

# Table of Contents

Introduction	1.1
Documentation	1.2
Implementation	1.2.1
Usage	1.2.2
Implementation	1.3
Installation	1.4

# Factions Extension Documentation

Add factions and relations in your game using C++ or Blueprints

This plugin is for Unreal Engine 4 and has support for versions **4.20**, **4.19** and **4.18**.

You can download this [Test project](#) to see and test the API

## Introduction

### What are "Factions"

When we say Faction we refer to a **group of entities or actors** that share something in common.

Almost every single type of game uses factions in different ways. For example, in Shooter games there will be enemies and friends, in RTS games every player will be a faction by itself, and in Open-World games you will have factions fighting each other while you run around.

This plugin fulfills the needs of this feature in UE4 with a very flexible tool that will make the implementation, editing and design of your own factions a 5 minutes thing.

### "What" can have a faction?

*Everything!*... well not everything, but pretty much. **All actors of any type can have a faction.**

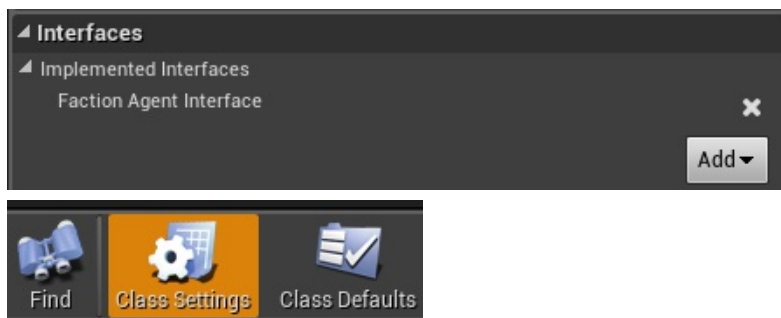
# Implementation

## Blueprints

Check the [Test Project](#) for an in-engine example

### Adding the interface

For an actor to have a faction, we need to add a FactionAgentInterface. This will allow the system to set and get the current interface.



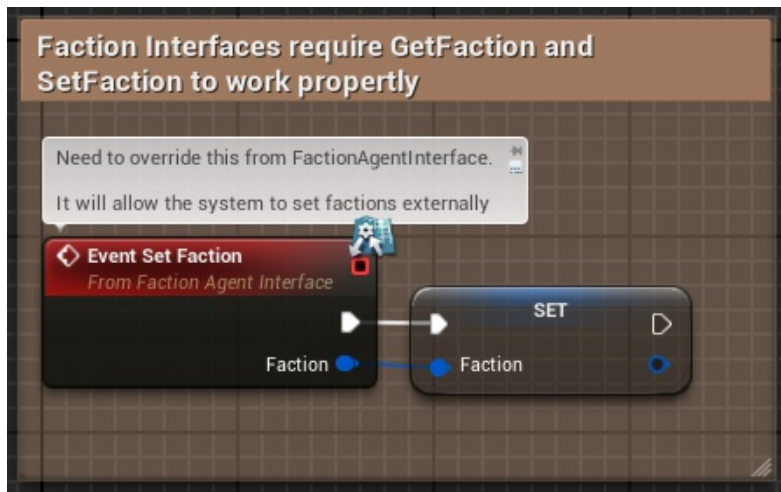
To add an interface we have to go into **Class Settings**

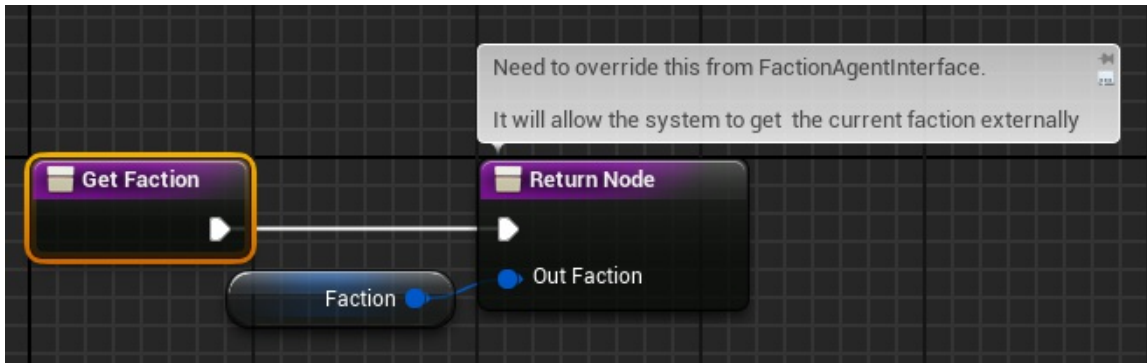
From here we click on **Add**, inside Interfaces and search for FactionAgentInterface

### Getting and Settings the Faction

Factions require your actor to define how to get and set a faction. That can be done overriding **GetFaction** and **SetFaction**.

