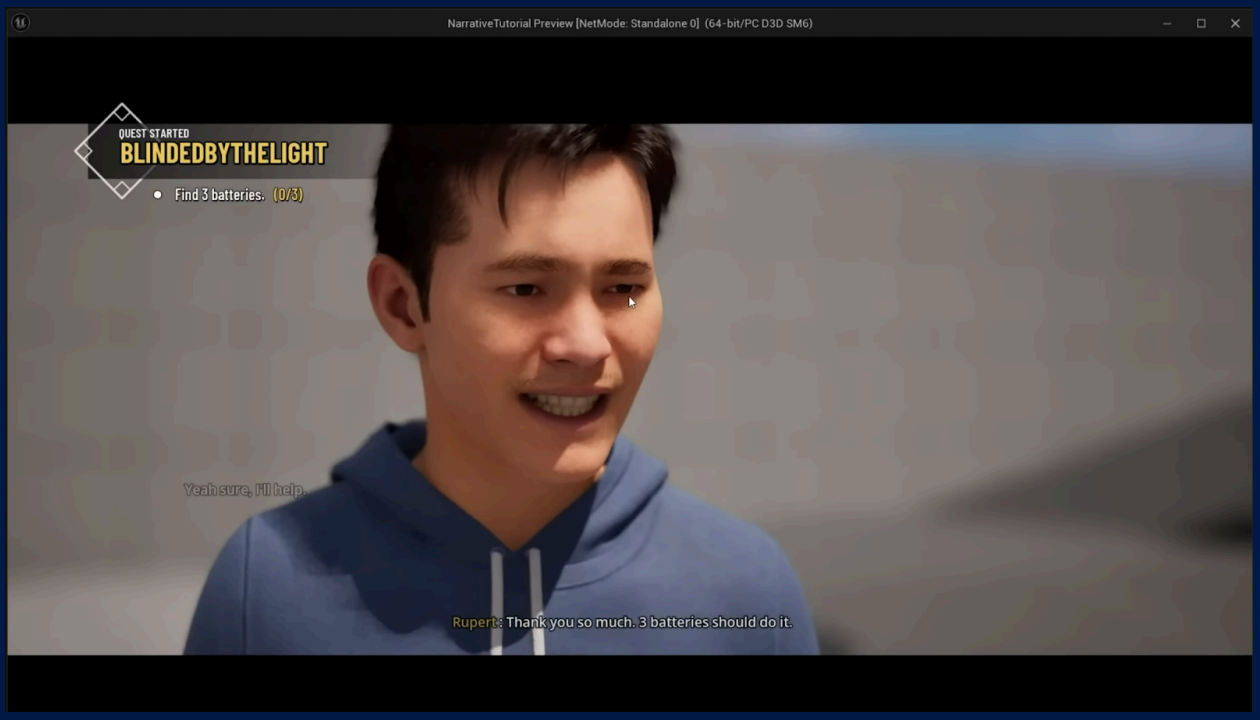
Narrative Pro
Quick Start Guide

Video Guide

The full written docs for Narrative Pro will be available on the 8th of September. In the meantime you will have to use the video tutorial.
https://www.youtube.com/watch?v=KCAqvnYe7_Y

Introduction



Welcome to the Narrative 3.5 quick start guide. This guide will show you how to create your first quest and dialogue with Narrative. After this guide, you will know how to add Narrative to your game with all the necessary functionality.

A note, Narrative is a big system. There are lots of moving parts that you can slowly work your way up to as and when you need them.

*Another note, Narrative 3.5 and future updates rely upon the **FREE** plugin **Narrative Common UI**. If you use Narrative 3.4 or lower, you do not need Common UI.*

A video guide is also available here:

https://www.youtube.com/watch?v=azheylJrvvk

An upgrade video guide is also available here to move to Common UI:

https://www.youtube.com/watch?v=lvH2QrYpOrs

Narrative also has an active Discord community with lots of helpful people:

https://discord.gg/qyVJmpQ2Pn

Note, you will need to verify your purchase by leaving a question OR review on at least 1 of the Narrative products to gain access to all the Discord's Narrative channels.

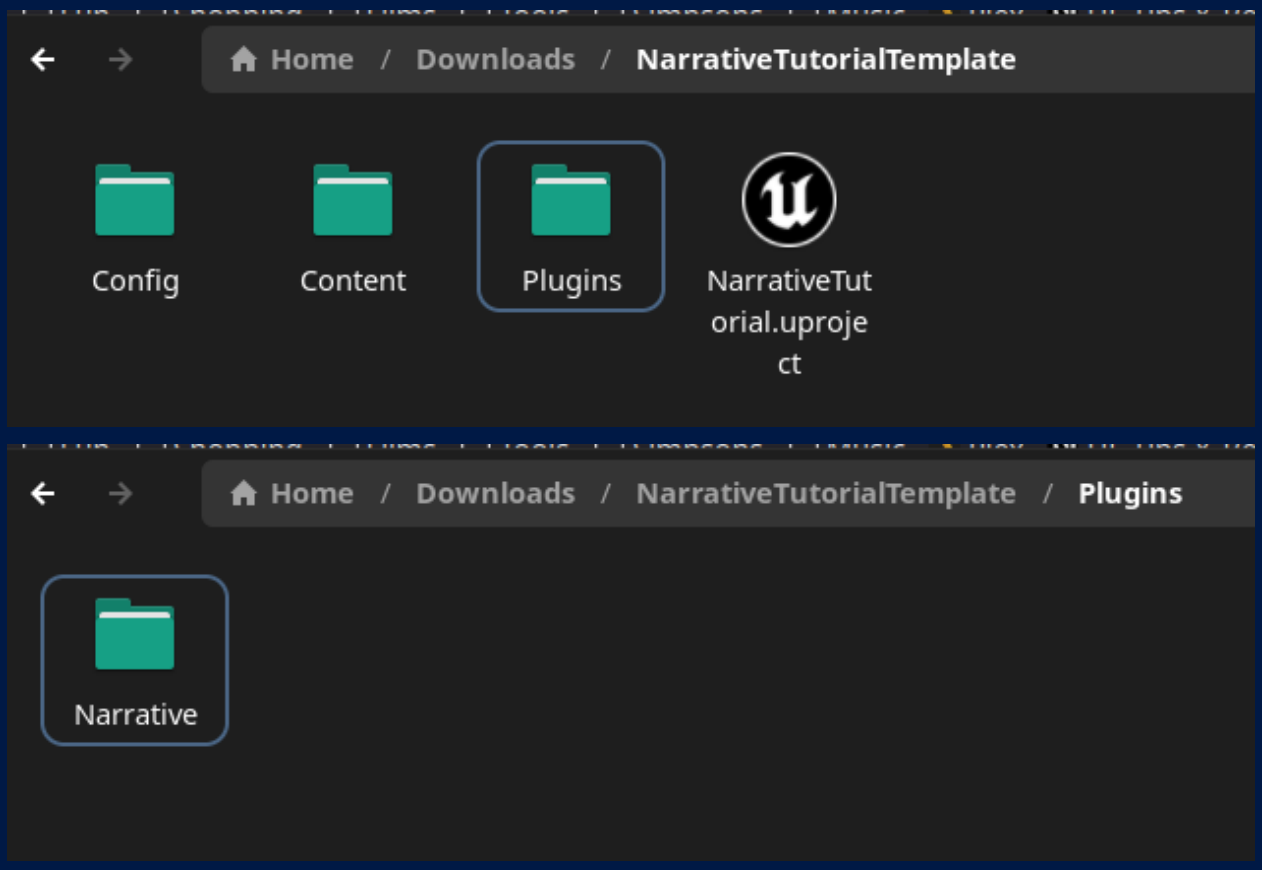
Installation

To get started, we will be using the **Narrative Tutorial Template project** and Unreal Engine version 5.3. This template is a duplication of the Third Person template and contains a bunch of pre-setup features such as Metahumans to make this tutorial easier.

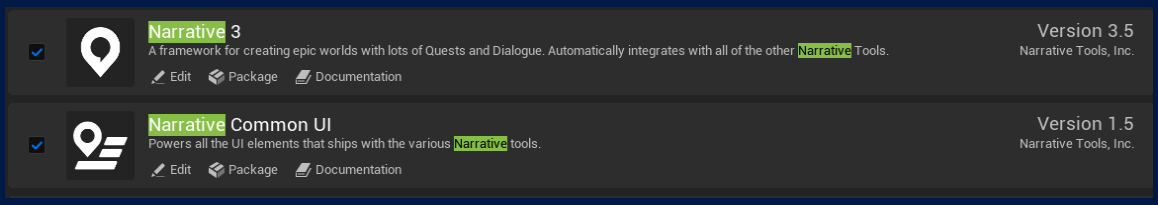
You can get the **Narrative Tutorial Template** here:
https://drive.google.com/file/d/1-Ta3DrBDQqp26Q9f3c5YJgeG1s0kMCqQ/view?usp=sharing

Alternatively, you can download the finished project here:
https://drive.google.com/file/d/1-UaJSERJ7OdpvLxzEYUvn_XCkn7gGo1r/view?usp=sharing

Extract the project then download Narrative from the Epic Marketplace and you can install it to the engine (default) or move it into your project's Plugins folder (you may have to create this folder).



Make sure to enable Narrative 3 and Narrative Common UI by going to edit->Plugins, and checking the box next to Narrative 3 and Narrative Common UI. (you may have to restart Unreal)



We are now ready to use Narrative.

Boilerplate setup

Common UI

Before we can move on using Narrative 3.5 we need to set up the Narrative Common UI plugin.

