Actor Communication

Overview of the different methods of Actor Communication.

Choose your implementation method

Blueprints C++

Blueprint scripting and C++ provide several ways to communicate and share information between Actors. This page gives you an overview of the different Actor communication methods available, as well as requirements and common use cases for each.

You will also find links to more detailed Quick Start guides for each Actor communication type.

Direct Communication

Direct Actor communication is the most common method of sharing information between Actors in your Level.

This method requires a reference to the target Actor so you can access its information from your working Actor. This communication type uses a one-to-one relationship between your working Actor and your target Actor.

When to Use It

Use this communication method if you have a reference to the specific Actor in your Level and you need to share information or trigger functionality within that specific Actor.

Examples

- Triggering an event on an Actor.
- Getting information from an Actor in your Level.

Casting

Casting is a common communication method where you take a reference to an Actor and try to convert it to a different class. If this conversion is successful, then you can use Direct communication to access its information and functionality.

This method requires a reference to the Actor in your Level as you can use the **Cast** node to try to convert it to a specific class. This communication method uses a one-to-one relationship between your working Actor and your target Actor.

When to Use It

Use this communication method if you have a reference to an Actor and want to check if the Actor is of a certain class to access its information.

Examples

- Using a volume to overlap all Pawns and cast the Pawn reference to a particular subclass, such as a Character, to access its information.
- Using an Actor's reference to cast to a common parent class and access its information.

Interfaces

Interfaces define a set of common behaviors or capabilities that can be implemented by different Actor classes. This communication method simplifies the process of implementing the same type of functionality on different Actor classes.

This method requires each Actor to implement the interface in order to access its common functions. You also need a reference to the Actor so you can call the interface function using that reference. This communication method uses a one-to-one relationship between your working Actor and your target Actor.

When to Use It

Use this method when you want to create common functionality that applies to different types of Actors.

Examples

- Creating an interaction system where each Actor does something different when the player interacts. For example, a door Actor will open when the player interacts with it, while a light Actor will activate when the player performs the same interaction.
- Applying damage to different Actors in your Level. Each Actor can react differently to the damage taken. For example, a wall Actor could break when taking damage, while a door Actor could open after taking damage.

Event Dispatchers

Event Dispatchers are a communication method where one Actor triggers an event and other Actors that are listening to that event get notified.

This method requires you to create Event Dispatchers on the working Actor and have your target Actors bind to them. This communication method uses a one-to-many relationship, where a single working Actor triggers the Event Dispatcher for many listening Actors.

When to Use It

Use this communication method when you want a single event to influence several different Actors.

Examples

- Creating a BossDied event in your game. Once a boss enemy dies, the event will trigger, causing a door Actor to open, the HUD Actor to display a message, and a chest Actor to open.
- Creating a day / night cycle where the DayStarted event tells your NPCs that they must start their daily routines.

Actor Communication Reference Table

Communication Type	When to Use It	Requirements	Example
Direct Communication	When communicating with a specific instance of an Actor in your Level.	Need a reference to the Actor in your Level.	Triggering an event on a specific Actor in your Level.
Casting	When you want to verify an Actor is of a certain class to access its properties.	Need a reference to an Actor in your Level to cast to the desired Actor class.	Accessing specific functionality of child Actors that share the same parent class.
Interfaces	When adding the same functionality to different Actor classes.	Need a reference to the Actor in your Level and the Actor needs to implement the interface.	Adding an interaction behavior to different types of Actors.

Communication Type	When to Use It	Requirements	Example
Event Dispatchers	When triggering an event from one Actor to many Actors.	Actors need to subscribe to the event to react to it.	Notifying many different types of Actors that an event has triggered.

Blueprint scripting and C++ provide several ways to communicate and share information between Actors. This page gives you an overview of the different Actor communication methods available, as well as requirements and common use cases for each.

You will also find links to more detailed Quick Start guides for each Actor communication type.

