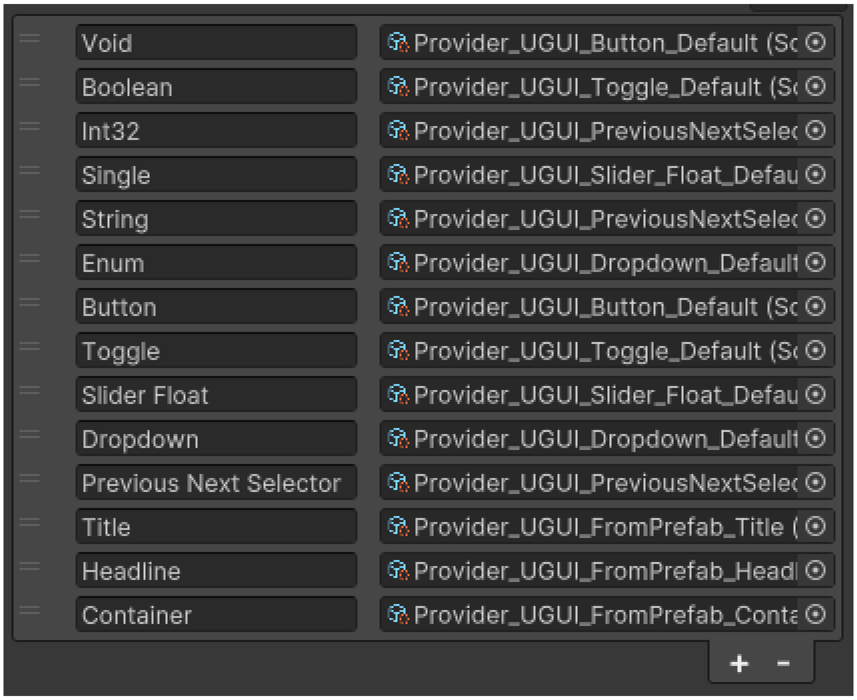
Input Element Providers

Overview

Input Element Provider Collections (UGUI & UI Toolkit) are scriptable objects containing a list of Input Element Provider objects that have a string associated with each provider. You can freely customize the list of providers but in most cases the default variable names can be used and only the providers are replaced to use different visuals.
There is one collection for UGUI and one for UI Toolkit.



Void	Provider_UGUI_Button_Default (Sc
Boolean	Provider_UGUI_Toggle_Default (S
Int32	Provider_UGUI_PreviousNextSelec
Single	Provider_UGUI_Slider_Float_Defau
String	Provider_UGUI_PreviousNextSelec
Enum	Provider_UGUI_Dropdown_Default
Button	Provider_UGUI_Button_Default (Sc
Toggle	Provider_UGUI_Toggle_Default (S
Slider Float	Provider_UGUI_Slider_Float_Defau
Dropdown	Provider_UGUI_Dropdown_Default
Previous Next Selector	Provider_UGUI_PreviousNextSelec
Title	Provider_UGUI_FromPrefab_Title (
Headline	Provider_UGUI_FromPrefab_Headl
Container	Provider_UGUI_FromPrefab_Conta

Creating An Input Element Provider Collection

You can create a new collection from the project window with Right Click > CitrioN > Settings Menu Creator > Provider Collection > …

What Is An Input Element Provider?

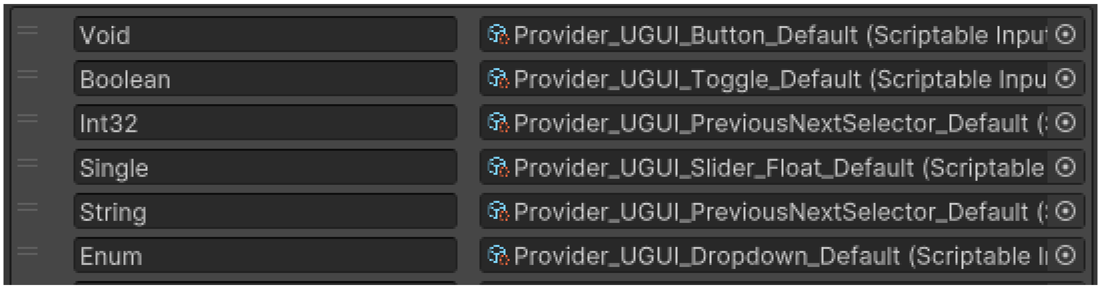
An Input Element Provider specifies how the input element (slider, toggle, etc.) is created and set up in the menu. An input element is for example a button or dropdown. In the case of the dropdown an Input Element Provider for a dropdown needs to be used. There are providers for all base input elements in UGUI and UI Toolkit. Those include providers for a button, toggle, slider, dropdown & previous next selector. An input element doesn't always have to process an input. There can also be providers for elements such as headlines or titles.

