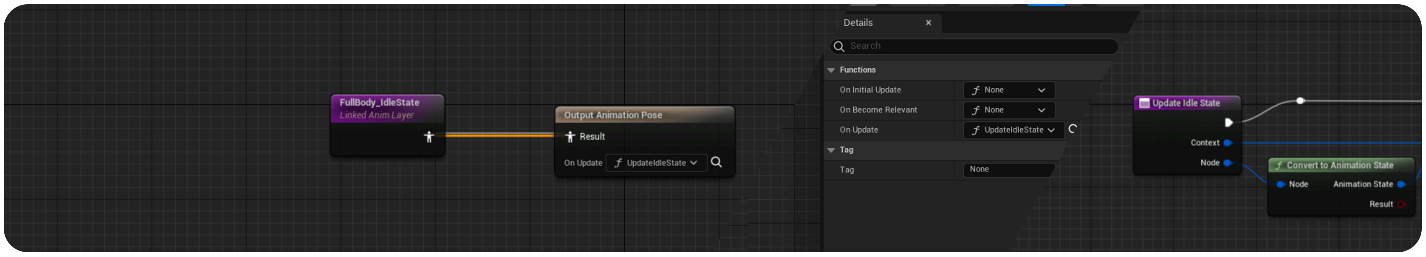
Animation Node Functions

Using Animation Node Functions.

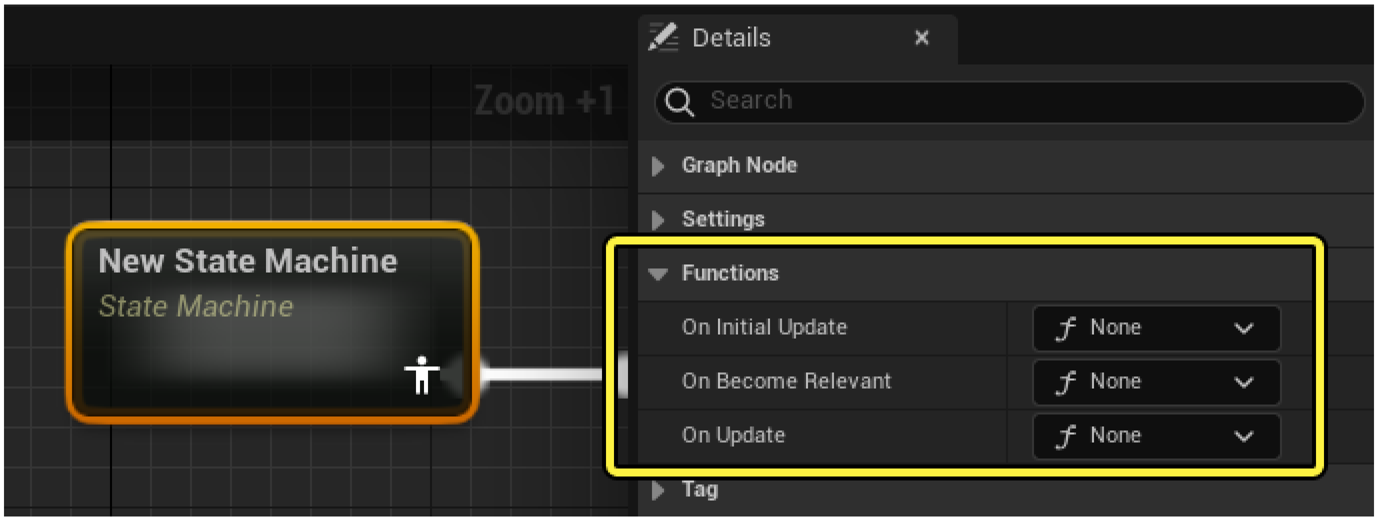


Animation Node Functions are function graphs that you can bind to specific Animation Blueprint nodes, at set points in your graph's **update** cycle, to perform relevant logic. You can use Animation Node Functions to set reference variables, determine a dynamic value, set animation states, and organize complex graphs. Additionally, by using Animation Node Functions, logic is only run at set points in the graph's evaluation, which can significantly increase your animation system's performance.

You can use this document to learn more about how to use Animation Node Functions in Unreal Engine.

Create a new Anim Node Function

To create a new Animation Node Function, select a node in the AnimGraph, and navigate to the node's **Details** panel to the **Functions** section.



Functions

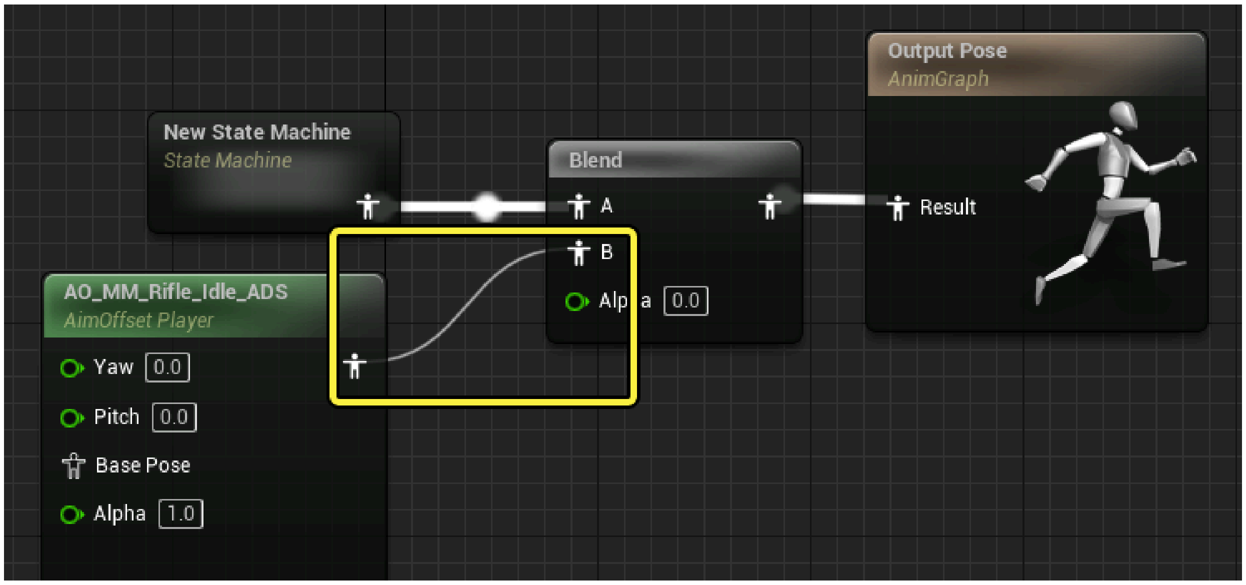
On Initial Update	None
On Become Relevant	None
On Update	None

You can choose to create a new function on any of the available bindings, depending on your project's needs. Here you can reference a list and description of each of the available functions:

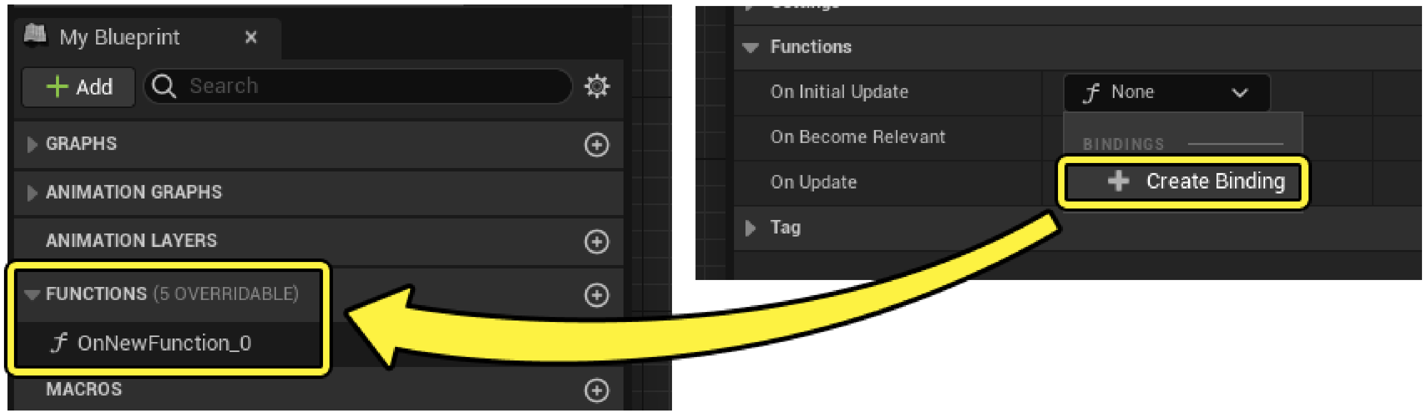
Function	Description
On Initial Update	The engine calls logic bound to this function's graph before updating the Animation Blueprint Node for the first time. You can use this function to set constants for the node that will not change, such as component references or static values.
On Become Relevant	The engine calls logic bound to this function's graph every time the node becomes relevant in the graph. You can use this function to set dynamic values the node requires, but will not update while the node is evaluating.
On Update	The engine calls logic bound to this function's graph every tick when it updates the node. You can use this function to set dynamic values the node requires during its update. For an example workflow using the **On Update** function binding, see the Distance Matching documentation.

The concept of **relevance** in the **AnimGraph** refers to whether or not the engine is evaluating a node. In cases when nodes are not being evaluated, such as when using Blend nodes or State Machines, some nodes may be completely inactive. When this occurs, the node is not relevant. Only nodes currently contributing to the **Output Pose** are considered relevant.

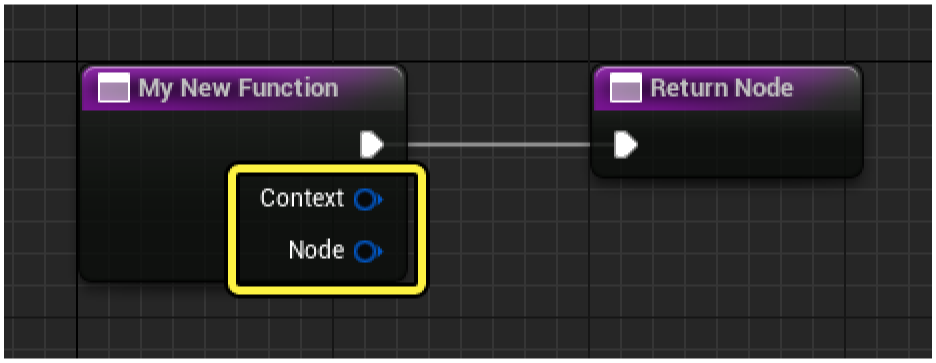
In this example, the Aim Offset node is not relevant because the Blend node is blended completely to input A.



After selecting when to bind a new function to a node's evaluation cycle, use the (**+**) **Create New Binding** option from the binding's drop-down menu to create a new function graph. The function graph will then appear in the **My Blueprints** panel, where you can name and open the function.



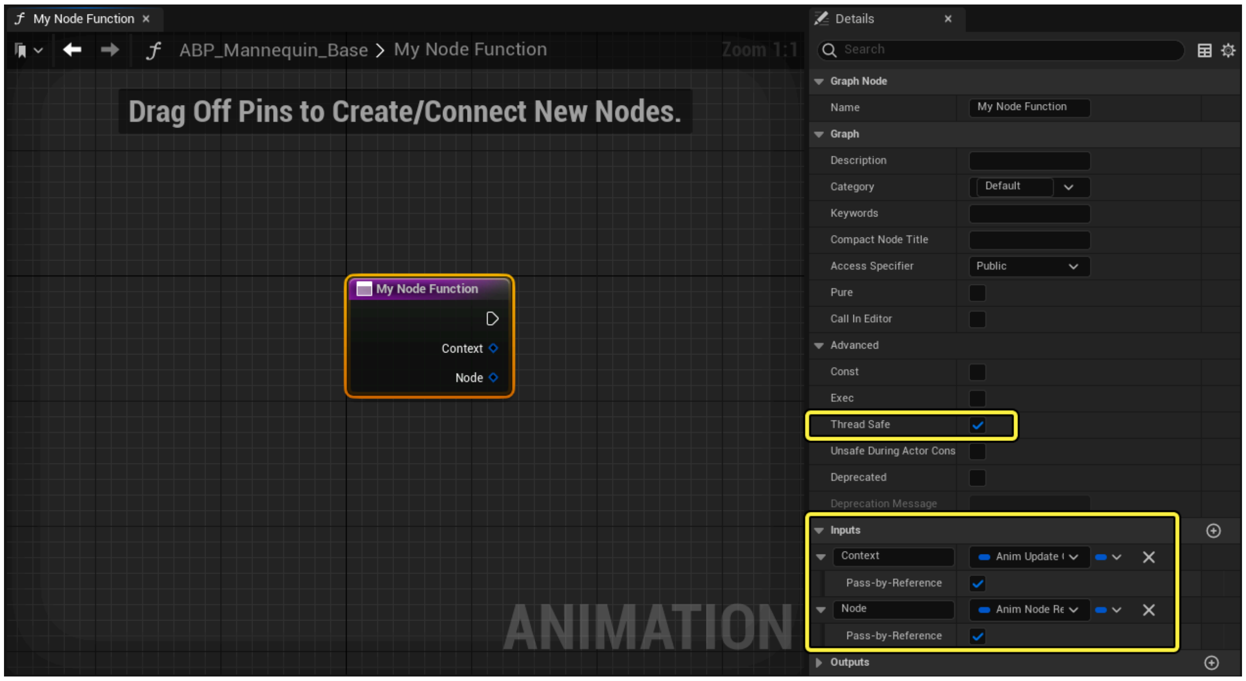
New Animation Node Functions will automatically create **Input** pins on the **Function** node which are used to pass through data from the bound Animation Blueprint node to the function's graph. Some function operations may not require the use of these pins, but other logic may require the data they provide, such as if you are using the function to read the current state of the node.



Input Pin	Description
Context	Allows the node to pass data through that is relevant to the node, such as Delta Time, or Inertialization requests.
Node	Allows the node to pass itself through this pin. Typically you will want to convert this pin to a specific node type using a Convert function.

