

Developer
/ Documentation
/ Unreal Engine ▾
/ Unreal Engine 5.4 Documentation
/ Creating User Interfaces
/ Plugins for UI Development
/ Common UI
/ Using CommonUI With Enhanced Input

Using CommonUI With Enhanced Input

Learn how to incorporate Enhanced Input with CommonUI's input system.



 Learn to use this **Experimental** feature, but use caution when shipping with it.

PREREQUISITE TOPICS



In order to understand and use the content on this page, make sure you are familiar with the following topics:

- [Common UI Overview](#)

The **CommonUI plugin** provides limited support for [Enhanced Input](#) actions.



As of UE 5.2, Enhanced Input support has not been tested as thoroughly as other features in CommonUI. We do not recommend attempting to ship titles with this feature at this time.

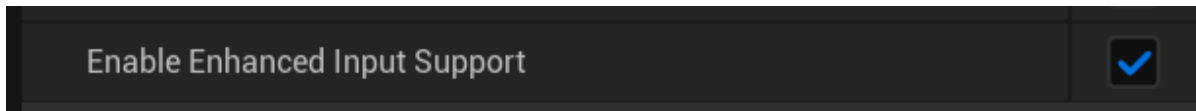
1. Required Setup

This page assumes that you have performed the following setup steps for CommonUI:

- Enable both the CommonUI and Enhanced Input plugins.
- Set your Viewport class to `CommonGameViewportClient`.
- Set up your InputData for Accept/Back actions.

2. Enable Enhanced Input in CommonUI

After enabling the CommonUI and Enhanced Input plugins, open **Project Settings**. Navigate to **Game > Common Input Settings** and set **Enable Enhanced Input Support** to **true**. This enables support for these two plugins to communicate.

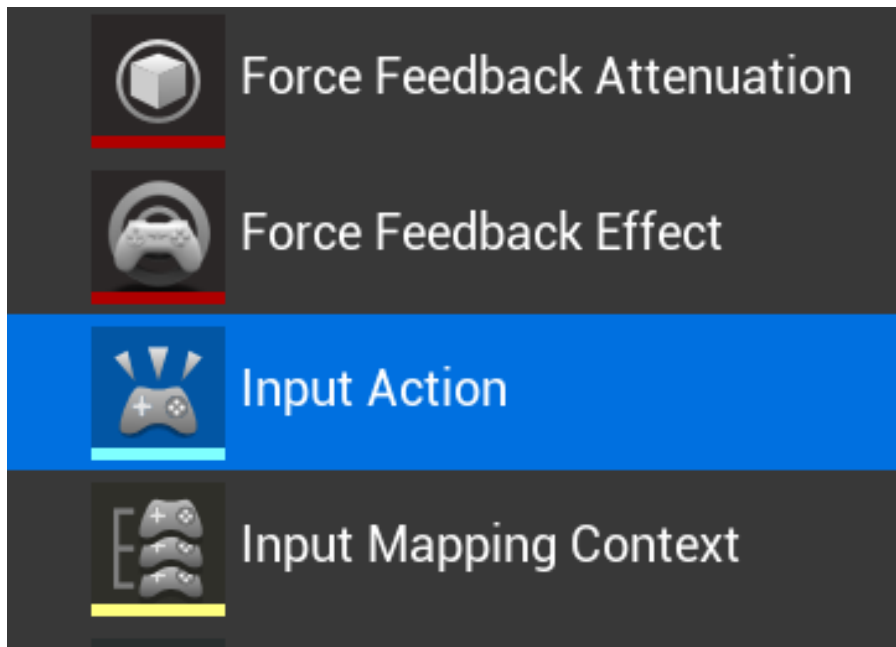


3. Set Up Enhanced Input Actions for CommonUI

Enhanced Input actions can bind input events anywhere, including specialized variants such as *ongoing* and *triggered* events. However, it is undesirable and chaotic to globally bind a majority of UI action bindings such as *FaceButtonTop*, *Accept*, or *Back*, as this can cause unintended events as a result of the user's input at unexpected times. CommonUI solves this problem with *generic actions*. Generic actions are bound to UI elements, but do not trigger their Enhanced Input Events in CommonUI.

