId", true);

  var image = new Texture2D(64,64);
  StreamDeckSettings.SetButtonImage(image, "MemberId", "GroupId", true);
}
```

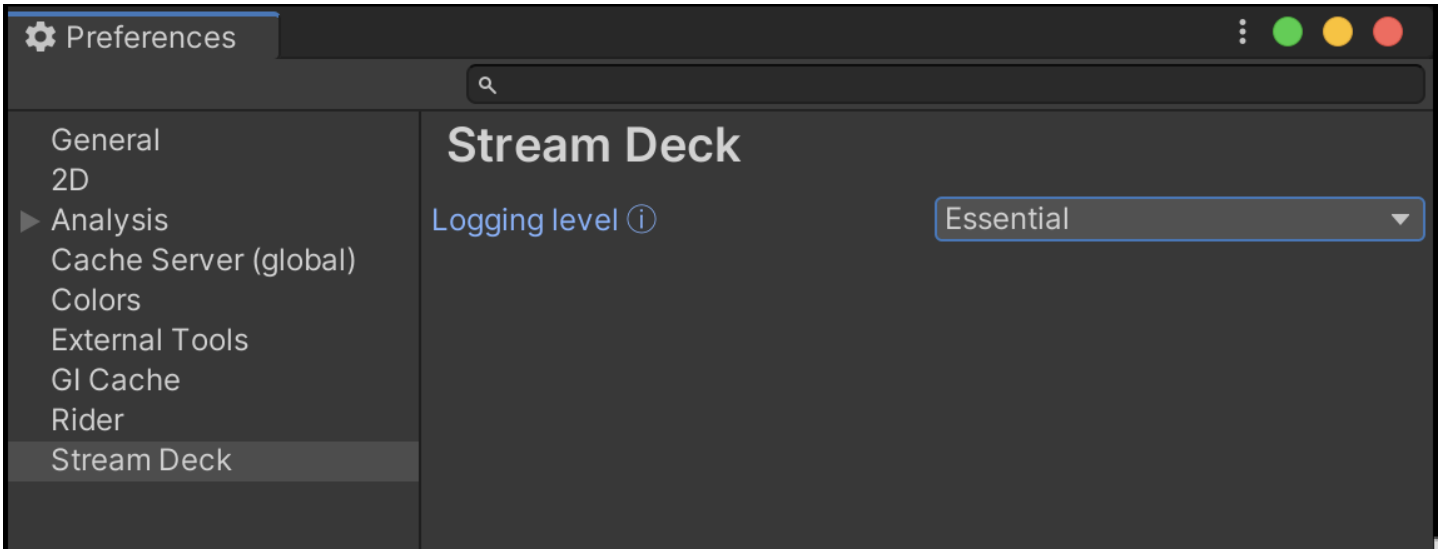
Preferences Window

The package includes configurable settings located in Unity's Preferences. These can be found under `Unity > Preferences... > Stream Deck`.

IMPORTANT

The preferences window is not available in **Unity 2018.2** or lower versions.

NAME	DEFAULT	DESCRIPTION
Logging Level	`Essential`	Package logging. `Critical` will only show errors, `Essential` will show warnings and other information, `All` will show debug, including message payloads.



NOTE

When opening the preferences window for the first time, a `StreamDeckPreferences` asset will be generated inside `Assets/Editor/`. This asset can be relocated anywhere inside the project, but should be kept on an Editor-only folder or assembly.

Plugin Manual

Quickstart

The Stream Deck plugin can be installed using the Elgato Marketplace.

The plugin can also be installed manually if needed. The compressed plugin can be found inside the package, named `com.adamcarballo.unity-integration.streamDeckPlugin`.

> **NOTE**
>
> The included plugin with package may not be the most updated version available. It's recommended to install the plugin using the Elgato Marketplace, which includes support for automatic updates.

For more information regarding Stream Deck actions, check the official documentation.

Each available action can be found under the "Unity" category. A documentation reference for each action can be found on the left sidebar under the Plugin Manual.

Play Action

This action allows to switch between play modes (Editor and runtime) on the Unity Editor. The action icon will also automatically update and reflect the current editor status on it's own.

Parameters

This action has no configurable parameters.

Pause Action

This action allows to switch between pause modes (Editor and runtime) on the Unity Editor. The action icon will also automatically update and reflect the current editor status on it's own.

> **IMPORTANT**
>
> This action is not available in **Unity 5.6** or lower versions.

> **NOTE**
>
> The Unity Editor can be paused regardless of it's play mode state. This is helpful to start a game paused and scroll through frames manually.

Parameters

This action has no configurable parameters.

Execute Menu Action

This action allows to execute any menu item (including custom items using `MenuItem` attributes) on the Unity Editor.

> **NOTE**
>
> This action uses the default Unity API to execute menu items. Any limitation you may found can't be fixed by the package itself.

Parameters

NAME	REQUIRED	DESCRIPTION