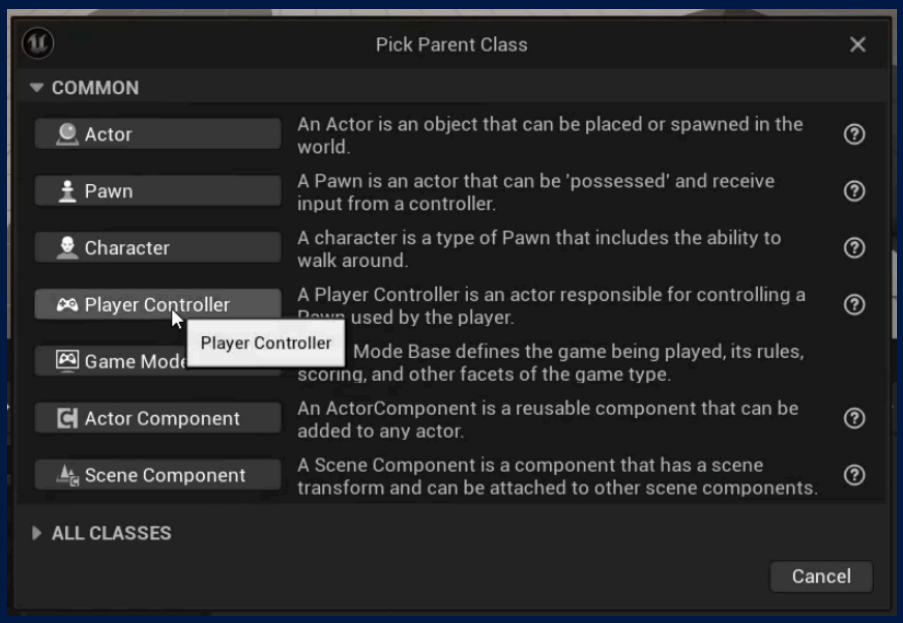
Follow the link below to set up Narrative Common UI.

https://docs.google.com/document/d/13IofD4cZqTpeGOM5x4HSKHWRx7ptJbav4FEnCMBbzQU

If you are using Narrative 3.4, you do not need to install Common UI.

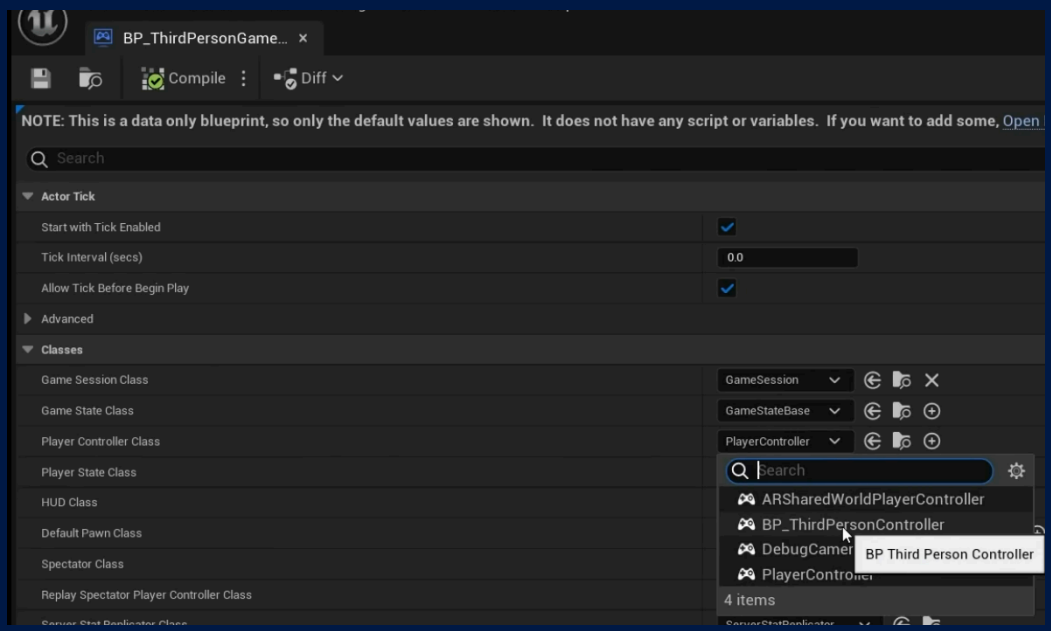
Player Controller

Before we can move on, we need to create a PlayerController. Simply right-click in your Content Drawer, select Blueprint Class and choose Player Controller. We will name this **BP_PlayerController**.

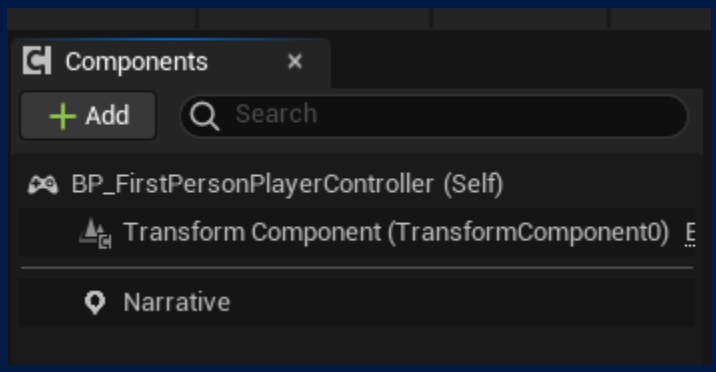


Inside the **BP_PlayerController**, add the Narrative component. If you can't find this, make sure the Narrative plugin is enabled above.

Now open the **BP_ThirdPersonGameMode** and set the **Player Controller class** to your newly created **BP_PlayerController.**



Now open your **Player Controller** and add the Narrative component to the components list.



Narrative UI

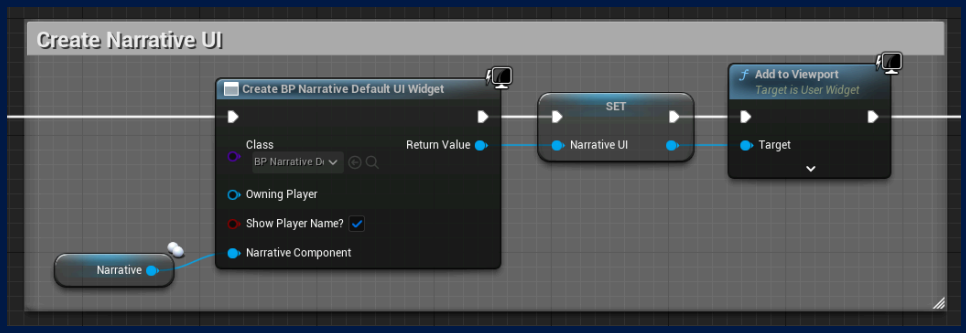
Narrative is packaged with a full UI system to display all the information of quests and dialogue for you to pick up and use right away. It's fully customisable and can be adapted, replaced and removed if you need to. An example of this would be to change how quests are displayed in your game.

Narrative 3.4

To initialize the Narrative UI, go into your **PlayerController** and where you want to create the UI (typically, **Event BeginPlay**) and call Unreal **CreateWidget**. Next, set the **Widget Class** to **BP_NarrativeDefaultUI**. Once it updates, it will require the Narrative component that controls that specific UI. Drag in your Narrative component and connect it.

It's highly recommended that you store this UI in a variable for later use. Drag from the **Return Value** and choose **Promote To Variable**. You can name this NarrativeUI.

Finally, we need to render the UI to the viewport. Drag from the output of the variable and choose **Add to viewport**.

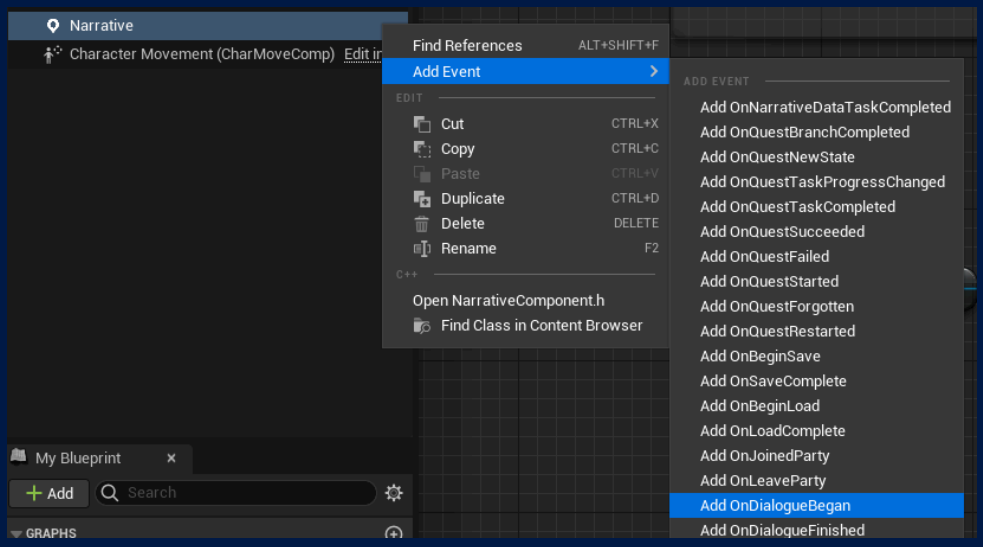


This will now render Narratives UI to the player's screen.

Narrative 3.5

To initialize the new Narrative UI, you will have set-up a HUD as part of the **Narrative Common UI setup**.

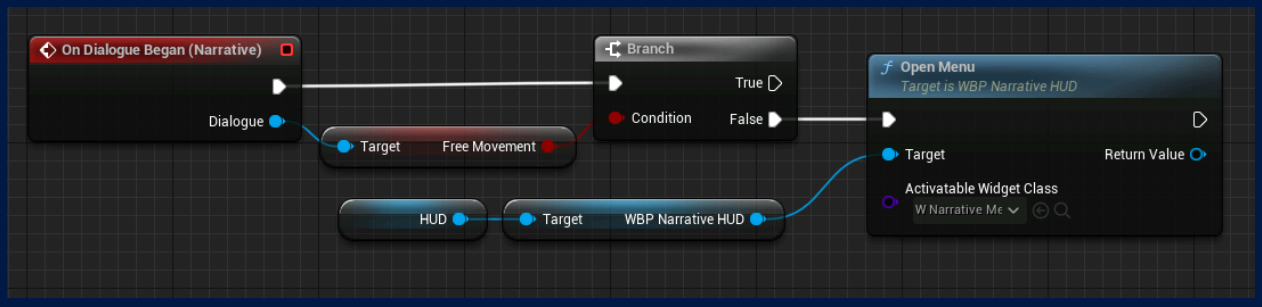
Next, right click your Narrative component, Add Event and select Add **OnDialogueBegan** to add the event into your graph.



Now drag from the **Dialogue** input variable and select free movement. Add a branch to this. Next add in your **HUD** and get the **WBP Narrative HUD** added in the **Narrative Common UI setup** and add an Open Menu call.

Set the **Activatable Widget Class** to **W_NarrativeMenu_Dialogue** option.

