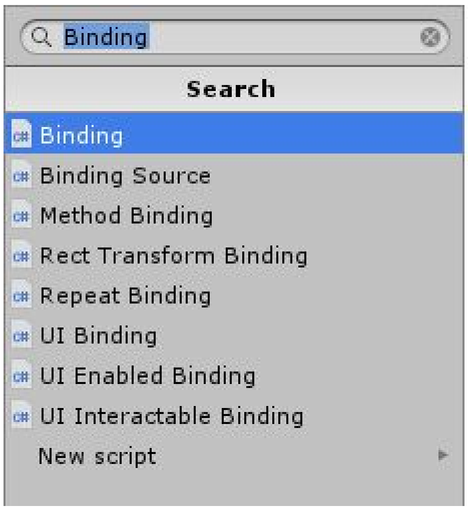
UBindr User Manual

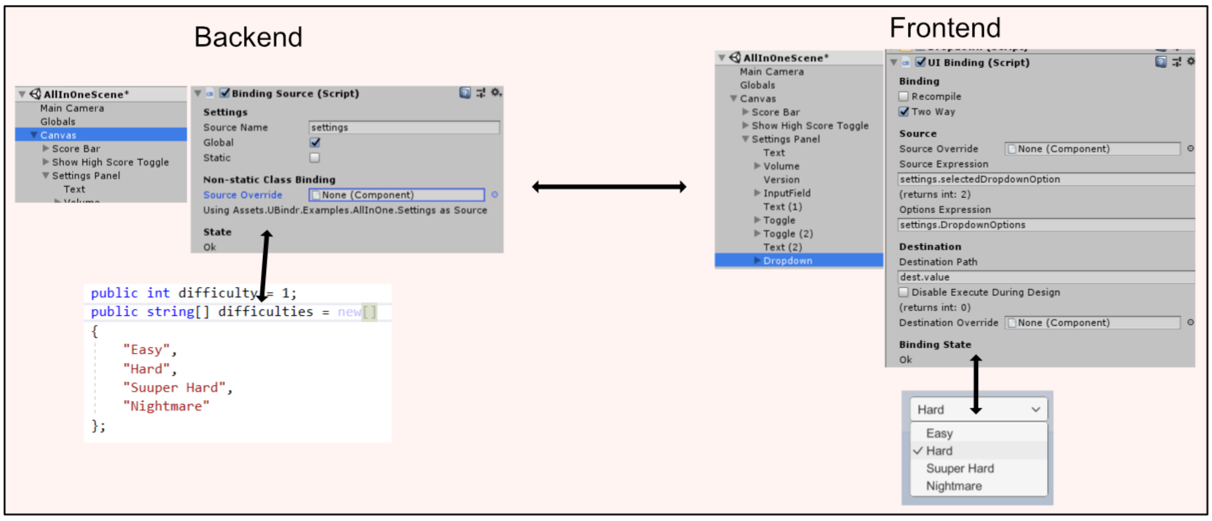
A user manual for the UBindr asset package. For support contact mattias.fagerlund@carretera.se. For a more hands-on approach, see;

- [Overview]()
- [Tutorial 1]()
- [Tutorial 2]()
- [Tutorial 3]()
- [Tutorial Movie]()

UBindr allows you to publish data from your C# code to the Unity UI with minimal to no code changes. It supports two way binding, so that changes made by the user are passed back to the C# code. It fills some of the same need that Angular, Knockout or React fills for Web applications. It makes building UIs easier and leaves the code cleaner.



Components



Binding Source (Script)
Settings
Source Name: settings
Global: ✓
Static:
Non-static Class Binding
Source Override: None (Component)
Using Assets.UBindr.Examples.AllInOne.Settings as Source
State
Ok

```
public int difficulty = 1;
public string[] difficulties = new[]
{
    "Easy",
    "Hard",
    "Suuper Hard",
    "Nightmare"
};
```

UI Binding (Script)
Binding
Recompile:
Two Way: ✓
Source
Source Override: None (Component)
Source Expression: settings.selectedDropdownOption
(returns int: 2)
Options Expression: settings.DropdownOptions
Destination
Destination Path: dest.value
Disable Execute During Design:
(returns int: 0)
Destination Override: None (Component)
Binding State
Ok

Hard
Easy
✓ Hard
Suuper Hard
Nightmare

UBindr comes with a number of components to provide binding functionality. The process typically looks like this;

- Provide a scripts of your own that you wish to bind to (Player, Highscore, ServerConnection etc)
- Create a Binding Source that publishes the values of the script.
- Create a UI component (Text, InputField etc)
- Add a UI Binding just below the UI component that binds a value from the script above to the value (or text) of the UI component.

Binding Source

UBindr binds Binding Sources, through Bindings, to Destinations. The BindingSource class represents the data sources where you collect the data you wish to bind to.

Source Name

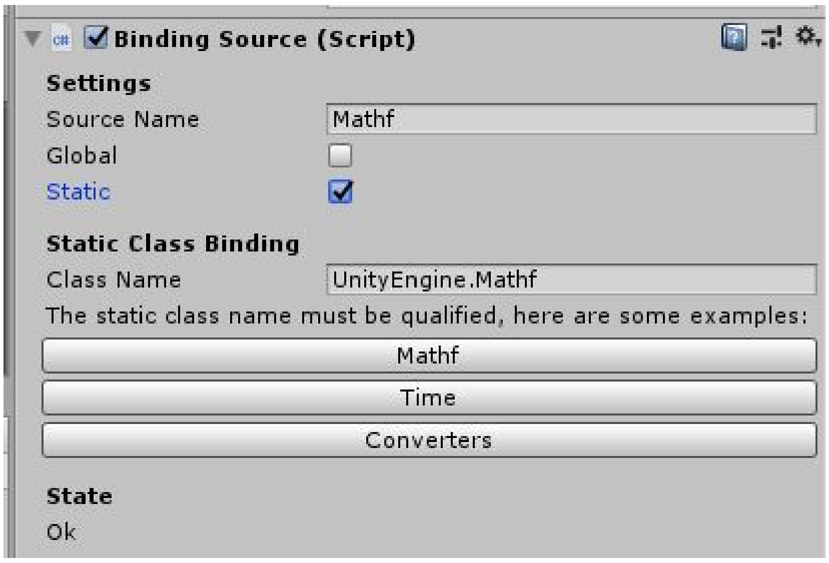
Bindings contain expressions - pieces of code that determine what value you wish to use in a binding (think “settings.player.fullName”). Binding Sources is made available to Bindings/binding expressions using their Source Name. In the binding expression “settings.serverName”, the “settings” part represents a Binding Source with the Source Name “settings”. The “serverName” part represents a property or field in the source with the name “serverName”. SourceNames don’t need to be unique, but you can only access one BindingSource per name - if there are multiple only one will be in scope.

Global

A Binding Source can be global (available anywhere in the hierarchy) or non global (available in the branch of the hierarchy where it’s found).

Static

A BindingSource can be static (publishes a static class with static properties, like “Math”) or non-static (publishes to an instance of a class).



Binding Source (Script)

Settings

Source Name: Mathf

Global: ☐

Static: ✓

Static Class Binding

Class Name: UnityEngine.Mathf

The static class name must be qualified, here are some examples:

Mathf

Time

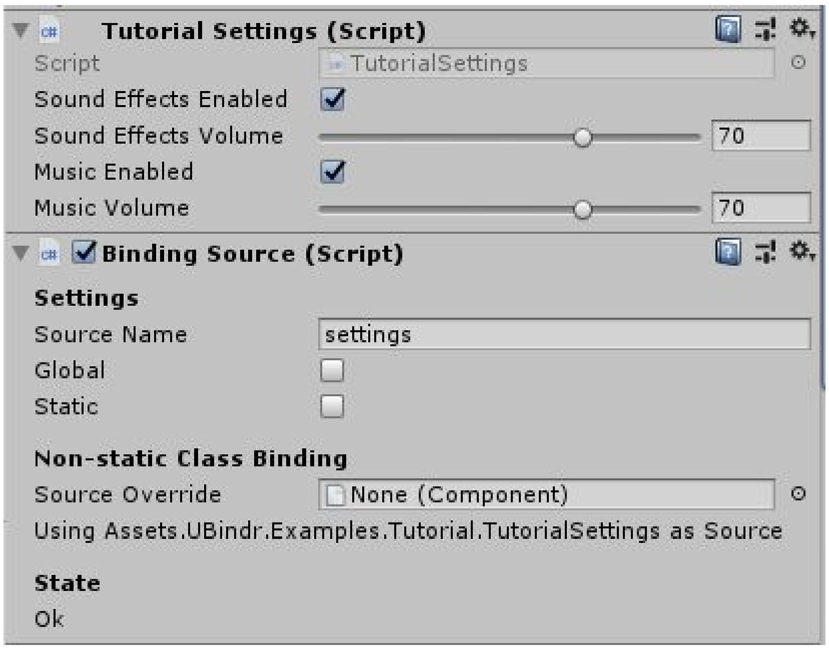
Converters

State

Ok

Source Override

By default, the Binding Source publishes the Component/Script that’s above it in the Game Object that it finds itself. In the example below, the Binding Source “settings” binds to the component “Tutorial Settings”.



