Types of Blueprints

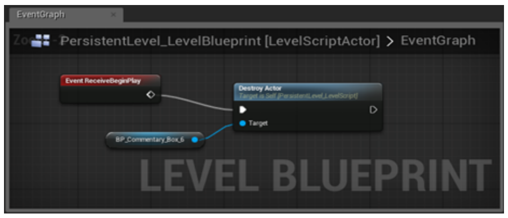
Landing page for information on different types of Blueprints.

Sometimes it is difficult to tell which type of **Blueprint** you should use at first glance, especially when it comes to **Blueprint Macro Libraries** and **Blueprint Classes**. A good rule of thumb is to ask yourself:

- *Are there multiple instances?*

If you will have more than one or two instances (e.g., a TV set that can be shot or turned on/off), then it probably makes sense to create a Blueprint Class, with associated code living there. If not, and you just want to have some helper functions (like find all Actors within X units), those are ideal to live in a Blueprint Macro Library.

Level Blueprint



A **Level Blueprint** is a specialized type of **Blueprint** that acts as a level-wide global event graph. Each level in your project has its own Level Blueprint created by default that can be edited within the Unreal Editor, however new Level Blueprints cannot be created through the editor interface.

Events pertaining to the level as a whole, or specific instances of Actors within the level, are used to fire off sequences of actions in the form of Function Calls or Flow Control operations. Those familiar with Unreal Engine 3 should be very familiar with this concept as this is very similar to how Kismet worked in Unreal Engine 3.

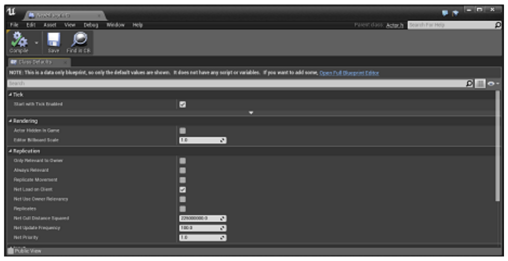
Level Blueprints also provide a control mechanism for level streaming and [Sequencer] as well as for binding events to Actors placed within the level.
For more information on this section, please see [Level Blueprint].

Blueprint Class

A **Blueprint Class**, often shortened as **Blueprint**, is an asset that allows content creators to easily add functionality on top of existing gameplay classes. Blueprints are created inside of Unreal Editor visually, instead of by typing code, and saved as assets in a content package. These essentially define a new class or type of Actor which can then be placed into maps as instances that behave like any other type of Actor.

For more information on this section, please see [Blueprint Class].

Data-Only Blueprint



NOTE: This is a data only blueprint, so only the default values are shown. It does not have any script or variables. If you want to add some, Open Full Blueprint Editor

Tick
Start with Tick Enabled ✓

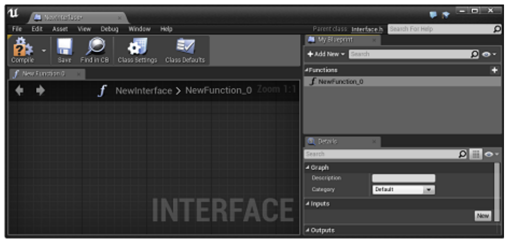
Rendering
Actor Hidden In Game
Editor Billboard Scale 1.0

Replication
Only Relevant to Owner
Always Relevant
Replicate Movement
Net Load on Client ✓
Net Use Owner Relevancy
Replicates
Net Cull Distance Squared 225000000.0
Net Update Frequency 100.0
Net Priority 1.0

A **Data-Only Blueprint** is a Blueprint Class that contains only the code (in the form of node graphs), variables, and components inherited from its parent. These allow those inherited properties to be tweaked and modified, but no new elements can be added. These are essentially a replacement for archetypes and can be used to allow designers to tweak properties or set items with variations.

Data-Only Blueprint are edited in a compact property editor, but can also be "converted" to full Blueprints by simply adding code, variables, or components using the full **Blueprint Editor**.
For more information on this section, please see Data-Only Blueprint.

Blueprint Interface



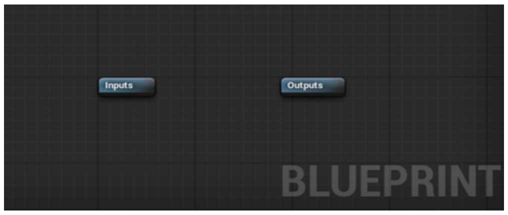
A **Blueprint Interface** is a collection of one or more functions - name only, no implementation - that can be added to other Blueprints. Any Blueprint that has the Interface added is guaranteed to have those functions. The functions of the Interface can be given functionality in each of the Blueprints that added it. This is essentially like the concept of an interface in general programming, which allows multiple different types of Objects to all share and be accessed through a common interface. Put simply, Blueprint Interfaces allow different Blueprints to share with and send data to one another.

Blueprint Interfaces can be made by content creators through the editor in a similar fashion to other Blueprints, but they come with certain limitations in that they cannot:

- Add new variables
- Edit graphs
- Add Components

For more information on this section, please see Blueprint Interface.

Blueprint Macro Library



A **Blueprint Macro Library** is a container that holds a collection of **Macros** or self-contained graphs that can be placed as nodes in other Blueprints. These can be time-savers as they can store commonly used sequences of nodes complete with inputs and outputs for both execution and data transfer.

Macros are shared among all graphs that reference them, but they are auto-expanded into graphs as if they were a collapsed node during compiling. This means that Blueprint Macro Libraries do not need to be compiled. However, changes to a Macro are only reflected in graphs that reference that Macro when the Blueprint containing those graphs is recompiled.

For more information on this section, please see Blueprint Macro Library.