

Sparrow Interface Field Documentation

Description

Summary

Sparrow is a lightweight system which allows you to show and use references to C# interfaces in Unity Editor via Drag-and-Drop.

Description

Sparrow Interface Field allows you to use Drag-and-Drop approach for C# interface field in Unity Inspector. It provides the possibility to use interfaces in code without more complex dependency injection principles. Now it is possible to use the interface fields references in a classic manner with the Drag-and-Drop like reference to an average class.

Besides this, Sparrow provides a mechanism of fast and easy initialization for every interface field. Beside a classic Drag-and-Drop, there are three initialization modes: Auto Initialization, Search in child components, and Search in scene components.

[Online Documentation](#)

Features:

- Classic Drag-and-Drop approach for C# interface initialization in Unity Inspector;
- Fast search target component with auto initialization mode in child components;
- Easily search target in child components with fast initialization mode;
- Easily search target in all scene components with fast initialization mode;
- Support arrays and container classes;
- Support Scriptable Objects;
- Support two work modes: via attributes and via special wrapper class.

Technical details

Support all platforms.

Support Unity 2019.4 and higher.

Read Documentation to configure the system for work with Unity 2020 and prior versions.

Using

Creation

First of all add Sparrow namespace:

```
CorD.SparrowInterfaceField
```

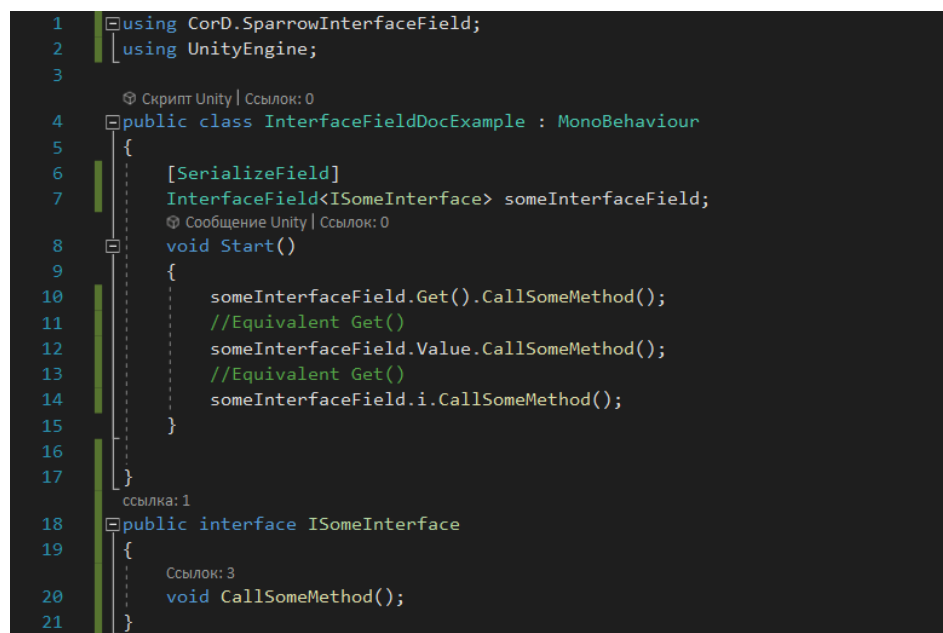
Next, to use Sparrow Interface Field, you need to declare a variable which will store reference to a target class. You can do this in two ways:

Way 1: Using Wrapper class (Recommended)

You need declare variable like this:

```
[SerializeField]
InterfaceField<T> fieldName;

//Where T is target interface
//fieldName - name of an Interface Field variable
```

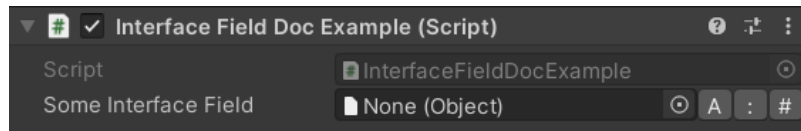


```
1 using CorD.SparrowInterfaceField;
2 using UnityEngine;
3
4 public class InterfaceFieldDocExample : MonoBehaviour
5 {
6     [SerializeField]
7     InterfaceField<ISomeInterface> someInterfaceField;
8     void Start()
9     {
10         someInterfaceField.Get().CallSomeMethod();
11         //Equivalent Get()
12         someInterfaceField.Value.CallSomeMethod();
13         //Equivalent Get()
14         someInterfaceField.i.CallSomeMethod();
15     }
16 }
17
18 public interface ISomeInterface
19 {
20     void CallSomeMethod();
21 }
```

Example of class which use Interface Field Wrapper

Interface "ISomeInterface" is an example of a simple interface.

