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.

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
```

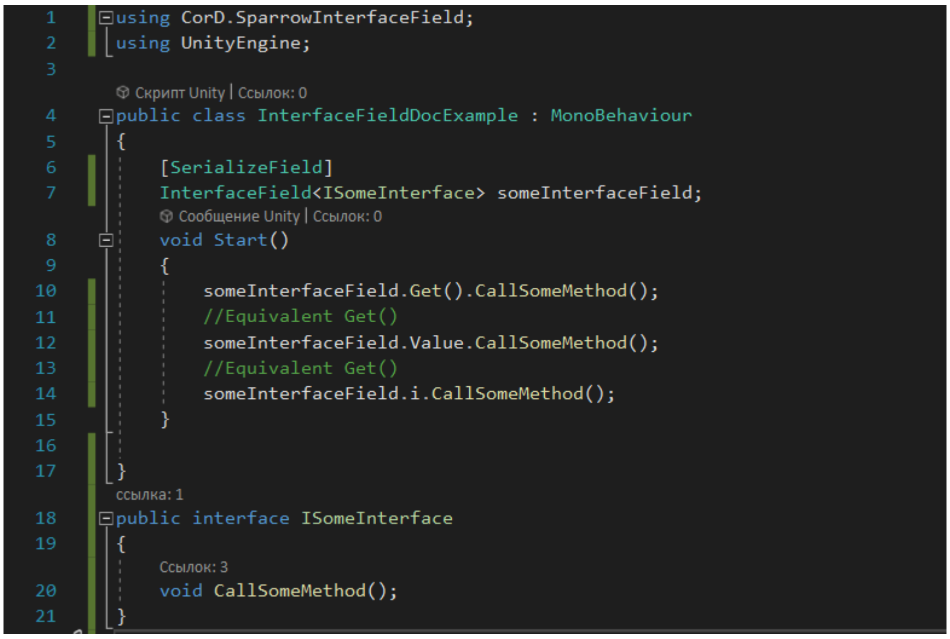
Next, to use Sparrow Interface Field, you need to declare a variable which will store reference to a target class. You can do this in two ways:

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
```



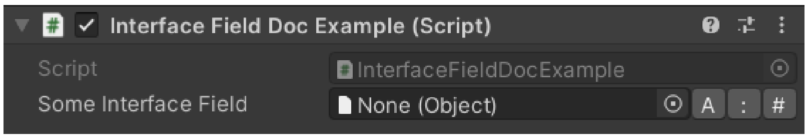
```
using CorD.SparrowInterfaceField;
using UnityEngine;

    Скрипт Unity | Ссылок: 0
public class InterfaceFieldDocExample : MonoBehaviour
{
    [SerializeField]
    InterfaceField<ISomeInterface> someInterfaceField;
    Сообщение Unity | Ссылок: 0
    void Start()
    {
        someInterfaceField.Get().CallSomeMethod();
        //Equivalent Get()
        someInterfaceField.Value.CallSomeMethod();
        //Equivalent Get()
        someInterfaceField.i.CallSomeMethod();
    }

}
    ссылка: 1
public interface ISomeInterface
{
    Ссылок: 3
    void CallSomeMethod();
}
```

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** - **i**nterface) are introduced, which are equivalent to the Get() method.



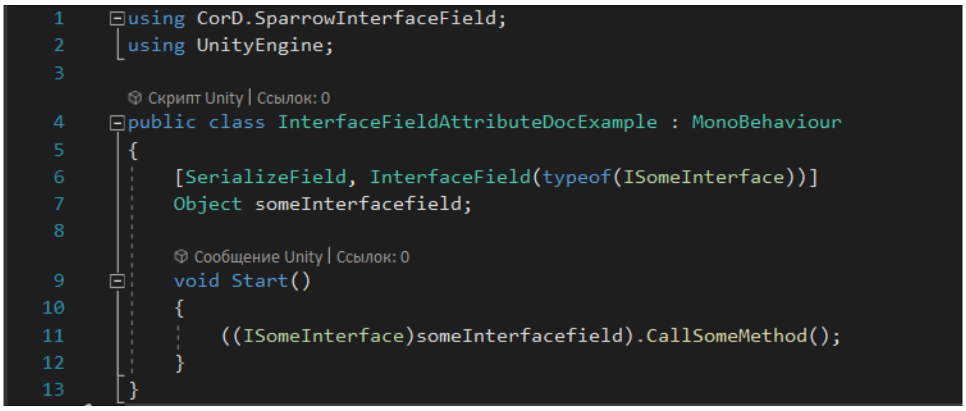
Example Sparrow Interface Field displaying in the Unity Inspector

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
```

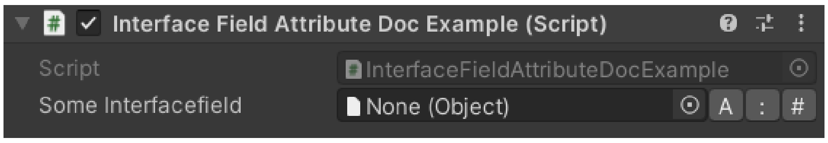