Direct Communication

Direct Actor communication is the most common method of sharing information between Blueprint Actors in your Level.

This method requires a reference to the target Actor class Blueprint so you can access its information from your working Actor. This communication type uses a one-to-one relationship between your working Actor and your target Actor.

When to Use It

Use this communication method if you have a reference to the specific Actor class Blueprint in your Level and you need to share information or trigger functionality within that specific Actor class Blueprint.

Examples

- Triggering an event on an Actor.
- Getting information from an Actor in your Level.

Casting With Blueprint Classes

Casting is a common communication method where you take a reference to an Actor class Blueprint and try to convert it to a different class. If this conversion is successful, then you can use Direct communication to access its information and functionality.

This method requires a reference to the Actor class Blueprint in your Level as you can use **casting** to try to convert it to a specific class. This communication method uses a one-to-one relationship between your working Actor and your target Actor.

When to Use It

Use this communication method if you have a reference to an Actor class Blueprint and want to check if the Actor class Blueprint is of a certain class to access its information.

Examples

- Using an AVolume to overlap all APawns and cast the Pawn reference to a particular subclass, such as an ACharacter, to access its information.
- Using an Actor class Blueprint's reference to cast to a common parent class and access its information.

Interfaces

Interfaces (UInterface) define a set of common method behaviors that can be implemented by different classes. This communication type simplifies the process of implementing the same type of functionality on different Actor class Blueprints.

This method requires each Actor to implement the interface in order to access its common methods. You will require a reference to the Actor class Blueprint so you can call the interface function using that reference. This communication method uses a one-to-one relationship between your working Actor and your target Actor.

Note: See Interfaces for additional documentation.

When to Use It

Use this method when you want to create common functionality that applies to different types of Blueprint Actors.

Examples

- Creating an interaction system where each Actor class Blueprint does something different when the player interacts. For example, a door Actor will open when the player interacts with it, while a light Actor will activate when the player performs the same interaction.
- Applying damage to different Actors in your Level. Each Actor class Blueprint can react differently to the damage taken. For example, a wall Actor could break when taking damage, while a door Actor could open after taking damage.

Delegates

Delegates can call methods in Actor class Blueprints in a type-safe way. A delegate can be bound dynamically to form a relationship where one Actor triggers an event on another Actor "listening" to be notified for that event.

Note: See Delegates for additional documentation.

When to Use It

Use this communication type when you want a single event to influence several different Actor class Blueprints.

Examples

- Creating a BossDied event in your game. Once a boss enemy dies, the event will trigger, causing a door Actor to open, the UMG Blueprint to display a message, and a chest Actor to open.
- Creating a day / night cycle where the DayStarted event tells your NPCs that they must start their daily routines.

Communicating with Blueprint Classes Reference Table

Communication Type	When to Use It	Requirements	Example
Direct Communication	When communicating with a specific instance of an Actor in your Level.	Need a reference to the Actor in your Level.	Triggering an event on a specific Actor in your Level.
Casting	When you want to verify an Actor is of a certain class to access its properties.	Need a reference to an Actor in your Level to cast to the desired Actor class.	Accessing specific functionality of child Actors that share the same parent class.
Interfaces	When adding the same functionality to different Actor classes.	Need a reference to the Actor in your Level and the Actor needs to implement the interface.	Adding an interaction behavior to different types of Actors.
Event Dispatchers	When triggering an event from one Actor to many Actors.	Actors need to subscribe to the event to react to it.	Notifying many different types of Actors that an event has triggered.

Quick Start Guides

Casting Quick Start Guide

Quick Start for the Casting communication method.

Direct Actor Communication Quick Start Guide

Quick Start for the Direct Actor Communication method.

Event Dispatchers and Delegates Quick Start Guide

Quick Start for the Event Dispatcher and Delegate communication method.

Interface Quick Start Guide

Quick Start for the Interface communication method.