

**PropertyInterface offline wiki**

# Introduction

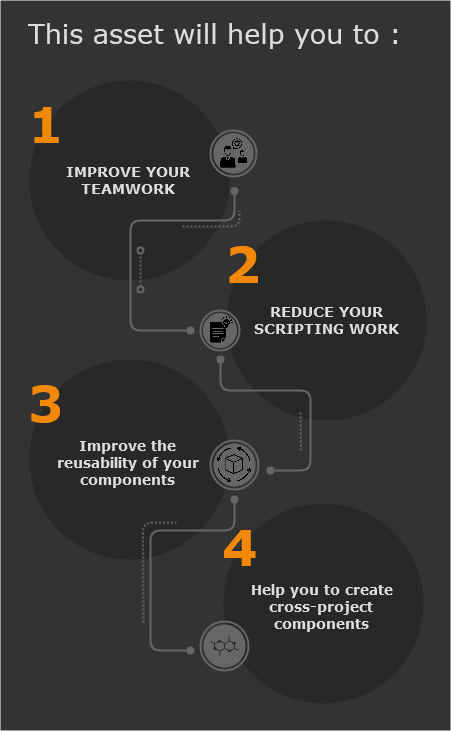
Welcome to the PropertyInterface wiki wich will help you to use and extand this powerfull asset.

## Required

To use this asset you need :

* At least Unity 2019.1.0f2
* Unity Player settings/Scripting Runtim Version => .NET 4.X

## Lets see what this asset can do for you**.**

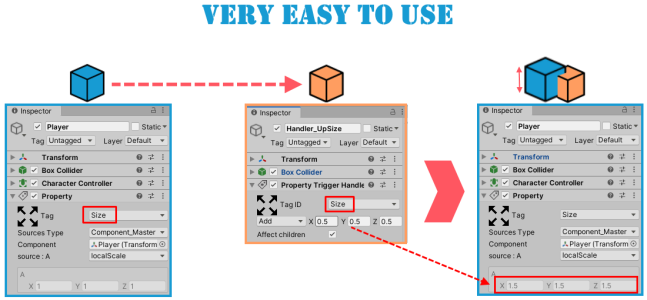


# First Look

## What does this Asset do ?

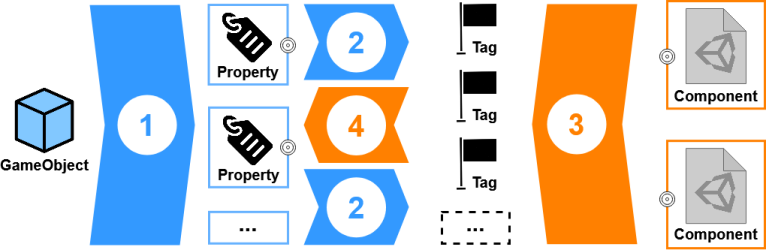
This asset allow you to easily add [Property](https://github.com/Nectunia/PropertyInterface/wiki/Property-Overview) to GameObject.

* Each [Property](https://github.com/Nectunia/PropertyInterface/wiki/Property-Overview) is a component so it can be linked to your other Components.
* Each [Property](https://github.com/Nectunia/PropertyInterface/wiki/Property-Overview) can be found by script via a powerfull and easy to use custom [Tag](https://github.com/Nectunia/PropertyInterface/wiki/Tag-Overview) system.
* Each [Property](https://github.com/Nectunia/PropertyInterface/wiki/Property-Overview) values can be filled with raw values or sourced from other Component.



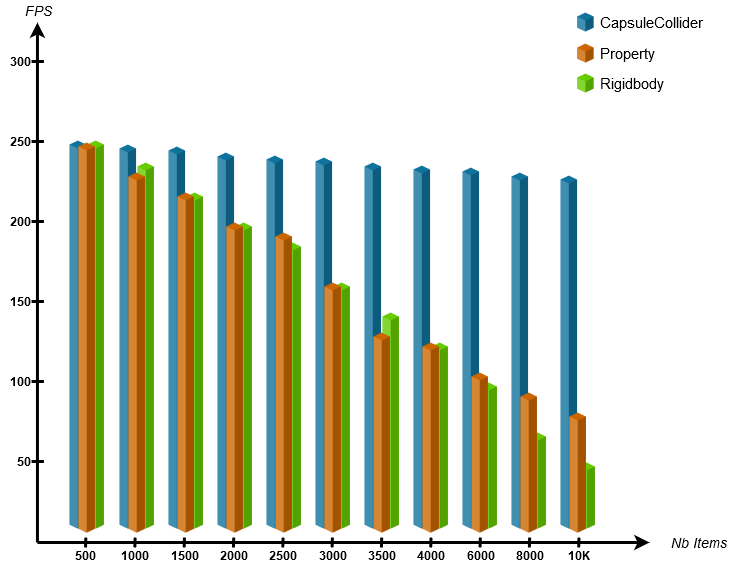
A complete and easy to use library allow you to handle [Property](https://github.com/Nectunia/PropertyInterface/wiki/Property-Overview) by script as well.