Tutorial Settings (Script)

Script: TutorialSettings

Sound Effects Enabled: ✓

Sound Effects Volume: 70

Music Enabled: ✓

Music Volume: 70

Binding Source (Script)

Settings

Source Name: settings

Global: ☐

Static: ☐

Non-static Class Binding

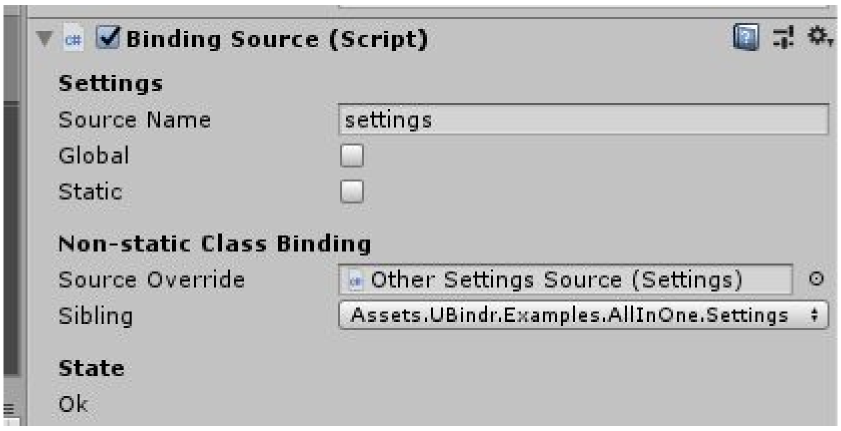
Source Override: None (Component)

Using Assets.UBindr.Examples.Tutorial.TutorialSettings as Source

State

Ok

But you can override this behaviour by populating the Source Override field.



Binding Source (Script)

Settings

Source Name: settings

Global: ☐

Static: ☐

Non-static Class Binding

Source Override: Other Settings Source (Settings)

Sibling: Assets.UBindr.Examples.AllInOne.Settings

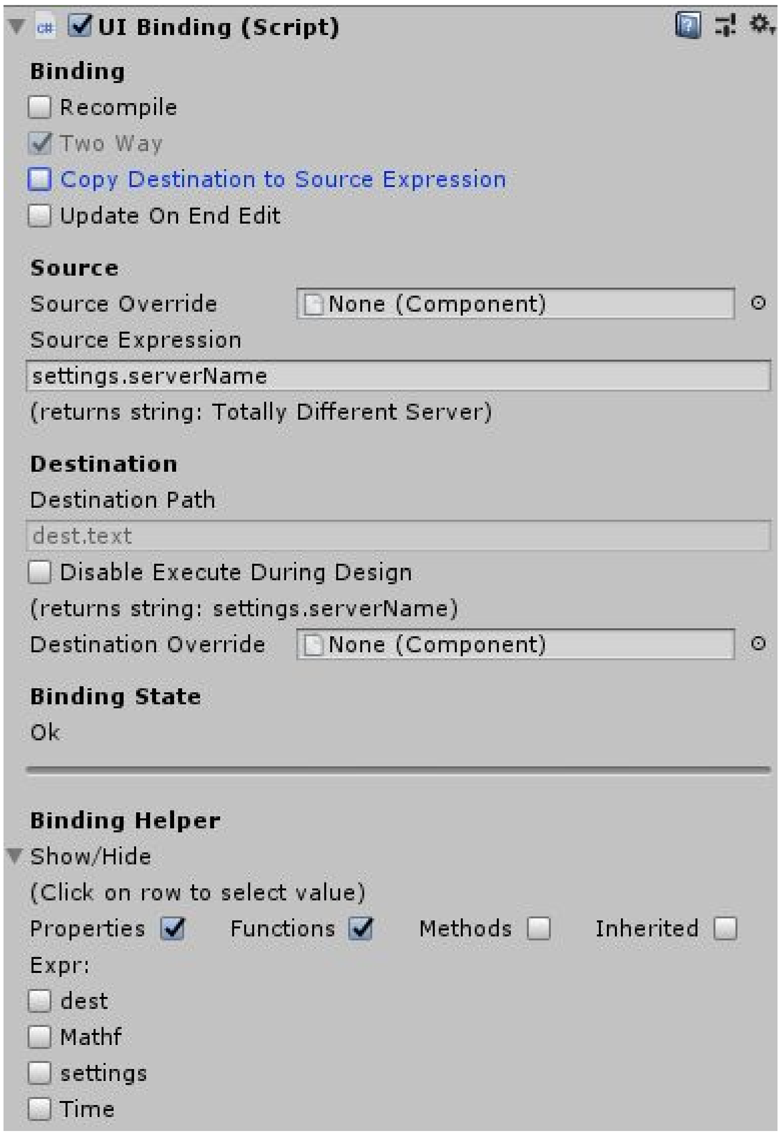
State

Ok

The “Sibling” dropdown appears when you populate Source Override - it helps you pick any script on the game object you’ve dropped into the Source Override field.

UI Binding

UI Binding is the main binding you’ll use when designing UIs. It’s a binding specialized for binding to UI Components (Toggle, Dropdown, InputField, Text, Slider, Scrollbar).


UI Binding (Script)

Binding

☐ Recommpile

✓ Two Way

☐ Copy Destination to Source Expression

☐ Update On End Edit

Source

Source Override	None (Component)

Source Expression

settings.serverName

(returns string: Totally Different Server)

Destination

Destination Path

dest.text

☐ Disable Execute During Design

(returns string: settings.serverName)

Destination Override	None (Component)

Binding State

Ok

Binding Helper

Show/Hide

(Click on row to select value)

Properties ✓ Functions ✓ Methods ☐ Inherited ☐

Expr:

☐ dest

☐ Mathf

☐ settings

☐ Time

Recompile

UBindr is designed to re-use compiled expressions to reduce the performance impact of bindings. If you for any reason would like to recompile the expression each frame, set this value to true.

Two Way

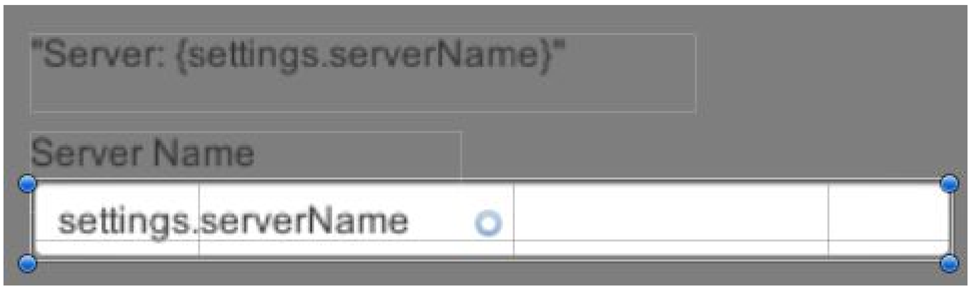
Determines if the binding should be one way or two way. A one way binding copies the value from the Source Expression to the Destination Path and ignores what's on the Destination Path. A two way binding copies changes that appear on the Destination Path back to the Source Expression. This only works if the Source Expression is a field or property. If the Source Expression is "1+2", then writing the value of the Destination Path doesn't make sense.

In the UI Binding component, this value cannot be altered as it's automatically set to what appropriate for the destination component.

Copy Destination to Source Expression

For text based components (Text or InputField), it's convenient to store the Source Expression in the ".text" field of the destination component. That way, you get a good sense, by looking at the UI, what expression is being bound. If "Copy Destination to Source Expression" is true, whatever value is found in the .text field when the application starts is used for Source Expression. The Source Expression field isn't visible when the value is true.