After declaring a variable, you can access the interface using the Get() method and use the target interface instance in your code. For convenience, the properties Value and i (i - interface) are introduced, which are equivalent to the Get() method.



Example Sparrow Interface Field displaying in the Unity Inspector

Way 2: Using attribute on Object field

You need declare variable like this:

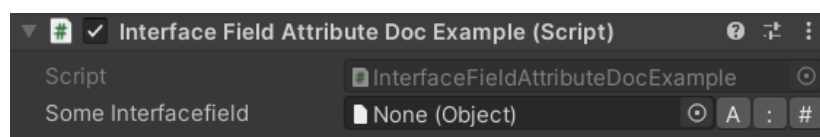
```
[SerializeField, InterfaceField(typeof(T))]
Object someInterfacefield;
```

//Where T is target interface

//someInterfacefield - name of an Interface Field base variable

```
1  using CorD.SparrowInterfaceField;
2  using UnityEngine;
3
4  public class InterfaceFieldAttributeDocExample : MonoBehaviour
5  {
6      [SerializeField, InterfaceField(typeof(ISomeInterface))]
7      Object someInterfacefield;
8
9      void Start()
10     {
11         ((ISomeInterface)someInterfacefield).CallSomeMethod();
12     }
13 }
```

Example of class which use Interface Field Attribute



Example Sparrow Interface Field Attribute displaying in the Unity Inspector

In general, these methods give the same results. However, in most cases a wrapper class may be preferable, as it provides get and initialization methods.

Using of the attribute is reasonable if you do not want to have explicit dependencies on Sparrow.

Using readonly property for convenience

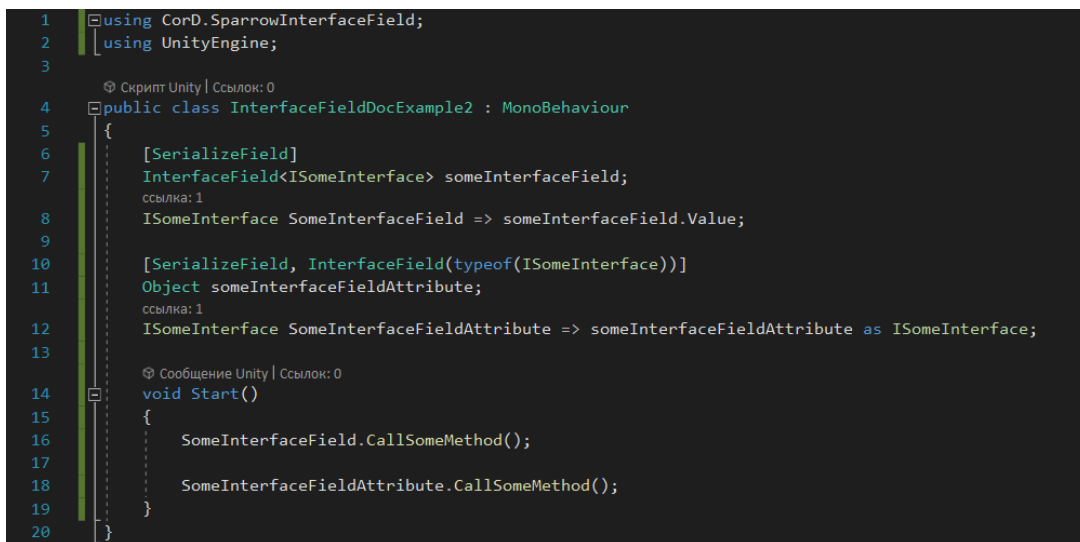
For convenience, you can declare properties for reading, which converts the type to the desired one:

1. For Wrapper:

```
[SerializeField]
InterfaceField<ISomeInterface> someInterfaceField;
ISomeInterface SomeInterfaceField => someInterfaceField.Value;
```

2. For Attribute:

```
[SerializeField, InterfaceField(typeof(ISomeInterface))]
Object someInterfaceFieldAttribute;
ISomeInterface SomeInterfaceFieldAttribute =>
someInterfaceFieldAttribute as ISomeInterface;
```



```
1 using CorD.SparrowInterfaceField;
2 using UnityEngine;
3
4 public class InterfaceFieldDocExample2 : MonoBehaviour
5 {
6     [SerializeField]
7     InterfaceField<ISomeInterface> someInterfaceField;
8     ISomeInterface SomeInterfaceField => someInterfaceField.Value;
9
10    [SerializeField, InterfaceField(typeof(ISomeInterface))]
11    Object someInterfaceFieldAttribute;
12    ISomeInterface SomeInterfaceFieldAttribute => someInterfaceFieldAttribute as ISomeInterface;
13
14    void Start()
15    {
16        SomeInterfaceField.CallSomeMethod();
17        SomeInterfaceFieldAttribute.CallSomeMethod();
18    }
19 }
20
```

