Actor Component Replication

Learn how to replicate actor-owned components.

An actor component extends an actor's behavior. An actor component is a special type of object that you can attach to an actor as a subobject. Actor components do not replicate by default, but you can configure any actor component to replicate as part of its owning actor. Actor components can replicate their own properties and subobjects, as well as call actor component class-defined remote procedure calls (RPCs) in the same way actors can.

To replicate an actor component as part of your actor, you must make sure that:

- The actor that owns your actor component is set to replicate.
- The actor component is set to replicate.

Types of Actor Components

Static Actor Components

A *static actor component* is an actor component spawned when the owning actor is spawned. Static components are created in an actor's C++ constructor as a default subobject or in the Blueprint Editor's Component Mode.

Replicate a Static Actor Component

To replicate an actor component created in your actor constructor, follow these steps:

1. In your actor constructor:
    - Set your actor to replicate with `bReplicates = true;`.
    - Create your actor component in your actor constructor with `CreateDefaultSubobject<T>`:

```
AMyActor::AMyActor()
{
bReplicates = true;
MyActorComponent = CreateDefaultSubobject<UMyActorComponent>
(TEXT("MyActorComponent"));
}
```

Copy full snippet

2. In you actor component constructor:
    - Set your actor component to replicate with `UActorComponent::SetIsReplicatedByDefault`:

```
UMyActorComponent::UMyActorComponent()
{
SetIsReplicatedByDefault(true);
}
```

Copy full snippet

Dynamic Actor Components

A *dynamic actor component* is an actor component spawned on the server at runtime. The creation or deletion of a dynamic actor component replicates to connected clients. Dynamic actor components work similarly to actors.

Clients can spawn their own, local, non-replicating dynamic actor components.

Replicate a Dynamic Actor Component

To replicate an actor component created dynamically at runtime, follow these steps:

1. In your actor constructor:
    - Set your actor to replicate with `bReplicates = true;`.
    - Create your actor component in your gameplay code with `NewObject<T>`:

```
MyActorComponent = NewObject<UMyActorComponent>();
```

Copy full snippet

2. When you want to replicate your new actor component:
    - Set your actor component to replicate with `UActorComponent::SetIsReplicated`:

```
if (MyActorComponent)
{
MyActorComponent->SetIsReplicated(true);
}
```