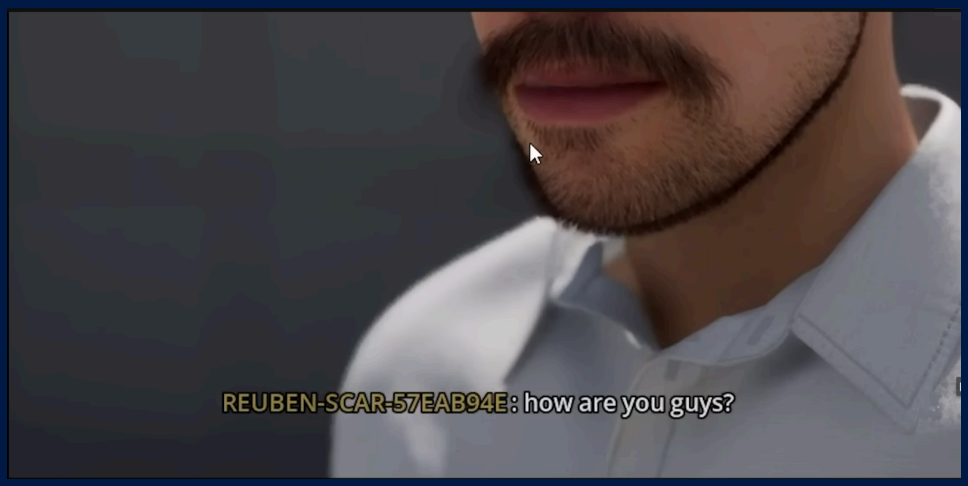
This code enables Narrative to spawn player options during dialogue if the dialogue has **Free Movement** turned off.



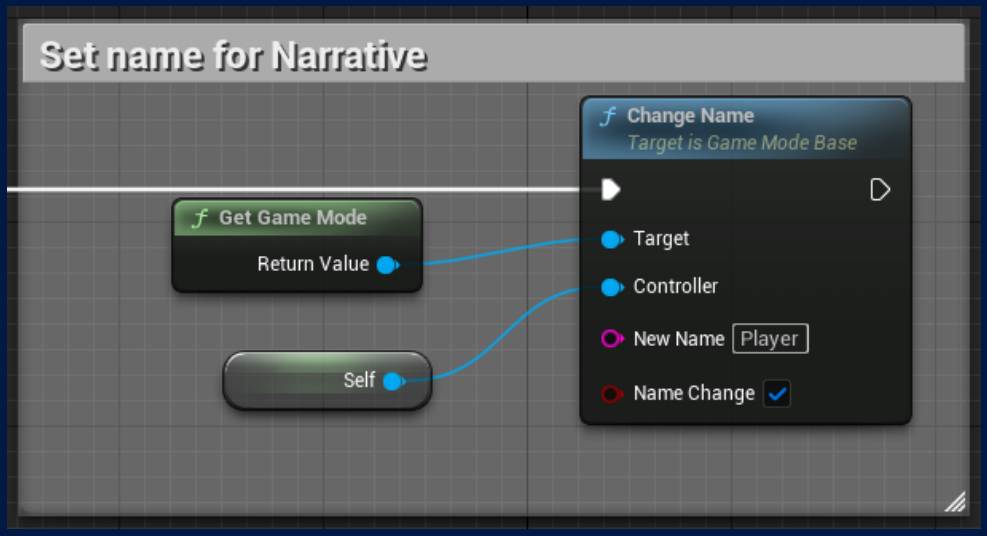
This will now render Narratives UI to the player's screen.

Show Player Name?

The NarrativeUI has an option for Showing the player's name or not. When ticked, the name of the player will be added before all player-selected options. By default, this will use your player's username, which is your workspace name in testing.



To set your default player name you can use the function **ChangeName** in Unreal.

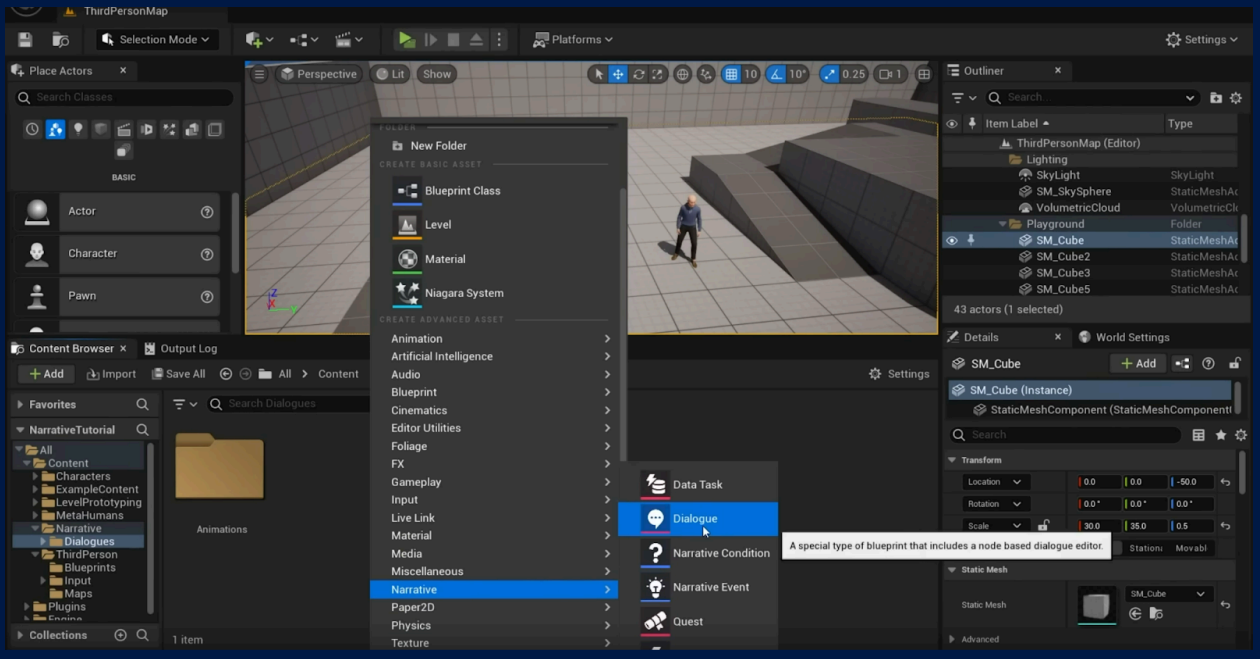


You can see a more in-depth look at **Narratives UI here**.

Dialogue

Creation

Create a new dialogue by right-clicking in the Content Drawer, Narrative and selecting Dialogue.



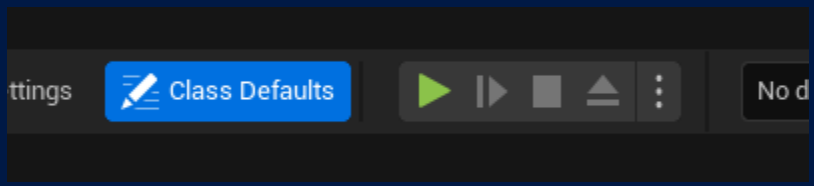
We will call it, Nick. Open this up.

Here you will see two tabs. The **Dialogue Graph** and the **Event Graph**.

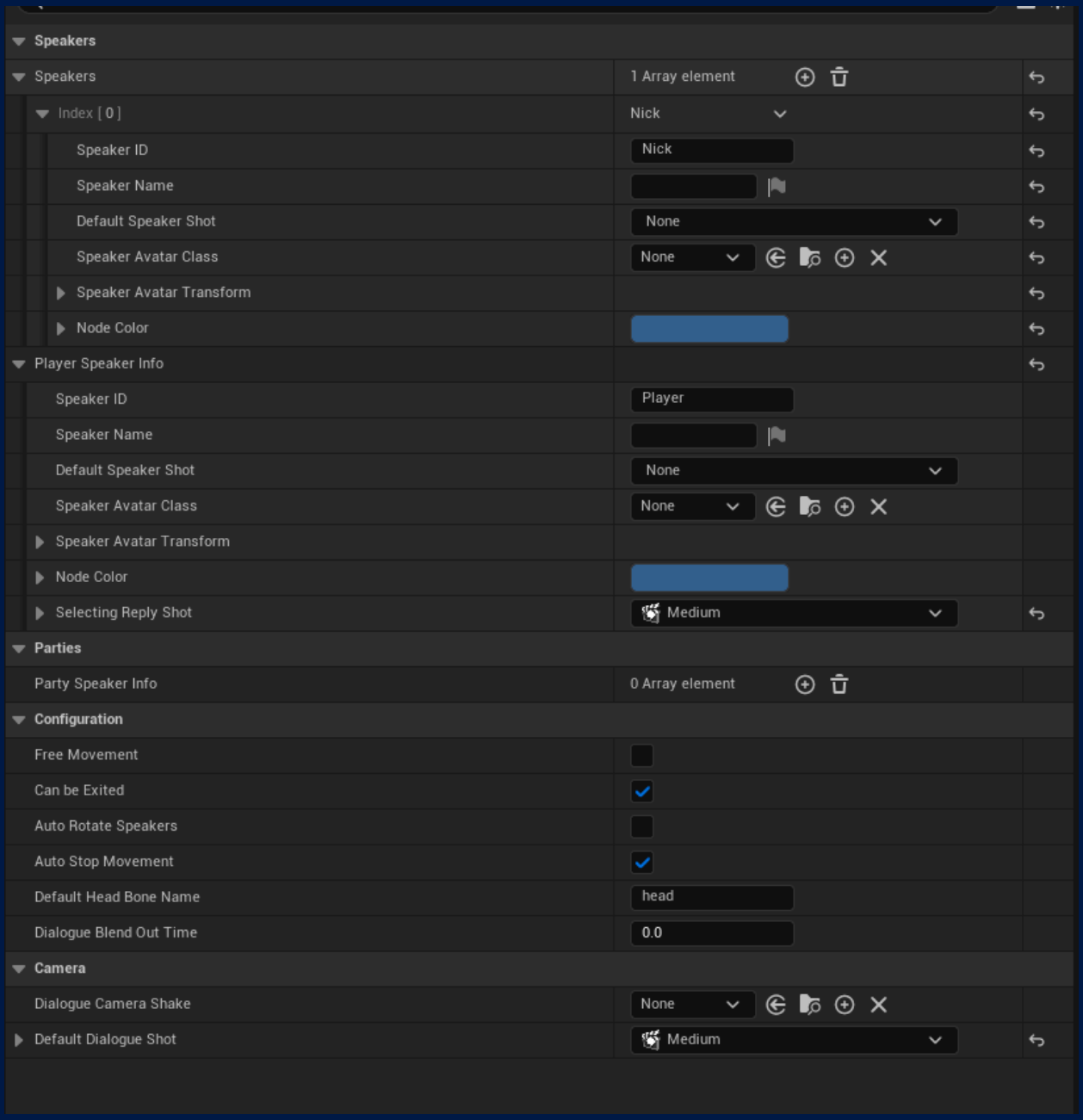
The **Event Graph** is where you can overwrite dialogue functions and add additional code to make your dialogue perform how you want.