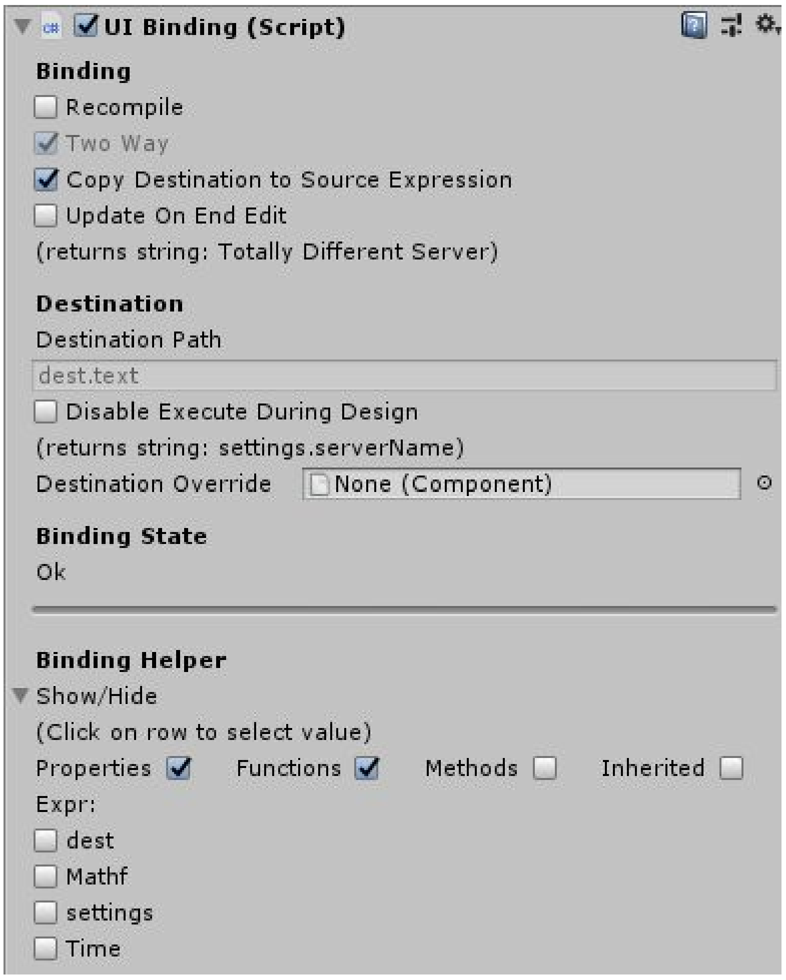
If it's set to false, the Source Expression becomes available and you can enter your expressions there. Any value found in the .text field will be ignored.


"Server: {settings.serverName}"

Server Name

settings.serverName

Binding Expressions in Text Based Components


UI Binding (Script)

Binding

- [ ] Recompile
- [✓] Two Way
- [✓] Copy Destination to Source Expression
- [ ] Update On End Edit

(returns string: Totally Different Server)

Destination

Destination Path

dest.text

- [ ] Disable Execute During Design

(returns string: settings.serverName)

Destination Override: None (Component)

Binding State

Ok

Binding Helper

Show/Hide

(Click on row to select value)

Properties [✓] Functions [✓] Methods [ ] Inherited [ ]

Expr:

- [ ] dest
- [ ] Mathf
- [ ] settings
- [ ] Time

No Source Expression Field is available when value is true

Update On End Edit

When the destination component is a InputField, this value determines if the binding should be continuous (update with every key-stroke) or non-continuous (update only when editing is done). When editing floats for instance, this value should be set to true.

Disable Execute During Design

Prevents the Binding from being executed during design time.

Source Override

Allows you to introduce a Binding Source that isn’t represented by a Binding Source component. Drop a component/script reference here to bind to any component in your hierarchy. The Source Override value will be presented with the source name “source”.

Source Expression

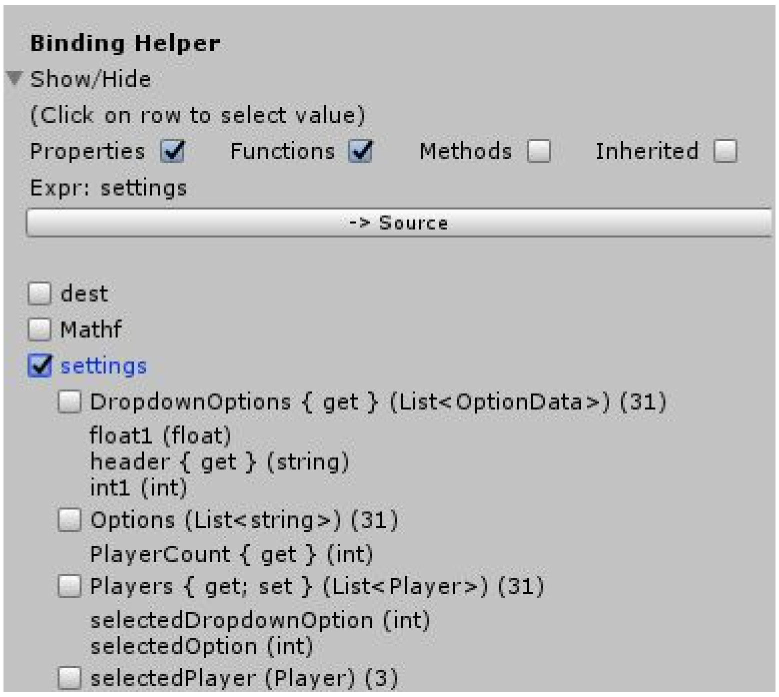
Here you can enter a simplified c# expression that will be evaluated each frame and the value will be copied to the Destination Path. Examples of acceptable source expressions are

- settings.selectedPlayer.name
- settings.GetFirstPlayer().name
- settings.players.Count>0?”Has players”:”No players found”

Destination Override

By default, the destination of a binding is the component above it. If you wish a binding to have a different Destination, you can populate this field.

Binding Helper



Binding Helper
Show/Hide
(Click on row to select value)
Properties ✓ Functions ✓ Methods ☐ Inherited ☐
Expr: settings
-> Source

☐ dest
☐ Mathf
✓ settings
  ☐ DropdownOptions { get } (List<OptionData>) (31)
  float1 (float)
  header { get } (string)
  int1 (int)
  ☐ Options (List<string>) (31)
  PlayerCount { get } (int)
  ☐ Players { get; set } (List<Player>) (31)
  selectedDropdownOption (int)
  selectedOption (int)
  ☐ selectedPlayer (Player) (3)

The Binding Helper is a tool that takes the guesswork out of creating simple expressions. It informs you which Binding Sources are available and allows you to drill down into them. You can filter the choices to include properties, functions, methods and inherited members.
It also provides you with methods to copy the expression you've designed into the Source Expression or Destination Path of the Binding.

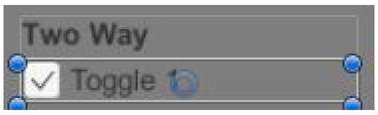
Nothing prevents you from typing in your expressions instead, but the risk of spelling issues makes it worthwhile to use the Binding Helper tool.

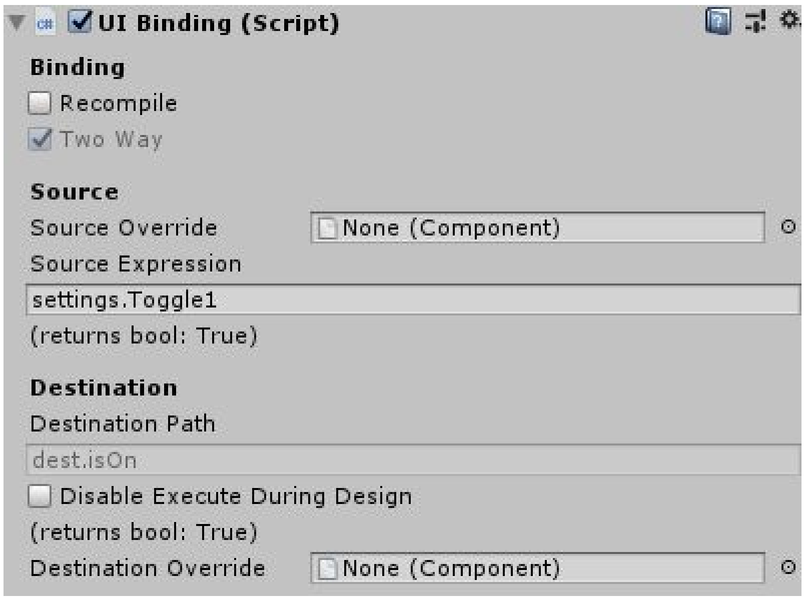
For the UI Binding the Destination Path is determined by the component type of the destination (.text for InputField or Text and .value for slider etc), so the Source Expression cannot be set.