How Providers Are Selected

There are different ways for the system to decide which provider is used to create the input element for each individual setting. This is the order in which the system checks for a valid provider:

1. A provider override is specified for a specific setting. This is done in the setting specific options under the Advanced tab on the SettingsCollection. Learn more about this setting option here.

2. A provider exists on the Input Element Providers Collection object that has the type name as its key. In the image below you can see the most common types and their associated Input Element Provider.



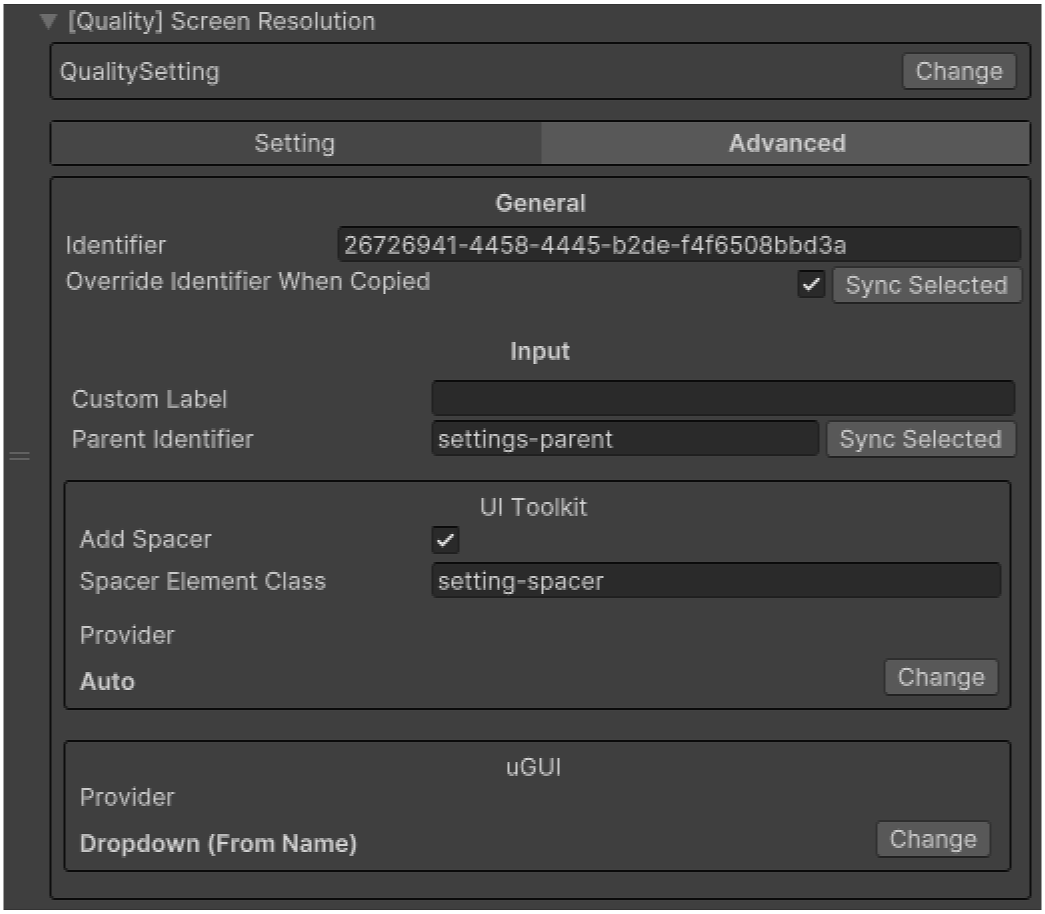
Void	Provider_UGUI_Button_Default (Scriptable Inpu
Boolean	Provider_UGUI_Toggle_Default (Scriptable Inpu
Int32	Provider_UGUI_PreviousNextSelector_Default (
Single	Provider_UGUI_Slider_Float_Default (Scriptable
String	Provider_UGUI_PreviousNextSelector_Default (
Enum	Provider_UGUI_Dropdown_Default (Scriptable I

3. A fallback function

If the fallback fails a warning is printed to the console and no input element will be created or set up.