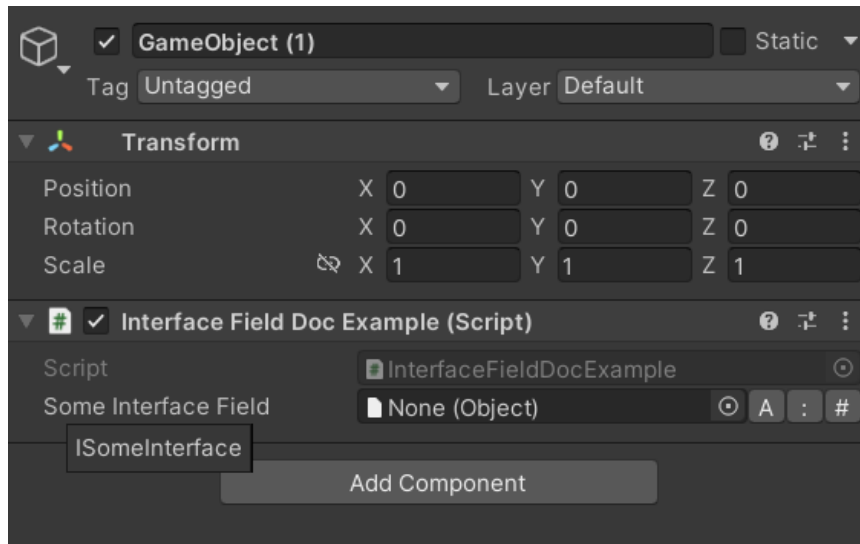
Example of using readonly property with Interface Field

This is especially recommended when using the Attribute variant, as unlike the Wrapper, there are no getter convenience methods.

Initialization

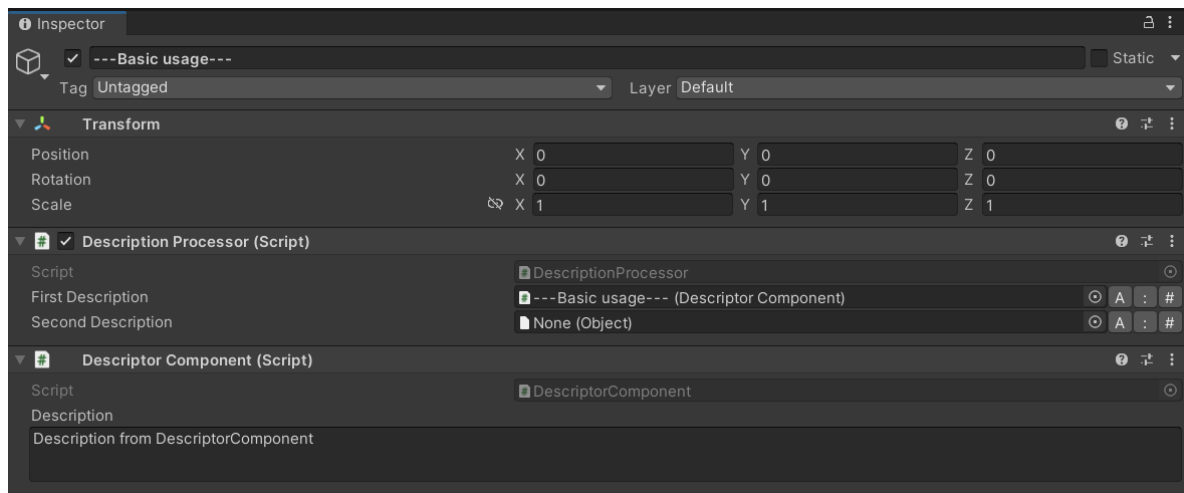
Editor

After declaring a variable, regardless of the method, you will see the standard drag and drop field in the unit editor, which is complemented by three buttons. When you hover over the field name, the required interface is displayed.



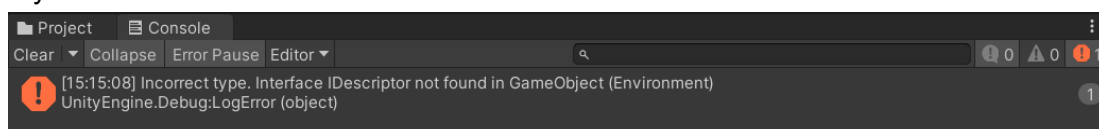
Example of displaying required interface

You can use the classic initialization mechanism by dragging the target class right in the editor. If the class is correct, it will be saved in this field, the dependency is set. Source code of Description Processor and Descriptor Component can be found in the Examples folder.