Path	Yes	The full menu item path with spaces included.

Set Field / Property Action

This action allows to set any exposed field or property value (with a supported type) on the Unity Editor and built projects.

Parameters

NAME	REQUIRED	DESCRIPTION
Group	No	Must be enabled if the field or property to set was exposed as part of a `StreamDeckGroup`.
Group ID	Yes (if group)	ID set in the `StreamDeckGroup` attribute. If no ID is set the class name is the exposed ID.
Member ID	Yes	ID of the field or property to set. If using `StreamDeckGroup` attribute or `StreamDeckButton` without a defined ID, use the field or property name.
Type	Yes	Type of the field or property to set. Supported values include: `Int`, `Float`, `Bool` and `String`.
Value	Yes	Value to be set in the field or property. Make sure the value matches a supported type.

Invoke Method Action

This action allows to invoke any exposed method with up to one parameter on the Unity Editor and built projects.

Parameters

NAME	REQUIRED	DESCRIPTION
Group	No	Must be enabled if the method to invoke was exposed as part of a `StreamDeckGroup`.
Group ID	Yes (if group)	ID set in the `StreamDeckGroup` attribute. If no ID is set the class name is the exposed ID.
Member ID	Yes	ID of the method to invoke. If using `StreamDeckGroup` attribute or `StreamDeckButton` without a defined ID, use the method name.
Parameter	No	Must be enabled if the method requires a parameter to be invoked. Only one parameter per method is supported. Supported values include: `Int`, `Float`, `Bool` and `String`.
Value	Yes (if parameter)	Parameter value to invoke the method. Make sure the value matches a supported type.

Scripted Action

This action allows to invoke a client-configured method on the Unity Editor and built projects.

> **NOTE**
>
> Scripted actions are fully managed by the integration on Unity's side (they don't expose any settings in the Stream Deck software), and must be configured by code.

Parameters

NAME	REQUIRED	DESCRIPTION
Member ID	Yes	ID of this action to be configurable by Unity.

Namespace F10.StreamDeckIntegration

Classes

StreamDeck

Handles adding and removing members that can be executed by Stream Deck.

StreamDeckRuntime

Keeps StreamDeckSocket updated when the integration is running on a build / not in the editor.

StreamDeckSettings

Handles changes to Stream Deck actions, like title and icon control.

StreamDeckSocket

Manages WebSocket connections and messages with the Stream Deck.

Enums

StreamDeckDialAction

Class StreamDeck

Handles adding and removing members that can be executed by Stream Deck.

Inheritance

object

StreamDeck

Namespace: F10.StreamDeckIntegration

Assembly: F10.StreamDeckIntegration.dll

Syntax