Specifying a provider for a single setting

On the SettingsCollection in the advanced tab for a setting a provider (UGUI & UI Toolkit) can be specified. By default it is set to auto which uses the type based approach explained above.



▼ [Quality] Screen Resolution

QualitySetting

General

Identifier: 26726941-4458-4445-b2de-f4f6508bbd3a

Override Identifier When Copied ☑

Input

Custom Label:

Parent Identifier: settings-parent

UI Toolkit

Add Spacer ☑

Spacer Element Class: setting-spacer

Provider: **Auto**

uGUI

Provider: **Dropdown (From Name)**

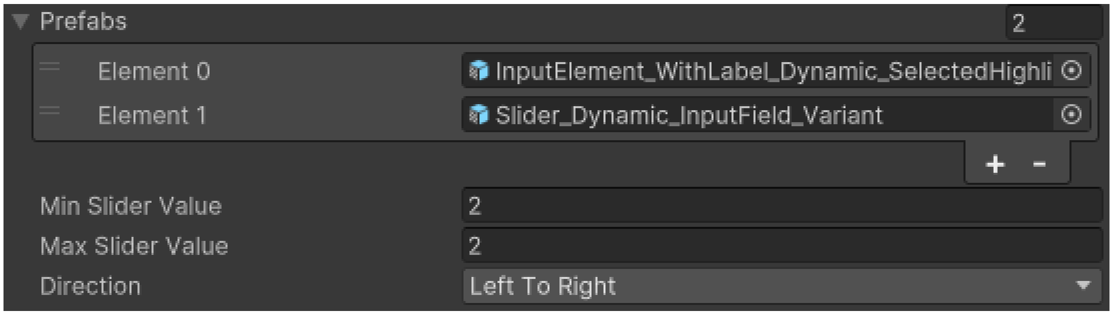
From Name

The ‘From Name’ provider will use the provider in the list that matches the name specified. There are multiple common names already selectable (button, dropdown etc.) or you can specify a custom one.

Auto

The auto option is selected by default which will try and find a provider matching the setting’s input type. If you for example have the ‘Allow HDR’ setting the Boolean provider will be used because the setting uses a boolean variable. This requires a Boolean entry to be specified on the Input Element Providers object and a boolean compatible provider specified to work. The sample Input Element Providers all have this set up.

For reference check out the list of compatible providers by type here.
Some providers have additional options available on the provider object itself like the slider provider here:



Prefabs 2

Element 0 InputElement_WithLabel_Dynamic_SelectedHighli

Element 1 Slider_Dynamic_InputField_Variant

Min Slider Value 2

Max Slider Value 2

Direction Left To Right

The min slider and max slider values will only be used if no variable is specified in the setting specific options. How to use setting options and variables can be found here.

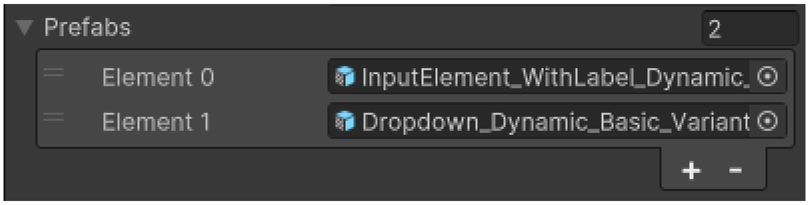
Depending on which UI System (UGUI or UI Toolkit) is used different objects need to be referenced for a provider to function properly. Let’s take a look at those in detail:

UGUI

A UGUI provider typically requires a prefab to be referenced that contains the correct script for the input element. In the case of a dropdown provider a Text Mesh Pro dropdown component needs to be attached to an object somewhere in the prefabs hierarchy.

Using Multiple Prefabs

To avoid prefab nesting multiple prefabs can be used to construct the final input element. In the provided demos most input elements come with a label alongside the input element. A prefab for both the label and the input element (slider, toggle etc) is referenced in the provider.