UI Binding Destinations

The UI Binding Binding contains code to handle a number of specific UI components;

Toggle




UI Binding (Script)

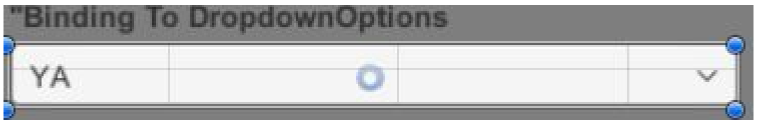
Binding
Recompile: ☐
Two Way: ✓

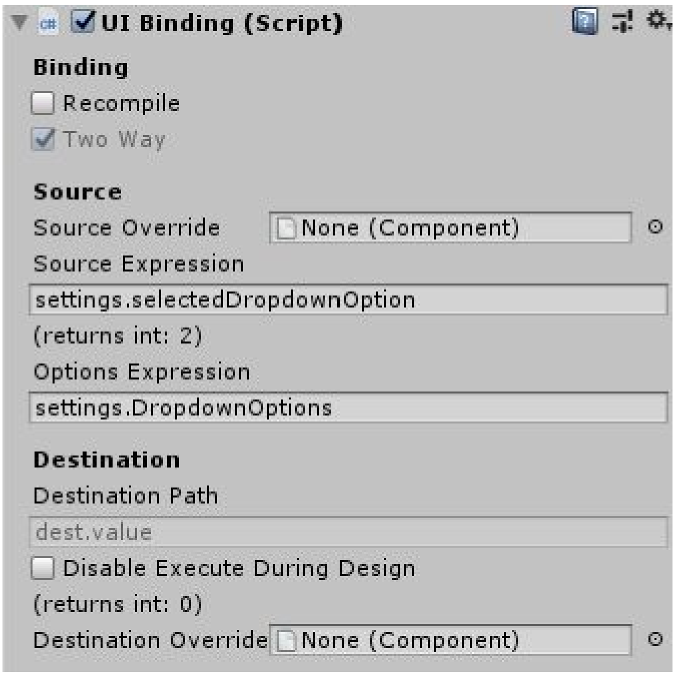
Source
Source Override: None (Component)
Source Expression
settings.Toggle1
(returns bool: True)

Destination
Destination Path
dest.isOn
Disable Execute During Design: ☐
(returns bool: True)
Destination Override: None (Component)

When placed on a Toggle, UI Binding will become a Two Way binding to the Toggle.isOn field.

Dropdown




UI Binding (Script)

Binding
Recompile: ☐
Two Way: ✓

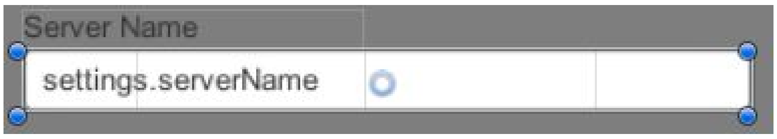
Source
Source Override: None (Component)
Source Expression
settings.selectedDropdownOption
(returns int: 2)
Options Expression
settings.DropdownOptions

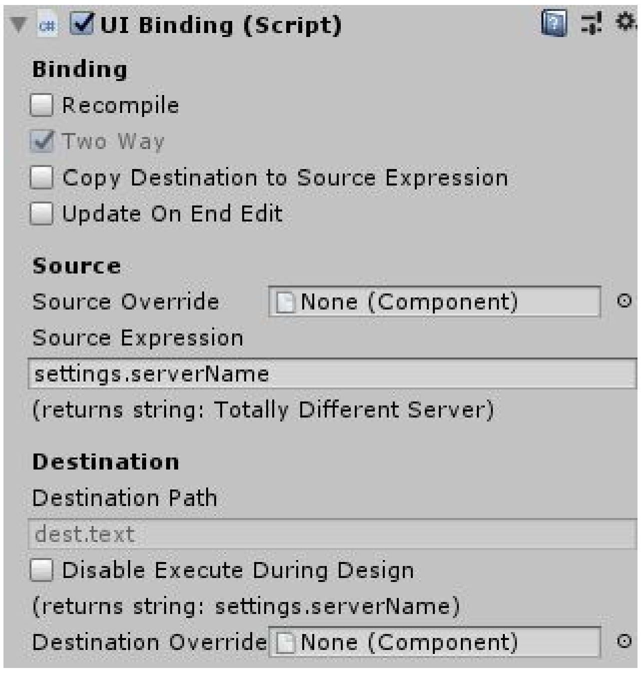
Destination
Destination Path
dest.value
Disable Execute During Design: ☐
(returns int: 0)
Destination Override: None (Component)

When placed on a Dropdown, UI Binding will become a Two Way binding to the Dropdown.value field.
It will also provide an **Options Expression** that can be used to populate the Dropdown.options field. The expression should evaluate to an IEnumerable of Dropdown.OptionData, which will be used as is, or a list of object, where each object will be added as an option with it's "ToString" value as the name.

Rev 1.0. For support contact mattias.fagerlund@carretera.se.

InputField




UI Binding (Script)

Binding

- [ ] Recompile
- [✓] Two Way
- [ ] Copy Destination to Source Expression
- [ ] Update On End Edit

Source

Source Override: None (Component)

Source Expression

```
settings.serverName
```

(returns string: Totally Different Server)

Destination

Destination Path

```
dest.text
```

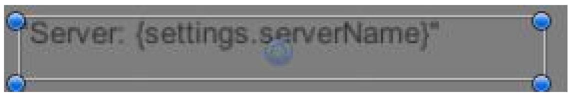
- [ ] Disable Execute During Design

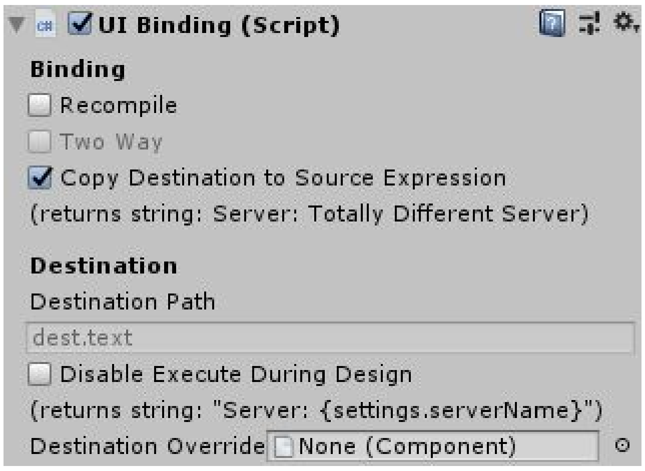
(returns string: settings.serverName)

Destination Override: None (Component)

When placed on a InputField, UI Binding will become a Two Way binding to the InputField.text field. By default, it will set Copy Destination to Source Expression to true, but you can set it to false as required.

Text




UI Binding (Script)

Binding

- [ ] Recompile
- [ ] Two Way
- [✓] Copy Destination to Source Expression

(returns string: Server: Totally Different Server)

Destination

Destination Path

```
dest.text
```

- [ ] Disable Execute During Design

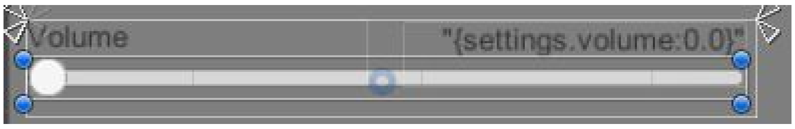
(returns string: "Server: {settings.serverName}")

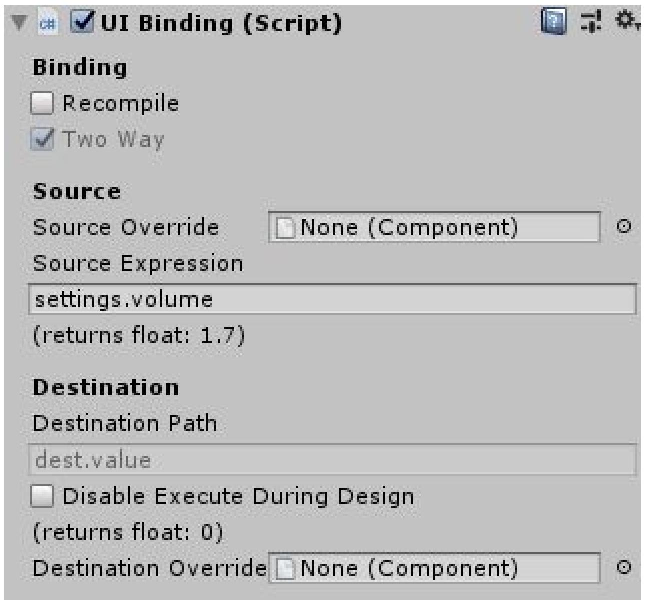
Destination Override: None (Component)

When placed on a Text, UI Binding will become a One Way binding to the Text.text field. By default, it will set Copy Destination to Source Expression to true, but you can set it to false as required.

Rev 1.0. For support contact mattias.fagerlund@carretera.se.