```
using CorD.SparrowInterfaceField;
using UnityEngine;

public class InterfaceFieldAttributeDocExample : MonoBehaviour
{
    [SerializeField, InterfaceField(typeof(ISomeInterface))]
    Object someInterfacefield;

    void Start()
    {
        ((ISomeInterface)someInterfacefield).CallSomeMethod();
    }
}
```

Example of class which use Interface Field Attribute


Interface Field Attribute Doc Example (Script)
Script
InterfaceFieldAttributeDocExample
Some Interfacefield
None (Object)
A
:
#

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.

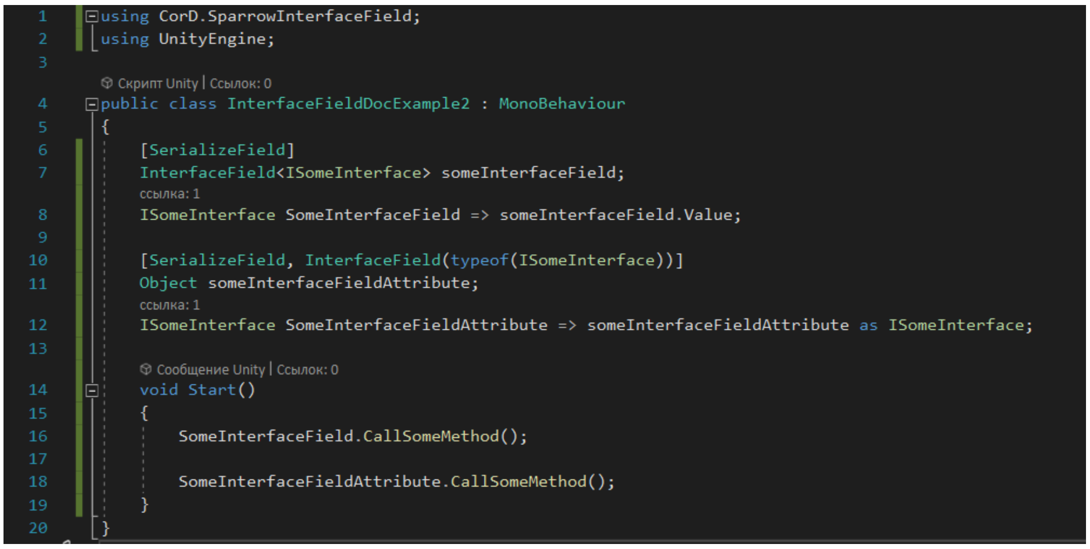
Using of the attribute is reasonable if you do not want to have explicit dependencies on Sparrow.

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;
```



```
using CorD.SparrowInterfaceField;
using UnityEngine;

Скрипт Unity | Ссылок: 0
public class InterfaceFieldDocExample2 : MonoBehaviour
{
    [SerializeField]
    InterfaceField<ISomeInterface> someInterfaceField;
    ссылка: 1
    ISomeInterface SomeInterfaceField => someInterfaceField.Value;

    [SerializeField, InterfaceField(typeof(ISomeInterface))]
    Object someInterfaceFieldAttribute;
    ссылка: 1
    ISomeInterface SomeInterfaceFieldAttribute => someInterfaceFieldAttribute as ISomeInterface;

    Сообщение Unity | Ссылок: 0
    void Start()
    {
        SomeInterfaceField.CallSomeMethod();

        SomeInterfaceFieldAttribute.CallSomeMethod();
    }
}
```

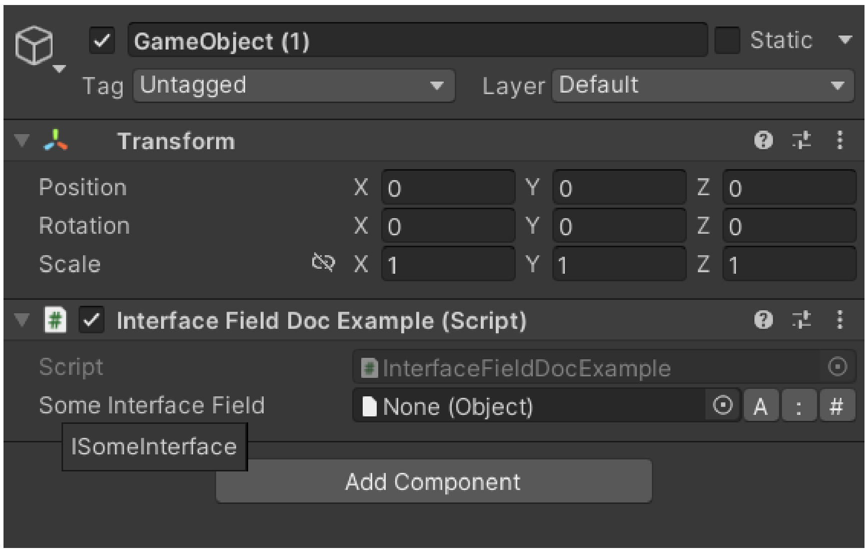
Example of using readonly property with Interface Field