To set up Enhanced Input Actions within CommonUI, follow these steps:

1. Create a generic **Input Action** in the Content Browser. Name it `IA_UI_GenericAccept`.



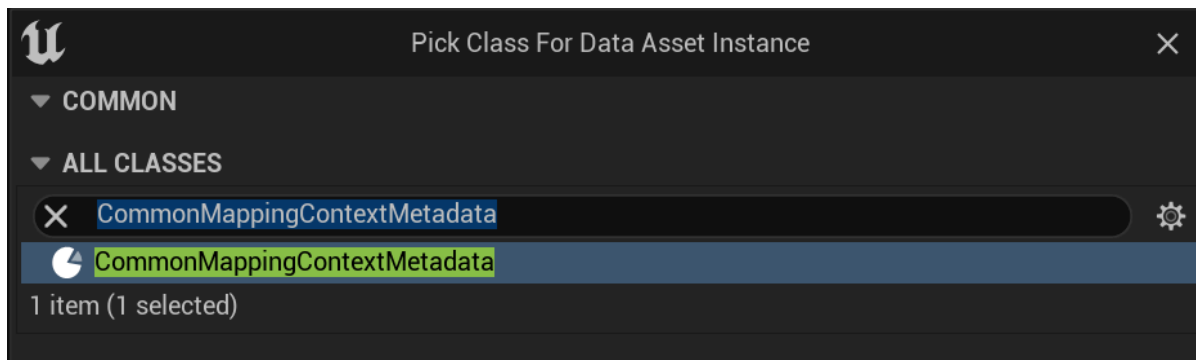
2. Add a **PlayerMappableKeySettings** to your Input Action.
3. In the **Settings** for your **PlayerMappableKeySettings**, set the **Metadata** field to an object that implements the `ICCommonMappingContextMetadataInterface`.

| | |
|--------------------------------|---|
| ▼ Description | |
| Action Description | Forward |
| ▼ Action | |
| Consume Input | <input checked="" type="checkbox"/> |
| Trigger when Paused | <input type="checkbox"/> |
| Reserve All Mappings | <input type="checkbox"/> |
| Value Type | Digital (bool) ▼ |
| Triggers | 0 Array elements |
| Modifiers | 0 Array elements |
| ▼ Input | |
| ▼ Settings | |
| ▼ Player Mappable Key Settings | Player Mappable Key Settings (Experimental) ▼ |
| ▼ Settings | |
| Metadata | UI_GenericMetadataMC ▼ |
| Name | Accept |
| Display Name | <input type="text"/> |
| Display Category | <input type="text"/> |

