

# Event Dispatchers

Allows a Blueprint Class to report on its state to the Level Blueprint.





An example on [Event Dispatchers](#) from the [Blueprint Actor Communication](#) page demonstrates both languages of Blueprint and C++ in a feature-oriented way. This page should be considered a supplemental helper for Blueprint Scripting.

By binding one or more events to an **Event Dispatcher**, you can cause all of those events to fire once the Event Dispatcher is called. These events can be bound within a Blueprint Class, but Event Dispatchers also allow events to be fired within the Level Blueprint.

## Creating Event Dispatchers

Event Dispatchers are created in the Blueprint Editor's [Blueprint Editor My Blueprint Panel](#) tab.

To create a new Event Dispatcher:

1. In the **My Blueprint** panel click  on the Event Dispatcher category: .
2. Enter a name for the Event Dispatcher in the name field that appears at the end of the list in the **My Blueprint** tab.

# Setting Properties

By selecting the Event Dispatcher in the **My Blueprint** panel, you can edit its properties in the **Details** panel. You can set the tooltip and category for your Event Dispatcher, as well as add inputs.

Adding inputs to your Event Dispatcher allows you to send variables to each event bound to your Event Dispatcher. This allows data flow not only within your Blueprint Class, but also between your Blueprint Class and the Level Blueprint.

The process to add inputs to your Event Dispatcher is similar to the workflow for adding inputs and outputs to functions, custom events, and macros. If you would like to use the same inputs as another event, you can use the **Copy Signature from** dropdown to indicate the event. To add your own inputs to the Event Dispatcher:

1. Click **New** in the **Inputs** section of the **Details** pane.
2. Name the new input and set its type using the dropdown menu. In this example, there is a String input parameter named **MyStringParam**.
3. You can also set a default value and indicate whether or not to pass the parameter by reference by expanding the entry for the parameter.



To change the location of the pin for this parameter on the edge of the node, use the up and down arrows in the expanded **Details** pane entry.

## Using Event Dispatchers

After creating the Event Dispatcher, you can add event nodes, bind nodes, and unbind nodes linked to it. Although you can double-click on the Event Dispatcher entry in the **My Blueprint**

tab to open the Event Dispatcher's graph, the graph is locked and you cannot modify the Event Dispatcher directly. The bind, unbind, and assign methods enable you to add events to the Event Dispatcher's event list, while the call method activates all the events stored in the event list.

Event, bind, and unbind nodes can be added in both the Blueprint Class and the Level Blueprint. Except for the Event node, each node has a **Target** input pin:

- In the Blueprint Class, this pin is automatically set to **Self**. This means that the event list is changed for the class, so every instance of the class will be changed.
- In the Level Blueprint, this pin must be connected to a reference to an instance of the class in the level. This means that the event list will only be changed for that particular instance of the class. The [Level Blueprint documentation](#) explains how to create any **Actor** references you might need.



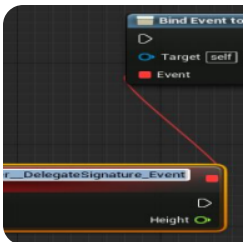
### Binding and Unbinding Events

Adding and removing events from an Event Dispatcher's events list.



### Calling Event Dispatchers

Calling the Event Dispatcher executes all of the currently bound events in the events list.



### Creating Dispatcher Events

Creating events that can be bound and added to the Event Dispatcher's events list.