Slider




UI Binding (Script)

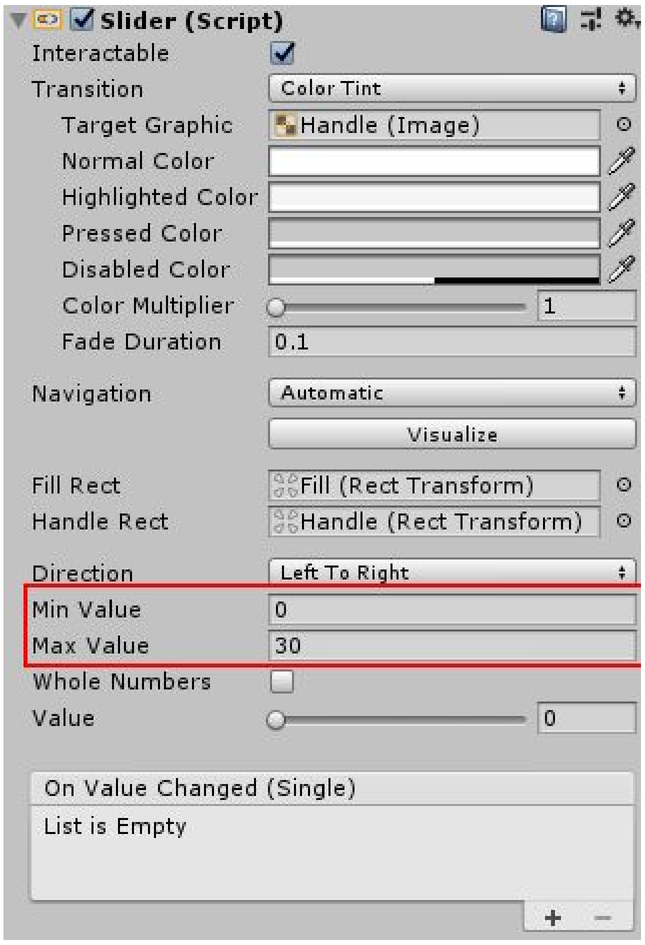
Binding
☐ Recompile
☑ Two Way

Source
Source Override: None (Component)
Source Expression
settings.volume
(returns float: 1.7)

Destination
Destination Path
dest.value
☐ Disable Execute During Design
(returns float: 0)
Destination Override: None (Component)

When placed on a Slider, UI Binding will become a One Way binding to the Slider.value field. If the Source Expression is the path to a field with “UnityEngine.RangeAttribute” attribute on it, the binding will update the min/max values to match the range attribute.

```
[Range(0, 30)]
public float volume;
```



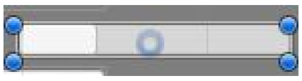
Slider (Script)

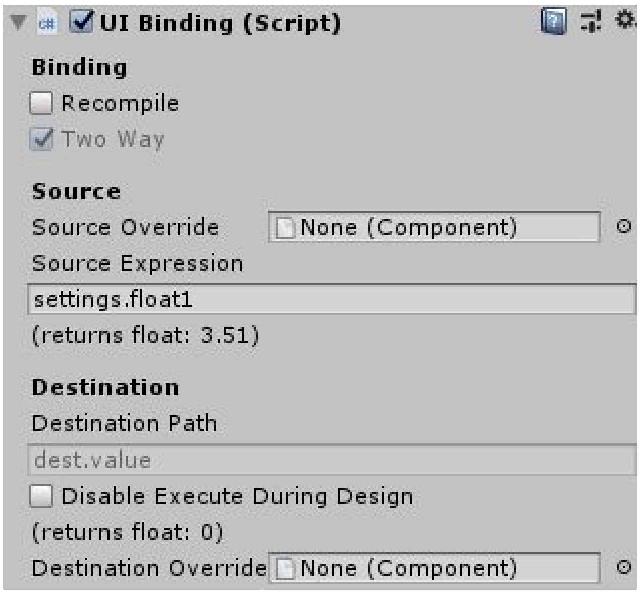
Interactable	✓
Transition	Color Tint
Target Graphic	Handle (Image)
Normal Color	
Highlighted Color	
Pressed Color	
Disabled Color	
Color Multiplier	1
Fade Duration	0.1
Navigation	Automatic
	Visualize
Fill Rect	Fill (Rect Transform)
Handle Rect	Handle (Rect Transform)
Direction	Left To Right
Min Value	0
Max Value	30
Whole Numbers	☐
Value	0

On Value Changed (Single)

List is Empty

Scrollbar





UI Binding (Script)

Binding

☐ Recompile

✓ Two Way

Source

Source Override	None (Component)

Source Expression

settings.float1

(returns float: 3.51)

Destination

Destination Path

dest.value

☐ Disable Execute During Design

(returns float: 0)

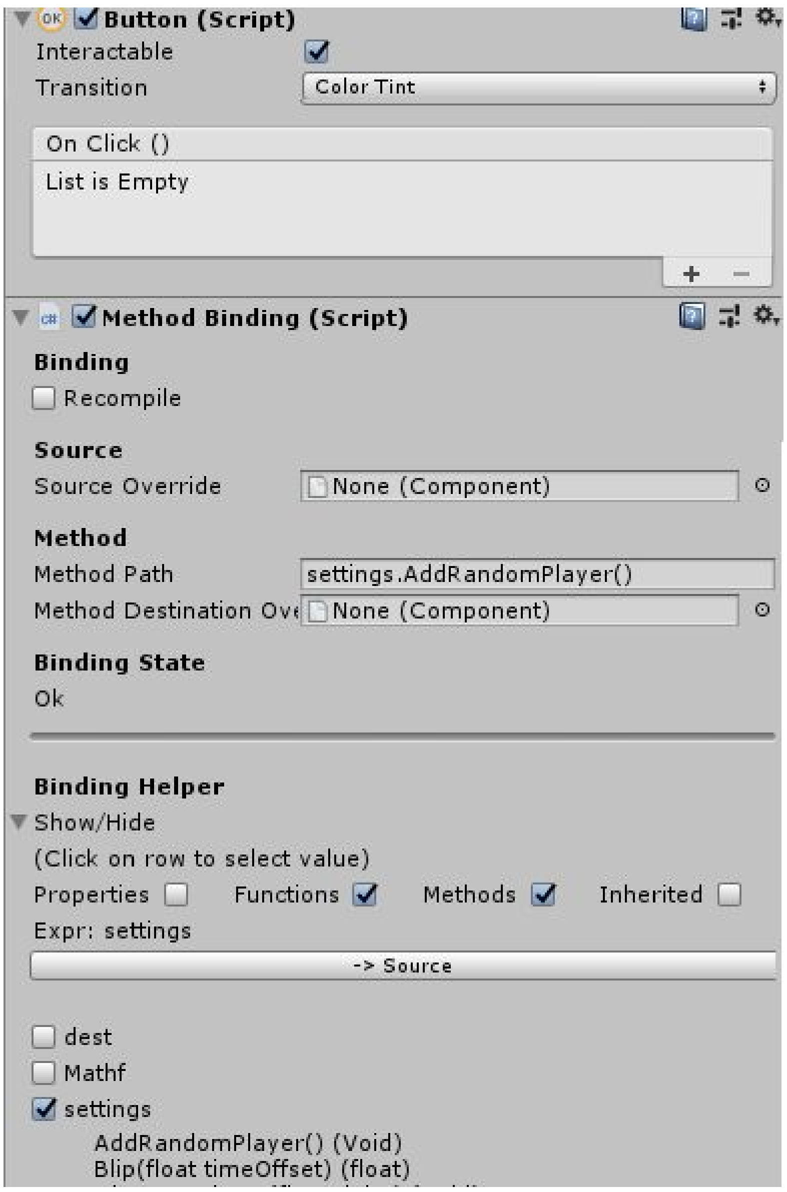
Destination Override	None (Component)

Rev 1.0. For support contact mattias.fagerlund@carretera.se.

When placed on a Scrollbar, UI Binding will become a Two Way binding to the Scrollbar.value field.