The **Dialogue Graph** is how you build up the dialogue with dialogue nodes.

Click the Class Defaults button at the top and we can now populate the Dialogues settings.



The class defaults give you access to modify how your dialogue will work. It lets you change the speakers in the dialogue, camera settings, character settings and more. We'll look into these later in the document.



Dialogue Properties

Each dialogue asset you create inherits from Narrative's Dialogue class.

This provides the dialogue with properties that you can use to control how the dialogue functions.

Speakers

Speakers are the key to dialogue. They store each actor who is within the dialogue (except the player) and their settings. Every actor you wish to be part of a dialogue should be added here.

Speaker ID

This is the ID and the tag of the speaker. This is how the speaker will show in the UI. For Narrative to stand the best chance of finding the right actors, the speaker ID should also be added as a tag to the actor.

Speaker Name

This is the Name of the speaker. This optional field will override the Speaker ID on the UI. This means your Speaker ID can be anything unique and the Speaker Name can be totally different.

Default Speaker Shot

An optional property where you can specifically assign a **dialogue shot** when this speaker is the one talking.

Speaker Avatar Class

An optional property where you can specify a blueprint to be spawned in place of the actor who is representing the speaker (see **speaker ID**). If this is set, the speaker will be hidden and this class will be spawned in its place. Upon exiting the dialogue, Narrative will also destroy this instance and show the speaker.

Speaker Avatar Transform

An optional property to be paired with the **Speaker Avatar Class** property, this is the world position vector in which you want to spawn the speaker.

Node Color

An editor-only option, but you can specify the color of the nodes for this speaker. Can often help when multiple speakers exist to quickly find all instances of them. Note, Narrative will automatically generate a new color for each new speaker added.

Player Speaker Info

A copy of the **Speakers** option but a single option specifically for the player. Contains additional properties.

Selecting Reply Shot

This optional property allows you to select a specific shot for when the player is waiting to select an option.

Free Movement

Check this option if you want the player to be allowed to move around during the dialogue. By default, Narrative will freeze the player during dialogue but this option can be checked to allow them to move around. Useful for following quests or barks.

Note: enabling this option stops the player from being able to select dialogue options because you can't move around in game and have UI control. If you require this, you need to modify your setup so you can select options via keys or some other input method.

Can be Exited

When you try to exit dialogue, you can call the node **TryExitDialogue** or press the exit key (default is ESCAPE). This option blocks these requests which is useful for mandatory dialogue that shouldn't be skipped.

Auto Rotate Speakers

Narrative can take over each speaker and rotate them to the correct positions when speakers are talking.

Auto Stop Movement

It is possible to start dialogue whilst the player still is in motion. This option stops the movement - useful if you have fixed camera dialogue or **Free Movement** unchecked.

Default Head Bone Name

When Narrative is positioning cameras, it will look to aim the camera directly at the speaker's head bone. This property is defaulted to the "head" name however, not everyone has this. This option is how you can select which bone you want the camera to look at.

Dialogue Blend Out Time

If a dialogue shot has been added, by default it will snap back to the camera used before Narrative took over. This option uses Unreal's View Mode Blending to smoothly move the camera back to the starting camera.

Dialogue Camera Shake

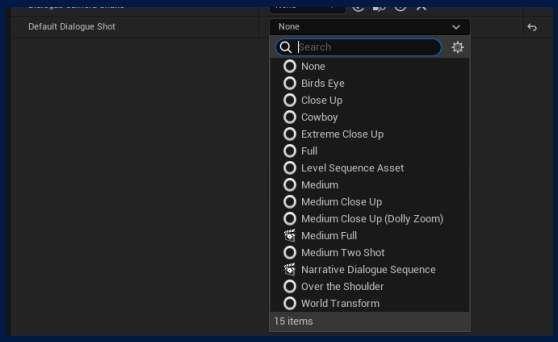
You can use this property to add mild camera shake to your cameras.

Default Dialogue Shot

This property will specify which dialogue shots will be used throughout your dialogue. If none is specified at any level (dialogue, speaker, node), then the camera will remain the one that started the dialogue. See **Dialogue Shots** to view the default list.

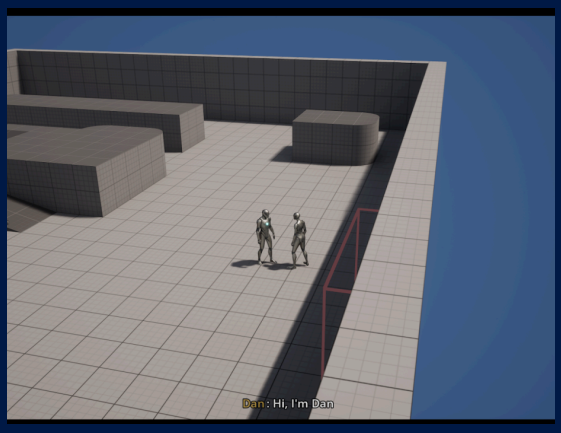
Dialogue Shots

Dialogue shots are a key part of archiving the cinematic look you want for your game. Narrative comes packaged with so many built-in shots for you to use. Narrative's shot system uses Unreals Cinematics to give you maximum flexibility when creating the dialogue you want.



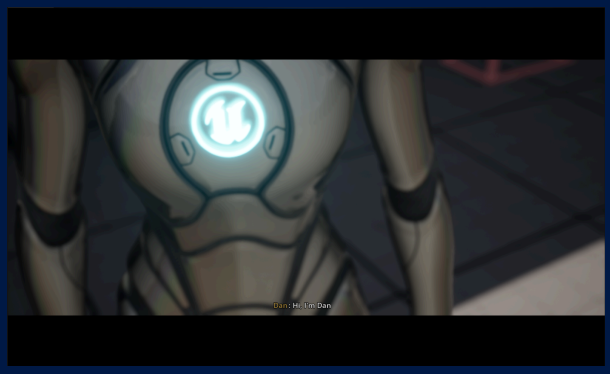
Birds Eye

This shot is used when you want to view two characters from a distance. Imagine a bird flying around looking down.



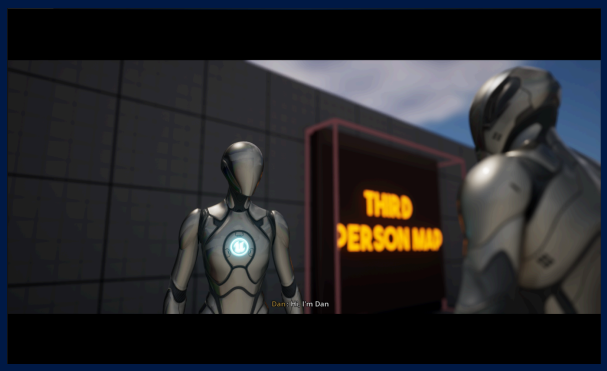
Close Up

This shot is for when you want to look at a specific area on a character.



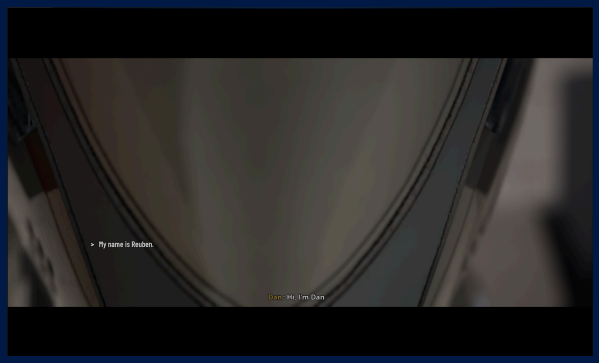
Cowboy

This shot is used when you want to view both speakers at a hip level.



Extreme Close Up

This shot is used when you want to really get into a character's face to show emotion.

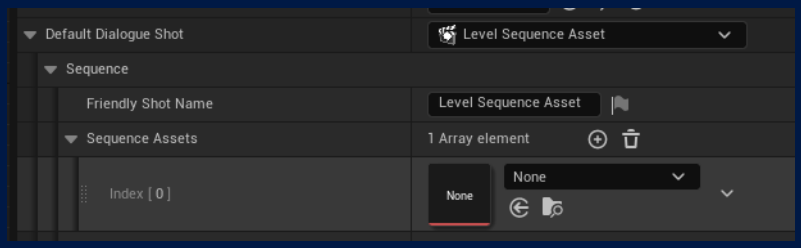


Full

This shot is used when you want to see the full bodies of the speaker and listener.

Level Sequence Asset

This shot is used when you want to use a custom sequence asset you have created. Really useful for cutscene-based dialogue.



Medium

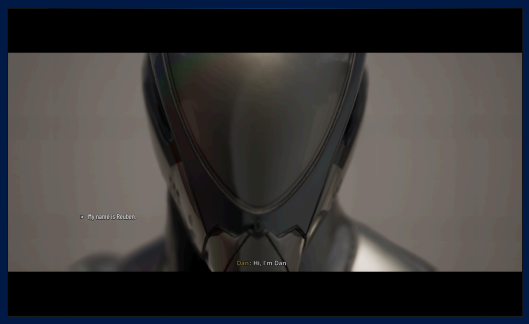
This shot is used when you want to view the body and head of the speaker.

Medium Close Up

This shot is when you are in between the speaker and listener and looking closely at the speaker's head.

Medium Close Up (Dolly Zoom)

This shot is animated. It's the same as the Medium Close Up, but slowly pans in for a few seconds. This is a great example to copy if you want to make your own animated shot.



