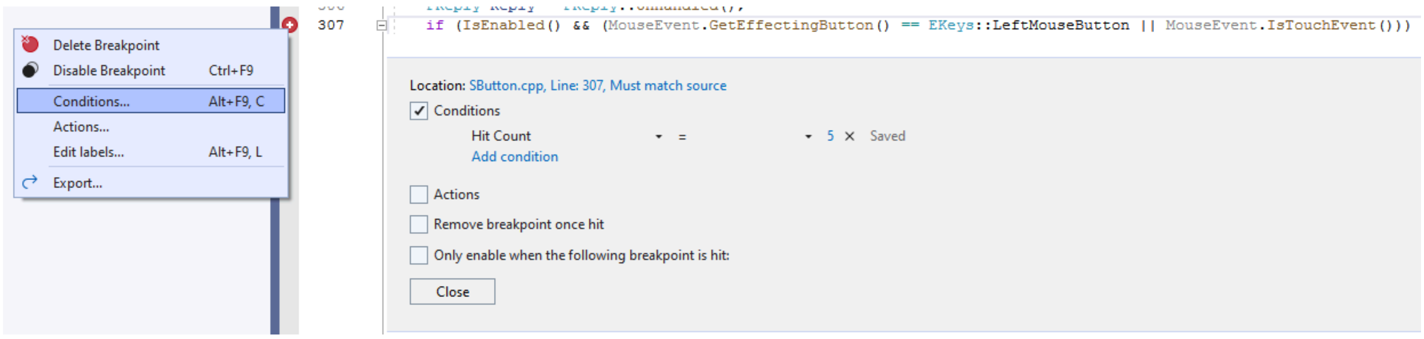
Input Debugging and Troubleshooting

Tips and tricks for debugging input code in CommonUI.

Debugging with breakpoints can trigger window focus changes, which in turn can affect widget focus in your application. This makes debugging input difficult, as the breakpoint can cause your UI to enter a different state than the one you intend to debug. To work around this limitation, try using **conditional breakpoints**.

Conditional Breakpoints With Hit Count

The simplest way to create a conditional breakpoint is to add a **hit count** to your breakpoint so that your breakpoint triggers after a few clicks.



```
if (IsEnabled() && (MouseEvent.GetEffectingButton() == EKeys::LeftMouseButton || MouseEvent.IsTouchEvent()))
```

Location: SButton.cpp, Line: 307, Must match source
[✓] Conditions
Hit Count = 5 Saved
Add condition
[ ] Actions
[ ] Remove breakpoint once hit
[ ] Only enable when the following breakpoint is hit:
Close

An example of a conditional breakpoint using the hit count condition in Visual Studio.

After you add a hit count, ensure that your widget of interest is clicked on the final click.

Refer to your IDE's documentation for more details about how to set up conditions such as Hit Counts.

Conditional Breakpoints With the Widget Reflector

You can use the Widget Reflector to get a conditional breakpoint for a specific widget. To do this, click **[CBP]** once you have identified the widget. This limits the breakpoint to only activate for that widget. For example, you can add a breakpoint to `SButton::OnMouseButtonDown` that only triggers for a specific button.

The [CBP] button in the Widget Reflector

CommonUI Input Troubleshooting FAQ

The following are some frequently-encountered issues or questions for developers using CommonUI:

How do I Work Around Ticking / Game Pauses Affecting UI?

CommonUI currently works with **LocalPlayerSubsystems**. If the game is paused, these subsystems won't tick If the subsystems do not tick, then CommonUI, including the CommonBoundActionBar, won't work. As a workaround, we recommend setting relevant Actors and Widgets to **tickable when paused** in cases where it will not contradict the intended functionality or performance of your UI. If necessary, derive custom child classes to do so.

Why am I getting analog inputs in my key handler methods?

In UE5 the *InputKey* and *InputAxis* behaviors are consolidated at the **PlayerController** level. Our intention is to make it easier to do fake input injections and bundle input parameters for easier updates in the future.

If you have code from before UE 5.0, analog inputs might now trigger key handler callbacks or vice versa. CommonUI should be correctly guarded to prevent this from causing any meaningful effects. However, if you are debugging early in the input pipeline, then you might notice some of this cross-triggering. For example, since `FCommonAnalogCursor` is an input processor, it interacts with earlier parts of the input pipeline. Because of this interaction, you might see cross-triggering in `FCommonAnalogCursor::HandleKeyDownEvent`.

Why am I getting game inputs when my input mode is Menu?

Per `UCommonUIActionRouterBase::CanProcessNormalGameInput`, UE5 allows game input in menu mode if the game viewport has mouse capture. This helps support manipulation of preview items and characters in the world while in menus. If you do not want this behavior, disable viewport mouse capture in your desired input config.