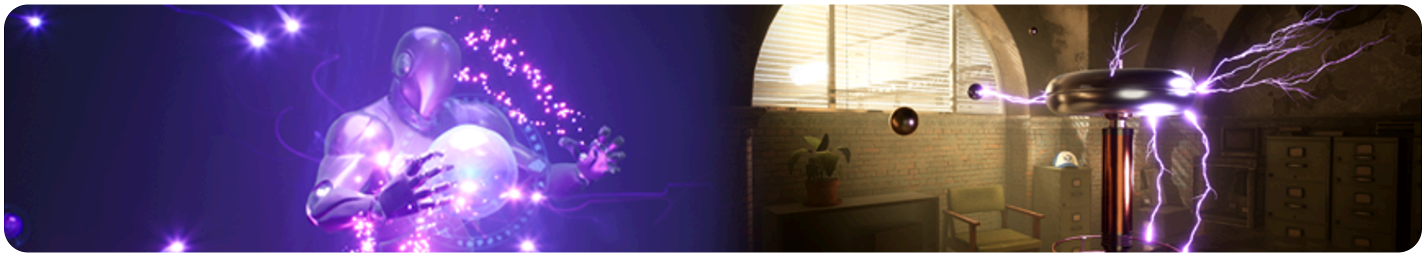
Particle Update Group

This page provides reference information for modules in the Particle Update group.



Particle Update modules are called every frame, per particle. Modules in this section should update new values for this frame. Modules are executed in order from the top to the bottom of the stack.

Each of the module types in the Particle 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, you can create custom modules for any part of a Niagara system or emitter. The ones listed here are just the ones that are automatically included with Unreal Engine.

Beam Modules

Module	Description
Beam Width Scale	This module scales the initial beam width by a user-defined scale factor.
Update Beam	This module places particles along a bezier spline, or simply along a line between two points. Useful for sprite facing along a beam-style path, or to be used with the ribbon renderer for a classic-style beam. **Update**

Module	Description
	Beam recalculates the spline positions in each frame, to allow for beams with dynamically changing start and end points.

Camera Modules

Module	Description
Camera Offset	This module offsets the particle along the vector between the particle and the camera.
Maintain in Camera Particle Scale	This retains the in-camera particle size by taking into account the camera's FOV, the particle's camera-relative depth, and the render target's size.

Collision Modules

Module	Description
Align Particles with Collision Plane	This module aligns a sprite to a plane over time. By default, the plane is defined by the collision module's returned collision plane. When using this module, the **Alignment** setting in the Sprite Renderer must be set to **Custom Alignment**; the **Facing Mode** must be set to **Custom Facing Vector**; and the **Custom Facing Vector Mask** must be set to **1**, **1**, **1**.
Collision	This module must be placed immediately before the solver module. When used in CPU emitters, the module will cast rays in order to calculate its collision with the world. When used with GPU emitters, the module will either use the scene depth or the global distance field to find collision surfaces.

Color Modules

Module	Description
Color	This module directly sets the **Particles.Color** parameter, with scale factors for the Float3 Color and Scalar Alpha components.
Scale Color	By default, this module accepts the initial color (as determined in the Particle Spawn group) and scales the RGB and Alpha components separately.
Scale Color by Speed	This module scales particle color according to the magnitude of the particle's velocity vector, with minimum and maximum speed thresholds.

Constraints Modules

Module	Description
Maintain a Set Distance Between Two Points	This module takes in two different positions: the particle's position, and the target position. The module will project a point in space away from the initial position, to a location on a vector between the two positions. This vector's length is set in **Ideal Distance**. For certain situations, you might want to switch the target position and particle position variables.
Pendulum Constraint	This module introduces a non-physically correct pendulum constraint that interacts with forces. Add the **Gravity Force** module to have it interact with gravity; add the **Acceleration Force** module to have it interact with an acceleration force other than gravity; Add the **Drag Force** module to have it interact with viscous friction drag.