Medium Full

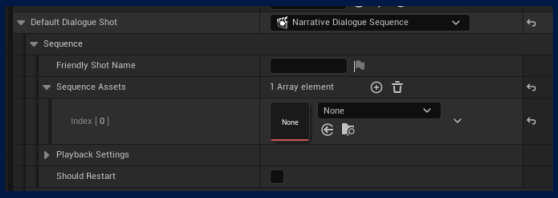
This shot is used when you want to view the body, top of the legs and head of the speaker.

Medium Two Shot

This shot is used when you want to see both listener and speaker up close.

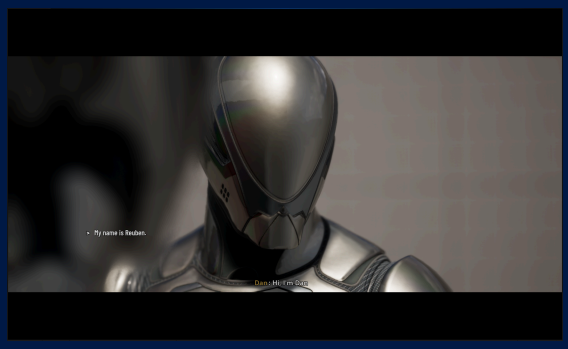
Narrative Dialogue Sequence

This shot is used when you want to use a custom sequence but let Narrative control features.



Over the shoulder

This shot is used when you want the camera to be resting on the shoulder of the listener.



World transform

This shot is used when you want to specify the coordinates of the camera to move it to any position you want. In this case, it has been set to a top-down-style camera.


Dan: Hi, I'm Dan

Custom Dialogue Shots

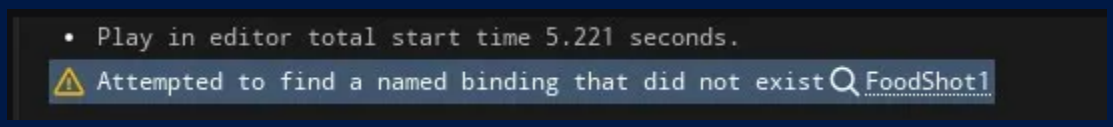
Narrative provides a bunch of dialogue shots you can use to get the shot you want, however, sometimes this just won't work. You may need something specific or different. This is where the power of the sequencer comes in.

You can add your own custom sequences to get the exact shot you want.

Named binding error / warning

When creating your own custom shots, if you have a camera within the shot it's common to receive a warning regarding a named binding.

Warning: Attempted to find a named binding that did not exist [Dialogue Name]

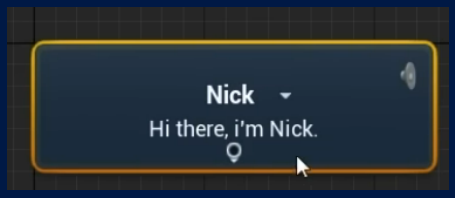


This is easily resolved by right clicking on your Cinecamin the sequencer, and adding a tag of **Cinecam.**

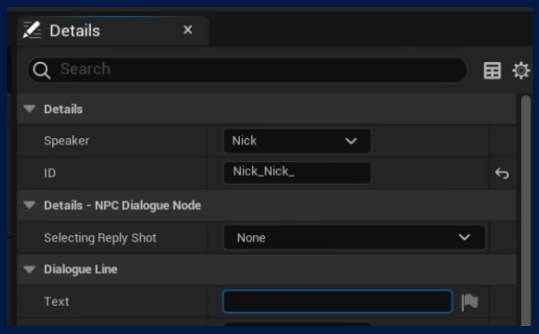
This tag is important as it allows Narrative to locate the camera and add any specific features, aim it in the right location and apply any transitions required.

Dialogue Nodes

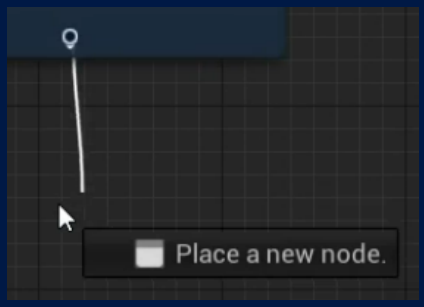
Inside the **Dialogue Graph** you will see the starting node for Dialogue.



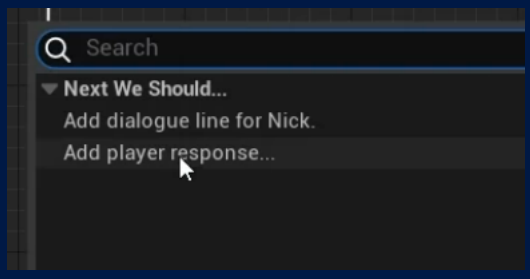
Typically, it's more flexible to remove the starting text from this node. Click the node and in the details panel, remove the Text.



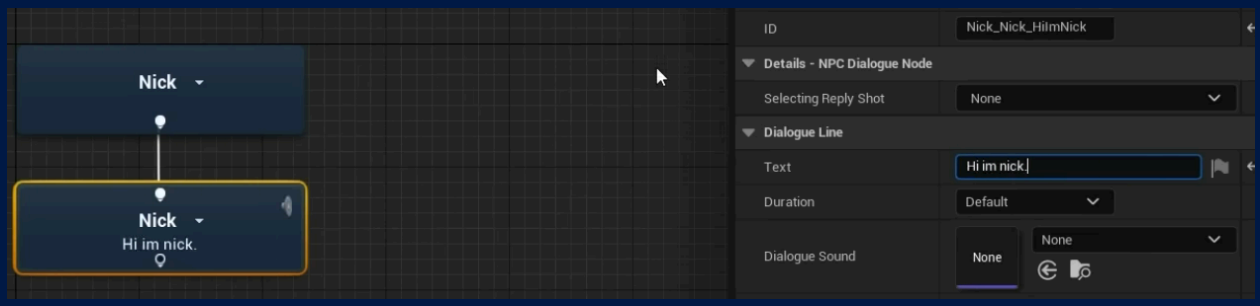
Now, if you click and drag from the white dot via the left mouse button:



You will be able to draw a path to place a new node. When you let go you will be presented with a menu of speakers you can add nodes for. If you select **Add dialogue line for Nick** it will create a new node for our speaker, Nick.


Search
Next We Should...
Add dialogue line for Nick.
Add player response...

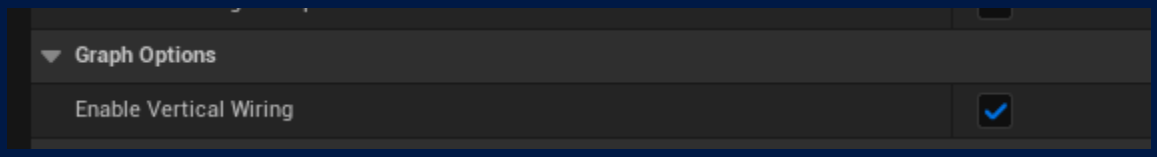
Click this new node and set the text. We will set it to **Hi I'm Nick.** You can drag this node into any formation you like to make the dialogue easy to follow.



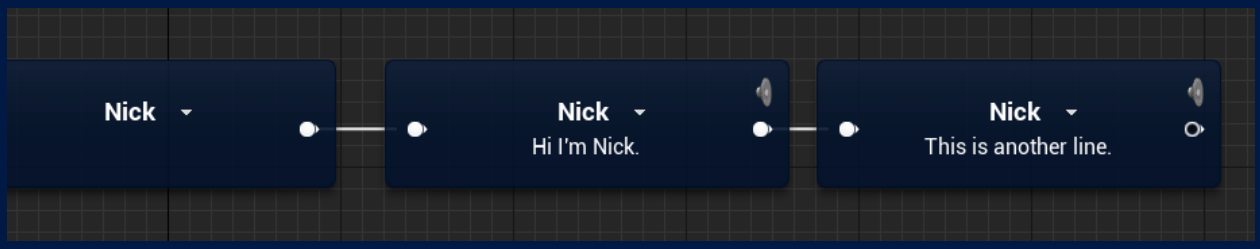
Vertical & Horizontal Nodes

By default Narrative will render the dialogue vertically but you also have the option to tell Narrative to render the dialogue Horizontally.

You can do this by going to **Edit** -> **Project Settings** -> **Narrative Dialogues - Editor** then unchecking Vertical Wiring.

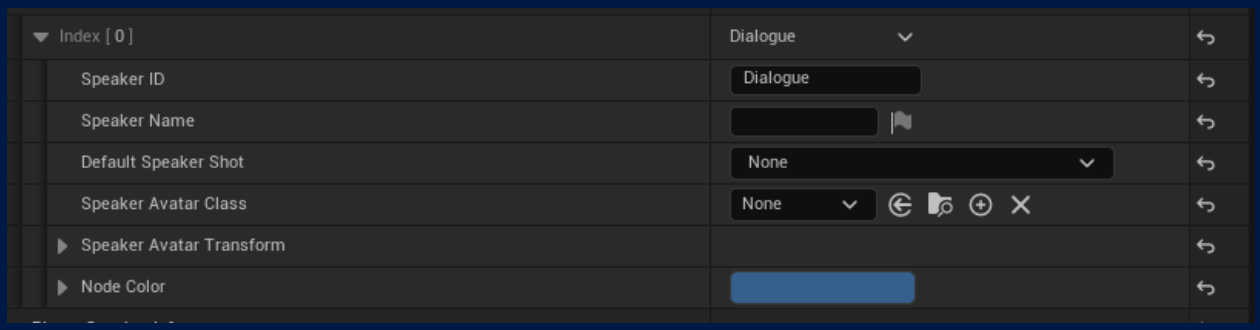


Changing this will result in the dialogue looking like the below:

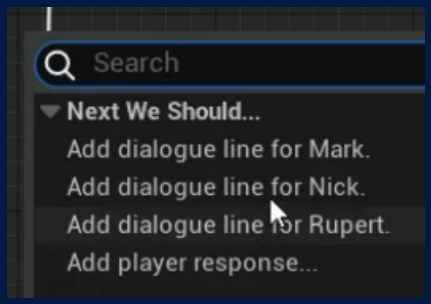


Adding new speakers

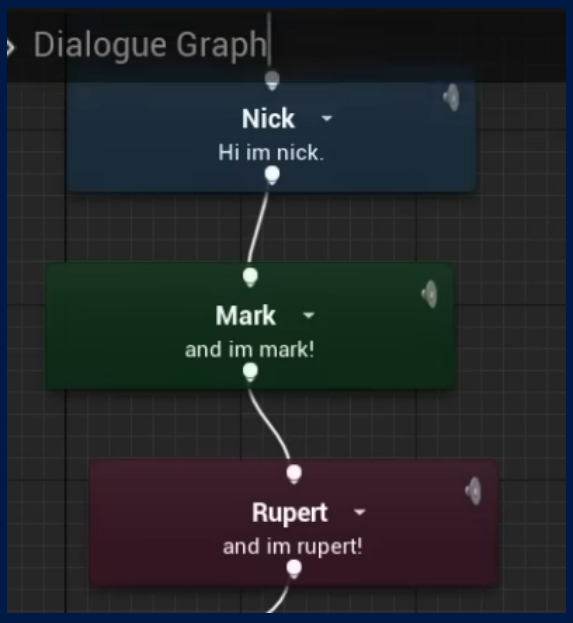
Go to the **Class Defaults** and duplicate or add a new speaker. We will call ours **Mark**. We are also going to add a third speaker called **Rupert**.