Example of correct initialization

Otherwise, the Interface Field value will be set to None (null) and an error message will be displayed in the console.



Example of error message

Important!

Only MonoBehaviour or ScriptableObject components can be used to initialize an interface field.

Buttons

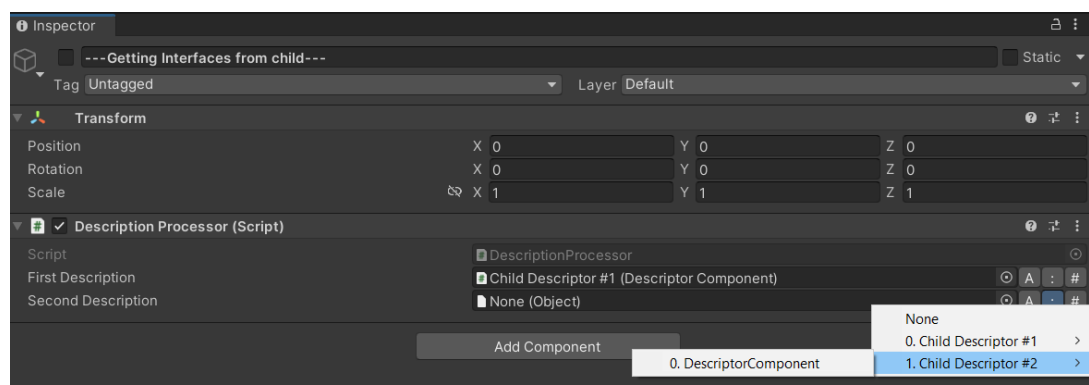
Buttons are needed to speed up work when initializing fields, because Drag And Drop is not always convenient.

Button “A” - Automatic initialization

A - Automatic initialization. When you click on this button, a suitable component is searched for on the current object, if not found, in child gameobjects. If a suitable component is found, it is assigned to this field. Otherwise, an error message is displayed and this field is set to None (null).

Button “:” - Search in children

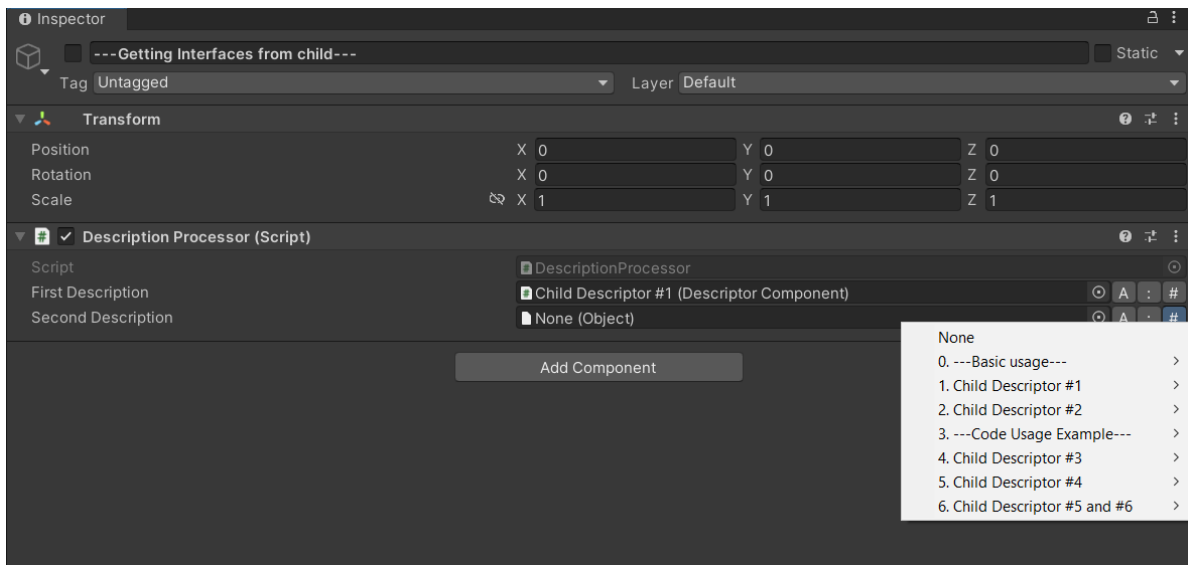
Search for the target component on the given object and its children. Serves to select the required component if the target interface is implemented by several components attached to this GO. When the button is clicked, a list of all components located on the given object and in its children that implement the target interface appears. There is also a None item, which is used to reset the current value.



Displaying dropdown list of suitable components on this game object and its children

Button “#” - Search in scene

Searches for the target component in the current scene. It is an analogue of the classic Object Picker. When you click on the button, a list of all the components on the stage that implement the target interface appears. There is also a None item, which is used to reset the current value.