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

Initialization

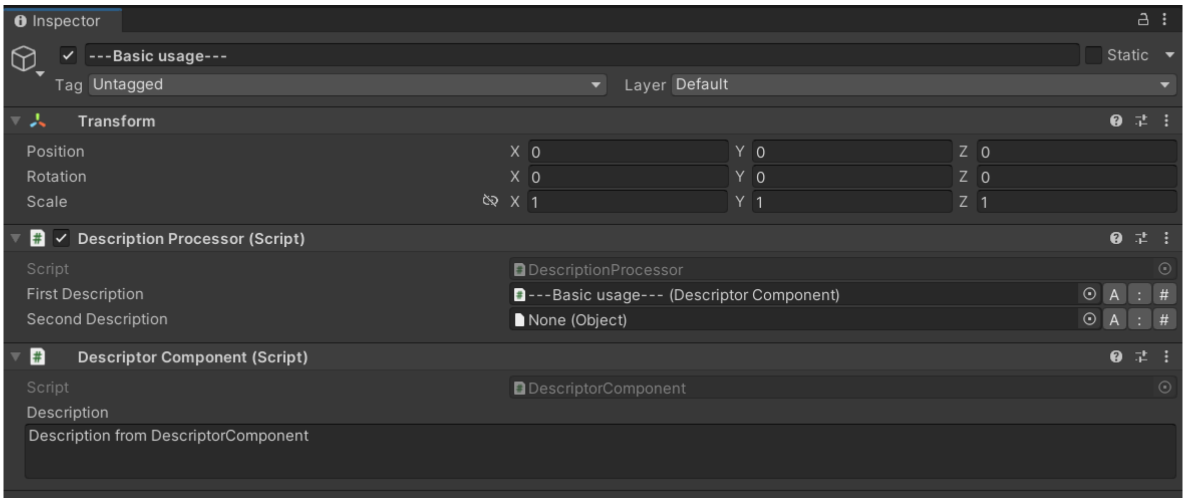
Editor

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.


GameObject (1)
Static
Tag Untagged
Layer Default
Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Interface Field Doc Example (Script)
Script InterfaceFieldDocExample
Some Interface Field None (Object)
ISomeInterface

Example of displaying required interface

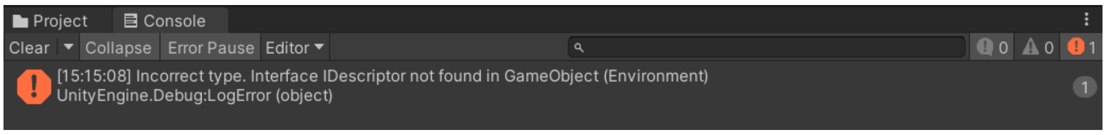
You can use the classic initialization mechanism by dragging the target class right in the editor. If the class is correct, it will be saved in this field, the dependency is set. Source code of Description Processor and Descriptor Component can be found in the Examples folder.



---Basic usage---
Static
Tag Untagged
Layer Default
Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Description Processor (Script)
Script DescriptionProcessor
First Description ---Basic usage--- (Descriptor Component)
Second Description None (Object)
Descriptor Component (Script)
Script DescriptorComponent
Description
Description from DescriptorComponent

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.



```
[15:15:08] Incorrect type. Interface IDescriptor not found in GameObject (Environment)
UnityEngine.Debug:LogError (object)
```

Example of error message

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

Buttons

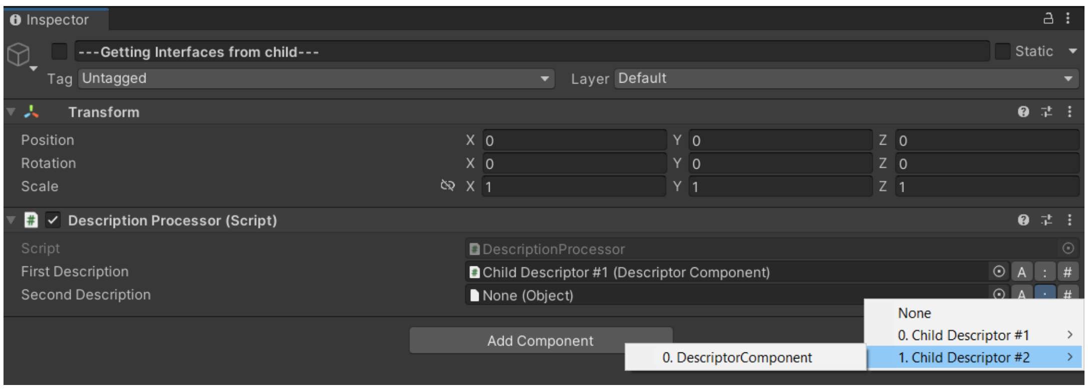
Buttons are needed to speed up work when initializing fields, because Drag And Drop is not always convenient.