You can also bind an existing function to an Animation Blueprint node, as long as it meets the following requirements:

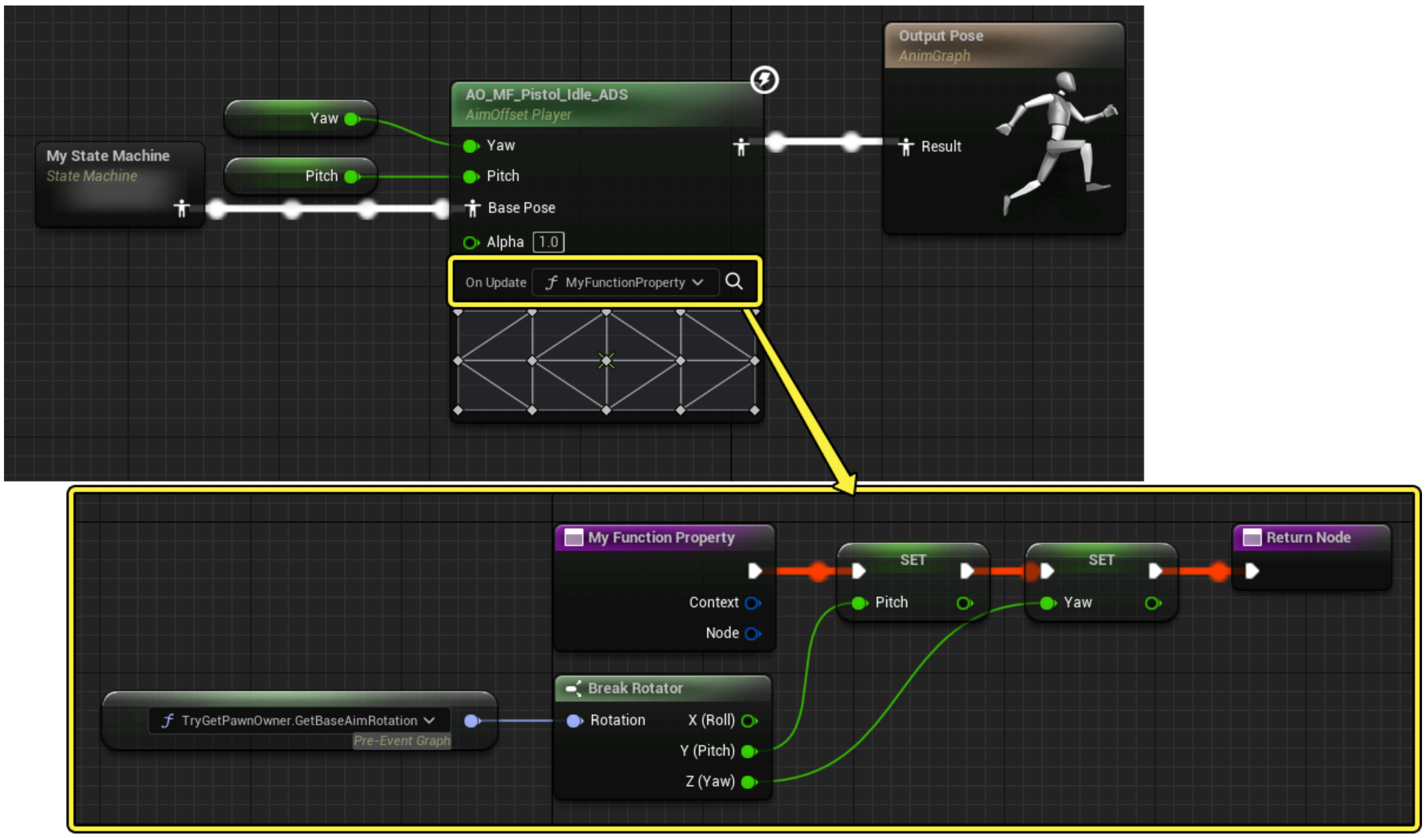
- The function's Thread Safe property is enabled.
- The function must contain two **Inputs**. The first being an **Anim Update Context Reference** type, and the second being an **Anim Node Reference** type. Each of these inputs must also have their **Pass-by-Reference** property enabled.



Name: My Node Function
Description
Category: Default
Keywords
Compact Node Title
Access Specifier: Public
Pure
Call In Editor
Const
Exec
Thread Safe ✓
Unsafe During Actor Cons
Deprecated
Deprecation Message
Context: Anim Update
Pass-by-Reference ✓
Node: Anim Node Re
Pass-by-Reference ✓

After creating or binding a new function to the Animation Blueprint node, the function will appear on the node in the AnimGraph.

In this example, the Aim Offset logic, for getting a character's rotation and setting the pitch and yaw values, is all contained within the function. The engine only executes this logic while updating the Aim Offset node in the AnimGraph, rather than every tick in the Event Graph, significantly reducing performance costs.



You can open an Animation Node Function directly from the associated Animation Blueprint node using the **Magnifying Glass** icon next to the function in the graph.

Graphing in Animation Node Functions

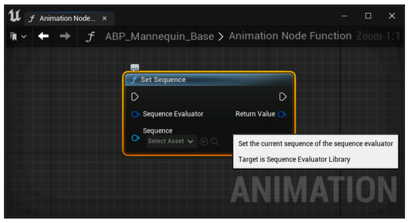
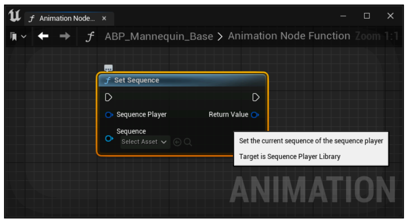
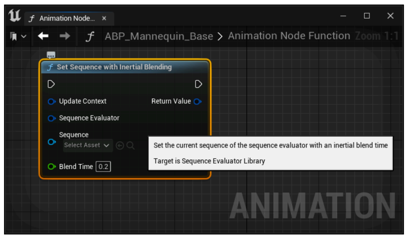
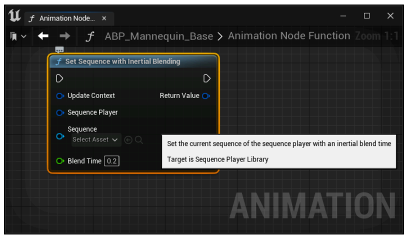
Graphing logic in Animation Node Functions is similar to any other function graph in Unreal Engine.

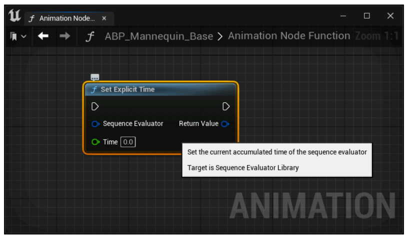
If you want to further increase your project's performance you can implement Animation Optimzation techniques, such as Property Access, to ensure the engine offloads the Animation Node Function onto the Worker Thread the animation update is executing on.

Sequence Player Nodes

When binding an Animation Node Function to a **Sequence Player** or **Sequence Evaluator** node, you can use **Sequence Player** nodes within the function to interface and play Animation Sequences directly using the Animation Node Function, providing gameplay teams with more control over animation playback.

Here you can reference a list of the Sequence Player nodes and a description of their function.