Adding new speakers will give your dialogue more NPCs who can talk.



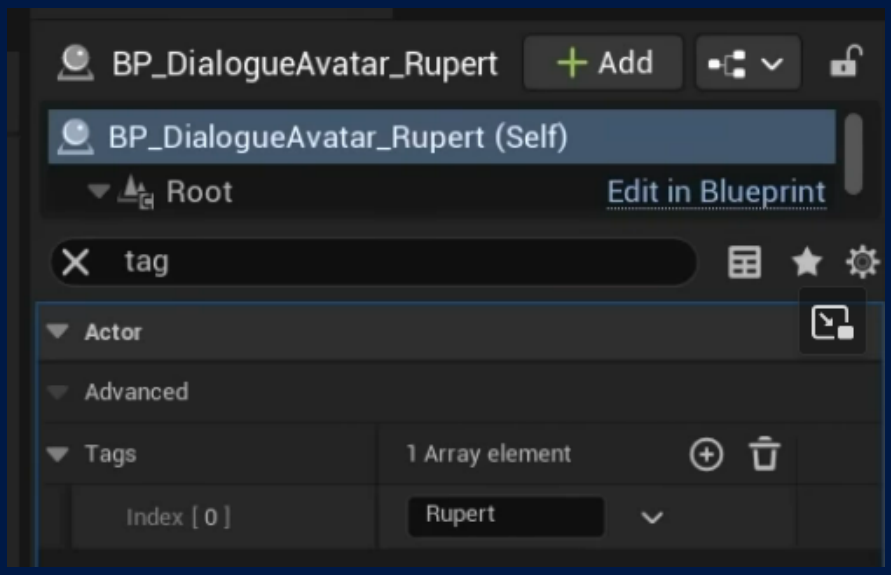
You can add more nodes simply by right-clicking or dragging from the white dots.



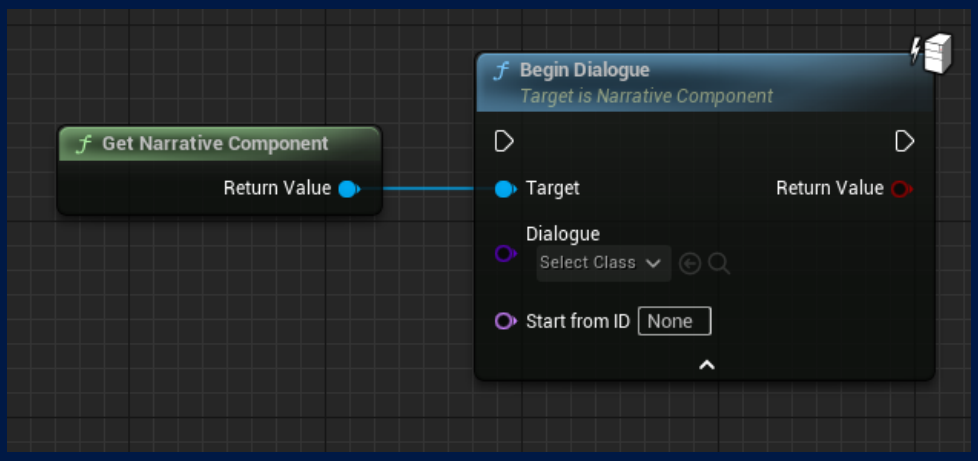
Finding the Speaker

One important feature of Narrative's dialogue is the method in which it tries to find the speaker. Since Narrative only spawns the dialogue as and when it needs it, you cannot reliably store references to world actors which could be removed, spawned in or not exist when they need to.

Narrative will try to find a **Speaker ID** in the **Tags** section of your actors. Simply add a tag to your character that is the same as your **Speaker ID**.



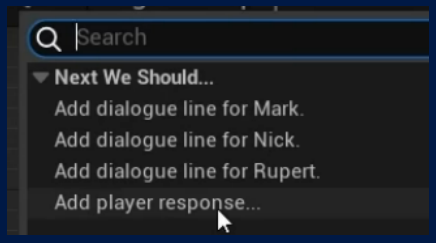
If Narrative fails to find an actor with the tag of the **Speaker ID** then it will use the **Default Speaker Avatar** when you call **BeginDialogue.**



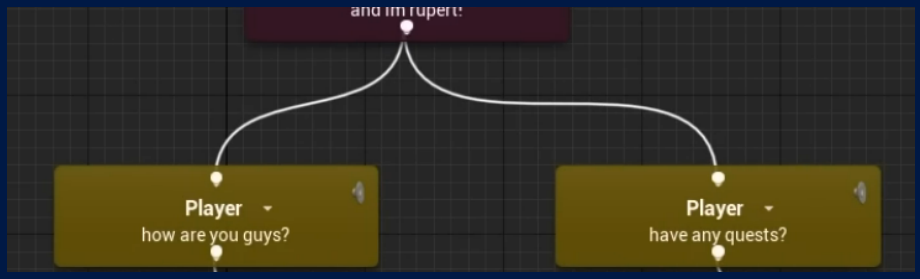
Note, if you are wanting to play animations, you will need to add extra tags to your characters. See the ***animation section*** *for more info.*

Player responses

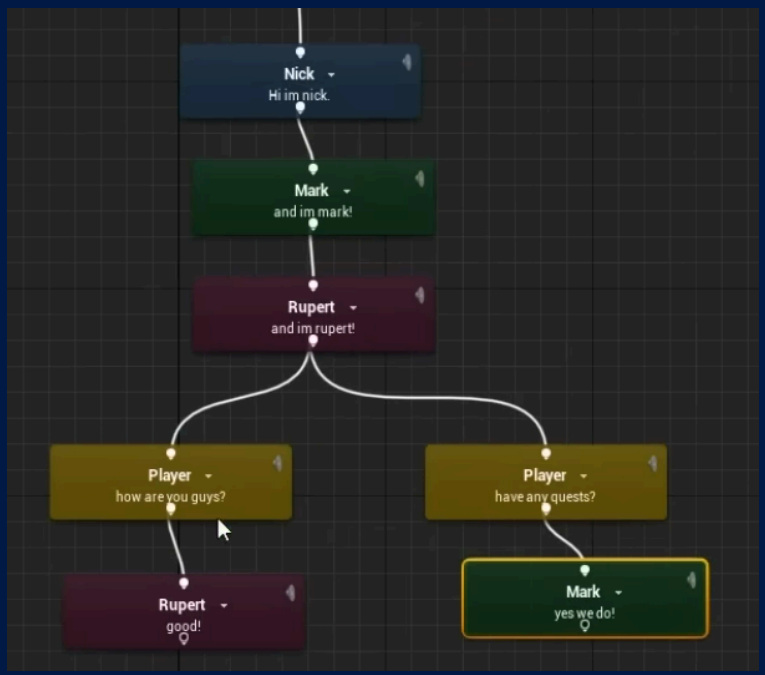
You can add player responses by simply clicking on **Add Player Response**.



A player response is where the user will select an option. The camera will focus on the player's character and display a dialogue selection UI for the user to make their decision.



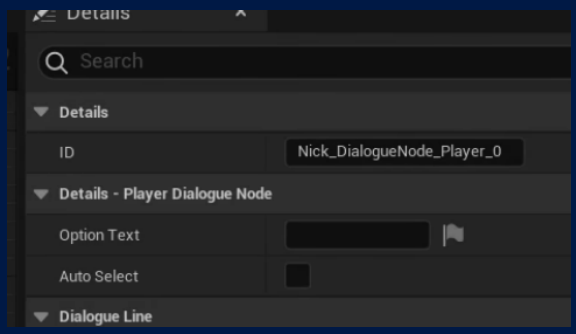
Using these simple methods you can build up dialogue to progress the story in your game.



Auto-selecting player options

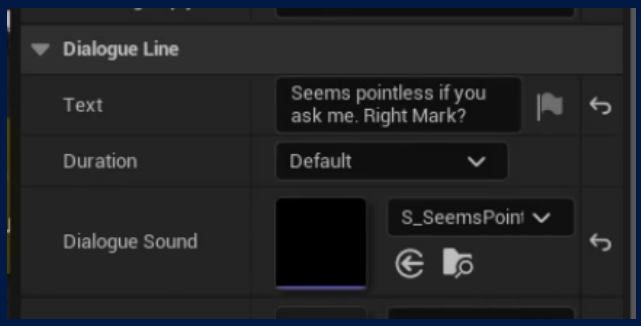
Player options have an additional property to allow them to be auto-selected. This will tell Narrative to skip rendering the UI and select the first Player option that is valid and can be taken. (See conditional dialogue below)

This is a great option for when you want the player's character to say something but the user does not have to select it.



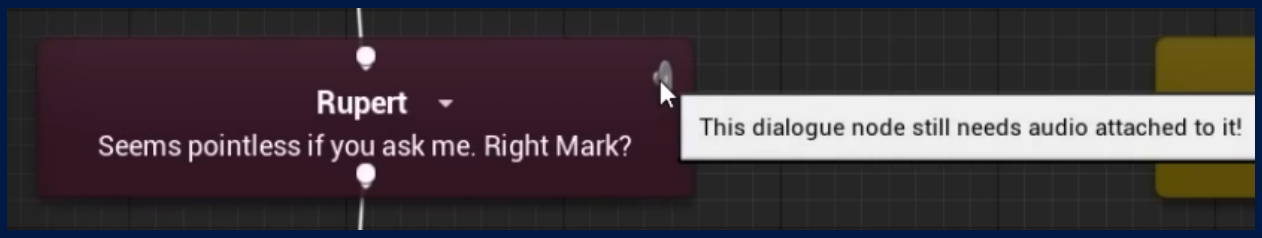
Audio

You may wish to use audio when your character speaks. Narrative has audio built in so you can assign a **Sound Base** file to each dialogue node to play when the node is reached.