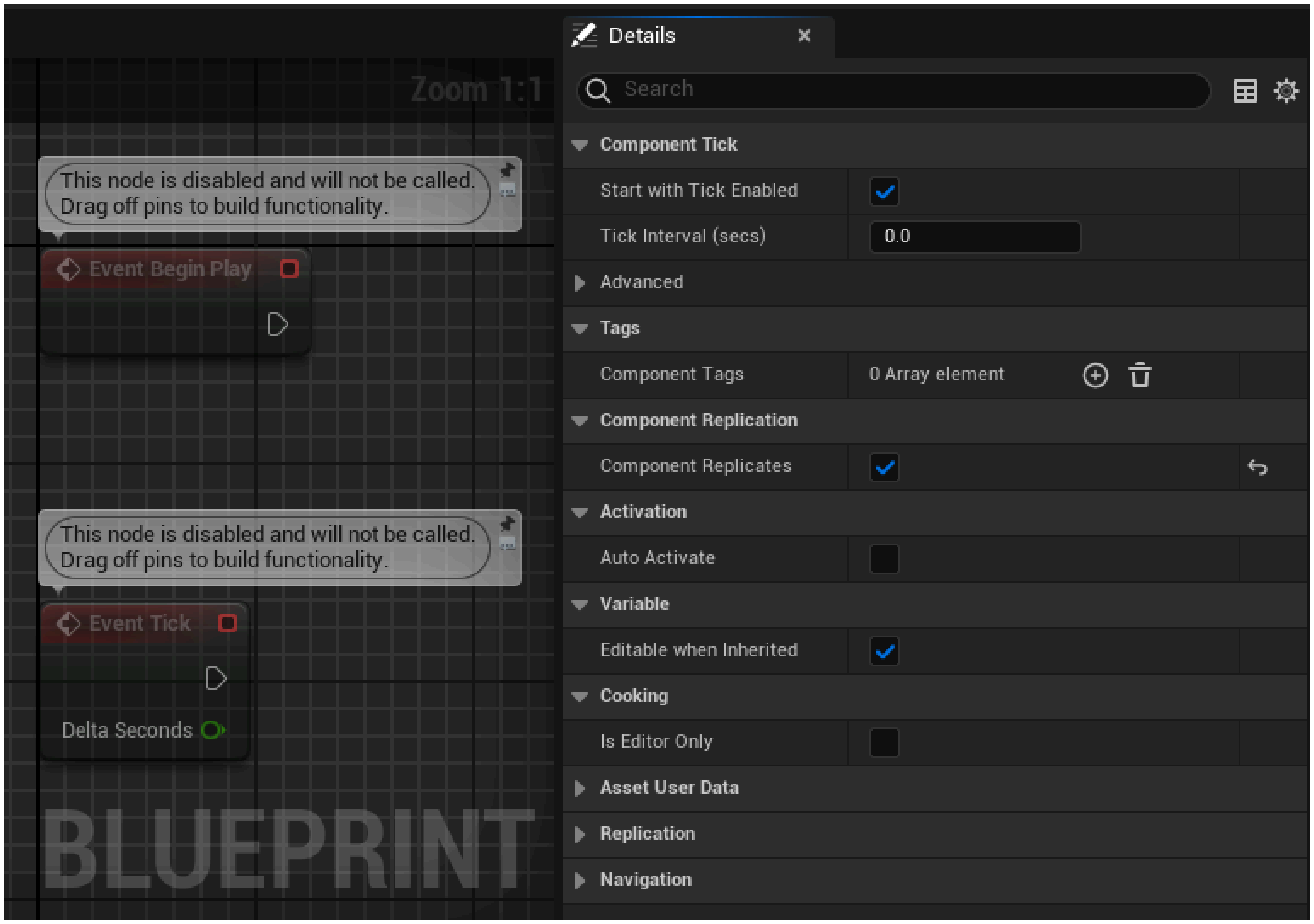
Copy full snippet

Blueprint Actor Components

You can spawn both static and dynamic actor components in Blueprint.

Replicate a Static Blueprint Actor Component

To replicate a static actor component in Blueprint, toggle the **Replicates** boolean field in your actor component's **Details Panel**. You only need to replicate an actor component if the component has properties or events that you need to replicate.



This node is disabled and will not be called.
Drag off pins to build functionality.

This node is disabled and will not be called.
Drag off pins to build functionality.

- Component Tick
  - Start with Tick Enabled: ✓
  - Tick Interval (secs): 0.0
  - Advanced
- Tags
  - Component Tags: 0 Array element
- Component Replication
  - Component Replicates: ✓
- Activation
  - Auto Activate: ☐
- Variable
  - Editable when Inherited: ✓
- Cooking
  - Is Editor Only: ☐
- Asset User Data
- Replication
- Navigation

You can set an actor component to replicate by default in the Component Replication section of the Details Panel.

The **Component Replication** section only appears on components that support some form of replication.

Replicate a Dynamic Blueprint Actor Component

To replicate a dynamic actor component in Blueprint, call the **Set Is Replicated** function with the **Should Replicate** field toggled on.

Copy code

Replicate Actor Component Properties

You can replicate actor component properties the same way that you replicate actor properties. For more information about how to replicate actor properties, see the Replicate Actor Properties documentation.

Actor Component Remote Procedure Calls

You can define remote procedure calls (RPCs) within your actor component class and call them the same way that you call actor RPCs. For more information about how to define, implement, and call RPCs, see the Remote Procedure Calls documentation.

Replicate Actor Component Subobjects

Actor Components can have their own replicated subobjects list in the same way as actors. They use the same API as actors for registering and unregistering their subobjects. Subobjects within an actor component can have replication conditions as well.

The owning component must be replicated to a connection before the conditions of its replicated subobjects are checked. For example, if a subobject has the `COND_OwnerOnly` condition, but is registered to a component that uses the `COND_SkipOwner` condition, the subobject never replicates, because the owner is skipped..

For more information on how to replicate subobjects, see the Replicate Actor Subobjects documentation.

Bandwidth Overhead

Each replicating actor component within an actor adds:

- A Network Globally Unique Identifier (NetGUID) header consisting of 4 bytes.
- All replicated properties and the space required.
- A footer consisting of approximately 1 byte.

When considering bandwidth overhead, there are three areas to be aware of:

- *Replication*: Compared to replicating an entire actor, the impact of replicating a property on an actor component is relatively low.

- *Calling an RPC*: Calling an RPC from an actor component has more overhead than calling from directly within an actor. To mitigate this, consider routing your actor component RPCs through your actor. For an example of this, see the character movement component.
- *Amount of actor components*: Actor components are relatively small. However, if you use a high number of components and component subobjects, your performance might be lower.