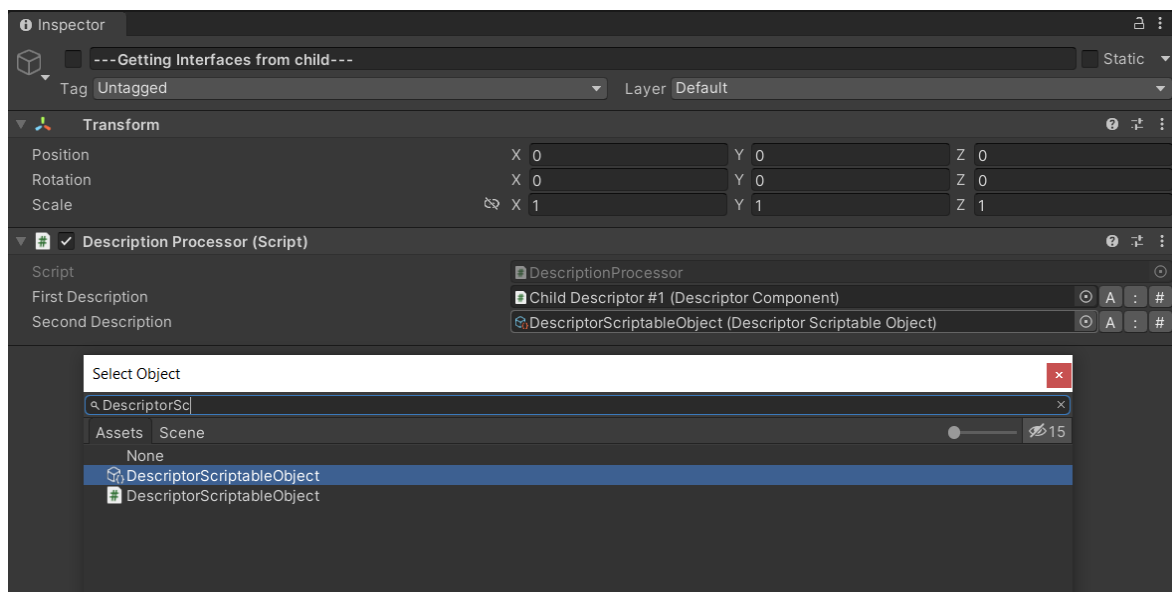
Displaying dropdown list of suitable components from current scene

Important!

Buttons : and # can be used to conveniently reset the current value set in the field.

Object Picker

You can also use a default object picker. However, due to the peculiarities of the system operation, it may be advisable to use the Object Picker only for assigning ScriptableObjects whose names you know. This is due to the fact that the object picker will display all available objects on the stage and in the project folder, regardless of whether they fit or not.



Example of usage Object Picker for assigning ScriptableObject

Code

Wrapper

This section applies to the Wrapper Interface Field.

When working with an interface field in code, you can assign values to the field using the Set() method.

```
1 using CorD.SparrowInterfaceField;
2 using CorD.SparrowInterfaceField.Test;
3 using UnityEngine;
4
5 Скрипт Unity | Ссылки: 0
6 public class InterfaceFieldDocExample3 : MonoBehaviour
7 {
8     [SerializeField]
9     InterfaceField<IDescriptor> interfaceField;
10     Сообщение Unity | Ссылки: 0
11     void Start()
12     {
13         IDescriptor descriptor = new DescriptorNonMonoBehaviour("NONMB");
14         interfaceField.Set(descriptor);
15         Debug.Log(interfaceField.i.GetDescription());
16     }
17 }
```

Example of assigning target interface instance by code

```
namespace CorD.SparrowInterfaceField.Test
{
    /// <summary>
    /// Simple example NonMonoBehavior realisation of IDescription interface
    /// </summary>
    Ссылки: 3
    public class DescriptorNonMonoBehaviour : IDescriptor
    {
        string description;
        Ссылки: 2
        public DescriptorNonMonoBehaviour(string description)
        {
            this.description = description;
        }
        Ссылки: 8
        public string GetDescription()
        {
            return description;
        }
    }
}
```

Simple non MonoBehaviour class which implements target interface

Also a TryInitInterface() method can be used. It tries to find the target component in the given game object. In case a component is not found, it tries to find it in children of the given game object. Can be handy for quick code initialization instead of using Drag And Drop or GetComponent().


```

1  using CorD.SparrowInterfaceField;
2  using CorD.SparrowInterfaceField.Test;
3  using UnityEngine;
4
5  public class InterfaceFieldDocExample4 : MonoBehaviour
6  {
7      [SerializeField]
8      InterfaceField<IDescriptor> interfaceField;
9      void Start()
10     {
11         //Search target interface in this gameObject and in child gameObjects
12         interfaceField.TryInitInterface(this);
13     }
14 }
15

```

Example of using TryInitInterface method

Attribute

You can set the Object field to the target component from code, if necessary, using an assignment. However, this is not convenient, so in such situations it is advisable to use the wrapper.