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

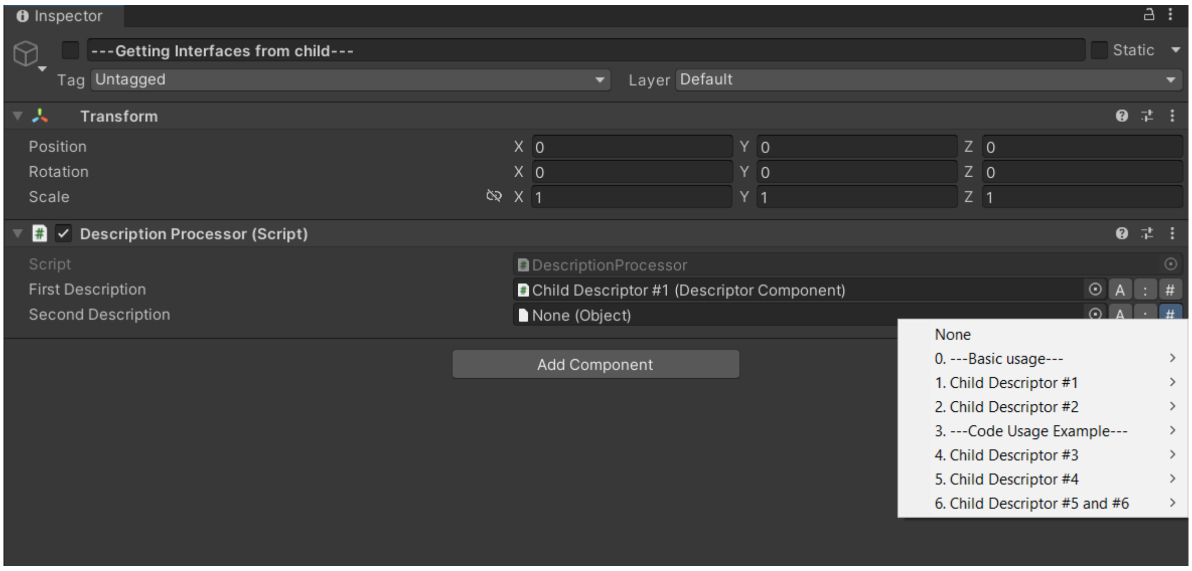
Search for the target component on the given object and its children. Serves to select the required component if the target interface is implemented by several components attached to this GO. When the button is clicked, a list of all components located on the given object and in its children that implement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


---Getting Interfaces from child---
Tag Untagged
Layer Default
Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Description Processor (Script)
Script DescriptionProcessor
First Description Child Descriptor #1 (Descriptor Component)
Second Description None (Object)

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.


Inspector

---Getting Interfaces from child---

Static

Tag Untagged

Layer Default

Transform

Position X 0 Y 0 Z 0

Rotation X 0 Y 0 Z 0

Scale X 1 Y 1 Z 1

Description Processor (Script)

Script DescriptionProcessor

First Description Child Descriptor #1 (Descriptor Component)

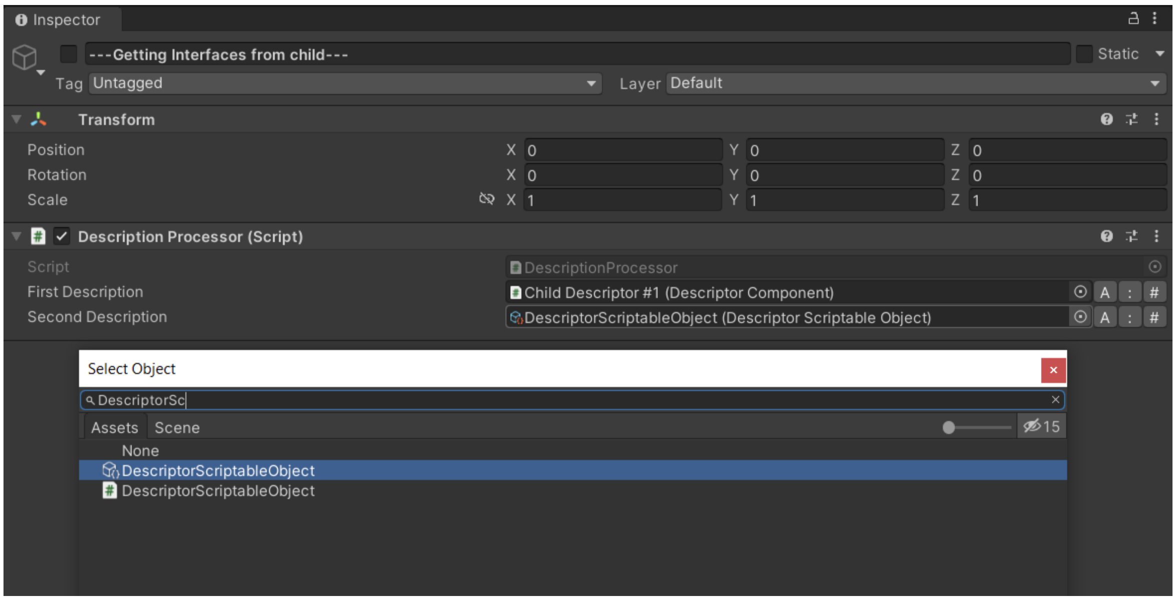
Second Description None (Object)

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.


Inspector

---Getting Interfaces from child---

Static

Tag Untagged

Layer Default

Transform

Position X 0 Y 0 Z 0

Rotation X 0 Y 0 Z 0

Scale X 1 Y 1 Z 1

Description Processor (Script)

Script DescriptionProcessor

First Description Child Descriptor #1 (Descriptor Component)

Second Description DescriptorScriptableObject (Descriptor Scriptable Object)

