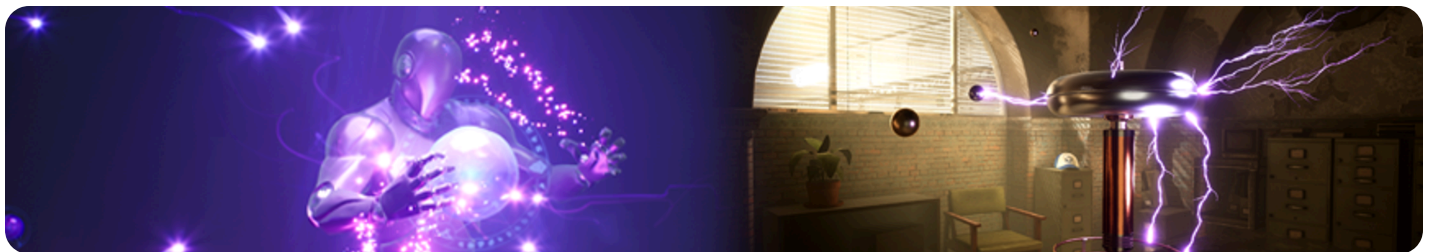


- Developer
- / Documentation
- / Unreal Engine ▾
- / Unreal Engine 5.4 Documentation
- / Creating Visual Effects
- / Niagara Reference
- / Niagara System and Emitter Module Reference
- / Emitter Update Group

Emitter Update Group

This document provides reference information for Emitter Update modules in a Niagara emitter.



Emitter Update modules occur every time the emitter ticks on the CPU. Modules in this group should compute values for Particle Spawn or Update parameters in this frame. Modules are executed in order from the top to the bottom of the stack.

Each of the module types in the Emitter Update group has its own section in this document, with tables that list and describe the default options available for that type of module. Keep in mind that you can create custom modules for any part of the Niagara emitter. The ones listed here are just the ones that are automatically included with Unreal Engine 4.

Beam Modules

Module	Description
Beam Emitter Setup	This module sets up and manages start points, endpoints and tangents for a beam emitter.

Chaos Modules

Module	Description
Spawn from Chaos	This module causes particles to spawn in response to a Chaos event.

Emitter State

Parameter	Description
Life Cycle Mode	<p>This setting determines whether life cycle (looping, age and death) is managed by the emitter itself, or by the system that owns the emitter. Settings are:</p> <ul style="list-style-type: none">• System: When you select this option, the owning system will calculate all life cycle functions. In most cases, having systems calculate life cycle increases optimization. Selecting this option hides the other fields.• Self: When you select this option, the emitter itself will calculate life cycle functions. Selecting this option makes the other settings below available.
Inactive Response	<p>This setting determines what happens when the emitter enters the Inactive state. Inactive means the emitter is dormant and no longer able to spawn or manage particles. Options are:</p> <ul style="list-style-type: none">• Complete: Particles will finish, then the emitter is killed.• Kill: Emitter and particles are both killed immediately.• Continue: Emitter is deactivated, but does not die until the system does.

Parameter	Description
Loop Behavior	<p>This determines the behavior of the emitter. You can select from the following:</p> <ul style="list-style-type: none"> • Once: the emitter plays the animation once. • Multiple: the emitter plays the animation a fixed number of times. • Infinite: the emitter plays the animation infinitely.
Loop Duration	This determines how long the loop lasts.
Loop Duration Mode	This determines whether the loop is finite or infinite.
Loop Delay	This setting delays the next loop by a given amount.
Scalability	
Scalability Mode	<p>This determines whether the emitter will take scalability settings from the system, or if it will have its own unique scalability settings. You can choose from the following:</p> <ul style="list-style-type: none"> • System: When you select this option, the owning system will calculate all life cycle functions. In most cases, having systems calculate life cycle increases optimization. Selecting this option hides the other fields. • Self: When you select this option, the emitter itself will calculate life cycle functions. Selecting this option makes the other settings below available.
Enable Distance Culling	Check this box to enable. This enables the culling of emitters based on the emitter's distance from the camera. Emitters can go to sleep, reawaken, be killed, and so on when they are a certain distance from the camera.
Enable Visibility Culling	Check this box to enable. This enables the culling of emitters based on whether they are visible to the camera. Emitters can go to sleep,

Parameter	Description
	reawaken, be killed, and so on based on whether they are onscreen or offscreen.
Reset Age on Awaken	Check this box to enable. When this emitter reawakens after being put to sleep by a scalability setting, this setting will reset the emitter's age. This means spawn bursts will refire, and the emitter's life cycle will restart.

Location Module

Module	Description
Spawn Particles in Grid	This spawns particles based on user-defined grid resolution settings.

MAX Scripts Module

Module	Description
Spawn MS Vertex Animation Tools Morph Target	This spawns and samples Morph Target textures that were created with the Vertex Animation Tool. The Vertex Animation Tool generates textures that represent Morph Target blend shapes. This module spawns one particle for every vertex that was captured by the tool. This module is meant to be used with the Update MS Vertex Animation Tools Morph Target module.