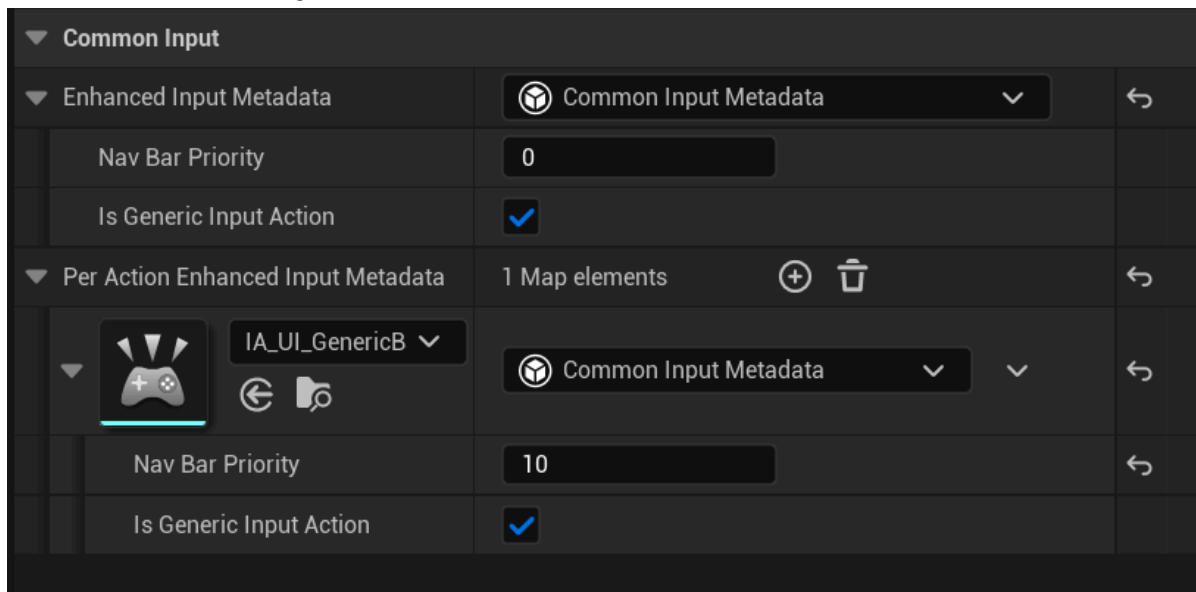


You might need your Input Action Metadata for something other than CommonUI. Therefore, we advise using a class that implements `ICommonMappingContextMetadataInterface` to ensure flexibility.

4. Right-click in the Content Browser, then click **Miscellaneous** > **Data Asset** to create a Data Asset.
5. Select a metadata class that implements `ICommonMappingContextMetadataInterface` as your Data Asset class. Name your Metadata UI_IA_GenericMetadata. You can use `UCommonMappingContextMetadata` as a default, or you can use a custom asset class.



6. Open UI_IA_GenericMetadata, then edit its settings as follows:
 - **Is Generic Input Action:** True
 - **Per Action Enhanced Input Metadata:** IA_UI_GenericAccept
 - **Nav Bar Priority:** 10




Check **Is Generic Input Action** to prevent CommonUI from broadcasting the input action.

This data asset provides a metadata object where you can set CommonUI action data. If you are already familiar with CommonUI, you might recognize the Nav Bar Priority setting from CommonUI's data tables. You can also extend your input actions with additional metadata by inheriting from *UCommonInputMetadata*.

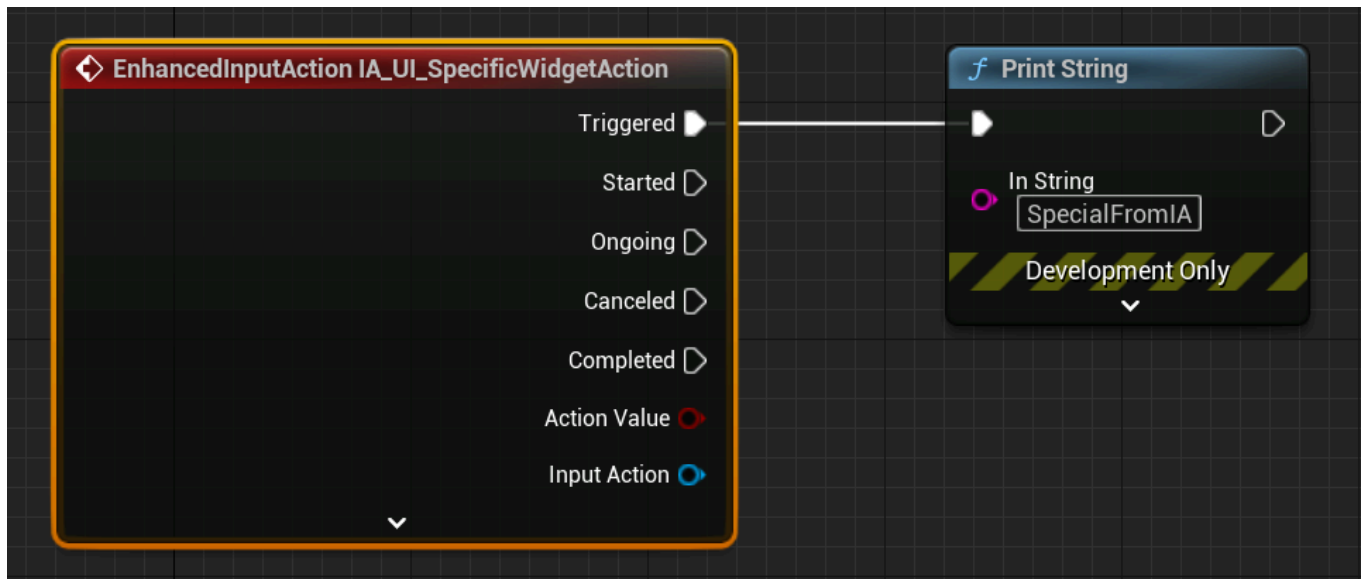


You can use the **Per Action Enhanced Input Metadata** to handle the metadata for multiple actions in a single asset rather than creating one asset for each action.