Restrictions

Type restriction

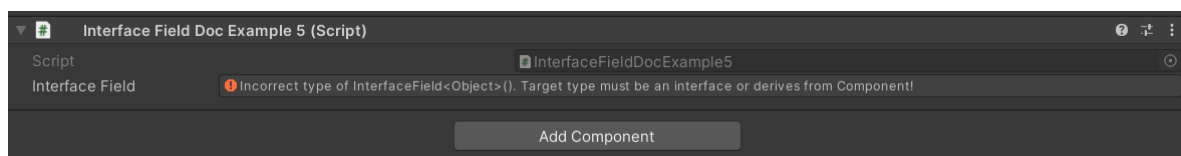
When creating an interface field, only interfaces or MonoBehaviour classes can be used as a type. If you try to set a different type, an error message will be displayed.

```

1  using CorD.SparrowInterfaceField;
2  using System.Collections;
3  using System.Collections.Generic;
4  using UnityEngine;
5
6  public class InterfaceFieldDocExample5 : MonoBehaviour
7  {
8      [SerializeField]
9      InterfaceField<Object> interfaceField;
10 }
11

```

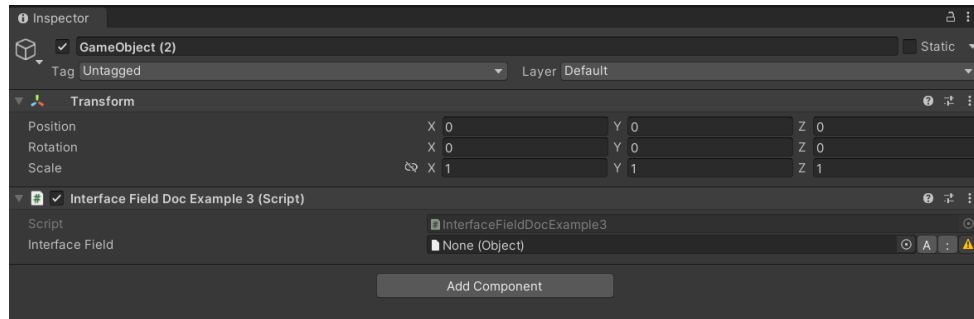
Example of incorrect type initialization



Example of incorrect type initialization

Prefab workflow

Sparrow Interface Field fully supports working with prefabs. When switching to the prefab editing mode, the operation of the “Search in current scene” button is blocked, since the prefab cannot refer to an object in a particular scene.



Search in current scene button is blocked

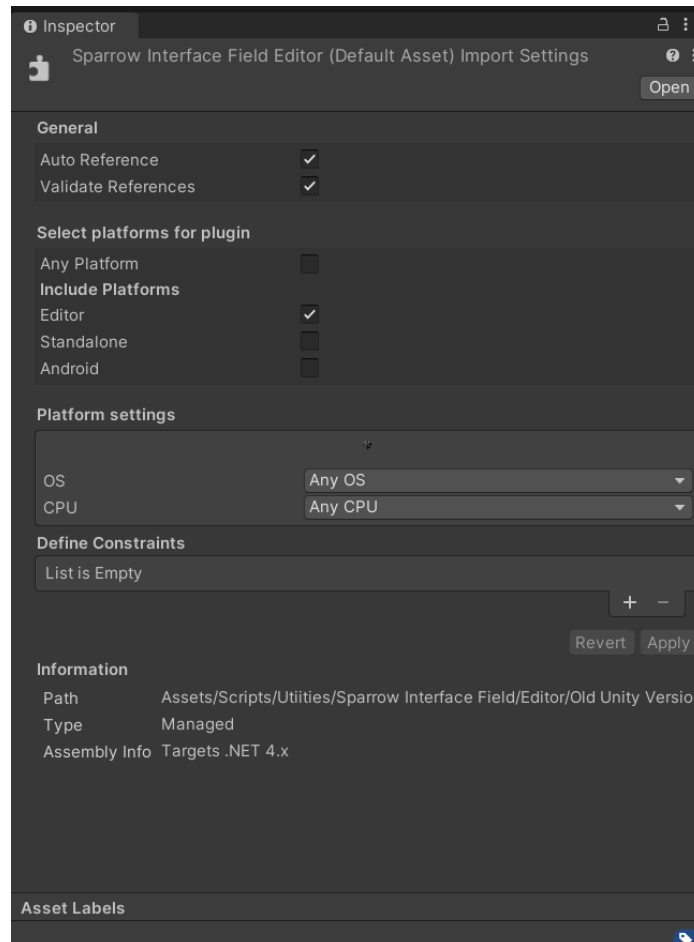
Important!