Module	Description
	You must have the **Pendulum Setup** module in order to use the **Pendulum Constraint** module.
Pendulum Setup	This module introduces a non-physically correct pendulum constraint that interacts with forces. Add the **Gravity Force** module to have it interact with gravity; add the **Acceleration Force** module to have it interact with an acceleration force other than gravity; Add the **Drag Force** module to have it interact with viscous friction drag.

Events Modules

Module	Description
Generate Collision Event	This module generates a collision event in the emitter. This can later be used by an Event Handler in another emitter, to cause actions in the system.
Generate Death Event	This module generates a death event in the emitter. This can later be used by an Event Handler in another emitter, to cause actions in the system.
Generate Location Event	This module generates a location event in the emitter. This can later be used by an Event Handler in another emitter, to cause actions in the system.

Forces Modules

Module	Description
Acceleration Force	This adds to the **Physics.Force** parameter, which will translate into acceleration within the solver.
Apply Initial Forces	This module converts rotational and linear forces, such as Curl Noise Force, into rotational and linear velocity.
Curl Noise Force	This adds to the **Physics.Force** parameter using a curl noise field. It samples a medium resolution, baked tiling curl noise field by default. However it can optionally sample a perlin-derived curl function directly at increased cost.
Drag	This applies the Drag value directly to particle velocity and rotational velocity, irrespective of mass. It accumulates into **Physics.Drag** and **Physics.RotationalDrag**, which are solved in the **Solve Forces and Velocity** and **Solve Rotational Forces and Velocity** modules.
Gravity Force	This applies a gravitational force (in cm/s) to the **Physics.Force** parameter.
Limit Force	This scales the **Physics.Force** parameter down to the magnitude specified, if it exceeds the **Force Limit**.
Line Attraction Force	This accumulates a pull toward the nearest position on a line segment. It then adds that value to **Transient.PhysicsForce**.
Linear Force	This adds a force vector (in cm/s) to the **Physics.Force** parameter in a specific coordinate space.
Mesh Rotation Force	This adds a rotational force as described by the newtons applied on the yaw, pitch and roll axes, and accumulates that value to the **Physics.RotationalForce** parameter.
Point Attraction Force	This accumulates a pull toward **AttractorPosition** into the **Physics.Force** parameter.

Module	Description
Point Force	This adds force from an arbitrary point in space with optional falloff. It uses the vector between the velocity origin and the **Particles.Position** parameter in order to determine the force vector. If positions have not been initialized (that is, the particle position and velocity origin are on top of each other), the module will inject random velocity. You should place this module after any location modules in the stack, to make sure that the particle positions have already been initialized.
Vector Noise Force	This introduces random noise into the **Physics.Force** parameter.
Vortex Force	This takes a velocity around a vortex axis (with an optional additional pull towards the vortex origin) and injects it into the **Physics.Force** parameter.
Wind Force	This applies a wind force to particles, with an optional **Air Resistance** parameter. If the particle is moving faster than the wind speed in the direction of the wind, no additional force is applied.

Kill Modules

Module	Description
Kill Particles	This switch kills all particles if set to True (the box is checked). It allows for particles to be dynamically killed based on this boolean, at any point in the execution stack.
Kill Particles in Volume	This module kills particles if they are inside an analytical shape. This shape can be a box, a plane, a slab (two planes facing inwards), or a sphere. The result can also be inverted. This module must be used with **Interpolated Spawn** enabled, or else any spawned particles will appear for one frame and then die.

Lifetime Module

Module	Description
Particle State	This module handles killing a particle from the simulation when the particle's lifetime has elapsed.