A common implementation would be to have a **class variable** (of type "Faction"), and then return it at GetFaction and set it at SetFaction:





## Factions in Controllers

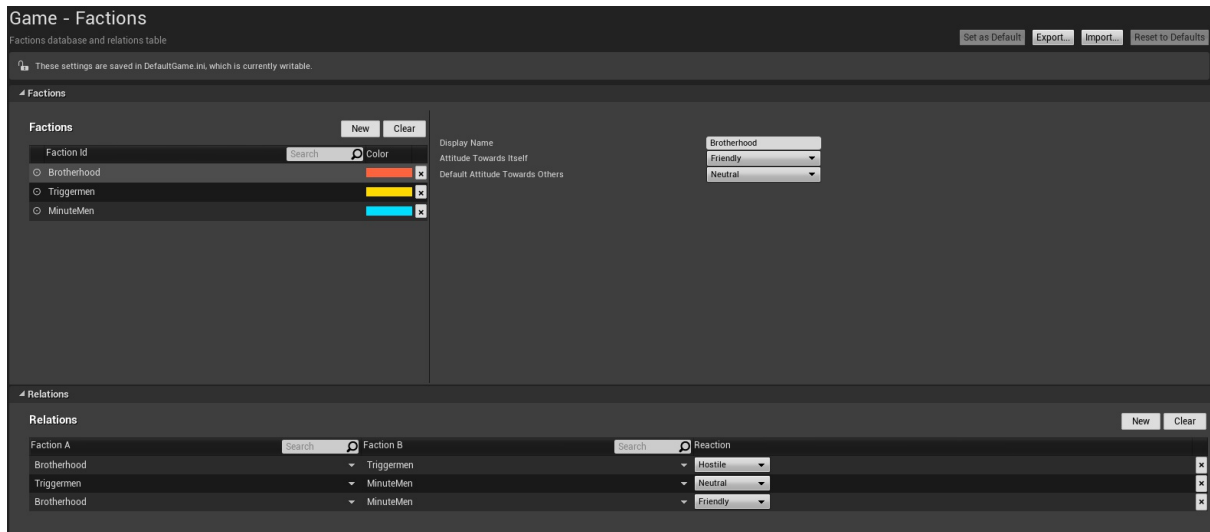
Sometimes you may want a controller to share a faction with its controlled pawn or character.

This process is exactly the same as before, except that instead of getting and setting a variable, we will get and set the faction from our pawn.

## C++

# Usage

All settings can be edited from **Project Settings -> Game -> Factions** tab.

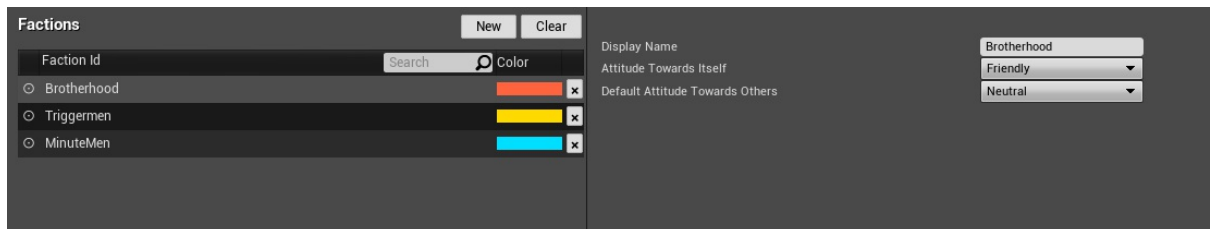


## Factions

Factions settings are defined by right and left areas.

**At the left** you have a list of all factions, with a little search box in case you have more than you can deal with.

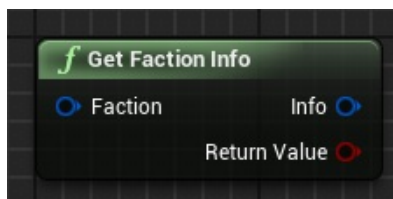
**At the right** you will find all the properties of a selected faction (which was selected by just clicking on it on the list).



You can also rename factions by double clicking on them.



All Faction information can be read on runtime too:



# Relations

Relations define how will two factions interact to each other.

In this example, Brotherhood and Triggermen are enemies:

The screenshot shows a 'Relations' window with two columns: 'Faction A' and 'Faction B'. Each column has a search bar and a dropdown menu. The 'Reaction' column has a dropdown menu with three options: 'Hostile', 'Neutral', and 'Friendly'. The interface is dark-themed.