- Repeat steps 5-6, but this time leave Is Generic Input Action unchecked. Name this metadata `UI_IA_SpecificMetadata`. This results in a metadata class you can use on any Input Action that is not generic.
- Open your Input Action. Set the Metadata to `UI_IA_GenericMetadata`. The Input Action now has all the information needed to function with CommonUI.

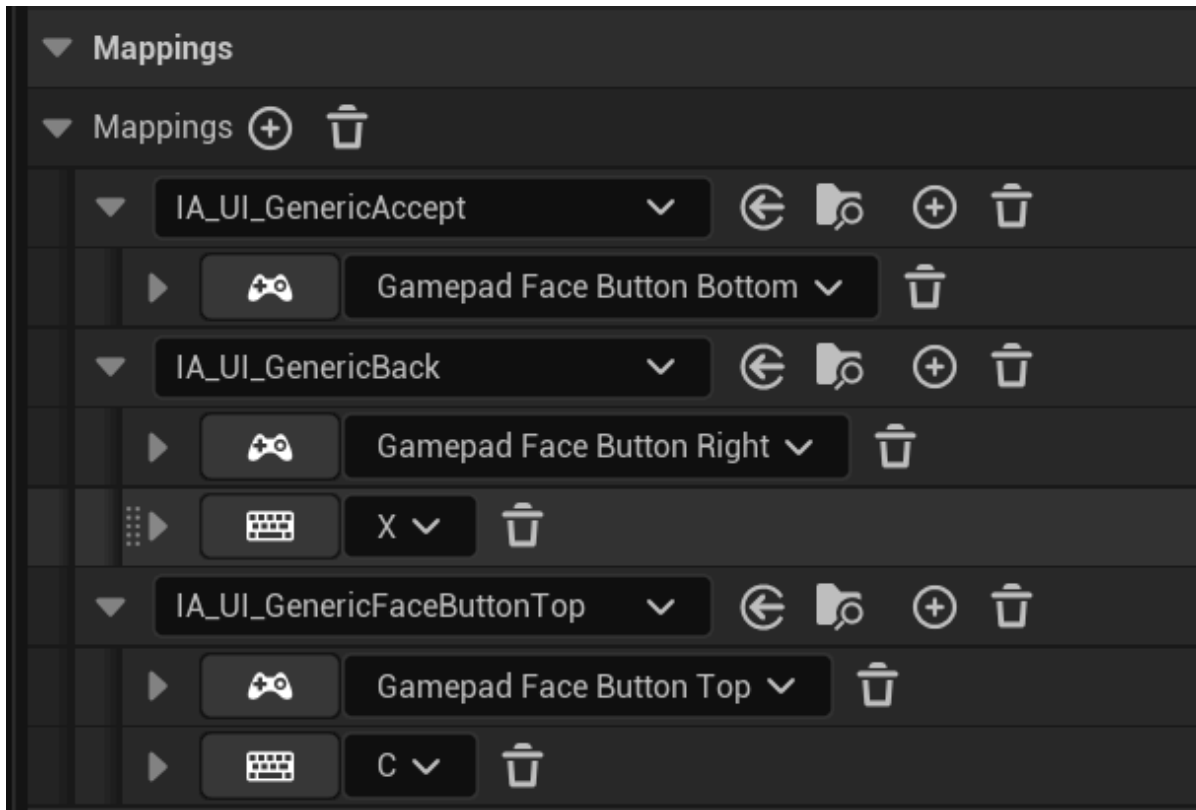
| | |
|------------------|---|
| Metadata |  UI_GenericMetadataIMC |
| Name | Accept |
| Display Name | <input type="text"/> |
| Display Category | <input type="text"/> |

- For Input Actions whose metadata does not have Is Generic Input Action enabled, you can bind input to events like any other Input Action.