Location Modules

Module	Description
Box Location	This spawns particles in a rectangular box shape.
Cone Location	This spawns particles in a cone shape.
Cylinder Location	This spawns particles in a cylinder shape, with lathe-style controls to modify the profile of the cylinder.
Grid Location	This spawns particles in an even distribution on a grid.
Jitter Position	This jitters a spawned particle in a random direction, on a delay timer.
Rotate Around Point	This module will find a position on a forward vector-aligned circle around a user-defined center point. The radius and position along the circle can be modified over time.
Skeletal Mesh Location	This places particles on a bone, socket, triangle or vertex of a Skeletal Mesh.
Sphere Location	This spawns particles in a spherical shape, with options for hemisphere shaping and density.
Static Mesh Location	This spawns particles from the surface of a static mesh.

Module	Description
System Location	This spawns particles from the system's location.
Torus Location	This spawns particles in a torus shape.

Mass Modules

Module	Description
Calculate Mass and Rotational Inertia by Volume	This calculates the mass and rotational inertia based on the particle's bounds and a density value. The density is measured in kg/m³.
Calculate Size and Rotational Inertia by Mass	This calculates the particle's scale and rotational inertia based on user-driven mass and density values. The density is measured in kg/m³.
Update Velocity on Mass Change	This alters the particle's angular and linear velocity based on the mass delta. In other words, if the particle's mass increases, the particle slows down; if the particle's mass decreases, the particle speeds up.

Materials Modules

Module	Description
Dynamic Material Parameters	These write to the Dynamic Parameter Vertex Interpolator node in the Material Editor. To use Indices 1-3, change the **Parameter Index** on the node itself in the Material Editor to the corresponding number. This allows the use of up to four unique dynamic parameter nodes in a given material.

Math/Blend Modules

Module	Description
Cone Mask	This module defines a cone in 3D space and checks if the position input lies inside the cone. If the position lies inside the cone, it returns **1**; otherwise it returns **0**.
Lerp Particle Attributes	This module enables linear interpolation (lerp) of all default particle parameters. You can select the specific parameter to interpolate each default particle parameter against, and also the interpolation factor for each default particle parameter.
Recreate Camera Projection	This module recreates the camera-relative world position of a scene capture 2D's pixel. The projector transform fields allow one to reposition and rotate the projected positions.
Temporal Lerp Float	This module performs a slow linear interpolation (lerp) according to the user's specified **Current Value** over time. The convergence rate is specified through the **Rate of Change** input.
Temporal Lerp Vector	This module performs a slow linear interpolation (lerp) according to the user's specified **Current Value** over time. The convergence rate is specified through the **Rate of Change** input.

Mesh Modules

Module	Description
Sample Skeletal Mesh Skeleton	This module samples the bone or socket positions of a skeletal mesh, and then writes those sampled values to particle parameters. These particle parameters can then be used later in the stack.

Module	Description
Sample Skeletal Mesh Surface	This module samples the surface of a skeletal mesh, then writes those sampled values to particle parameters. These particle parameters can then be used later in the stack.
Sample Static Mesh	This module samples a static mesh, and then writes those sampled values to particle parameters. These particle parameters can then be used later in the stack.
Update Mesh Reproduction Sprite	This module is used along with the Initialize Mesh Reproduction Sprite module. To recreate the effect in the Niagara level of Content Examples, follow these steps: 1. Place the **Initialize Mesh Reproduction Sprite** module in the **Particle Spawn** group. 2. Place the **Update Mesh Reproduction Sprite** module into the **Particle Update** group. 3. In the Sprite Renderer, set **Alignment** to **Custom Alignment**; set **Facing Mode** to **Custom Facing Vector**; set **Custom Facing Vector Mask** to **1, 1, 1**. 4. In your Material, use **Niagara Mesh Reproduction Sprite UVs** to sample the mesh's UVs. 5. If **Module.OverwriteIntrinsicVariables** is set to **False**, make sure that this module's output variables drive the particle's attributes (such as position, alignment and so on).