Faction A	Faction B	Reaction
Brotherhood	Triggermen	Hostile
Triggermen	MinuteMen	Neutral
Brotherhood	MinuteMen	Friendly

By default there are three possible attitudes: **Hostile**, **Neutral** and **Friendly**

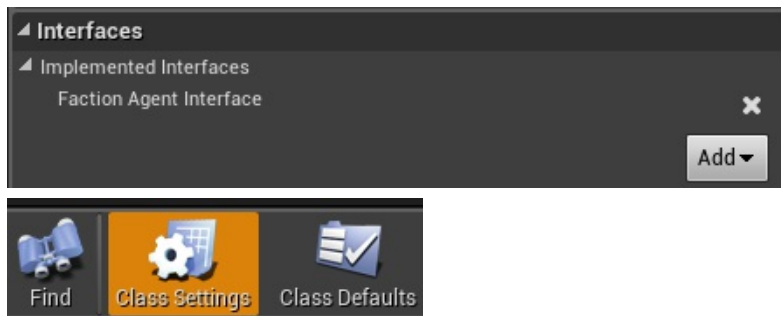
# Implementation

## Blueprints

Check the [Test Project](#) for an in-engine example

### Adding the interface

For an actor to have a faction, we need to add a FactionAgentInterface. This will allow the system to set and get the current interface.



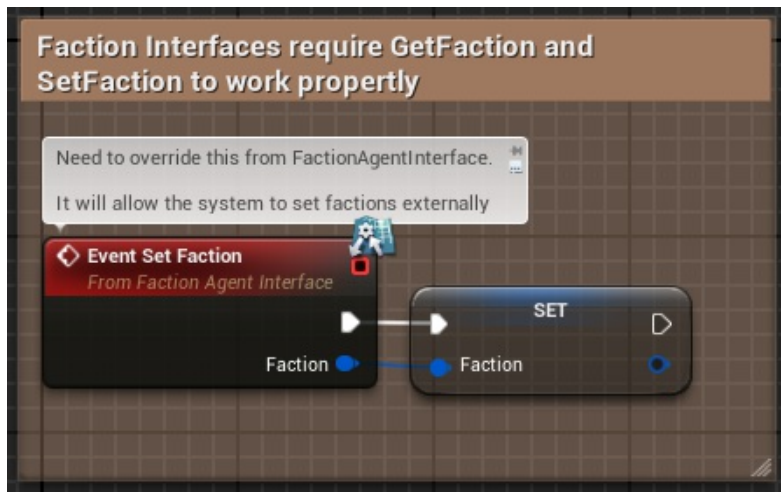
To add an interface we have to go into **Class Settings**

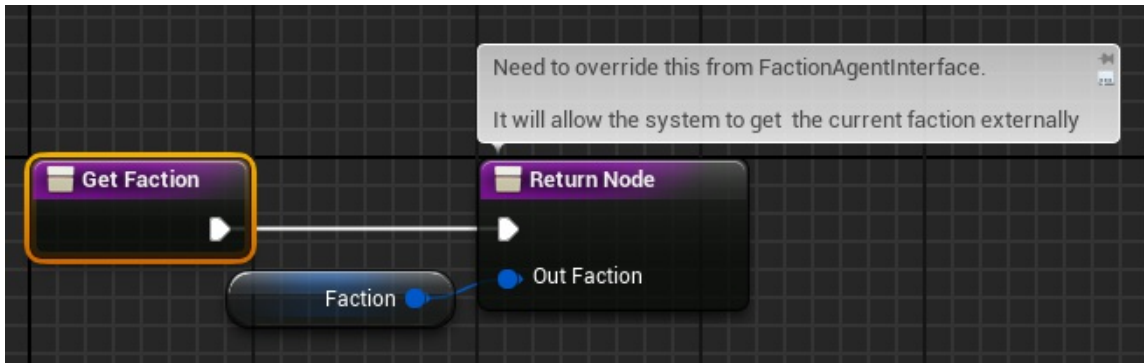
From here we click on **Add**, inside Interfaces and search for FactionAgentInterface

### Getting and Settings the Faction

Factions require your actor to define how to get and set a faction. That can be done overriding **GetFaction** and **SetFaction**.

A common implementation would be to have a **variable** (of type "Faction"), and then return it with GetFaction and set it with SetFaction:





## Factions in Controllers

Sometimes you may want a controller to share a faction with its controlled pawn or character.

This process is exactly the same as before, except that instead of getting and setting a variable, we will get and set the faction from our pawn.

## C++



# Installation

## Manually

This are the general steps for installing the plugin into your project:

1. Download the last release from [here](#)

*Make sure you download the same version than your project*

2. Extract the folder "FactionsExtension" into the **Plugins folder** of your existing project (e.g "MyProject/Plugins")

2. Done! You can now open the project

## From Marketplace

Install from the launcher: [AVAILABLE HERE](#)