Select Object

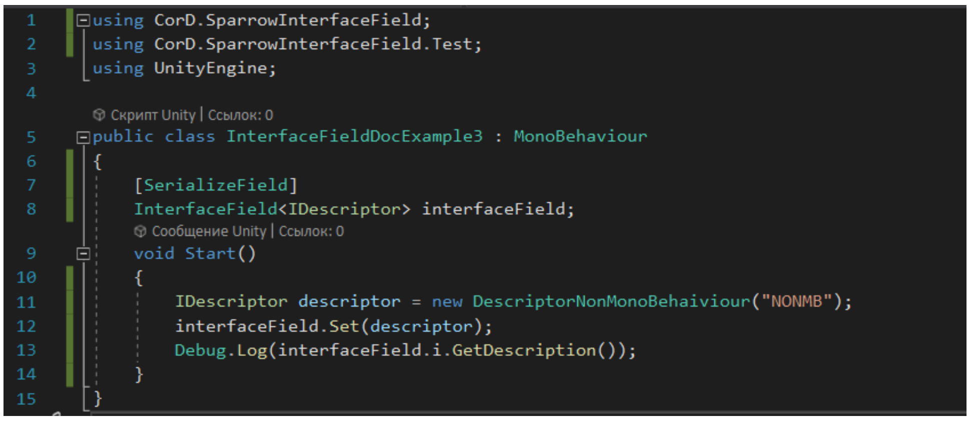
DescriptorSc

Example of usage Object Picker for assigning ScriptableObject

Code

Wrapper

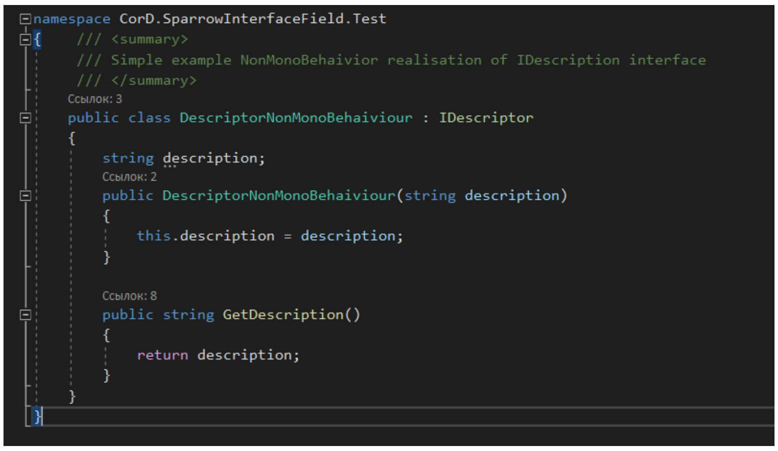
This section applies to the Wrapper Interface Field.
When working with an interface field in code, you can assign values to the field using the Set() method.



```
using CorD.SparrowInterfaceField;
using CorD.SparrowInterfaceField.Test;
using UnityEngine;

Скрипт Unity | Ссылок: 0
public class InterfaceFieldDocExample3 : MonoBehaviour
{
    [SerializeField]
    InterfaceField<IDescriptor> interfaceField;
    Сообщение Unity | Ссылок: 0
    void Start()
    {
        IDescriptor descriptor = new DescriptorNonMonoBehaiviour("NONMB");
        interfaceField.Set(descriptor);
        Debug.Log(interfaceField.i.GetDescription());
    }
}
```

Example of assigning target interface instance by code

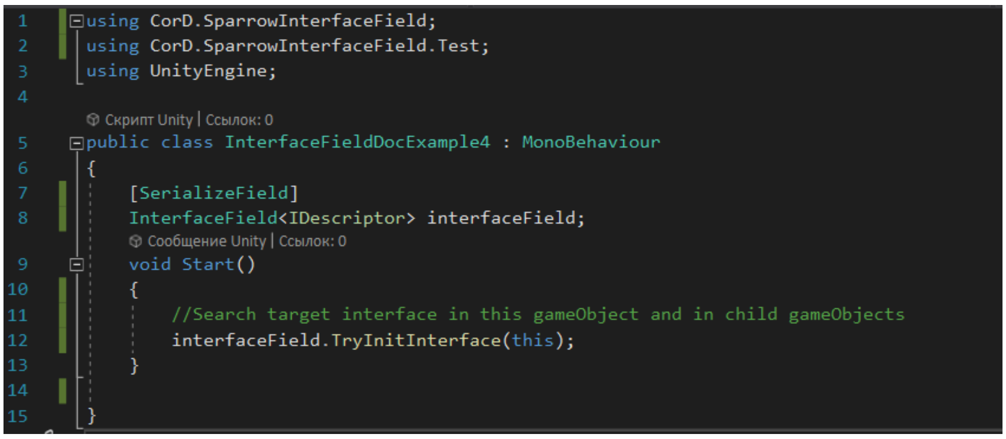


```
namespace CorD.SparrowInterfaceField.Test
{
    /// <summary>
    /// Simple example NonMonoBehavior realisation of IDescription interface
    /// </summary>
    Ссылок: 3
    public class DescriptorNonMonoBehaiviour : IDescriptor
    {
        string description;
        Ссылок: 2
        public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

        Ссылок: 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().



```
using CorD.SparrowInterfaceField;
using CorD.SparrowInterfaceField.Test;
using UnityEngine;

Скрипт Unity | Ссылок: 0
public class InterfaceFieldDocExample4 : MonoBehaviour
{
    [SerializeField]
    InterfaceField<IDescriptor> interfaceField;
    Сообщение Unity | Ссылок: 0
    void Start()
    {
        //Search target interface in this gameObject and in child gameObjects
        interfaceField.TryInitInterface(this);
    }