Orientation Modules

Module	Description
Align Sprite to Mesh Orientation	This module aligns sprites to a mesh particle's orientation. This enables you to use the **Mesh Rotation** and **Rotational Velocity** modules to control a sprite's alignment in relation to the world. Make sure that the **Alignment** and **Facing Mode** settings in the Sprite Renderer are set to **Custom Alignment**

Module	Description
	and **Custom Facing**. Set your **Custom Facing Vector Mask** to **1, 1, 1**.
Orient Mesh to Vector	This module aligns a mesh to an input vector.
Sprite Rotation Rate	This module rotates a sprite over time. The default input is in degrees. You can add the Dynamic Input **Normalized Angle to Degrees** if you want an input range from 0-1.
Update Mesh Orientation	This module rotates the Mesh Orientation parameter over time.

Physics Modules

Module	Description
Add Rotational Velocity	This module adds to the **Rotational Velocity** value in a user-defined space.
Find Kinetic and Potential Energy	This module returns the following: 1. A particle's kinetic energy, based on the particle's velocity. 2. A particle's potential energy, which is the sum of all force modules that write to **Physics.PotentialEnergy**. 3. The sum of 1 and 2.

Post-Solve Modules

Module	Description
Calculate Accurate Velocity	This module calculates an accurate velocity from a previous position to the current position. This is useful if a constraint was

Module	Description
	applied that altered the expected position of a particle outside the framework of the solver.

Ribbon Modules

Module	Description
Scale Ribbon Width	This controls the width of the spawned ribbon, and writes to **Particles.RibbonWidth**.

Size Modules

Module	Description
Scale Mesh Size	This module takes the initial mesh size scale, as set in the Particle Spawn group, and increases the scale by a user-set factor.
Scale Mesh Size by Speed	This module scales the initial mesh size scale by the magnitude of the velocity vector.
Scale Sprite Size	This module takes the initial sprite size scale, as set in the Particle Spawn group, and increases the scale by a user-set factor.
Scale Sprite Size by Speed	This module scales the **Particles.SpriteSize** parameter by the magnitude of the velocity vector.

SubUV Modules

Module	Description
SubUVAnimation	Some sprites are made in a grid, with each individual sprite representing an animation frame. This module accepts the total number of sprites that are to be animated, and plots those sprites along a curve so that they animate smoothly.

Texture Modules

Module	Description
Sample Pseudo Volume Texture	This module samples the color of a pseudo volume texture, based on UVW coordinates.
Sample Texture	This samples a texture at a specific UV location, and returns the color of that part of the texture. This module is only supported in GPU simulations.
Sub UV Texture Sample	This module samples a single texture pixel in a subUV fashion.
World Aligned Texture Sample	This module samples a texture's color based on particle position, much like a world-aligned texture does in the Material Editor.

Utility Modules

Module	Description
Do Once	This module keeps track of whether its trigger condition has ever been true in a previous frame. If not, **Particles.Module.Execute** returns True. If the trigger condition has returned True in a previous frame, **Particles.Module.Execute** returns False.
Generate Grid Ribbon IDs	You can use this module to produce the output particle parameters used to produce a 3D grid with 3 ribbon emitters. In the Ribbon Renderer, replace **RibbonID** with **Particles.RibbonID1**, **Particles.RibbonID2**, and **Particles.RibbonID3**. Set the **Ribbon Link Order** to **1**, **2** and **3**. For a 2D grid, do the above, but with only 2 ribbon emitters.
Increment Over Time	This module increases a value each frame. The counter variable increments using the tick delta value and multiplying it by a user-specified rate.
Time Based State Machine	This module outputs a float parameter (**Particles.Module.OnOffPercentage**) that indicates if the particle is in the On state (1) or Off state (2).
Update MS Vertex Animation Tools Morph Targets	This module reads morph target texture data, and outputs positions and normal vectors for a given pixel-per-particle index. Connect the module's world space normal output into the asset's material, with tangent space normals disabled within the material in order to reproduce the mesh's surface. For more information on generating morph target textures, see Vertex Animation Tool. This module can directly set a particle's position. If you use it this way, do not use another module that directly sets a particle's position.