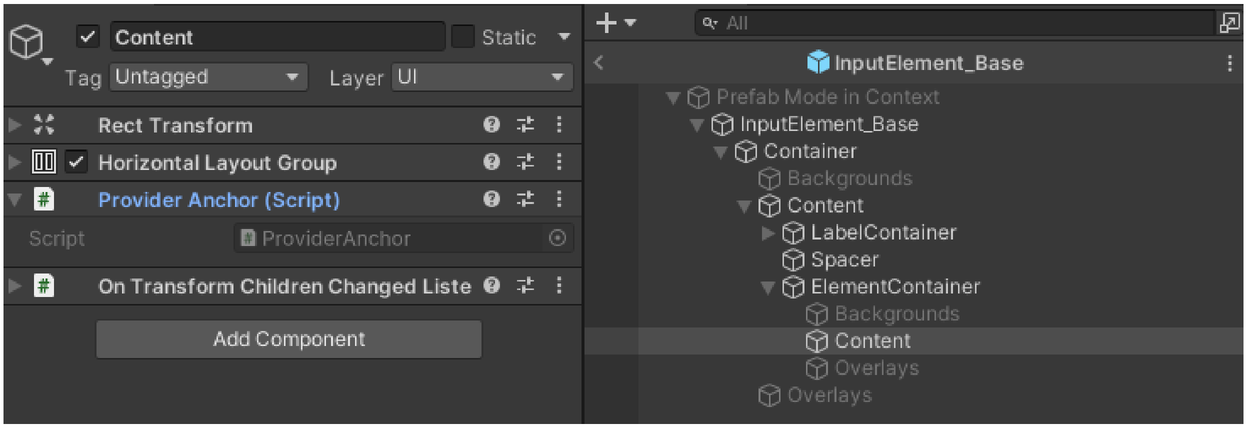


Prefabs 2

Element 0 InputElement_WithLabel_Dynamic_

Element 1 Dropdown_Dynamic_Basic_Variant

This way the same label prefab can be used for the various providers making customization easier. The input element is then created by creating instances of the referenced prefabs. The prefabs are created in the order they appear in the list and then parented to the previous object. To specify where to parent the object the ProviderAnchor script can be attached to any object in the prefabs hierarchy. If no ProviderAnchor is found the prefab will be parented to the previous object's root. Take a look at the InputElement_WithLabel prefabs hierarchy. The GameObject highlighted (Content) has the ProviderAnchor script attached to tell the provider to parent the following prefab to this GameObject. Without this script the following prefab would be attached to the root called InputElement_Base which is not desired in this case.