Name	Image	Description
 Set the current sequence of the sequence evaluator Target is Sequence Evaluator Library	**Set Sequence** (**Evaluator Library**)	Set the current Animation Sequence to be played by the connected **Sequence Evaluator** node.
 Set the current sequence of the sequence player Target is Sequence Player Library	**Set Sequence** (**Player Library**)	Set the current Animation Sequence to be played by the connected **Sequence Player** node.
 Set the current sequence of the sequence evaluator with an inertial blend time Target is Sequence Evaluator Library	**Set Sequence with Inertial Blending** (**Evaluator Library**)	Set the current Animation Sequence to be played by the connected **Sequence Evaluator** node using an **Inertial Blend** with the specified duration.
 Set the current sequence of the sequence player with an inertial blend time Target is Sequence Player Library	**Set Sequence with Interior Blending** (**Player Library**)	Set the current Animation Sequence to be played by the connected **Sequence Player** node using an **Inertial Blend** with the specified duration.

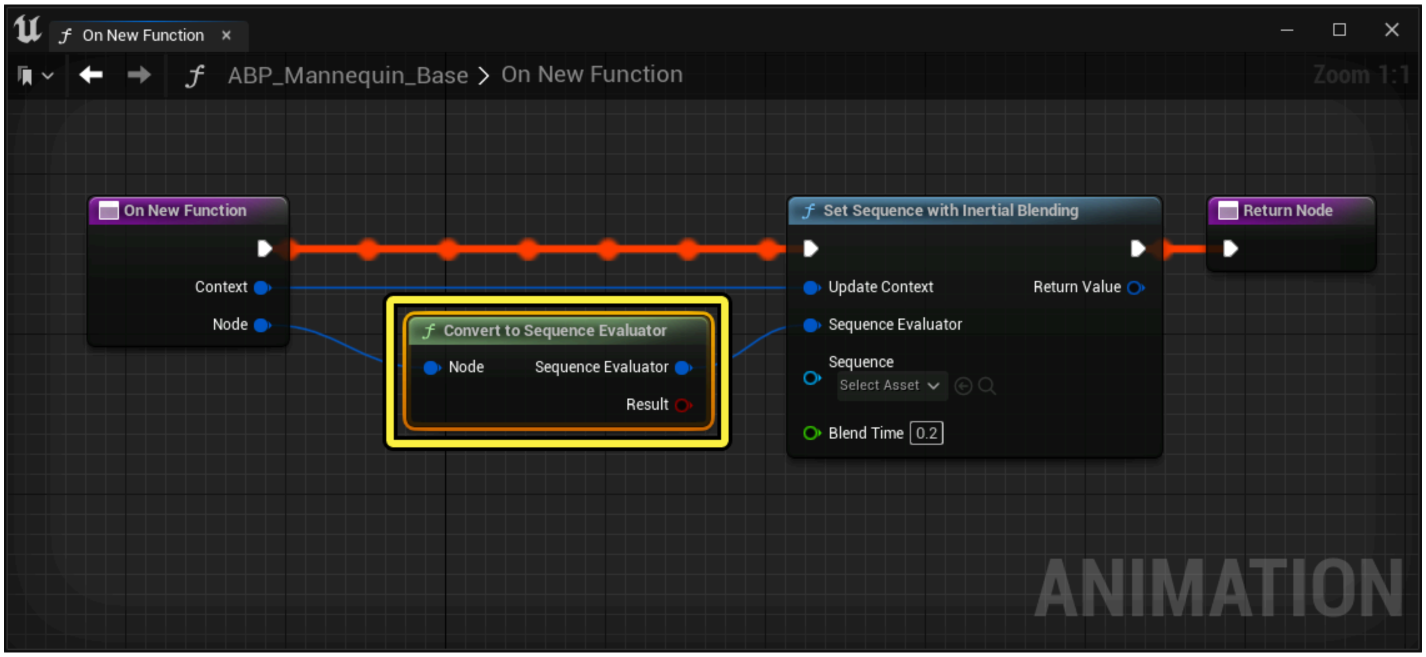
Name	Image	Description
Set Explicit Time (**Evaluator Library**)	 Set the current accumulated time of the sequence evaluator Target is Sequence Evaluator Library	Set the current accumulated time of the connected **Sequence Evaluator** node.

After creating a Sequence Player node in your animation function, and adding it to the graph, you will need to use a **Convert** node to provide the player node with the proper data.

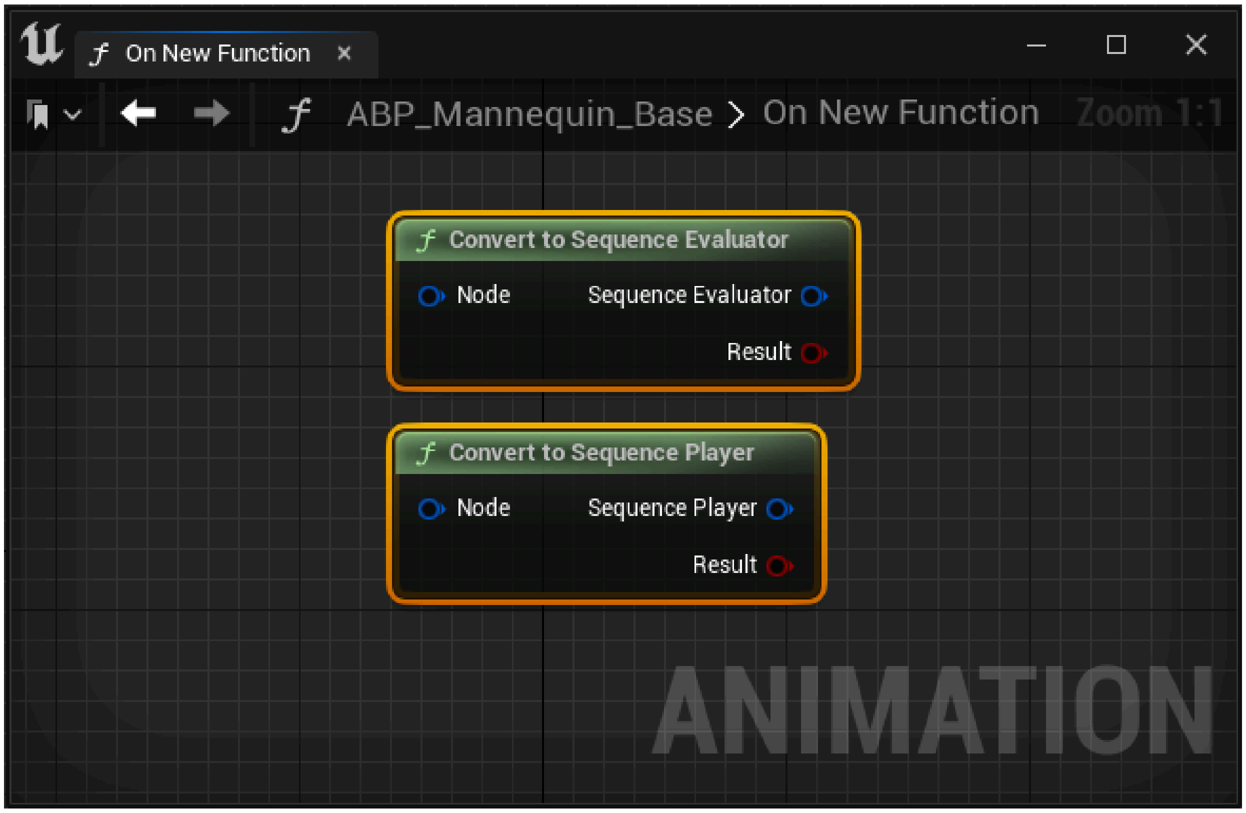
Convert to Sequence Player and Evaluator Nodes

You can use Convert nodes to pass through data from the context Sequence Player and Evaluator Nodes to your Animation Node Function, in order to play animations using the Animation Blueprint Function.