Spawning Modules

Module	Description
Spawn Burst Instantaneous	This module spawns a burst of particles spontaneously.
Spawn Per Frame	This module spawns a burst of particles in each frame.
Spawn Per Unit	This module spawns particles based on distance traveled in Unreal units.
Spawn Rate	This module spawns particles continuously at a particular rate.

Utility Module

Module	Description
Emitter Frame Counter	Enabling Increment Counter in this module sets up a counter which increments with each frame in the emitter's animation.

New Scratch Pad Module

Selecting this item in the **Add** (Plus sign) menu opens the **Scratch Pad** panel (by default this docks next to the **System Overview**) and places a **Scratch Pad module** in the **Selection** panel. You can also open the Scratch Pad panel by using **Windows > Scratch Pad**. However, by placing a Scratch Pad module in the stack, any modules or dynamic inputs you create in the Scratch Pad are automatically connected to your script. If you open the Scratch Pad panel using the Windows menu, any items you create there will have to be added to your script manually.

Set New or Existing Value Directly

Selecting this item from the **Add** menu places a **Set Parameter** module in the **Selection** panel. Click the **Plus sign (+)** icon to select **Add Parameter** or **Create New Parameter**.

Add Parameter

When you select **Add Parameter**, you select from the parameters listed. This adds that parameter to the **Set Parameter** module in the Emitter Update group.



Some of these parameters can be set or modified in other modules. Some are only set using a Set Parameter module.

Parameter	Description
Emitter.Age	This parameter defines the age of this emitter.
Emitter.CurrentLoopDelay	This parameter defines the current amount of delay before the emitter's current loop repeats.
Emitter.CurrentLoopDuration	This parameter defines the duration of the current emitter loop.
Emitter.ExecutionState	This affects the state of the emitter. Valid value choices are: <ul style="list-style-type: none">• Active• Inactive• InactiveClear• Complete
Emitter.ExecutionStateSource	This indicates the source of an execution state setting. It is used to allow scalability to change the state, but only if the state has not been defined by something with higher precedence.
Emitter.LocalSpace	This parameter defines whether the position of particles is respective to the world origin or the owning Niagara Component's location. <ul style="list-style-type: none">• False: Particle position is in WorldSpace and will be relative to the World origin. A particle with position 0,0,0 will render at the World origin.

Parameter	Description
	<ul style="list-style-type: none"> • True: Particle position is in LocalSpace and will be relative to the owning Niagara Component's location. A particle with position 0,0,0 will render at the owning Niagara Component's location.
Emitter.LoopCount	This parameter defines how many times the emitter's loop repeats.
Emitter.LoopedAge	This parameter calculates the age of the emitter relative to its current loop. For example, if an emitter has been active for 8 seconds and it loops every 5 seconds, the emitter's LoopedAge will be 3 seconds . LoopedAge returns to 0 every time an emitter loops.
Emitter.NormalizedLoopAge	This parameter calculates the age of the emitter relative to its current loop, normalized from 0 to 1 . NormalizedLoopAge is expressed as LoopedAge divided by CurrentLoopDuration . If an emitter has been active for 8 seconds and it loops every 5 seconds, the emitter's LoopedAge will be 3 . The emitter's NormalizedLoopAge will be 0.6 .
System.ExecutionState	This affects the state of the system. Valid value choices are Active , Inactive , InactiveClear , Complete , Disabled , and Num .
System.ExecutionStateSource	This indicates the source of a system execution state setting. It is used to allow scalability to change the state, but only if the state has not been defined by something with higher precedence.

Create New Parameter


When you select **Create New Parameter**, you select from the parameters listed. This adds a **Set Parameter** module to the Emitter Update group.


Parameter	Type	Description
Audio Oscilloscope	Data interface	This adds a new Audio Oscilloscope data interface module to the emitter. The Audio Oscilloscope module can directly access the waveform data of the audio signal.
Audio Spectrum	Data interface	This adds a new Audio Spectrum data interface module to the emitter. The Audio Spectrum module can drive a visualization based on how loud the audio is at specific frequencies.
Bool	Primitive	This adds a Set Variable module that has a true/false checkbox.
Camera Query	Data Interface	This adds a new Camera Query data interface module to the emitter. This data interface can be used to retrieve camera information (camera position, rotation, FOV, etc) for the specified controller index.
Collision Query	Data Interface	This adds a collision data interface to the emitter stack. This is usually used in conjunction with Collision modules.
Curl Noise	Data Interface	This adds a curl noise data interface to the emitter stack. If you use this in conjunction with Curl Noise Force modules, this data interface injects different types of noise into your simulation.
Curve for Colors	Data Interface	This adds a four-channel color curve data interface for the simulation. This curve can be sampled by dynamic