## How does it work ?



1. Add [Property](https://github.com/Nectunia/PropertyInterface/wiki/Property-Overview) to GameObject.
2. Select a [Tag](https://github.com/Nectunia/PropertyInterface/wiki/Tag-Overview) for the [Property](https://github.com/Nectunia/PropertyInterface/wiki/Property-Overview).
3. [Make your own tagged component](https://github.com/Nectunia/PropertyInterface/wiki/Create-my-tagged-component).
4. Select a [Tag](https://github.com/Nectunia/PropertyInterface/wiki/Tag-Overview) in your custom component. It will now be able to interact with all [Property](https://github.com/Nectunia/PropertyInterface/wiki/Property-Overview) component in the scene wich have the same [Tag](https://github.com/Nectunia/PropertyInterface/wiki/Tag-Overview).

# What about performance ?

As you can see below, [Property](https://github.com/Nectunia/PropertyInterface/wiki/Property-Overview) is not the lightest component. However it will take a significant amount of [Property](https://github.com/Nectunia/PropertyInterface/wiki/Property-Overview) before it really affects the performance of your game.  


All tests have been done with the following configuration :

* OS : Windows 10 Familly 64Bits
* UNITY : 2019.3.0f6
* CPU : Intel Core I7-4800MQ (4 x 2.7GHz)
* GPU : Nvidia GeForce GTX870m 3GB GDDR5
* RAM : 8GB DDR3 800Mhz
* HD : Sata 7200rpm

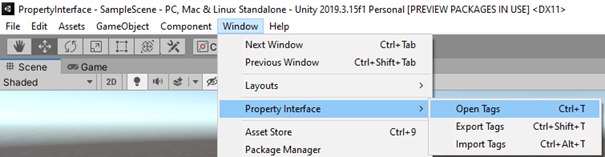
# Tag Overview

## What is it for ?

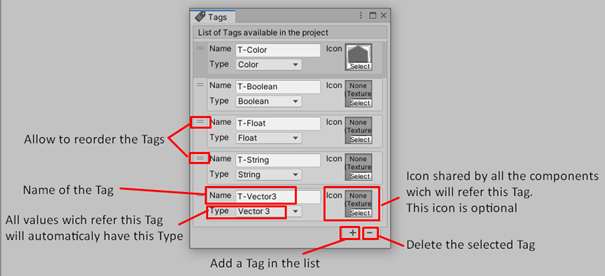
Tags are made to link your [tagged components](https://github.com/Nectunia/PropertyInterface/wiki/Create-my-tagged-component) and the [Property](https://github.com/Nectunia/PropertyInterface/wiki/Property-Overview) components that you will add to your GameObjects.

warning***Tag are not built. You shouldn't access them in game play.***

### Opening the Tags window (Ctrl+T by default):



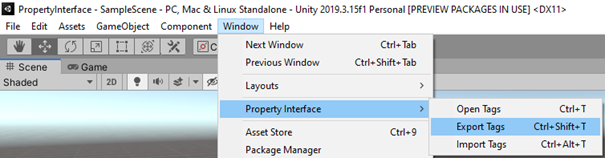
## Setting Tags :



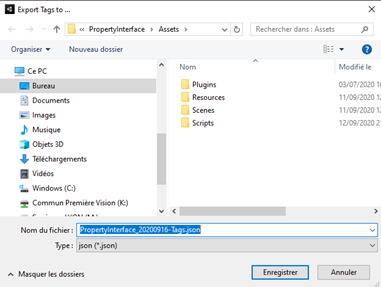
# Export Tags

It's possible to export a Tag list to backup or share it.

In Unity editor, go to menu "Windows/PropertyInterface/Export Tags" *(Ctrl+Shift+T by default)*:



Choose in your explorer where to export your Tags :



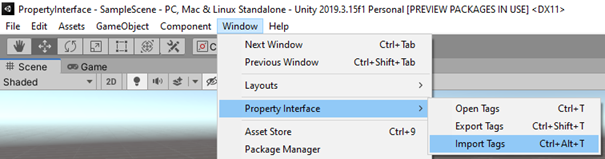
You can now share it to another project or use it as backup.