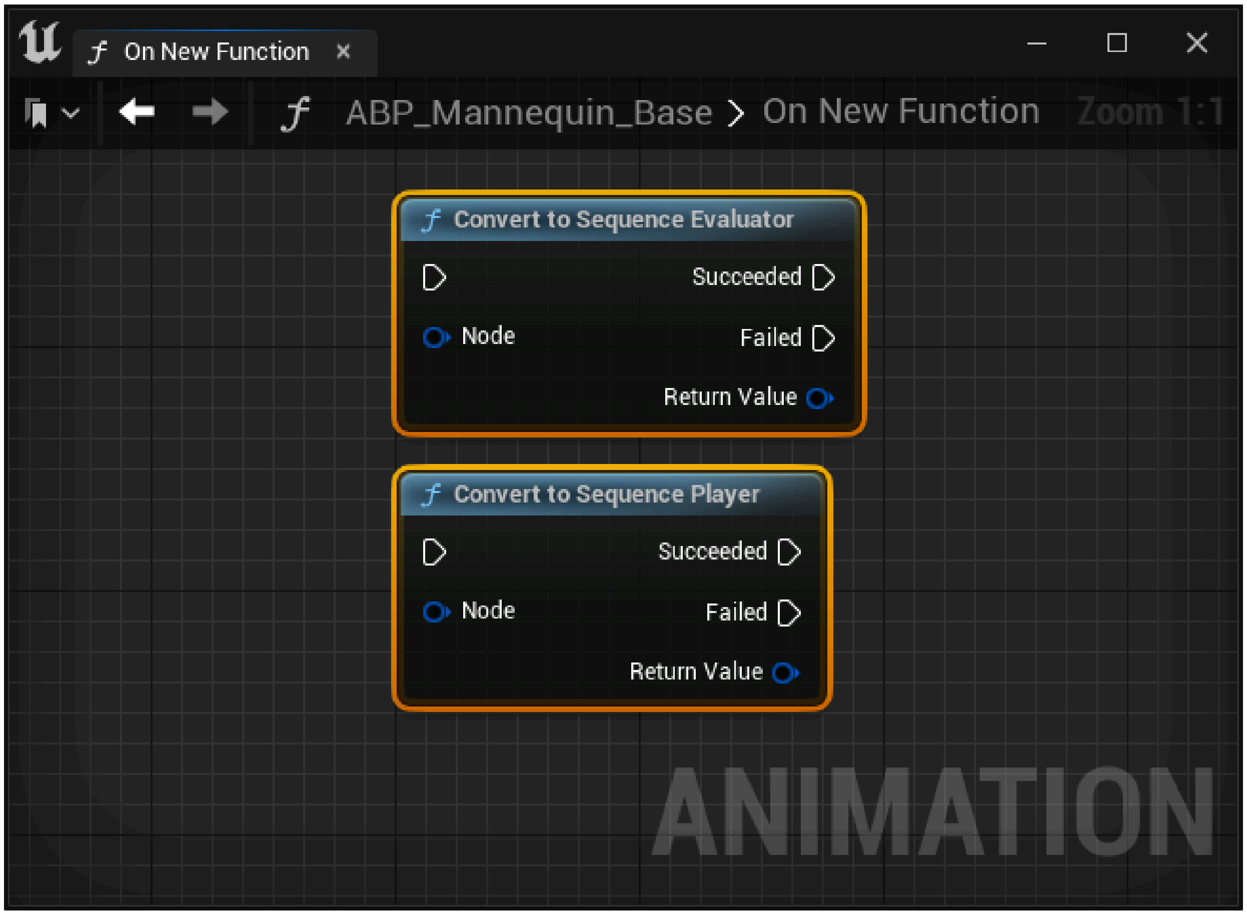
After adding a Sequence Player node to your Animation Node Function graph, add a Convert node from the Function node's Node pin, and connect the output to the Player node's **Sequence Player** or **Sequence Evaluator** input.



When building an Animation Node Function directly on a Sequence Evaluator or Player node, you can use the pure function Convert nodes (green).


On New Function
ABP_Mannequin_Base > On New Function
Zoom 1:1
Convert to Sequence Evaluator
Node
Sequence Evaluator
Result
Convert to Sequence Player
Node
Sequence Player
Result
ANIMATION

When creating an Animation Node Function on different node types, for example a state machine, you must use the function convert nodes (blue) to properly convert the node to a Sequence Player or Evaluator.