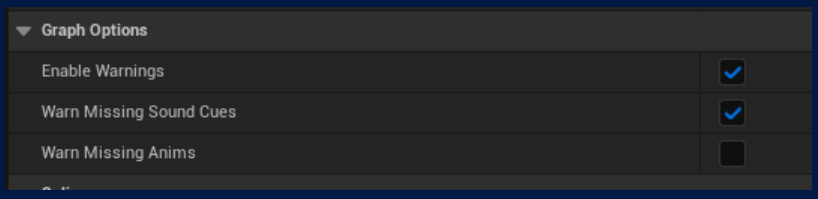


Audio missing error

Narrative will display an icon when a node is missing audio. This is a helpful hint to tell the user when they have missed it.

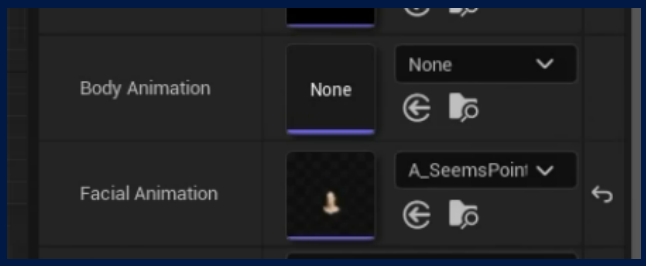


This can be disabled by going to **Edit** -> **Project Settings** -> **Narrative Dialogues - Editor** and unchecking **Warn Missing Sound Cues**



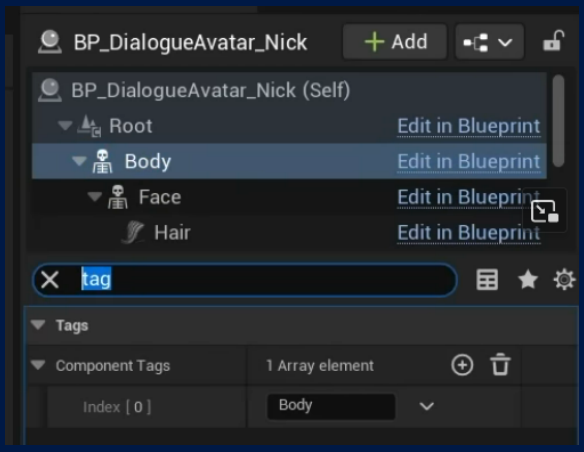
Animations

Narrative also has the functionality to play animations on both the body and face of your character. This is useful for when you want a character to play an idle breathing animation, but talking using facial mocap. This is optional and will not stop the dialogue from working if they are not populated.

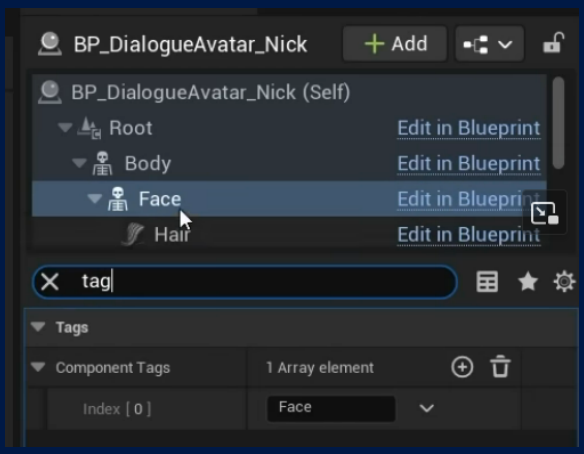


A key thing to note is that Narrative will *try* to find the correct meshes to play the animations on. However, it's not always easy. The best way to avoid any headaches is to tag your characters so Narrative understands.

Open your character blueprint and select **Mesh**. This will typically be the character's body. In the details panel, search for **Tags** and set a **Component Tag** of **body**.



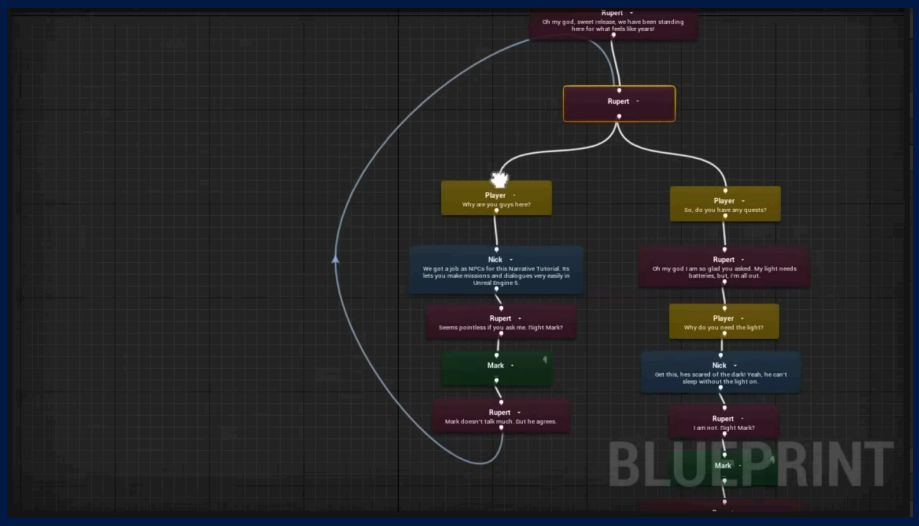
Now select the **Head Mesh** and add a **Component Tag** of **head**.



Setting these two tags will ensure Narrative knows exactly which Skeletal Mesh to play the animations on.

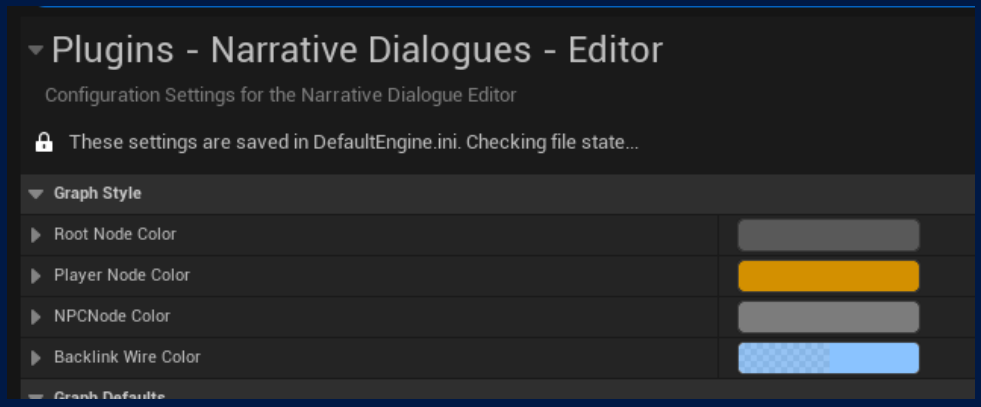
Backlinking

It's highly common for dialogue to often return back to a set of nodes. For example, you could be questioning an NPC and after you have selected the first question, it should take you back to the list of questions. Backlinking is the feature where you connect a lower node back up to the parent to redirect the player.



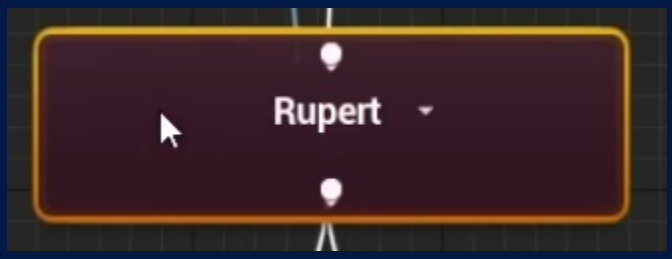
Color

You can change the colors of Narratives backlinking and other nodes from the project settings by going to **Edit** -> **Project Settings** -> **Narrative Dialogues - Editor** and changing the colors.



Empty nodes

Empty nodes are useful to keep so you can better connect Dialogue. It's often used before a list of player options so you can connect multiple entries into a single node instead of every player option, but it can be used to also just make the dialogue more readable.



Collapsed nodes

Sometimes empty nodes can be used to organise dialogue. You can tick a node's **Collapsed** option in the details in order to change it into a smaller, closed version in the editor.

Copy and pasting nodes

Dialogue can be a time-consuming task to write and often, writers will not use Unreal to write the story and script. Narrative supports a copy-and-paste standard to help create dialogue faster.

You can create Dialogue simply by copying the speaker ID and text in the following format.

SpeakerID: Text

For example
Reubs: Hey! Welcome to Narrative
RandomJoe: Thanks!
Player: Who is RandomJoe?
Reubs: It's Joe! You know Joe!

Narrative will take this and automatically create the speakers if they do not exist and join all the dialogue the best it can.

Narrative does not let you import from Excel files, CSV's or any other media.

*Note, that copying and pasting **existing** nodes have been disabled in Narrative dialogue until further notice. It was proving to be problematic and until it can be provided up to the Narrative standard, it has been removed.*

Has Dialogue Available

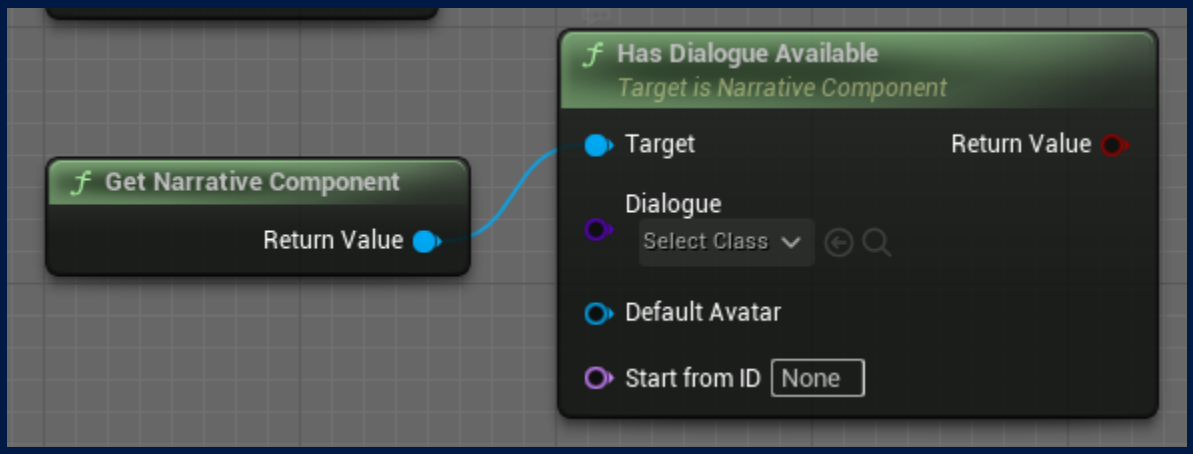
Whilst creating dialogue it is common to have dialogue assets that are not used anymore due to **conditions** blocking all routes - and this is okay.