Vector Field Modules

Module	Description
Apply Vector Field	This module takes the vector samples by a vector field sampler, and applies it as a force or velocity.
Sample Vector Field	This module samples a vector field, applying a per-particle intensity factor and an optional falloff factor, that fades the influence of the vector field towards the edges of the bounding box. This can optionally apply a local translate, rotate or scale transformation.

Velocity Modules

Module	Description
Add Velocity	This module assigns a velocity to spawned particles. You can add various dynamic inputs to modify the values you enter in this module.
Add Velocity from Point	This module adds velocity from an arbitrary point in space, with optional falloff. It uses the vector between the velocity origin and the particle's position to determine the velocity vector. If particle positions have not been initialized (causing the particle position and velocity origin to be too close together), the module will inject random velocity. For the most accurate results, place this module below any location modules in the stack. This ensures that particle positions have been initialized.
Add Velocity in Cone	This module adds velocity to the **Particles.Velocity** parameter in a cone shape, with parameters for cone angle, and for velocity distribution along the cone axis.

Module	Description
Align Velocity to Random Axis	This module takes a velocity vector and maintains its magnitude, while picking a random axis to align it to, with an optional time interval between updates (in seconds).
Inherit Velocity	This module adds inherited velocity from another source. This defaults to the position of the system that owns the current emitter.
Scale Velocity	This module multiplies **Particles.Velocity** by a separate vector in a specific coordinate space.
Static Mesh Velocity	This module adds velocity based on the normals from a static mesh, as well as adding the inherited velocity of a static mesh.
Vortex Velocity	This module calculates an angular velocity around a vortex axis, and injects it into the **Particles.Velocity** parameter. This is added to the initial velocity the particle has when spawned.

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**(Plus sign) 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 Particle 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
DataInstance.Alive	This parameter is used to determine whether or not this particle instance is still valid, or if it can be deleted.
Particles.Age	This defines the age of a particle.
Particles.CameraOffset	This sets the Camera Offset for a particle. The Camera Offset determines the distance between the particle and the camera.
Particles.Color	This directly sets the color of the particle.
Particles.DynamicMaterialParameter	This is a four-float vector used to send data to the renderer.
Particles.DynamicMaterialParameter 1	This is a four-float vector used to send data to the renderer.
Particles.DynamicMaterialParameter 2	This is a four-float vector used to send data to the renderer.
Particles.DynamicMaterialParameter 3	This is a four-float vector used to send data to the renderer.

Parameter	Description
Particles.ID	This is an engine-managed attribute that provides a persistent ID to each spawned particle.
Particles.Initial.Color	This sets the initial color of a sprite used by a particle.
Particles.Lifetime	This is the lifetime of a spawned particle in seconds.
Particles.LightRadius	This parameter determines the radius of emitted light when you are using a Light Renderer.
Particles.Mass	This parameter determines the mass of a spawned particle.
Particles.MaterialRandom	This parameter is used to drive the **Particle Random** node in the Material Editor. When this is not set, any Particle Randoms will get **0.0**.
Particles.MeshOrientation	This determines the axis-angle rotation that is applied to a spawned mesh particle.
Particles.NormalizedAge	This is the value of **Particles.Age** (in seconds) divided by the value of **Particles.Lifetime** (in seconds). This is useful for animation, because the value generated is between **0** and **1**.
Particles.Position	This sets the position of a spawned particle.
Particles.PreviousVelocity	This is used with the **Solve Forces And Velocity** module to calculate a particle's position in response to force and velocity. Previous velocity is needed to solve for acceleration.
Particles.RibbonFacing	This sets the facing vector of the ribbon at the ribbon particle's position, or the side vector that the ribbon's