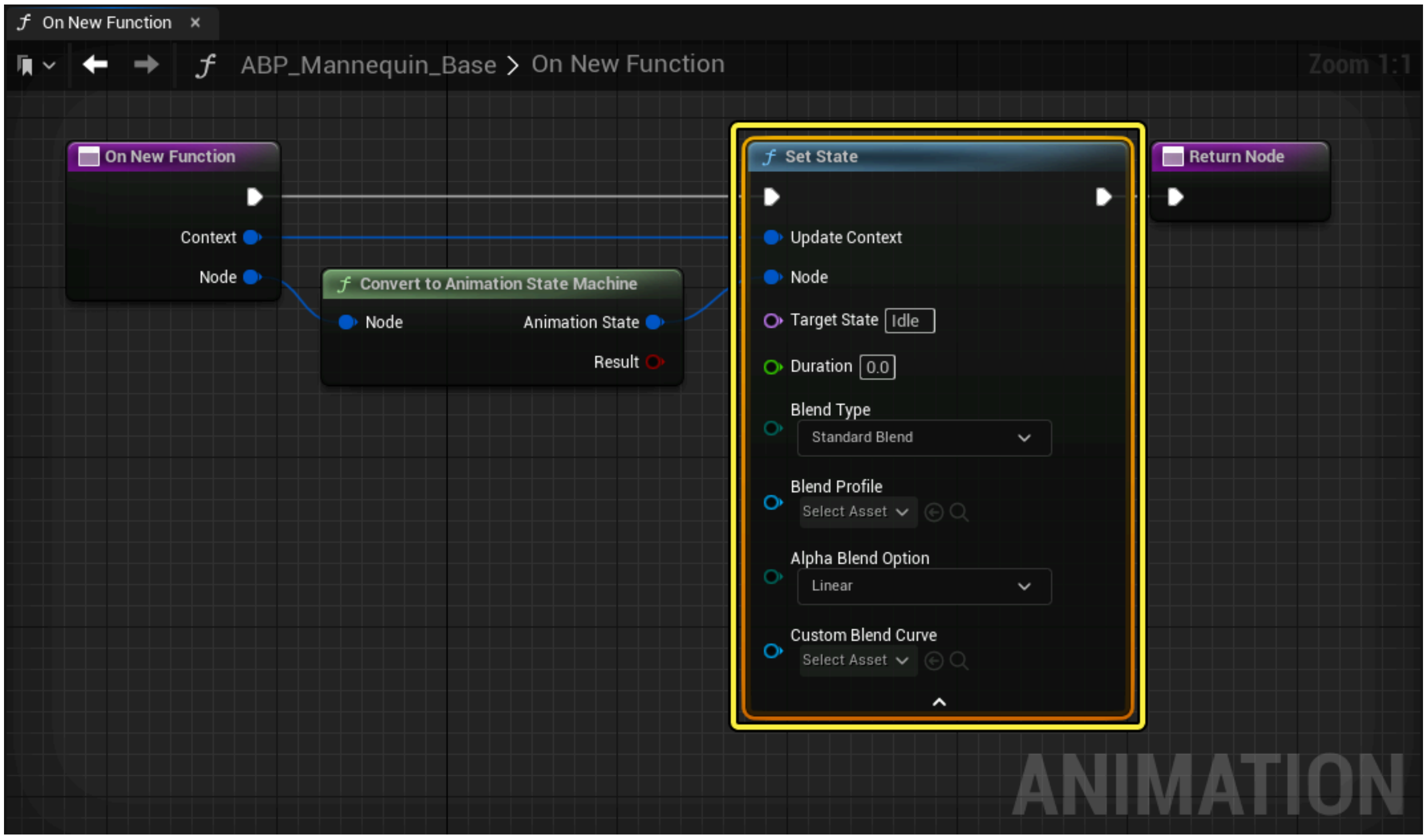


State Machines

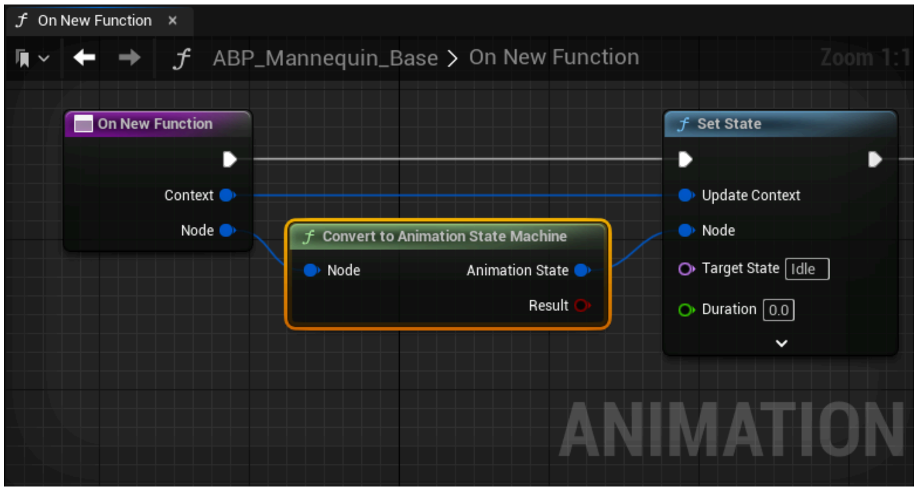
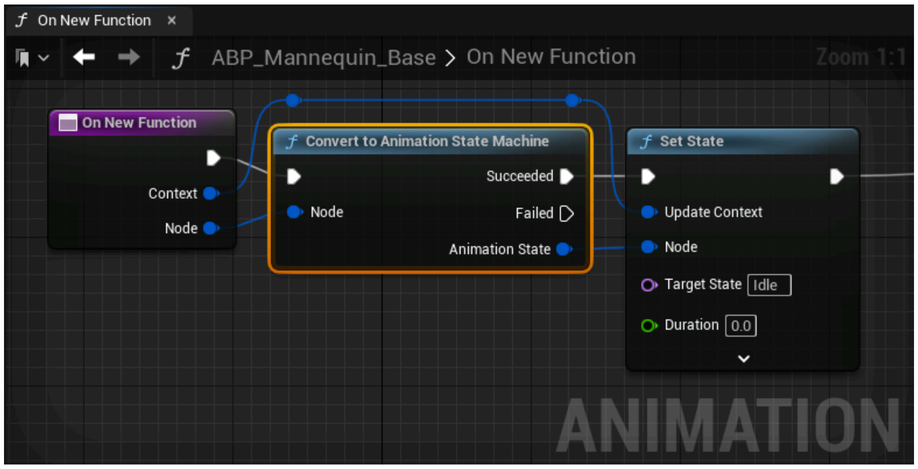
You can create Animation Blueprint Function Nodes on State Machine nodes, and use them to set Animation States of the State Machine. For example, you can use an On Update function with a **Set State** node to set an Animation State Machine State without the need to set up transition logic.

Set State Node

You can use Set State nodes to directly set an animation state using an Animation Node Function, without the need to set up transition logic. Here, you can reference a list of the Set State node's properties and a description of their function.



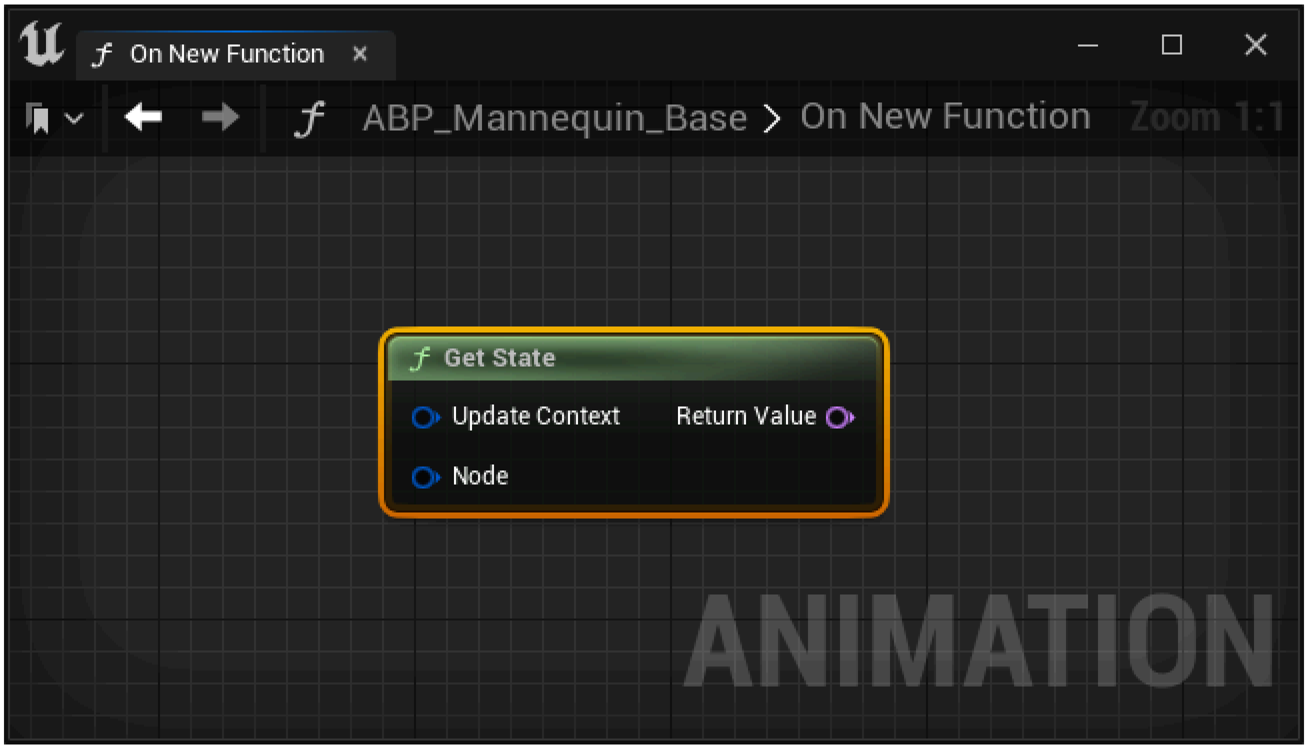
Property	Description
Update Context	Connect the Function node's Context pin to provide the Set State node with the requisite context for its operation, such as the delta time, and the current place in AnimGraph.