# Import Tags

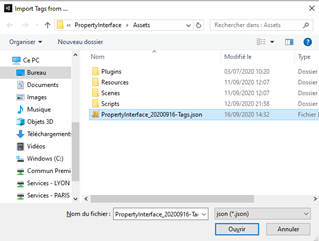
It's possible to import a Tag list from a backup or another project.

**warningThis action will replace all your Tags by thoose imported**

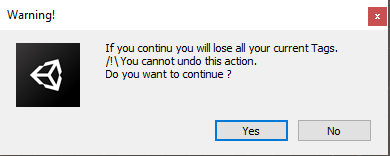
In Unity editor, go to menu "Windows/PropertyInterface/Import Tags" *(Ctrl+Alt+T by default)*:



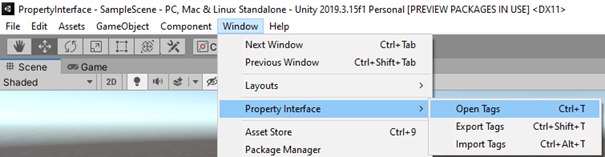
Choose a previous exported Tags file in your explorer *(See* [*Export Tags*](https://github.com/Nectunia/PropertyInterface/wiki/Export-Tags)*)* :



**warningThis action can't be undone.** If you are sure, click on the **Yes** button in the warning popup :



Open the Tags window to see your changes *(Ctrl+T by default)*:



# Property Overview

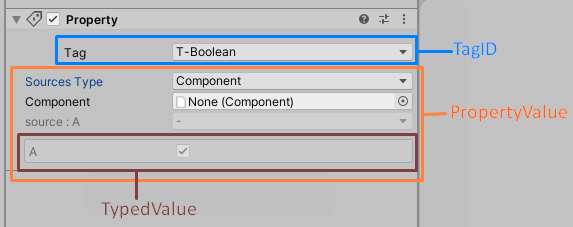
## What is it for ?

A Property is a component wich allow to add properties in GameObjects.

## Property's elements

Property components are composed of 2 elements :

* A [TagID](https://nectunia.github.io/PropertyInterface/struct_nectunia_1_1_property_interface_1_1_tag_i_d.html) (used to refer a Tag)
* A [PropertyValue](https://nectunia.github.io/PropertyInterface/class_nectunia_1_1_property_interface_1_1_property_value.html) (where all the Property settings are stored)

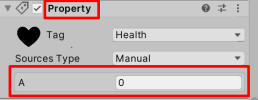
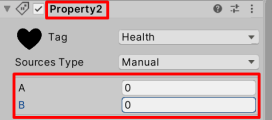
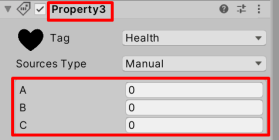


The [TypedValue](https://nectunia.github.io/PropertyInterface/class_nectunia_1_1_property_interface_1_1_typed_value.html) is stored in the PropertyValue and will be define by the Type set in the Tag.  
In the above example, the Type is set to ***Boolean*** in the Tag named "T-Boolean". So the TypedValue is a checkbox.

Property composition

## Property's values

As Vector fields, Property components can have several values. By default they can have 3 values maximum :

* Property => one value (A) 
* Property2 => two values (A, B) 
* Property3 => three values (A, B, C) 

warning***Property, Property2 and Property3 are differents components.***

Property2 and Property3 exist because it can be usefull sometime to have distinct values for the same Property. For example, if you take the Health characteristic for your player. With Property2, you can set the current Health (A) and the max Heatlh (B). With Property3, you can set the min Health (A), the current Health (B) and the max Health (C).

If you need more values for your Property, you can make your own Property component. See [Property inheritance](https://github.com/Nectunia/PropertyInterface/wiki/Property-inheritance).

## How does it works?

In editor mode, you will [setting the Property's value](https://github.com/Nectunia/PropertyInterface/wiki/Setting-a-Property).  
At runtime, the value will be updated by your custom [tagged components](https://github.com/Nectunia/PropertyInterface/wiki/Create-my-tagged-component).

## Add a Property

A Property is a component. So you can add it as any other component.  
Select a GameObject in the scene view and click on "**Add component**" button in the inspector view :