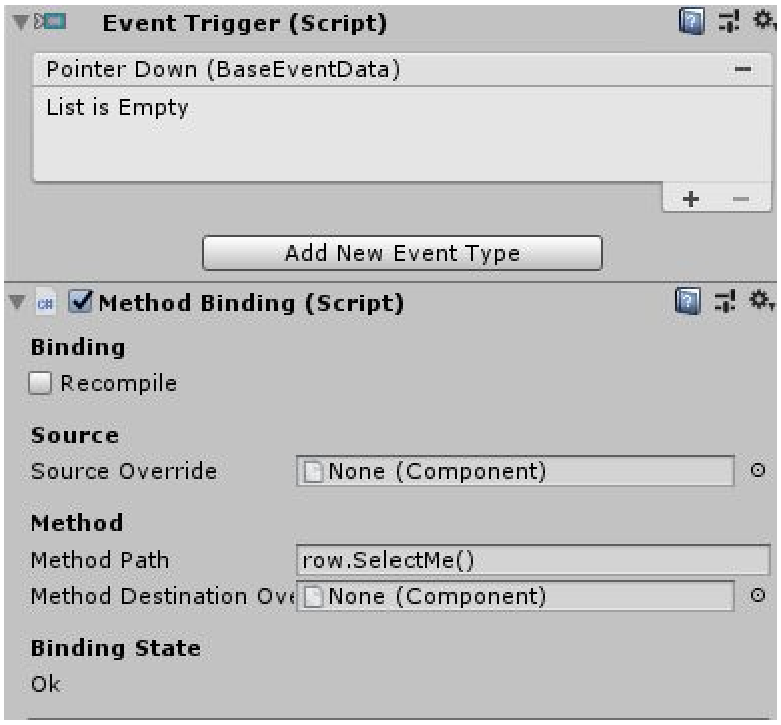
Method Binding

Method Bindings bind Event Triggers to method expressions on Binding Sources. For instance, when a button is clicked, “settings.AddRandomPlayer()” is executed. The advantage over using the Unity Event Triggers directly is that Method Binding supports complex binding expressions, such as “settings.GetSelectedGame().AddRandomPlayer()”. And if you’re using Binding Sources for the other parts of the UI, Method Bindings are a natural extension.


Button (Script)
Interactable ✓
Transition: Color Tint
On Click ()
List is Empty
Method Binding (Script)
Binding
☐ Recompile
Source
Source Override: None (Component)
Method
Method Path: settings.AddRandomPlayer()
Method Destination Ove: None (Component)
Binding State
Ok
Binding Helper
Show/Hide
(Click on row to select value)
Properties ☐ Functions ✓ Methods ✓ Inherited ☐
Expr: settings
-> Source
☐ dest
☐ Mathf
✓ settings
AddRandomPlayer() (Void)
Blip(float timeOffset) (float)

If you for instance want to bind the “Pointer Down” event of a panel to a method, add an Event Trigger component to the Panel. Click “Add New Event Type” and select “Pointer

Down". Then add a Method Binding after the Event Trigger and set the Method Path to the method you wish to invoke;



Binding
Recompile

Source
Source Override: None (Component)

Method
Method Path: row.SelectMe()
Method Destination Ove: None (Component)

Binding State
Ok

Method Path

The path to the method to invoke, for instance "settings.OpenServerUi()" or "player.SelectedItem.DropItem()".

Method Destination Override

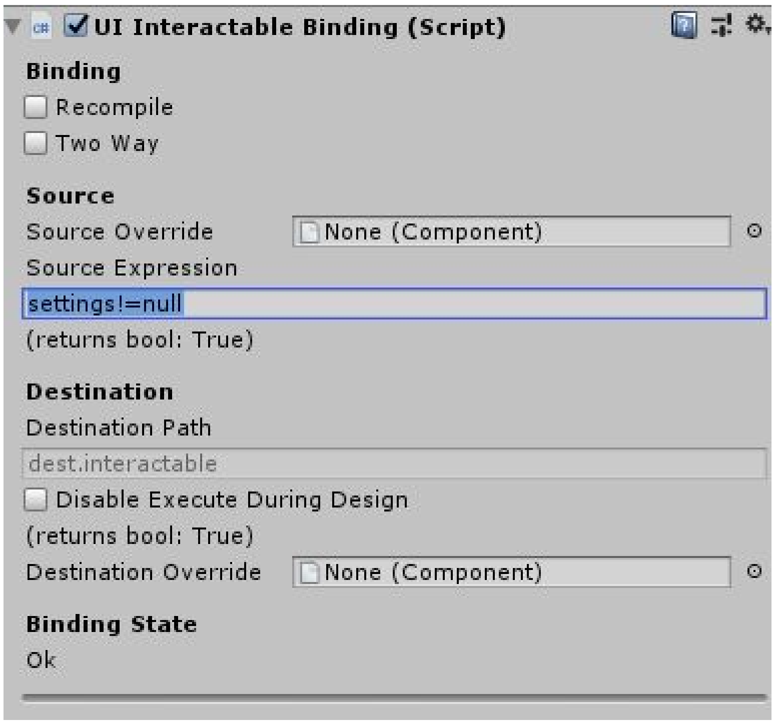
Usually a Binding will include the component just above it as a Binding Source named "dest". Populate this field if you wish to publish another component as "dest".

Other Fields

See UI Binding.

UI Interactable Binding

When you're using ui components, sometimes you wish to make them read only. All UI components has a field called "interactable" which is TRUE when the component is NOT readonly. And FALSE when the component is readonly. So note how it's inverted. UBindr provides a specialized binding called UI Interactable Binding that takes an expression returns a boolean and binds that to the interactable field of the destination.


UI Interactable Binding (Script)

Binding

Recompile

Two Way

Source

Source Override: None (Component)

Source Expression

```
settings!=null
```

(returns bool: True)

Destination

Destination Path

```
dest.interactable
```

Disable Execute During Design

(returns bool: True)

Destination Override: None (Component)

Binding State

Ok

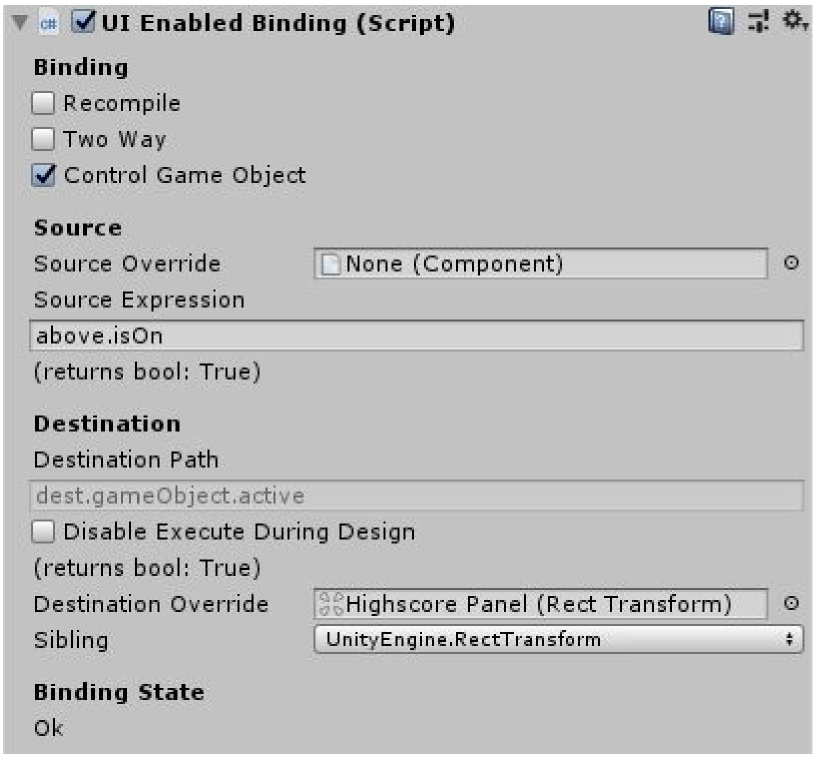
Source Expression

Should return true if the destination component should be interactable, and false if it shouldn't be. Notice how this is the opposite of “readonly”. See UI Binding for more details.

Other Fields

See UI Binding.

UI Enabled Binding


UI Enabled Binding (Script)

Binding

☐ Recompile

☐ Two Way

☑ Control Game Object

Source

Source Override: None (Component)

Source Expression: above.isOn

(returns bool: True)

Destination

Destination Path: dest.gameObject.active

☐ Disable Execute During Design

(returns bool: True)

Destination Override: Highscore Panel (Rect Transform)

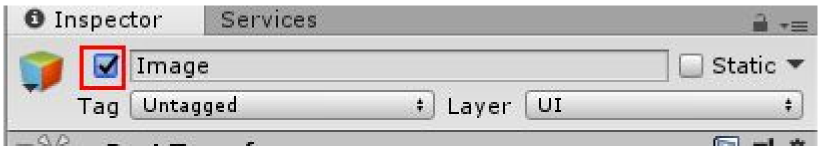
Sibling: UnityEngine.RectTransform

Binding State

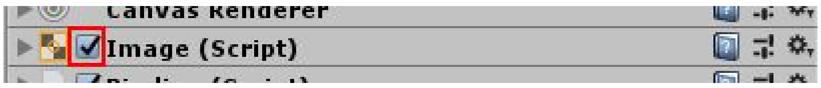
Ok

If you wish to disable an entire GameObject (which will hide it) or a particular script on a game object (will prevent it from updating), you can do that using a UI Enabled Binding component.