Property	Description
Node	Set the node reference for the Set State node to use as a binding, the playrate, and the current playback time. Ensure this node is connected to a Convert to Animation State Machine node to properly set the State Machine from the function graph. When the function is bound directly to a State Machine node, you can use the pure Convert to Animation State Machine node (green).  On New Function ABP_Mannequin_Base > On New Function Zoom 1:1 On New Function Context Node Convert to Animation State Machine Node Animation State Result Set State Update Context Node Target State Idle Duration 0.0 ANIMATION When the node is not connected directly to a state machine node, you must use the Convert to Animation State Machine function node (blue).  On New Function ABP_Mannequin_Base > On New Function Zoom 1:1 On New Function Context Node Convert to Animation State Machine Succeeded Node Failed Animation State Set State Update Context Node Target State Idle Duration 0.0 ANIMATION
Target State	Set the State the engine plays when this node becomes active. You can bind this pin using a dynamic value, or input a State's name using

Property	Description
	the field on the node.
Duration	Set the duration of the transition blend from the Current State to the Target State. A value of `0.0` will result in an immediate transition, while larger values will result in slower transitions.
Blend Type	Set the type of blend the node will use to transition the animations between states. You can select from the following options: • **Standard Blend**: Performs a linear blend. • **Inertialization**: Performs an Inertialized blend.
Blend Profile	Set a blend profile to apply to the blend.
Alpha Blend Option	Set the Alpha blend option to customize the animation blending.
Custom Blend Curve	Set a curve asset to be used as a blend curve.

Get State Node

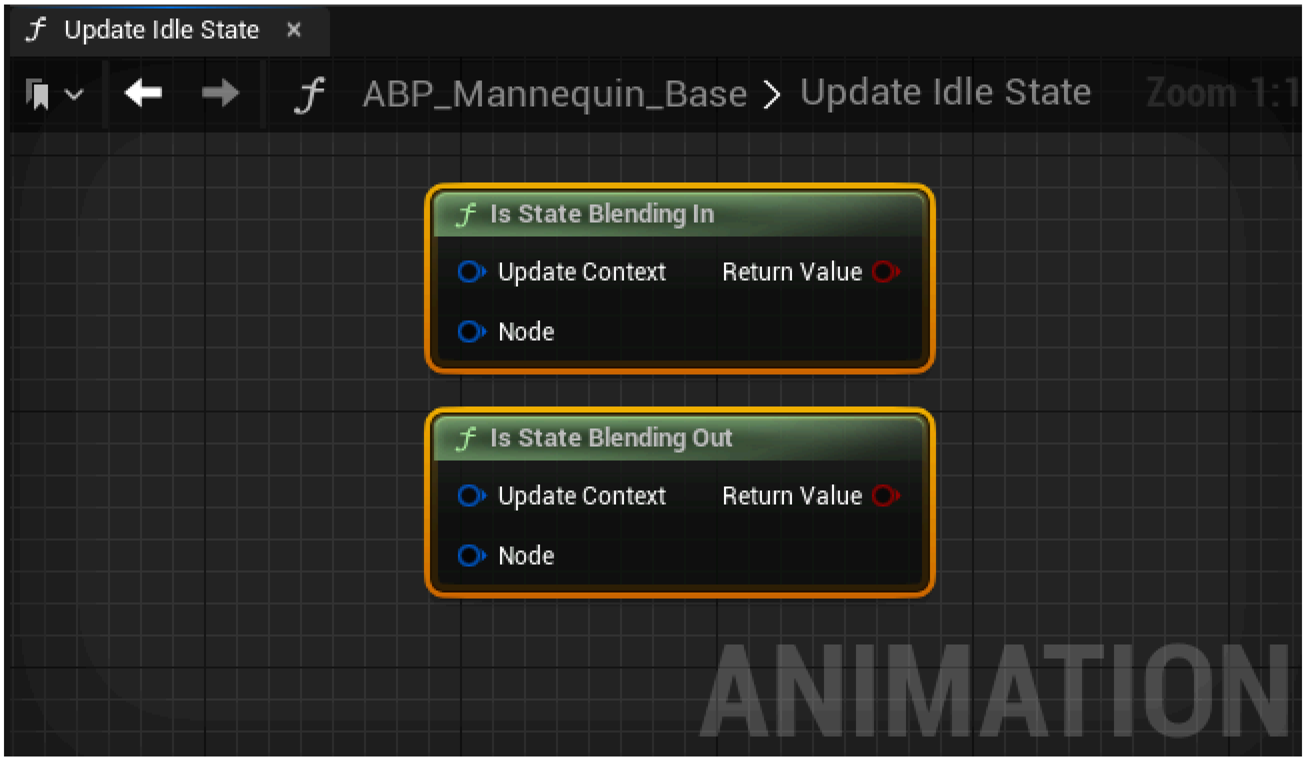
You can use the Get State node to determine what Animation State is currently active. Here you can reference a list of the Get State node's Input and Output pins, and a description of their function.



Pin	Description
Update Context	Set the Get State node's context in the graph using the Function node's Context pin. This pin transfers information such as the delta time, and the current place in AnimGraph.
Node	Set the node reference for the Set State node to use as a binding, the playrate, and the current playback time.
Return Value	Returns the current Animation State's name as a string value.