When working with prefabs, to avoid unintended behavior, it is necessary to switch to editing the prefab in isolation mode. When switching to the prefab context editing mode, if the prefab instance was changed, due to the peculiarities of Unity, the value of the interface field from the instance, and not from the prefab, will be displayed, which may lead to incorrect saving.

Run on Unity versions below 2021

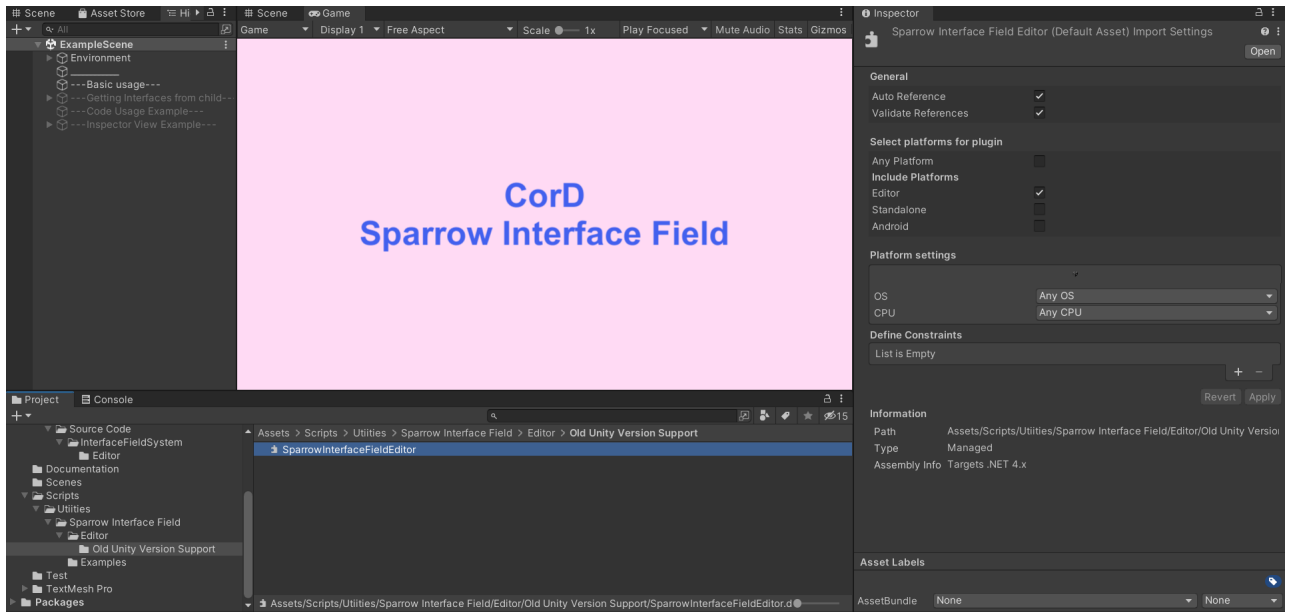
Important!

To run Sparrow Interface Field on Unity prior to 2021, you must use the DLL from the Old Unity Version Support folder. This folder contains a DLL compiled for use with older versions of Unity. To start using it, you need to set the following flags: Auto Reference, Validate References, Editor. In turn, on a DLL compiled for new versions of Unity, you need to remove these flags or completely delete this DLL.