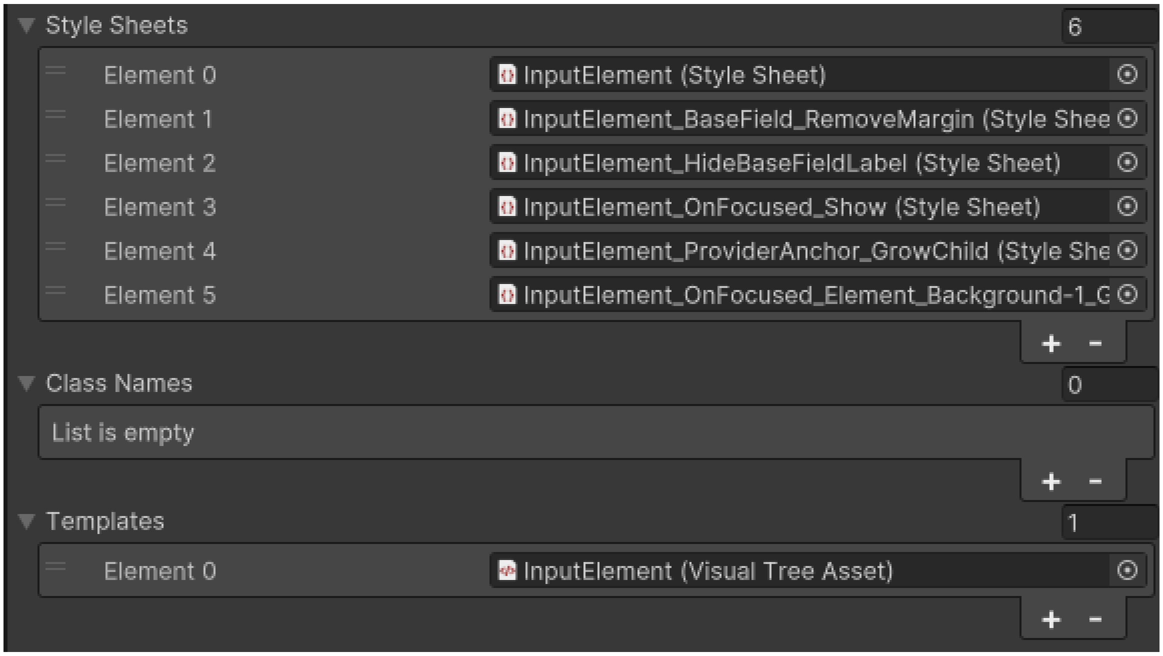
☑ Content □ Static

Tag Untagged Layer UI

Script ProviderAnchor

There are many prefabs for input elements in the project window under **Packages/CitrioN - Settings Menu Creator/Content/Prefabs/InputElements**. There are prefab variants for all the common input elements (toggle, slider, dropdown etc.) which are fully compatible with all the features of the settings menu creator such as support for using style profiles. It is recommended to create prefab variants of those and customize the variants to create your own style of input elements that fits your project's requirements. This is also done if you have your resources generated using the ResourceCreatorProfile. You can of course create and use your own prefabs too.

UI Toolkit



Style Sheets 6

Element 0 InputElement (Style Sheet)

Element 1 InputElement_BaseField_RemoveMargin (Style Shee

Element 2 InputElement_HideBaseFieldLabel (Style Sheet)

Element 3 InputElement_OnFocused_Show (Style Sheet)

Element 4 InputElement_ProviderAnchor_GrowChild (Style She

Element 5 InputElement_OnFocused_Element_Background-1_G

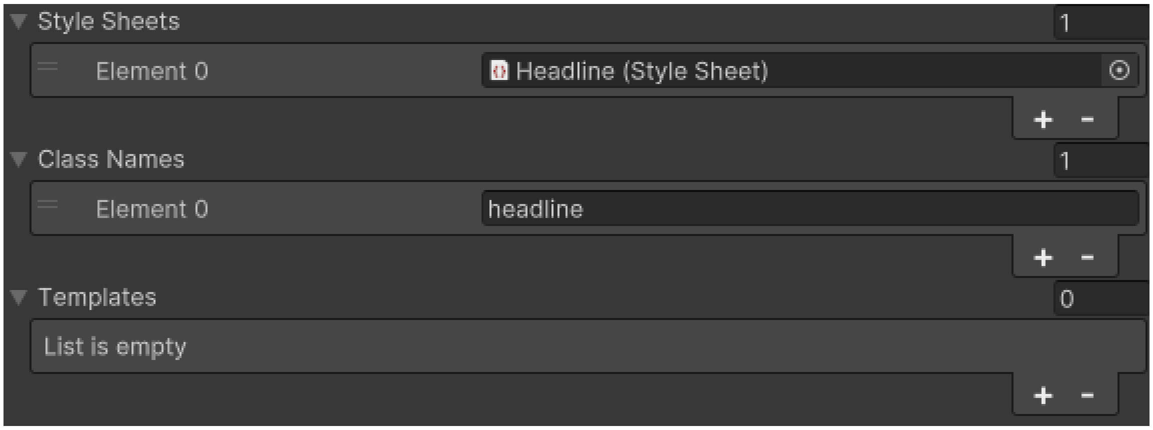
Class Names 0

List is empty

Templates 1

Element 0 InputElement (Visual Tree Asset)

UI Toolkit providers typically allow the specifications of stylesheets, class names and uxml files. The resulting provider will use a similar approach as the one for UGUI. The uxml files will be used to construct the final input element. To correctly parent elements the provider-anchor class can be used. UI Toolkit providers will generate the correct elements if no uxml file is specified. For example the dropdown provider for UI Toolkit will automatically create a dropdown if no dropdown is available in any of the provided uxml files. You would only want to specify a dropdown if you have elements in the dropdown hierarchy that are not on the default dropdown by Unity. Specified class names and style sheets will be added to the resulting input element during generation. You can use class names to make style sheets compatible with your resulting element without having to provide a uxml file. An example is the headline on the default Input Element Provider Collection where no uxml file and only a stylesheet is referenced that is set up to work with the headline class.