Control Game Object



If this field is true, the binding will control if the Game Object is active or not. Setting a Game Object to Active=false means it will be hidden and no scripts on it will be executed. So if you use Control Game Object, the Binding should placed outside the GameObject in question, or it will never be set to active once it's been set to inactive.



If Control Game Object is false, the binding will control a component instead. If you have an Image or a Sprite Renderer component on a Game Object, setting it to disabled will prevent the graphics from rendering. All other functionality of the Game Object will continue to execute.

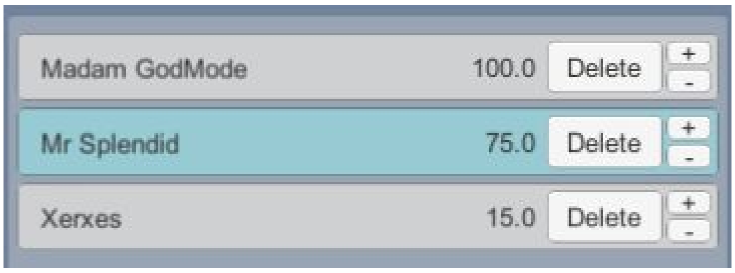
Source Expression

Should return true if the destination should be active/enabled and false if it should be disabled/inactive.

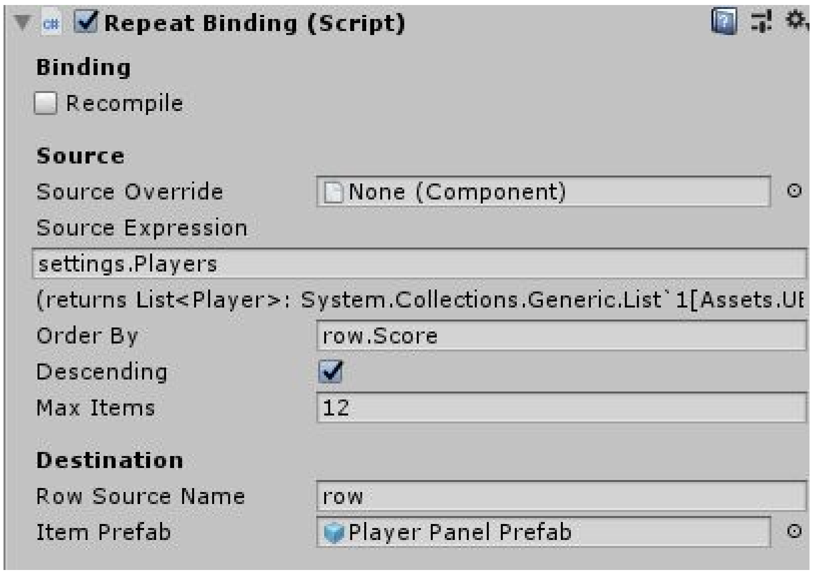
Other Fields

See UI Binding.

Repeat Binding



A binding for generating lists of objects. It will iterate over the values retrieved when executing the source expression and it will instantiate a copy of the "Item Prefab" GameObject for each item it finds. The source expression must return an IEnumerable. If new items are added to the IEnumerable returned by the source expression, a new GameObjects will be instantiated. If existing items are removed, the matching GameObject will be destroyed.
All instantiated Game Objects will be parented to the transform on which the RepeatBinding exists. If you need to handle the positioning of the instantiated objects, you can either do that by the regular UI positioning scripts (Vertical Layout Group etc) or by using a RectTransformBinding and binding the x/y using expressions.
The binding supports sorting and you can limit how many rows are generated.



Repeat Binding (Script)

Binding

☐ Recompile

Source

Source Override	None (Component)
Source Expression	

settings.Players

(returns List<Player>: System.Collections.Generic.List`1[Assets.UI

Order By	row.Score
Descending	✓
Max Items	12

Destination

Row Source Name	row
Item Prefab	Player Panel Prefab

Rev 1.0. For support contact mattias.fagerlund@carretera.se.

Source Expression

Should return an IEnumerable of objects. The Repeat Binding will generate an instance of the Item Prefab for each item in the IEnumerable. As new items appear

Order By

An expression that's used to sort the IEnumerable before it's used. For instance "row.score" or "row.Values.Average()"

Descending

Determines the sort order IEnumerable should ascending or descending. Used with orderBy.

Max Items

Allows the user to limit how many items from the IEnumerable should be included. -1 means all items are included.

Row Source Name

Each child GameObject instantiated for a row in the bound IEnumerable will have access to a binding source representing the row. The name of that Binding Source will be whatever rowSourceName contains. If you have RepeatBindings within RepeatBindings, you can use different names for this property to access destinations higher up in the hierarchy. If all use "row", the lowest accessible RepeatBinding will hide those higher up in the hierarchy.

Item Prefab

The GameObject that should be instantiated for each item in the IEnumerable. The instantiated GameObject will be parented to the Transform that the RepeatBinding exists on.

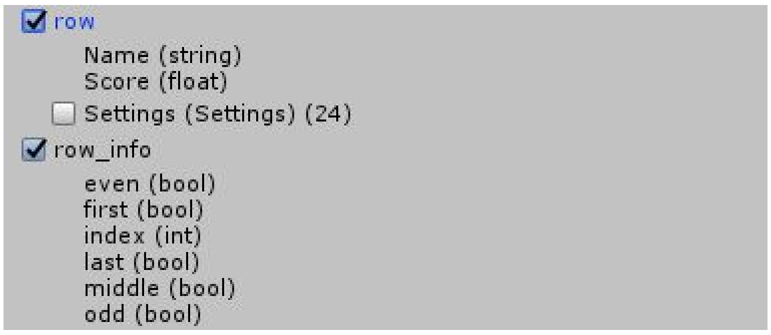
Other Fields

See UI Binding.

Repeat Binding Rows

For each item in the IEnumerable that the Repeat Binding Source Expression contains it will instantiate a copy of the Item Prefab Game Object called a row. Each row will have access to two Repeat Binding specific Binding Sources - one called "row" (assuming the Repeat Binding has the Row Source Name "row") and one called "row_info". If the Repeat Binding has the Row Source Name "player" the two Binding Sources will be called "player" and "player_info".

The "row" Binding Source publishes the item from the IEnumberable - so you can bind to any field on that item. The "row_info" Binding Source contains fields that represent the positioning of the row within the IEnumerable. Note that these values can change over time as the IEnumerable is modified.



```
☑ row
    Name (string)
    Score (float)
  ☐ Settings (Settings) (24)
☑ row_info
    even (bool)
    first (bool)
    index (int)
    last (bool)
    middle (bool)
    odd (bool)
```

even

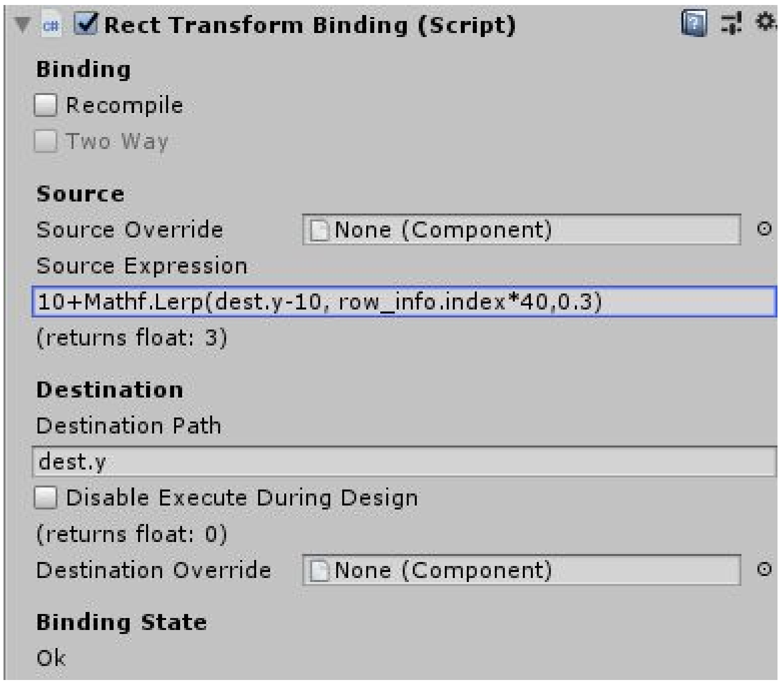
Is true when index is an even number, can be used to create a "striped" look in a table.

first

Is true when the row is the first row in the Repeat Binding.

index

The index/position of the row in the Repeat Binding. Can be used to position Game Object using a Rect Transform Binding, for example: "10+Mathf.Lerp(dest.y-10, row_info.index*40,0.3)";



```
☑ Rect Transform Binding (Script)
Binding
☐ Recompile
☐ Two Way

Source
Source Override        None (Component)
Source Expression
10+Mathf.Lerp(dest.y-10, row_info.index*40,0.3)
(returns float: 3)

Destination
Destination Path
dest.y
☐ Disable Execute During Design
(returns float: 0)
Destination Override   None (Component)

Binding State
Ok
```

last

Is true when the row is the last row in the Repeat Binding.

middle

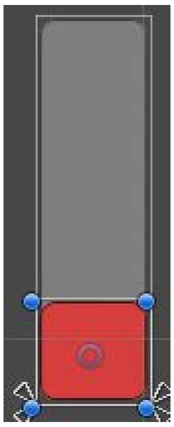
Is true when the row is not either first or last.