}
```

Example of using TryInitInterface method

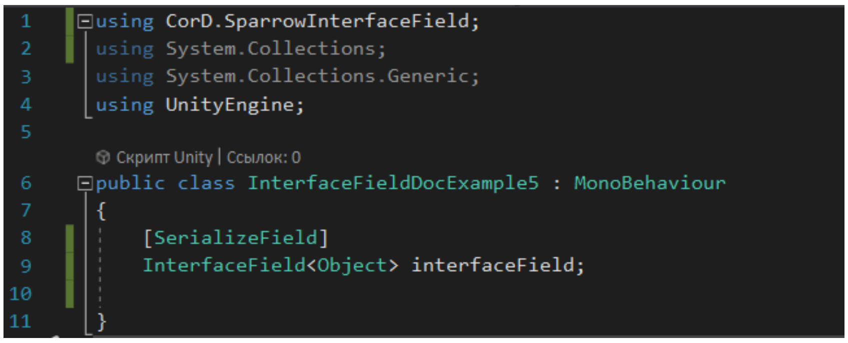
Attribute

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.

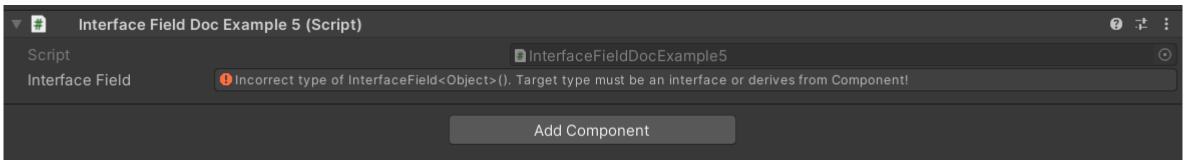


```
using CorD.SparrowInterfaceField;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

Скрипт Unity | Ссылок: 0
public class InterfaceFieldDocExample5 : MonoBehaviour
{
    [SerializeField]
    InterfaceField<Object> interfaceField;

}
```

Example of incorrect type initialization

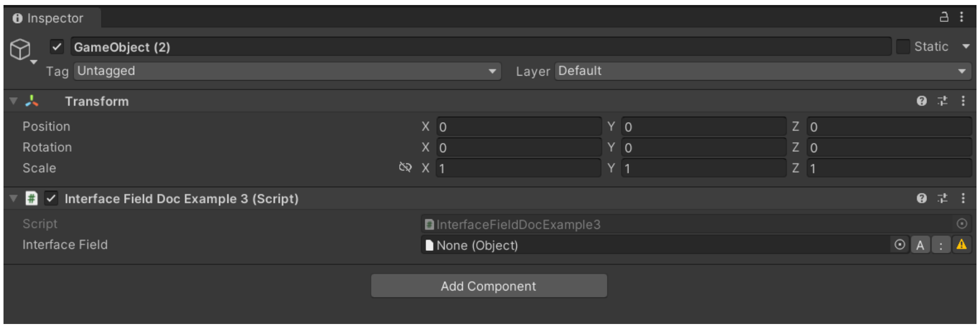