Style Sheets 1

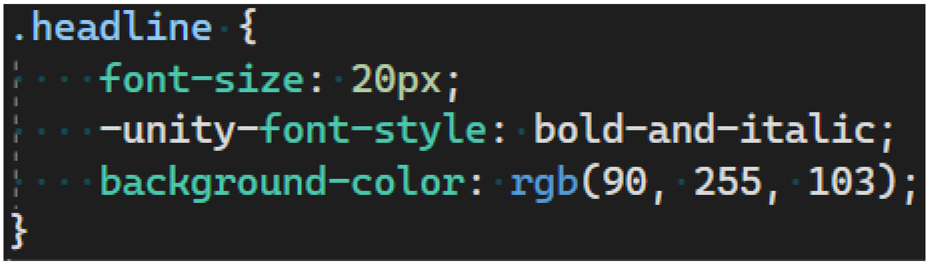
Element 0 Headline (Style Sheet)

Class Names 1

Element 0 headline

Templates 0

List is empty



```
.headline {
    font-size: 20px;
    -unity-font-style: bold-and-italic;
    background-color: rgb(90, 255, 103);
}
```

There are a lot of example uxml files and style sheets provided that you can duplicate and customize to fit your project's needs. You can find them in the project window under Packages/CitrioN - Settings Menu Creator/Content/UI Toolkit/USS or UXML/Elements or InputElement. The ResourceCreatorProfile also uses this file duplication approach.

You can then use the preset system to quickly set up your references or use them as a foundation to customize it to your needs.

Special UI Elements

There are a couple elements that serve a special purpose in a settings menu. By default those are headlines, titles and groups. They are UI elements with the purpose of styling/organizing a menu and don't have any particular functionality connected. To add such a custom UI element it only needs an entry in the collection and a setting (preferably empty setting) that specifies the use of that UI element. Check out the various demo collections to see how that looks.

Creating a Scriptable Input Element Provider

In the project window Right Click > CitrioN > Settings Menu Creator > Input Element Provider > …

Writing a custom provider

UGUI

To write your own UGUI provider you need to derive your class from either the ScriptableInputElementProvider_UGUI or the InputElementProvider_UGUI. The scriptable one is used on the Input Element Provider Collection and the other one is used in the setting specific provider options on the settings collection. Typically the (Scriptable)InputElementProvider_UGUI_FromPrefab class is used which extends the base class.

You need to at least override the following methods:

public abstract Type GetInputFieldParameterType(SettingsCollection settings);

Specifies the parameter type for the input element.
Example: The Toggle providers have the boolean type specified.

public abstract Type GetInputFieldType(SettingsCollection settings);

Specifies the Input Element Type.
Example: The dropdown providers have the dropdown component type specified.

When using the base class (only recommended in rare instances and when you know what you are doing):
public abstract RectTransform GetInputElement(string settingIdentifier, SettingsCollection settings);

Functionality on how to provide/create the input element. Check the available providers code for reference such as the ScriptableInputElementProvider_UGUI_FromPrefab_Button.cs

UI Toolkit

To create a new UI Toolkit provider you need to derive your class from either the ScriptableInputElementProvider_UIT or the InputElementProvider_UIT. The scriptable one is used on the Input Element Provider Collection and the other one is used in the setting specific provider options on the settings collection. Typically the (Scriptable)InputElementProvider_UIT_FromUXML class is used which extends the base class.

You need to at least override the following methods:

public abstract Type GetInputFieldParameterType(SettingsCollection settings);

Specifies the parameter type for the input element.
Example: The Toggle providers have the boolean type specified.

public abstract Type GetInputFieldType(SettingsCollection settings);

Specifies the Input Element Type.
Example: The dropdown providers have the dropdown element type specified.

When using the base class (only recommended in rare instances and when you know what you are doing):
public abstract VisualElement GetInputElement(string settingIdentifier, SettingsCollection settings);

Functionality on how to provide/create the input element. Check the available providers code for reference like the ScriptableInputElementProvider_UIT_FromUXML_Button.cs