Parameter	Type	Description
		inputs or other modules to create a time-varying color.
Curve for Floats	Data Interface	This adds a single-channel curve data interface for the simulation. This curve can be sampled by dynamic inputs or other modules to create a time-varying float value.
Curve for Vector 2Ds	Data Interface	This adds a two-channel curve data interface for the simulation. This curve can be sampled by dynamic inputs or other modules to create a time-varying pair of floats.
Curve for Vector 3s	Data Interface	This adds a three-channel curve data interface for the simulation. This curve can be sampled by dynamic inputs or other modules to create a time-varying set of floats.
Curve for Vector 4s	Data Interface	This adds a four-channel curve data interface for the simulation. This curve can be sampled by dynamic inputs or other modules to create a time-varying set of floats.
ENiagaraBooleanLogicOps	Enum	<p>This is an enumeration used by various modules and dynamic inputs that want to test using boolean logic:</p> <ul style="list-style-type: none"> • Greater Than • Greater Than Or Equal To • Equal To • Not Equal To

Parameter	Type	Description
ENiagaraCoordinateSpace	Enum	<p>This is an enumeration used by various modules and dynamic inputs to distinguish between coordinate spaces:</p> <ul style="list-style-type: none"> • Simulation: If the emitter is set to Local, use Local. Otherwise, use World. • World: In the world space of the game. • Local: In the coordinate space of the owning component.
ENiagaraExecutionState	Enum	<p>This enumeration type is used by parameters that manage system or emitter execution states, such as Emitter.ExecutionState or System.ExecutionState.</p>
ENiagaraExecutionStateSource	Enum	<p>This indicates the source of an execution state setting. It is used to allow scalability to change the state, but only if the state has not been defined by something with higher precedence.</p>
ENiagara ExpansionMode	Enum	<p>This enumeration is used by location modules to determine where the origin point of expansion is:</p> <ul style="list-style-type: none"> • Inside • Centered • Outside
ENiagaraOrientationAxis	Enum	<p>This is an enumeration used by several modules to determine which axis to do calculations with:</p> <ul style="list-style-type: none"> • X Axis • Y Axis

Parameter	Type	Description
		<ul style="list-style-type: none"> • Z Axis
ENiagaraRandomnessMode	Enum	<p>This sets the type of random number generation used by this emitter. Valid choices are:</p> <ul style="list-style-type: none"> • Simulation Defaults • Deterministic • Non-Deterministic
Float	Primitive	This creates a float value variable.
Grid2D Collection	Data Interface	This is used with simulation stages. It enables the user to read or write to a 2D array of data, and then iterate over each pixel in the grid during a simulation stage.
Int32	Primitive	This creates an integer variable.
Linear Color	Primitive	This creates an RGBA color variable, represented as a color picker.
Matrix	Primitive	This creates a 4×4 matrix variable.
Mesh Tri Coordinate	Struct	This is a simple struct containing a triangle index along with a barycentric coordinate on the face of that triangle.
Neighbor Grid 3D	Data Interface	This is used with simulation stages. It enables the user to read or write to a 3D array of data, and then iterate over each pixel in the volume during a simulation stage.

Parameter	Type	Description
Niagara ID	Struct	<p>This is a two-part struct used to track particles. Index in the indirection table for this particle. It allows fast access to this particle's data. It is always unique among currently living particles, but will be reused after the particle dies.</p> <p>AcquireTag is a unique tag for when this ID was acquired. It allows us to differentiate between particles when one dies and another particle reuses the dead particle's Index.</p>
Occlusion Query	Data Interface	<p>This adds a new Occlusion Query data interface module to the emitter. The data interface is used to read depth buffer occlusion information.</p> <div>  <p>This can only be used with GPU emitters.</p> </div>
Particle Attribute Reader	Data Interface	<p>This adds a new Particle Attribute Reader data interface to the emitter. The data interface can be used to query particle payload values from other emitters, and can sometimes be easier to use than Events.</p>
Quat	Primitive	<p>This creates a quaternion variable, used to represent rotations.</p>
Simple Counter	Data Interface	<p>This adds a new Simple Counter data interface module to the emitter. This data interface enables you to increment a thread-safe counter.</p>

Parameter	Type	Description
		<div>  <p>This can only be used with CPU emitter.</p> </div>
Skeletal Mesh	Data Interface	This is a data interface with functions to interact with a skeletal mesh's bones or sockets and skinned geometry.
Spawn Info	Struct	This is a structure used in spawning to specify the Count of particles to create, InterpStartDt or offset from the current frame begin time to start spawning, IntervalDt defining the time gap between particles being spawned, and SpawnGroup allowing spawned particles to belong to different categories.
Spline	Data Interface	This is a data interface that interacts with a Spline Asset.
Static Mesh	Data Interface	This is a data interface with functions to interact with a static mesh's surface.
Texture Sample	Data Interface	This is a data interface with functions to interact with a texture on the GPU.
Vector	Primitive	This creates a three-channel set of floats.
Vector 2D	Primitive	This creates a two-channel set of floats.
Vector 4	Primitive	This creates a four-channel set of floats.

Parameter	Type	Description
Vector Field	Data Interface	This is a DataInterface with functions to query a vector field.
Volume Texture Sample	Data Interface	This adds a new Volume Texture data interface module to the emitter. You can use this to sample a volume texture.