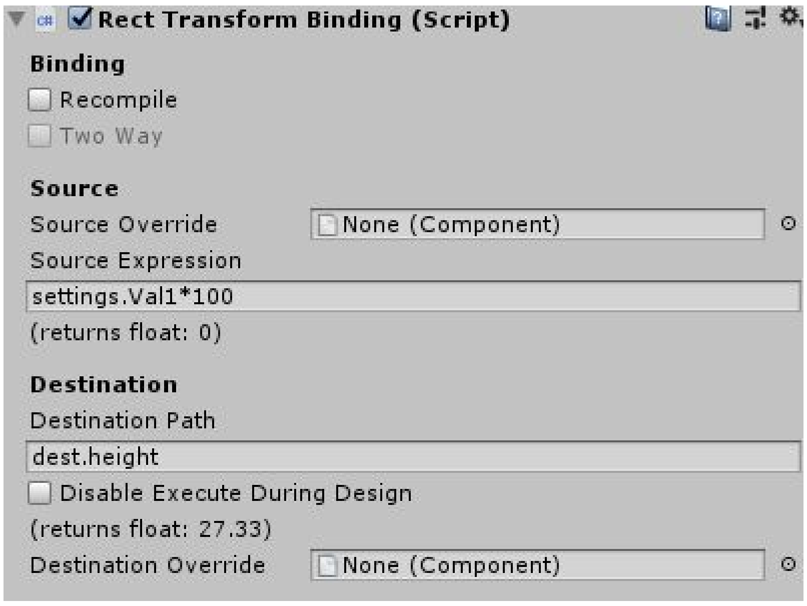
odd

Is true when even isn't.

Rect Transform Binding

Rect Transform Binding is used to bind to RectTransform specific fields, fields that aren't specifically exposed by RectTransforms. Rect Transform Binding adds a Binding Source called "Dest" that contains the fields specified below.





Rect Transform Binding (Script)

Binding
- [ ] Recompile
- [ ] Two Way

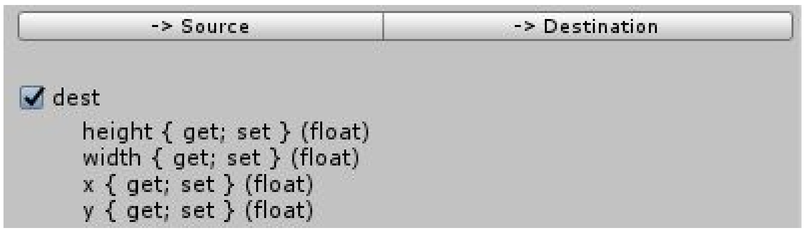
Source
Source Override: None (Component)
Source Expression
```
settings.Val1*100
```
(returns float: 0)

Destination
Destination Path
```
dest.height
```
- [ ] Disable Execute During Design

(returns float: 27.33)
Destination Override: None (Component)

Other Fields

See UI Binding.



```
☑ dest
    height { get; set } (float)
    width { get; set } (float)
    x { get; set } (float)
    y { get; set } (float)
```

height

The height value of the Rect Transform.

width

The width value of the Rect Transform.

x

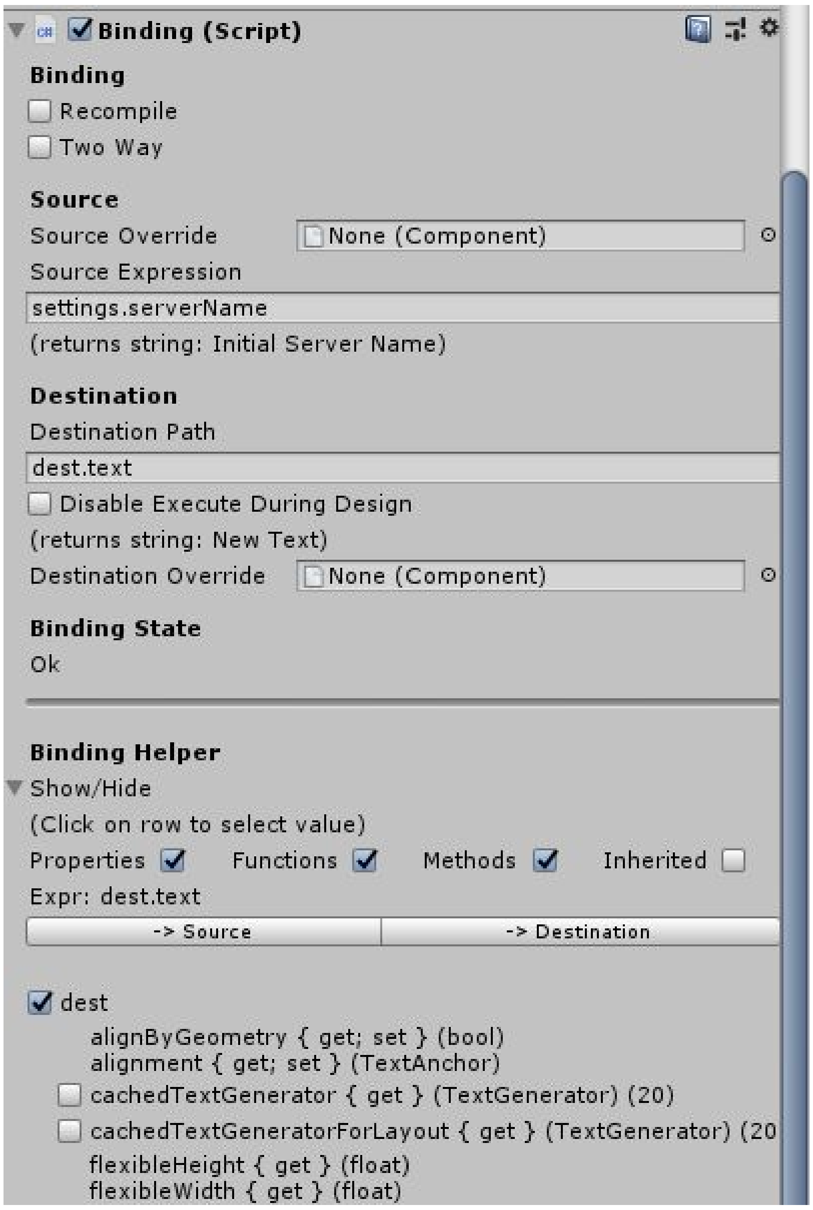
The x position of the Rect Transform - computed from the left edge of the parent Rect Transform.

y

The y position of the Rect Transform - computed from the top edge of the parent Rect Transform.

Binding

A raw binding that doesn’t know anything about the destination you’re binding to - allowing you to bind to any Destination Path you require.



Binding (Script)

Binding

☐ Recompile

☐ Two Way

Source

Source Override: None (Component)

Source Expression

```
settings.serverName
```

(returns string: Initial Server Name)

Destination

Destination Path

```
dest.text
```

☐ Disable Execute During Design

(returns string: New Text)

Destination Override: None (Component)

Binding State

Ok

Binding Helper

Show/Hide

(Click on row to select value)

Properties ✓ Functions ✓ Methods ✓ Inherited ☐

Expr: dest.text

-> Source | -> Destination

✓ dest

```
    alignByGeometry { get; set } (bool)
    alignment { get; set } (TextAnchor)
☐  cachedTextGenerator { get } (TextGenerator) (20)
☐  cachedTextGeneratorForLayout { get } (TextGenerator) (20
    flexibleHeight { get } (float)
    flexibleWidth { get } (float)
```

Destination Path

The UI* components control the Destination Path for you, but on the Binding Component you can set it to anything you like. You can replicate the behaviour of UI Binding using a Binding - but it takes more work. The Destination Path typically takes the form of “dest.text”, but you could bind “settings.playerName” to “settings.otherPlayerName” and the Binding would copy the value of “settings.playerName” to “settings.otherPlayerName” every frame.

Other Fields

See UI Binding.