4. Create an Input Mapping Context (IMC) for CommonUI

Input Mapping Contexts (IMCs) for CommonUI behave the same as other IMCs. To create an Input Mapping Context, right-click in the Content Browser, then click **Input > Input Mapping Context**. The following image is an example of what an IMC used with CommonUI might look like:

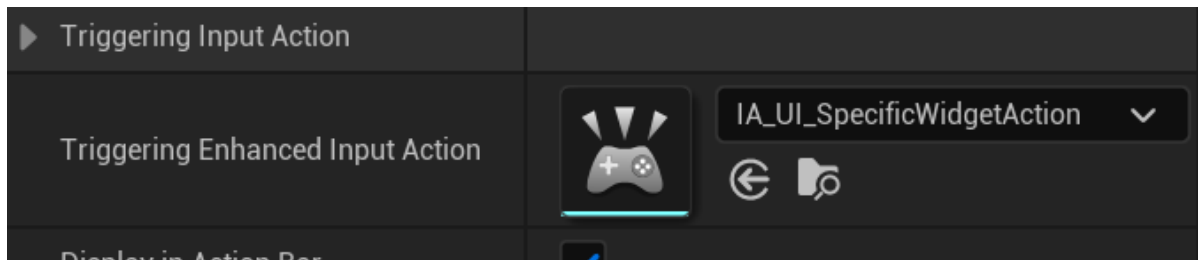


For the sake of clarifying that your IMC is used for your UI, we recommend naming it `IMC_UI_GenericActions` or something similar.

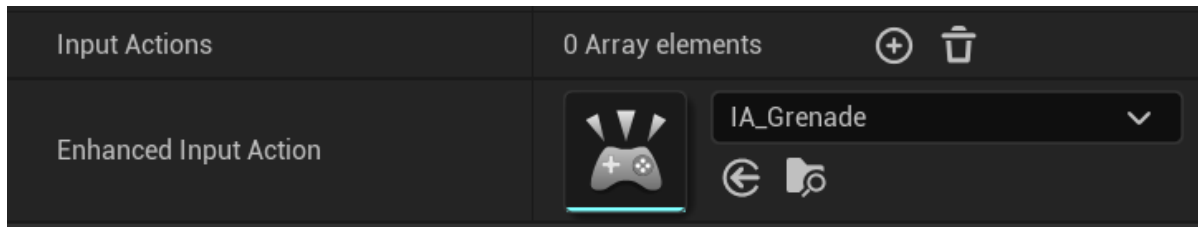
5. Use Input Actions and Input Mapping Contexts in CommonUI

You can use Input Actions anywhere you previously used **DataTableRows** to specify input information. The following are typical examples:

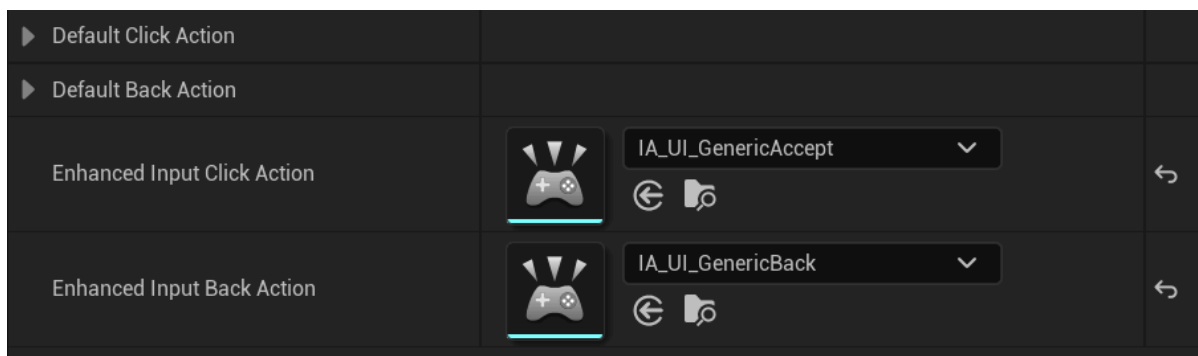
- `CommonButtonBase`.



- CommonActionWidget. This can show keys for Non-UI Input actions.



- Common UI Input Data. This is where default navigation actions are defined.



If these settings do not appear, check that the [Enable Enhanced Input in CommonUI](#) setting is set to true.

In Activatable Widgets, you can specify IMCs to apply and remove on activation and deactivation. For better organization, we suggest applying your generic UI IMC wherever you apply other top-level game IMCs.

