

Table 1: Outlets for UAffinitiesComponent

GetUnspentPoints	
► Before	<code>const uint8 OriginalPoints, uint8& ReturnedPoints</code>
► After	<code>const uint8 OriginalPoints, const uint8 ReturnedPoints</code>

SetUnspentPoints	
► Before	<code>const uint8 OriginalPoints, const uint8 InputPoints, uint8& SetPoints</code>
► After	<code>const uint8 OriginalPoints, const uint8 InputPoints, const uint8 SetPoints</code>

— • • • —

Table 2: Outlets for ULevelComponent

GetBaseExpYield	
► Before	<code>const float OriginalYield, float& ReturnedYield</code>
► After	<code>const float OriginalYield, const float ReturnedYield</code>

GetCXP	
► Before	<code>const uint32 OriginalCXP, int32& ReturnedCXP</code>
<i>Note:</i>	ReturnedCXP is <code>int32&</code> instead of <code>uint32&</code> for Blueprint compatability.
► After	<code>const uint32 OriginalCXP const int32 ReturnedCXP</code>
<i>Note:</i>	ReturnedCXP is <code>const int32</code> instead of <code>const uint32</code> for Blueprint compatability.

Continued on next page

Table 2: Outlets for ULevelComponent (Continued)

GetExpYield	
► Before	<pre>const float OriginalYield, float& ReturnedYield, const uint16 DefeatedLevel, const uint16 VictoriousLevel</pre>
<i>Note:</i>	“Defeated” and “Victorious” levels are provided for flexibility (e.g., in case you want to yield exp differently based on level difference, although technically you could always back-calculate the level difference based on the equation and <code>OriginalYield</code>).
► After	<pre>const float OriginalYield, const float ReturnedYield, const uint16 DefeatedLevel, const uint16 VictoriousLevel</pre>
<i>Note:</i>	“Defeated” and “Victorious” levels are provided for symmetry with respect to the <code>Before</code> delegate (since <code>ReturnedValue</code> is already calculated, I can’t think of why you would need them, but you never know!).
GetMaxLevel	
► Before	<pre>const uint16 DefaultMax, int32& AttemptedMax</pre>
<i>Note:</i>	<p><code>DefaultMax</code> is defined in the code. It should normally be 100, but may change for certain subclasses (e.g., a <code>UBossLevelComponent</code> may have a max of 200 instead).</p> <p><code>AttemptedMax</code> is <code>int32&</code> instead of <code>uint16&</code> for Blueprint compatability.</p>
► After	<pre>const uint16 DefaultMax const int32 ReturnedMax</pre>

Continued on next page

Table 2: Outlets for ULevelComponent (Continued)

GetMinLevel	
► Before	<code>const uint16 DefaultMin, int32& AttemptedMin</code>
<i>Note:</i>	<code>DefaultMin</code> is defined in the code. It should normally be 1, but may change for certain subclasses (e.g., a <code>UEggLevelComponent</code> may have a min of 0 instead for whatever reason). <code>AttemptedMin</code> is <code>int32&</code> instead of <code>uint16&</code> for Blueprint compatability.
► After	<code>const uint16 DefaultMin const int32 ReturnedMin</code>
<i>Note:</i>	<code>ReturnedCXP</code> is <code>const int32</code> instead of <code>const uint32</code> for Blueprint compatability.
GetBaseExpYield	
► Before	<code>const float OriginalYield, float& ReturnedYield</code>
► After	<code>const float OriginalYield, const float ReturnedYield</code>
SetBaseExpYield	
► Before	<code>const float OldYield, const float InputYield, float& AttemptedYield</code>
► After	<code>const float OldYield const float InputYield, const float NewYield</code>
<i>Note:</i>	<ul style="list-style-type: none"> ▷ <code>OldYield</code> is the yield prior to calling <code>SetBaseExpYield</code>, ▷ <code>InputYield</code> is the original, unmodified input to <code>SetBaseExpYield</code>, ▷ <code>AttemptedYield</code> is the modified value that will be used to set the base exp yield.

Continued on next page

Table 2: `Outlets` for `ULevelComponent` (Continued)

<code>SetCXP</code>	
► Before	<pre>const uint32 OldCXP, const int32 InputCXP, int32& AttemptedCXP</pre>
<i>Note:</i>	<code>AttemptedCXP</code> is <code>int32&</code> instead of <code>uint32&</code> for Blueprint compatability.
► After	<pre>const uint32 OldCXP const int32 InputCXP, const uint32 NewCXP</pre>
<i>Note:</i>	<p><code>UStatsComponent</code> subscribes to this in order to change stats on level change.</p> <ul style="list-style-type: none"> ▷ <code>OldCXP</code> is the cumulative experience points prior to calling <code>SetCXP</code>, ▷ <code>InputCXP</code> is the original, unmodified input to <code>SetCXP</code>, ▷ <code>AttemptedCXP</code> is the modified value that will be used to set the cumulative experience points.

• • •

Table 3: `Outlets` for `UStatsComponent`

<code>ModifyStat</code>	
► Before	<pre>const EStatEnum TargetStat, const EStatValueType ValueType, const EModificationMode Mode, const float OriginalValue, float& AttemptedValue</pre>
► After	<pre>const EStatEnum TargetStat, const EStatValueType ValueType, const EModificationMode Mode, const float OriginalValue, const float NewValue</pre>
<i>Note:</i>	All “ModifyStat” functions from <code>UStatsComponent</code> (such as <code>ModifyStatsUniformly</code> or <code>RandomizeStats</code>) go through <code>ModifyStatInternal</code> , which calls this <code>Outlet</code> .

Continued on next page

Table 3: `Outlets` for `UStatsComponent` (Continued)

RandomizeStats	
► Before	<pre>const EStatEnum TargetStat, const FStatRandParams OriginalParams, FStatRandParams& ParamsToBeUsed</pre>
► After	<pre>const EStatEnum TargetStat, const FStatRandParams OriginalParams, const FStatRandParams UsedParams</pre>
<i>Note:</i>	The <code>EStatEnum</code> is not the acutal <code>FStat</code> . To get the <code>FStat</code> (such as <code>FHealth</code>), use <code>UStatsComponent::GetStat(EStatEnum)</code>
RecalculateStats	
► Before	<pre>const EStatEnum TargetStat, const bool bResetCurrent, const float OriginalCurrent, const float OriginalPermanent</pre>
► After	<pre>const EStatEnum TargetStat, const bool bResetCurrent, const float OriginalCurrent, const float OriginalPermanent</pre>