```
Interface Field Doc Example 5 (Script)
Script            InterfaceFieldDocExample5
Interface Field   Incorrect type of InterfaceField<Object>(). Target type must be an interface or derives from Component!
                  Add Component
```

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.



GameObject (2) Static
Tag Untagged Layer Default

Transform

Position	X 0	Y 0	Z 0
Rotation	X 0	Y 0	Z 0
Scale	X 1	Y 1	Z 1

Interface Field Doc Example 3 (Script)

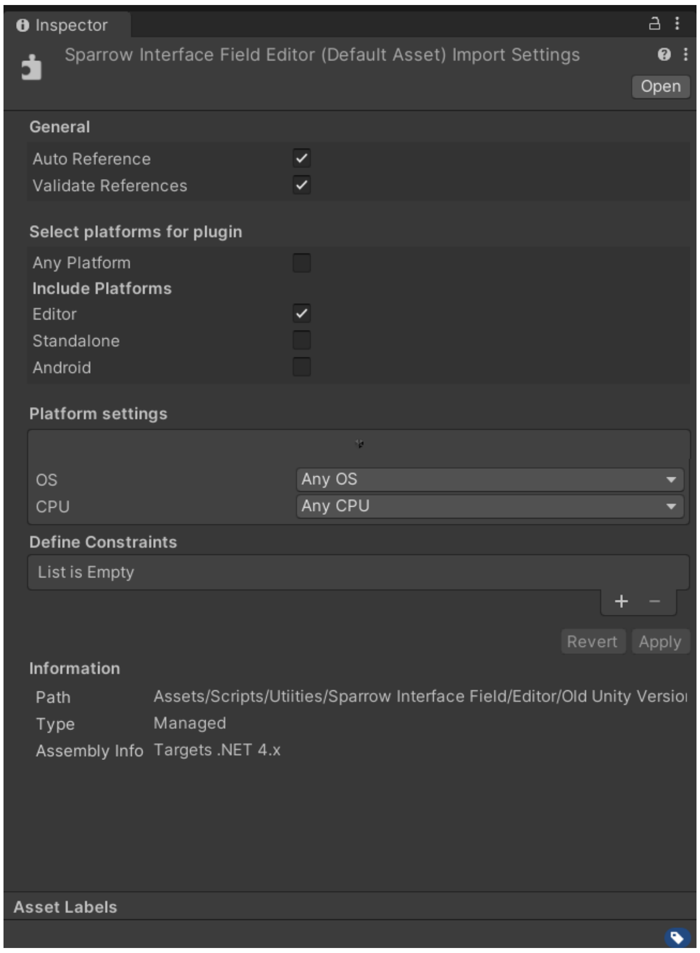
Script	InterfaceFieldDocExample3
Interface Field	None (Object)

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 <u>isolation mode</u>. 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.



Sparrow Interface Field Editor (Default Asset) Import Settings

General
Auto Reference ✓
Validate References ✓

Select platforms for plugin
Any Platform ☐
Include Platforms
Editor ✓
Standalone ☐
Android ☐

Platform settings
OS: Any OS
CPU: Any CPU

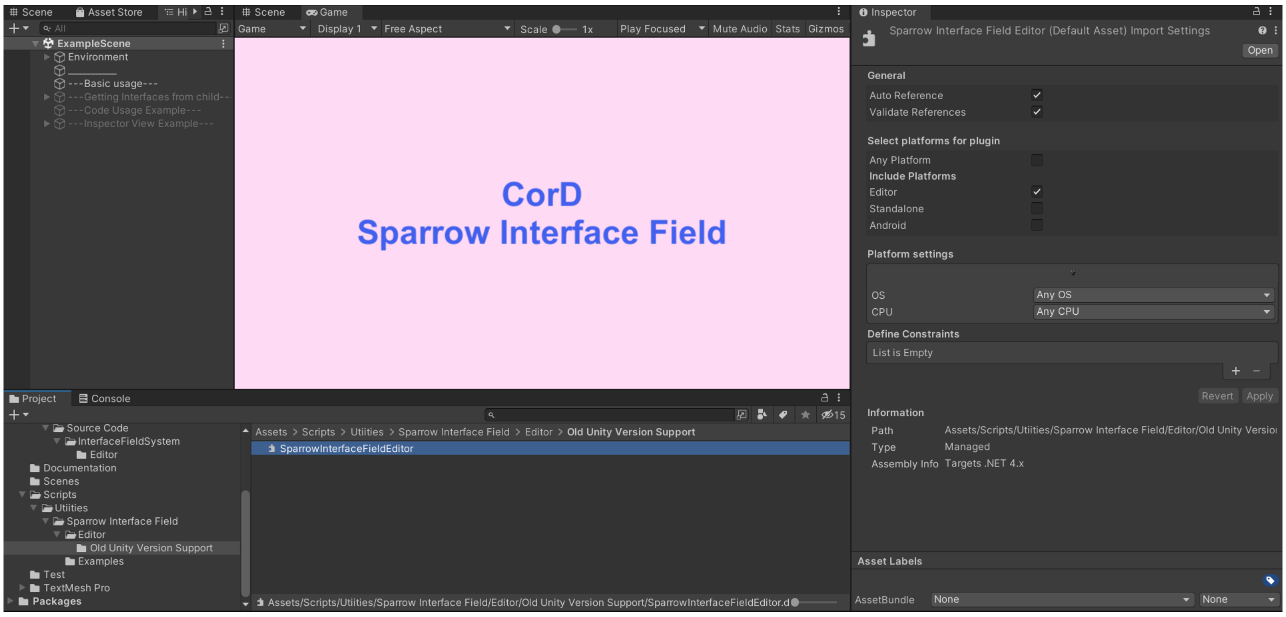
Define Constraints
List is Empty

Information
Path: Assets/Scripts/Utiities/Sparrow Interface Field/Editor/Old Unity Versio
Type: Managed
Assembly Info: Targets .NET 4.x

Asset Labels

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.



Sparrow Interface Field Editor (Default Asset) Import Settings

General

Auto Reference ✓
Validate References ✓

Select platforms for plugin

Any Platform
Include Platforms
Editor ✓
Standalone
Android

Platform settings

OS Any OS
CPU Any CPU

Define Constraints

List is Empty

Information

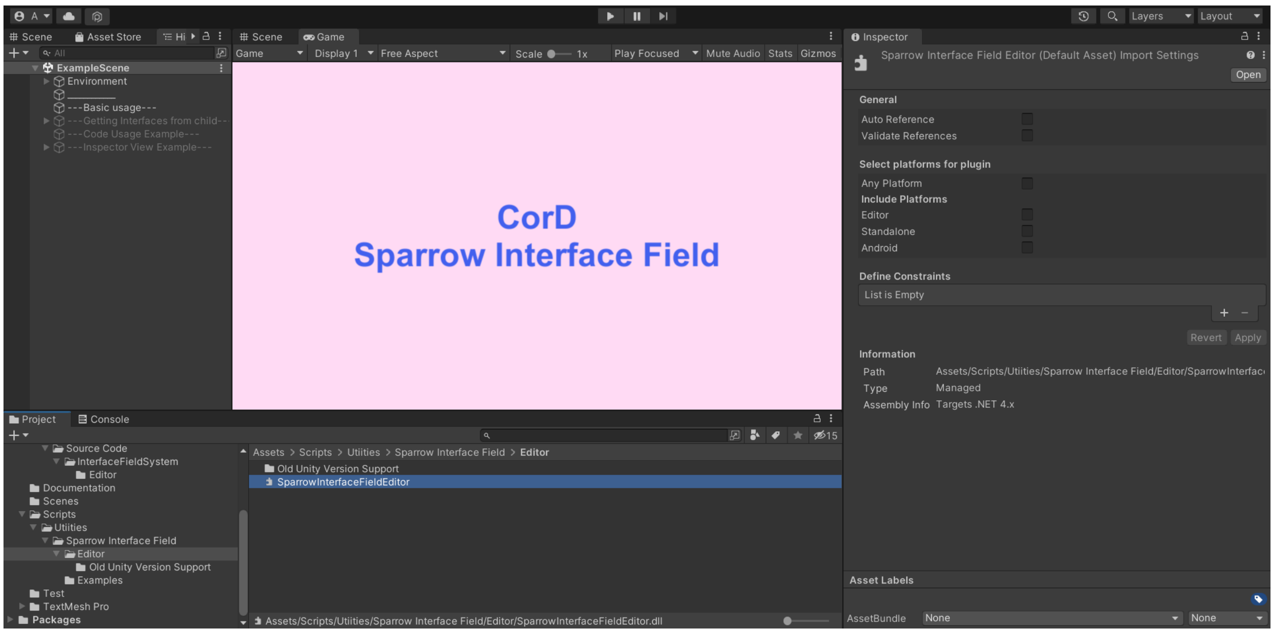
Path Assets/Scripts/Utiities/Sparrow Interface Field/Editor/Old Unity Versio
Type Managed
Assembly Info Targets .NET 4.x

Assets > Scripts > Utiities > Sparrow Interface Field > Editor > Old Unity Version Support