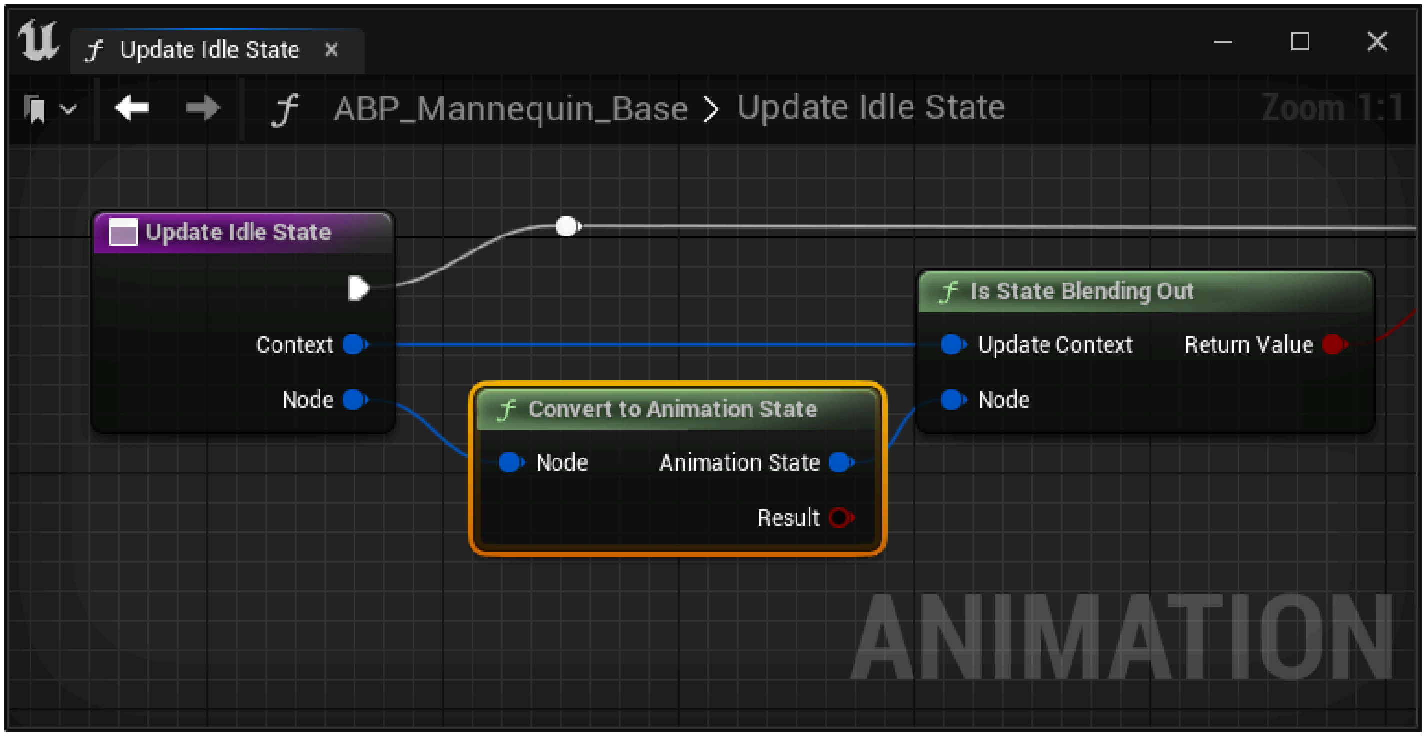
Is State Blend In and Out Nodes

You can use the **Is State Blending In** or **Out** nodes to determine if the current animation state is blending in or out, in order to build logic to drive blending behaviors. Here you can reference a list of the Is State Blending In or Out nodes pins and a description of their function.



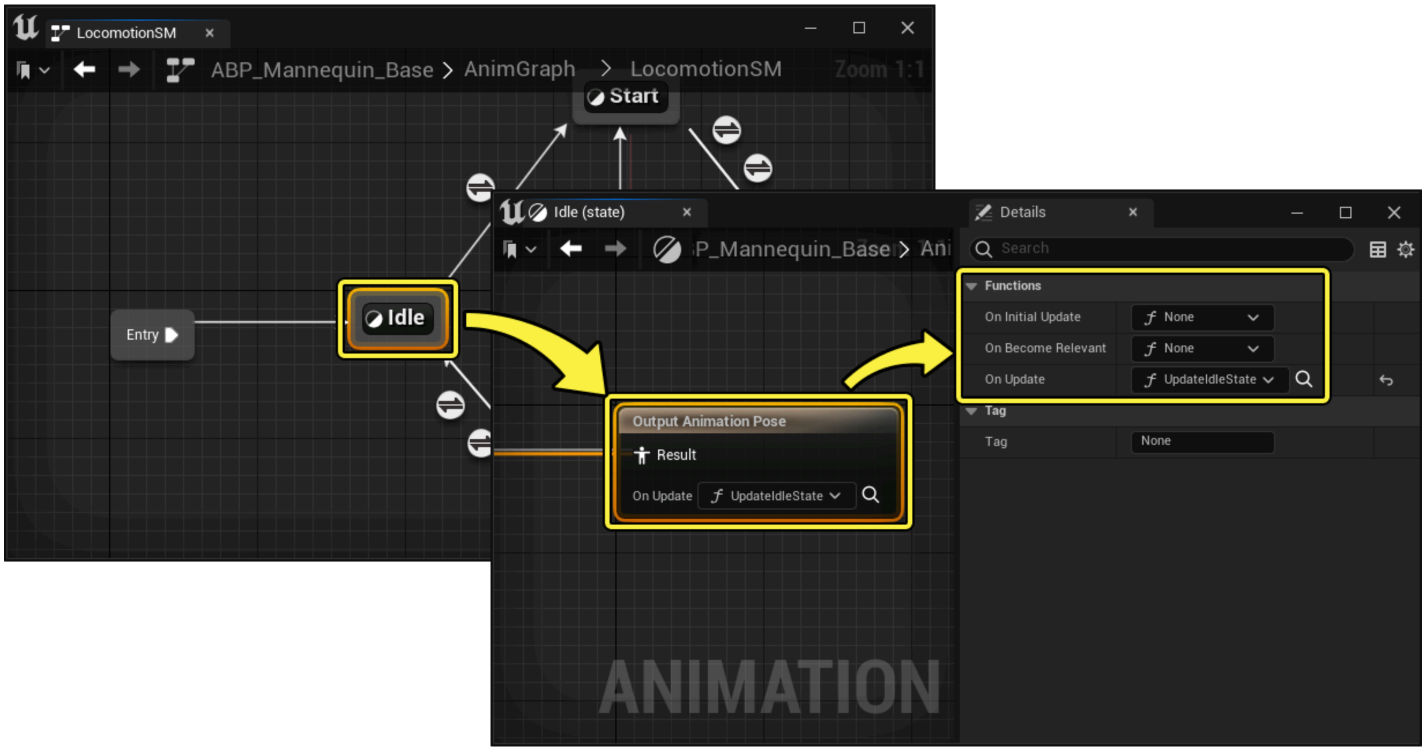
Pin	Description
Update Context	Set the **Get State** node's context in the graph using the **Function** node's **Context** pin. This pin transfers information such as the delta time, and the current place in AnimGraph.
Node	Set the node reference for the **Set State** node to use as a binding, the playrate, and the current playback time.
Return Value	Returns a boolean of whether or not the state is blending **In** or **Out**.

When using the Is State Blending In or Out nodes, you must use a convert to Animation State node to provide the proper Node context data. When updating a function directly on an Animation State output node, you can use the pure **Convert to Animation State** node (green), for all other applicable instances, use the **Convert to Animation State** function node (blue).



Binding a Function to States

You can also bind Animation Node Functions directly to Animation states. However, function graphs can only be bound to Output Nodes within the States AnimGraph.



Additional Resources

For more information about using Animation Node Functions please see the Lyra example project.

Animation In Lyra

An overview of the Animation System in Lyra

To reference an example workflow of setting up an Animation Node Function, see the Distance Matching documentation.

Distance Matching

An in depth look at Distance Matching in Unreal Engine with an example workflow implementation.