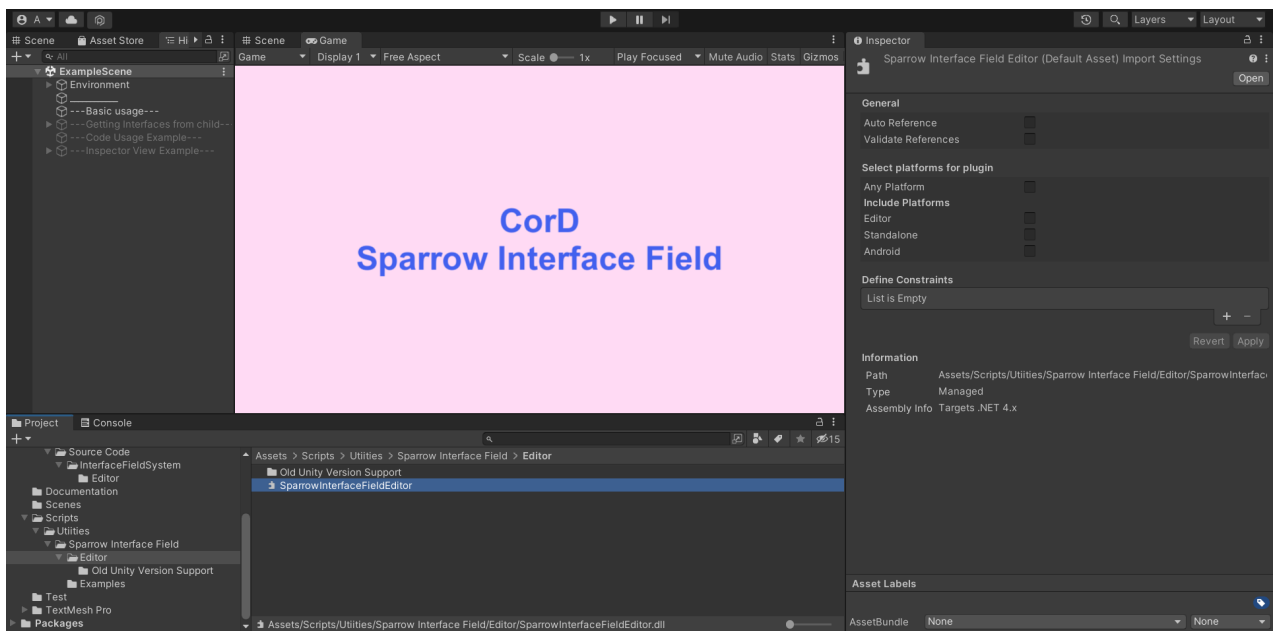


These flags must be set to use the DLL

On older versions of Unity, there are no methods for determining the stage of work with the prefab. Therefore, when using a DLL for older versions, the component search button in the current scene will not be blocked.



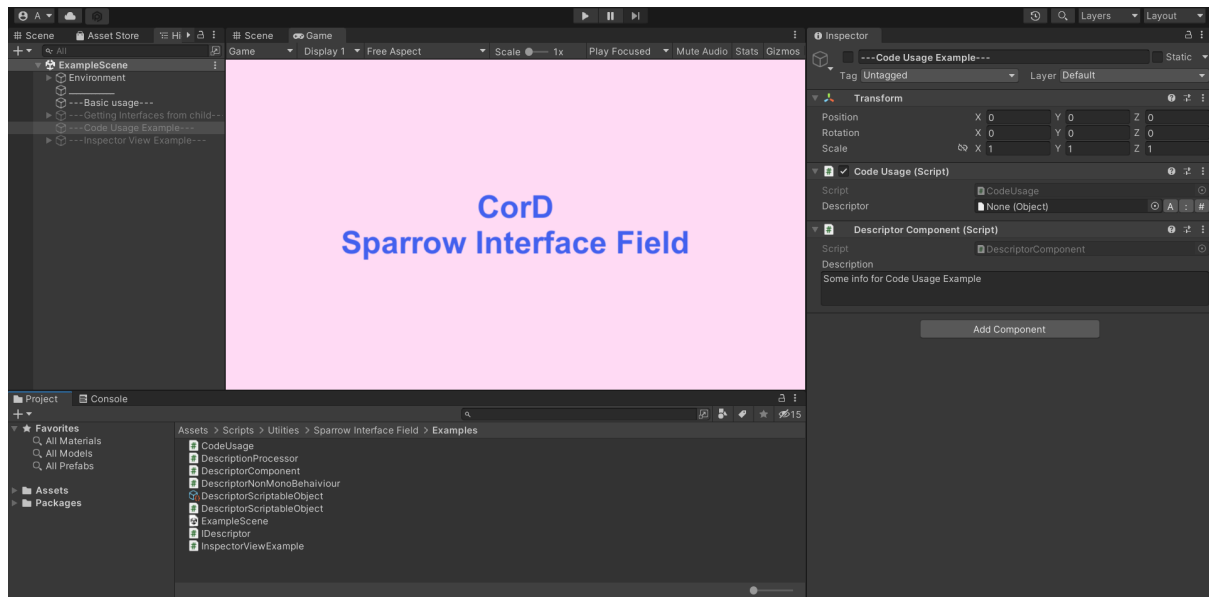
DLL activation to work with older versions of Unity



DLL activation to work with older versions of Unity

Example Scene

In the Examples folder you can find a scene with simple sample scripts that show examples of using the system.



Folder with code examples

Contacts

Thanks for purchasing Sparrow Interface Field!

If you have any questions or suggestions for improvement Sparrow feel free to contact us at coredevelopmentstudio@gmail.com.

Any feedback you can give us on [Sparrow Interface Field](#) at Asset Store would be highly appreciated!

Have a nice day!