SparrowInterfaceFieldEditor

Assets/Scripts/Utiities/Sparrow Interface Field/Editor/Old Unity Version Support/SparrowInterfaceFieldEditor.d

DLL activation to work with older versions of Unity



Sparrow Interface Field Editor (Default Asset) Import Settings

General

Auto Reference
Validate References

Select platforms for plugin

Any Platform
Include Platforms
Editor
Standalone
Android

Define Constraints

List is Empty

Information

Path Assets/Scripts/Utiities/Sparrow Interface Field/Editor/SparrowInterfac
Type Managed
Assembly Info Targets .NET 4.x

Assets > Scripts > Utiities > Sparrow Interface Field > Editor

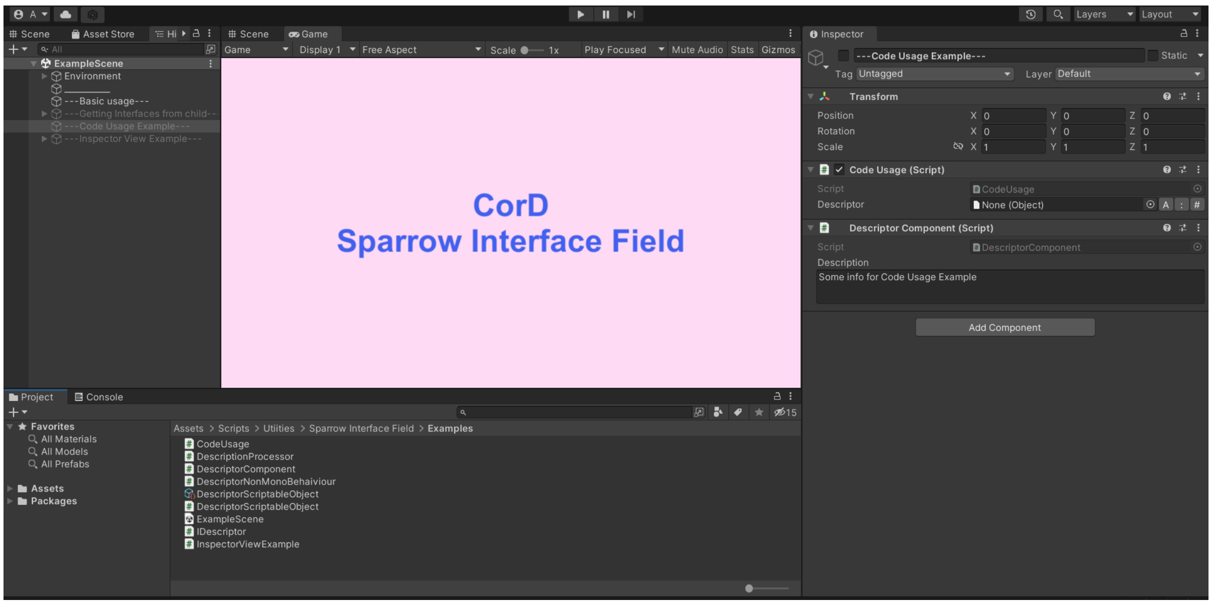
Old Unity Version Support
SparrowInterfaceFieldEditor

Assets/Scripts/Utiities/Sparrow Interface Field/Editor/SparrowInterfaceFieldEditor.dll

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.



```
ExampleScene
  Environment
  ________
  ---Basic usage---
  ---Getting Interfaces from child--
  ---Code Usage Example---
  ---Inspector View Example---
```

```
---Code Usage Example---                Static
Tag Untagged            Layer Default

Transform
Position    X 0   Y 0   Z 0
Rotation    X 0   Y 0   Z 0
Scale       X 1   Y 1   Z 1

Code Usage (Script)
Script       CodeUsage
Descriptor   None (Object)   A : #

Descriptor Component (Script)
Script       DescriptorComponent
Description
Some info for Code Usage Example
```

```
Favorites
  All Materials
  All Models
  All Prefabs
Assets
Packages
```

```
Assets > Scripts > Utiities > Sparrow Interface Field > Examples
  CodeUsage
  DescriptionProcessor
  DescriptorComponent
  DescriptorNonMonoBehaiviour
  DescriptorScriptableObject
  DescriptorScriptableObject
  ExampleScene
  IDescriptor
  InspectorViewExample
```

Folder with code examples