```
[ExecuteInEditMode]
public static class StreamDeck
```

Methods

Add(object)

Adds the given object member instance to the list of available StreamDeckButtonAttribute and StreamDeckGroupAttribute.

Declaration

```
public static void Add(object obj)
```

Parameters

TYPE	NAME	DESCRIPTION
object	obj	Boxed member to add.

AddStatic(Type)

Adds the given type as a static member to the list of available StreamDeckButtonAttribute and StreamDeckGroupAttribute.

Declaration

```
public static void AddStatic(Type type)
```

Parameters

TYPE	NAME	DESCRIPTION
Type	type	Type of the static member to add.

Remove(object)

Removes the given object member instance to the list of available StreamDeckButtonAttribute and StreamDeckGroupAttribute.

Declaration

```
public static void Remove(object obj)
```

Parameters

TYPE	NAME	DESCRIPTION

TYPE	NAME	DESCRIPTION
object	obj	Boxed member to remove.

Enum StreamDeckDialAction

Namespace: F10.StreamDeckIntegration

Assembly: F10.StreamDeckIntegration.dll

Syntax

```
public enum StreamDeckDialAction
```

Fields

NAME	DESCRIPTION
Press	
Rotation	

Class StreamDeckRuntime

Keeps StreamDeckSocket updated when the integration is running on a build / not in the editor.

Inheritance

object
Object
Component
Behaviour
MonoBehaviour
StreamDeckRuntime

Namespace: F10.StreamDeckIntegration

Assembly: F10.StreamDeckIntegration.dll

Syntax

```
[DisallowMultipleComponent]
public class StreamDeckRuntime : MonoBehaviour
```

Class StreamDeckSettings

Handles changes to Stream Deck actions, like title and icon control.

Inheritance

object

StreamDeckSettings

Namespace: F10.StreamDeckIntegration

Assembly: F10.StreamDeckIntegration.dll

Syntax

```
[PublicAPI]
public static class StreamDeckSettings
```

Methods

RemoveScriptedButton(string)

Remove the configuration of the Stream Deck scripted action linked by the given ID.

Declaration

```
public static void RemoveScriptedButton(string id)
```

Parameters

TYPE	NAME	DESCRIPTION
string	id	Member ID of the targeted action.

SetButtonImage(Texture2D, string, string, bool)

Set a new image / icon on the Stream Deck action linked to the passed member.
If the target action has a custom icon (manually set icon on the Stream Deck software) changes won't be visible.

Declaration

```
public static void SetButtonImage(Texture2D image, string id, string groupId = null, bool immediate = false)
```

Parameters

TYPE	NAME	DESCRIPTION
Texture2D	image	Texture2D to set as the action's image / action.
string	id	Member ID of the targeted action.
string	groupId	Optional. Group ID of the targeted action. Defaults to null for actions without groups.
bool	immediate	Optional. Allows the setting to be sent immediately, instead of being added to the queue. Defaults to false.

SetButtonTitle(string, string, string, bool)

Set a new title on the Stream Deck action linked to the passed member.
If the target action has a custom title (manually set title on the Stream Deck software) changes won't be visible.

Declaration

```
public static void SetButtonTitle(string title, string id, string groupId = null, bool immediate = false)
```

Parameters

TYPE	NAME	DESCRIPTION
string	title	Text to set as the action's title.
string	id	Member ID of the targeted action.
string	groupId	Optional. Group ID of the targeted action. Defaults to null for actions without groups.
bool	immediate	Optional. Allows the setting to be sent immediately, instead of being added to the queue. Defaults to false.

SetButtonValue(object, string, string, bool)

Set the value of the parameter on the Stream Deck action linked to the passed member.

Declaration

```
public static void SetButtonValue(object value, string id, string groupId = null, bool immediate = false)
```

Parameters

TYPE	NAME	DESCRIPTION
object	value	A supported type value to set on the targeted action.
string	id	Member ID of the targeted action.
string	groupId	Optional. Group ID of the targeted action. Defaults to null for actions without groups.
bool	immediate	Optional. Allows the setting to be sent immediately, instead of being added to the queue. Defaults to false.

SetScriptedButton(string, Action, string, Texture2D, bool)

Configure the Stream Deck scripted action linked by the given ID.

Declaration

```
public static void SetScriptedButton(string id, Action callback, string title = null, Texture2D image = null, bool immediate = false)
```

Parameters

TYPE	NAME	DESCRIPTION
string	id	Member ID of the targeted action.
Action	callback	Method to invoke on action trigger.
string	title	Text to set as the action's title.
Texture2D	image	Texture2D to set as the action's image / action.
bool	immediate	Optional. Allows the setting to be sent immediately, instead of being added to the queue. Defaults to false.

Class StreamDeckSocket

Manages WebSocket connections and messages with the Stream Deck.

Inheritance

object

StreamDeckSocket

Namespace: F10.StreamDeckIntegration

Assembly: F10.StreamDeckIntegration.dll

Syntax

```
public static class StreamDeckSocket
```

Properties

IsConnected

Is the WebSocket connected and alive.

Declaration

```
[PublicAPI]
public static bool IsConnected { get; }
```

Property Value

TYPE	DESCRIPTION
bool	

Methods

Connect()

Try to connect or re-connect to any available Stream Deck.

Declaration

```
[PublicAPI]
public static void Connect()
```

Disconnect()

Disconnect from any connected Stream Deck.

Declaration

```
[PublicAPI]
public static void Disconnect()
```

OnUpdate()

Handle one queued received and / or sent message.

Declaration