Parameter	Description
	width is extended along, depending on which **Facing Mode** is selected.
Particles.RibbonID	This assigns a **Ribbon ID** to a ribbon particle. Particles with the same Ribbon ID are connected into a ribbon.
Particles.RibbonLinkOrder	This sets an explicit order for linking particles within a ribbon. Particles with the same Ribbon ID are connected into a ribbon in incrementing order according to this value.
Particles.RibbonTwist	This sets the amount of twist a ribbon particle has, in degrees.
Particles.RibbonWidth	This sets the width of a ribbon particle, in UE4 units.
Particles.Scale	This sets the XYZ scale of a non-sprite particle.
Particles.SpriteAlignment	This makes the texture point toward the sprite's selected alignment axis. When using this parameter, the Sprite Renderer's **Alignment** must be set to **Custom Alignment**.
Particles.SpriteFacing	This makes the surface of the sprite face towards a custom vector. To use this parameter, the Sprite Renderer's **Facing Mode** must be set to **Custom Facing Vector**, and values must be provided in the Sprite Renderer's **Custom Facing Vector Mask** setting.
Particles.SpriteRotation	This sets the screen-aligned roll of the particle, in degrees.
Particles.SpriteSize	This determines the size of the sprite particle's quad.
Particles.SubImageIndex	This sets a value which ranges from 0 to a value equal to the number of entries in the table of SubUV images.

Parameter	Description
Particles.UniqueID	This is an engine-managed attribute that serves as a unique ID for each spawned particle. The ID is incremented for each new particle spawned.
Particles.UVScale	This is used to multiply the generated UVs for Sprite Renderers.
Particles.Velocity	This determines the velocity of a particle in centimeters per second (cm/s).

Create New Parameter

When you select **Create New Parameter**, you select from the parameters listed. This adds that parameter to the **Set Parameter** module in the Particle 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

Parameter	Type	Description
		camera information (such as camera position, rotation, or FOV) 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 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.

Parameter	Type	Description
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**	This is an enumeration used by various modules and dynamic inputs that want to test using boolean logic: • **Greater Than** • **Greater Than Or Equal To** • **Equal To** • **Not Equal To**
ENiagaraCoordinateSpace	**Enum**	This is an enumeration used by various modules and dynamic inputs to distinguish between coordinate spaces: • **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.

Parameter	Type	Description
ENiagaraExecutionState	**Enum**	This enumeration type is used by parameters that manage system or emitter execution states, such as Emitter.ExecutionState or System.ExecutionState.
ENiagaraExecutionStateSource	**Enum**	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.
ENiagaraExpansionMode	**Enum**	This enumeration is used by location modules to determine where the origin point of expansion is: • **Inside** • **Centered** • **Outside**
ENiagaraOrientationAxis	**Enum**	This is an enumeration used by several modules to determine which axis to do calculations with: • **X Axis** • **Y Axis** • **Z Axis**
ENiagaraRandomnessMode	**Enum**	This sets the type of random number generation used by this emitter. Valid choices are: • **Simulation Defaults** • **Deterministic** • **Non-Deterministic**

Parameter	Type	Description
Float	**Primitive**	This creates a float value variable.
Grid 2D 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 Grid3D	**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.
Niagara ID	**Struct**	This is a two-part struct used to track particles. 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. **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.

Parameter	Type	Description
Occlusion Query	**Data Interface**	This adds a new Occlusion Query data interface module to the emitter. The data interface is used to read depth buffer occlusion information. This can only be used with GPU emitters.
Particle Attribute Reader	**Data Interface**	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.
Quat	**Primitive**	This creates a quaternion variable, used to represent rotations.
Simple Counter	**Data Interface**	This adds a new Simple Counter data interface module to the emitter. This data interface enables you to increment a thread-safe counter. This can only be used with CPU emitters.
Skeletal Mesh	**Data Interface**	This is a data interface with functions to interact with a skeletal mesh's bones/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

Parameter	Type	Description
		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.
Vector Field	**Data Interface**	This is a data interface 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.