```
[PublicAPI]
public static void OnUpdate()
```

Namespace F10.StreamDeckIntegration.Attributes

Classes

StreamDeckButtonAttribute

Attribute to mark specific fields, properties and / or methods as executable by Stream Deck.

StreamDeckDialAttribute

Attribute to mark specific methods as executable by Stream Deck dials.

StreamDeckGroupAttribute

Attribute to mark specific classes with all it's members as executable by Stream Deck.

Class StreamDeckButtonAttribute

Attribute to mark specific fields, properties and / or methods as executable by Stream Deck.

Inheritance

object

Attribute

StreamDeckButtonAttribute

Namespace: F10.StreamDeckIntegration.Attributes

Assembly: F10.StreamDeckIntegration.dll

Syntax

```
[PublicAPI]
[MeansImplicitUse]
[AttributeUsage(AttributeTargets.Method|AttributeTargets.Property|AttributeTargets.Field)]
public class StreamDeckButtonAttribute : Attribute
```

Constructors

StreamDeckButtonAttribute(string)

Marks this field, property or method as executable by the Stream Deck.

Declaration

```
public StreamDeckButtonAttribute(string id = null)
```

Parameters

TYPE	NAME	DESCRIPTION
string	id	Custom ID. Defaults to the name of the member.

Class StreamDeckDialAttribute

Attribute to mark specific methods as executable by Stream Deck dials.

Inheritance

object

Attribute

StreamDeckDialAttribute

Namespace: F10.StreamDeckIntegration.Attributes

Assembly: F10.StreamDeckIntegration.dll

Syntax

```
[PublicAPI]
[MeansImplicitUse]
[AttributeUsage(AttributeTargets.Method)]
public class StreamDeckDialAttribute : Attribute
```

Constructors

StreamDeckDialAttribute(string)

Marks this field, property or method as executable by the Stream Deck.

Declaration

```
public StreamDeckDialAttribute(string id = null)
```

Parameters

TYPE	NAME	DESCRIPTION
string	id	Custom ID. Defaults to the name of the member.

Class StreamDeckGroupAttribute

Attribute to mark specific classes with all it's members as executable by Stream Deck.

Inheritance

object

Attribute

StreamDeckGroupAttribute

Namespace: F10.StreamDeckIntegration.Attributes

Assembly: F10.StreamDeckIntegration.dll

Syntax

```
[PublicAPI]
[MeansImplicitUse]
[AttributeUsage(AttributeTargets.Class)]
public class StreamDeckGroupAttribute : Attribute
```

Constructors

StreamDeckGroupAttribute(string)

Marks all the fields, properties and methods inside the class as executable by the Stream Deck.

Declaration

```
public StreamDeckGroupAttribute(string id = null)
```

Parameters

TYPE	NAME	DESCRIPTION
string	id	Custom ID. Defaults to the name of the class.

Namespace F10.StreamDeckIntegration.Demo

Classes

StreamDeckExample

Example class using StreamDeckButtonAttribute attributes and runtime support.

Class StreamDeckExample

Example class using StreamDeckButtonAttribute attributes and runtime support.

Inheritance

object
Object
Component
Behaviour
MonoBehaviour
StreamDeckExample

Namespace: F10.StreamDeckIntegration.Demo

Assembly: Assembly-CSharp.dll

Syntax

```
public class StreamDeckExample : MonoBehaviour
```

Fields

_field

Declaration

```
[Header("Check the source code for commented API!")]
[StreamDeckButton(null)]
public int _field
```

Field Value

TYPE	DESCRIPTION
int	

Methods

ExampleMethod()

Declaration

```
[StreamDeckButton("ExampleMethod")]
public void ExampleMethod()
```

Namespace F10.StreamDeckIntegration.Demo.Editor

Classes

StreamDeckStaticExample

Example class using StreamDeckGroupAttribute attributes and static support.

Class StreamDeckStaticExample

Example class using StreamDeckGroupAttribute attributes and static support.

Inheritance

object

StreamDeckStaticExample

Namespace: F10.StreamDeckIntegration.Demo.Editor

Assembly: Assembly-CSharp-Editor.dll

Syntax

```
[ExecuteInEditMode]
[InitializeOnLoad]
[StreamDeckGroup(null)]
public static class StreamDeckStaticExample
```

Methods

ExampleStaticFields()

Declaration

```
public static void ExampleStaticFields()
```

ExampleStaticMethod()

Declaration

```
public static void ExampleStaticMethod()
```