Sometimes you want to be able to check if a dialogue can be started before calling **BeginDialogue**.

This is where the function; HasDialogueAvailable can be used.

The HasDialogueAvailable executes the initialize method of the provided dialogue and does not start the dialogue. This will return you a boolean to whether calling **BeginDialogue** would successfully enter the dialogue.

Simply provide the **NarrativeComponent**, Dialogue, Default avatar (optional) and StartFromID to have a boolean returned.



Starting Dialogue

There are so many ways to start dialogue and it all stems from calling the **BeginDialogue** function.

Each method of starting Dialogue has the same 3 parameters.

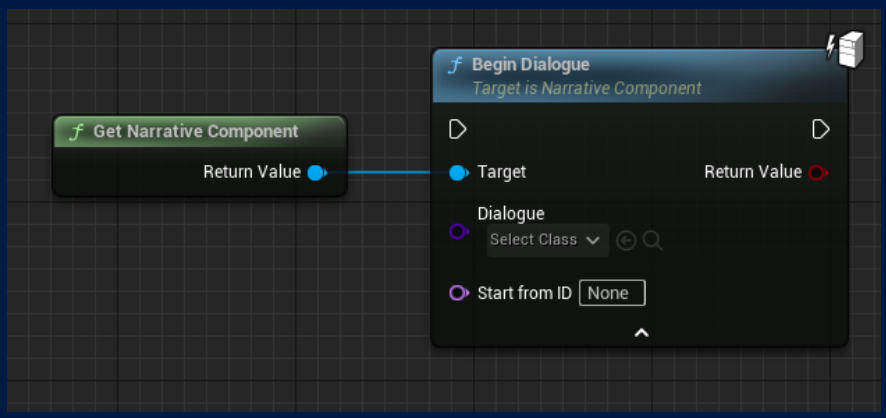
Dialogue - the dialogue class you wish to invoke.

DefaultNPCAvatar - the NPC class that started the dialogue

Start from ID - the ID of the node you wish to start from. None means Narrative will start from the root node. Providing the ID will make Narrative (if possible!) go to that node and start from there.

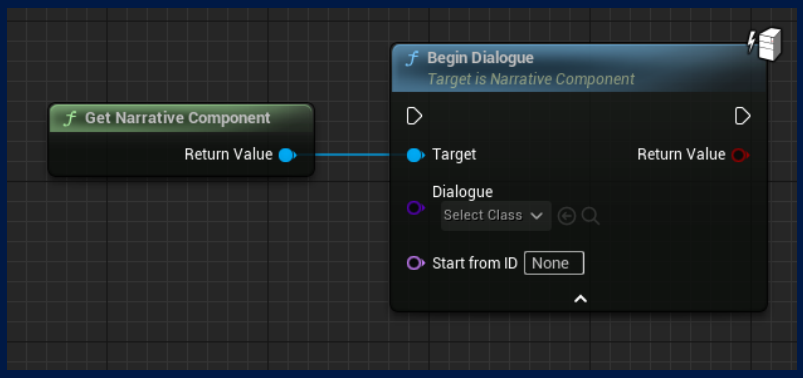
Narrative Component

If you have direct access to the Narrative component (if you are starting a dialogue on your PlayerController for example), then you can just drag the Narrative component in and choose **BeginDialogue.**



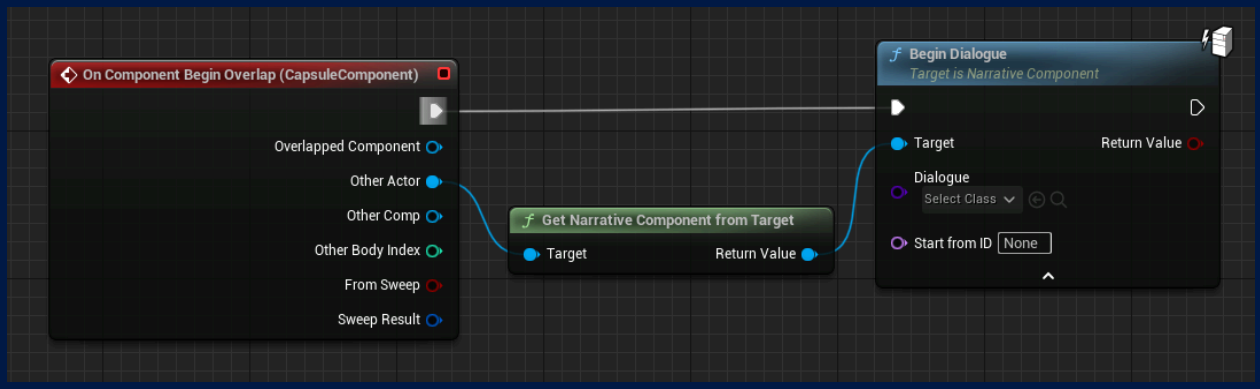
Get Narrative Component

However, more often, you won't have direct access to the component. In this case, you can just use the helper function **GetNarrativeComponent** and then add **BeginDialogue.**



GetNarrativeComponentFromTarget

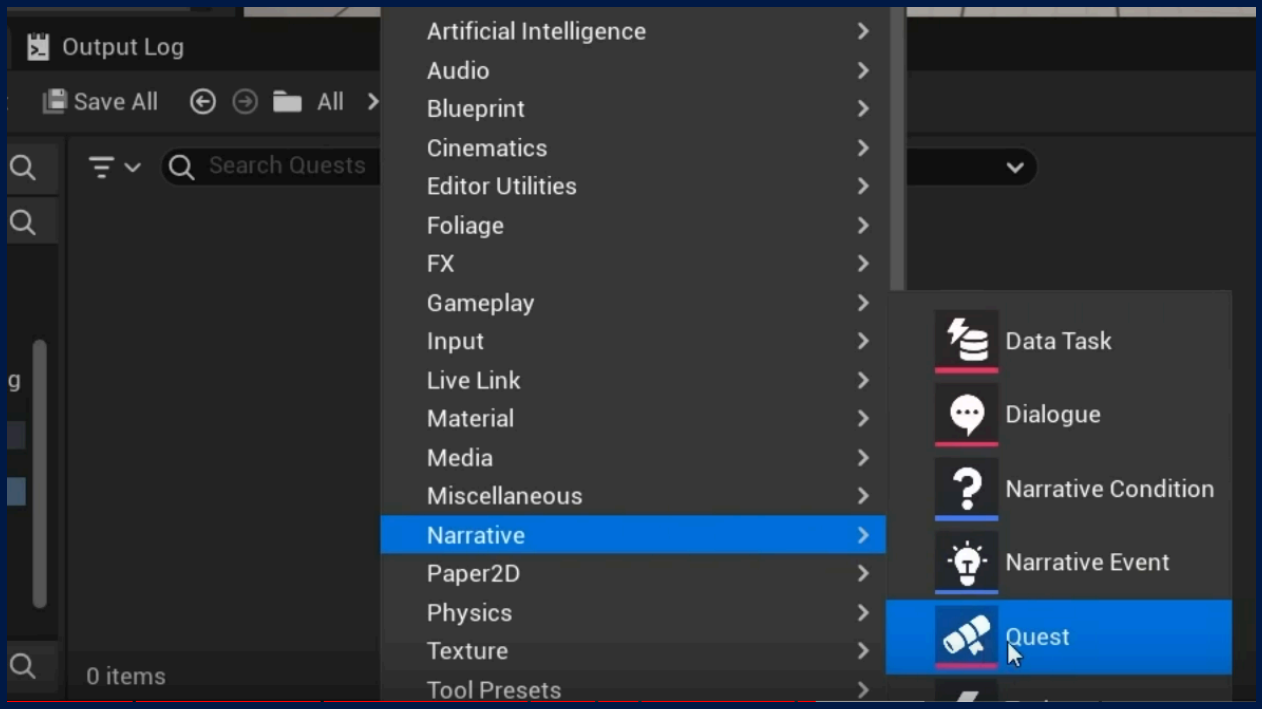
Another method you can use to get Narrative which can be more useful for multiplayer games is to use the **GetNarrativeComponentFromTarget**. This is useful when you have multiple players, each with their own Narrative to track their own quests.



Quests

Creation

Let's create our first quest by right-clicking in the Content Drawer then hovering over Narrative and selecting **Quest**.



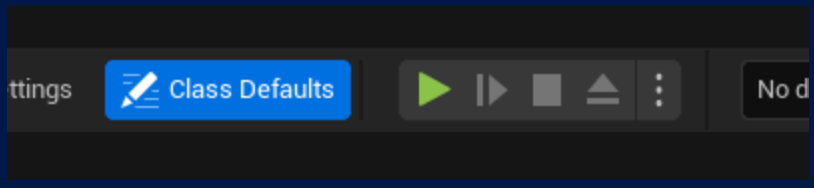
In this example, we'll name it **QB_BlindedByTheLight** and open it up.

Here you will see two tabs: the **Quest Graph** and the **Event Graph**.

The **Event Graph** is where you can overwrite dialogue functions and add additional code to make your dialogue perform how you want.

The **Quest Graph** is how you build up the quest with the branches and states.

Click the **Class Defaults** button at the top and we can now populate the Quest's **Name** and **Description**.



Copy and pasting nodes

Note, copying and pasting nodes have been disabled in Narrative quests until further notice. It was proving to be problematic and until it can be provided up to the Narrative standard, it has been removed.

Quest Logic

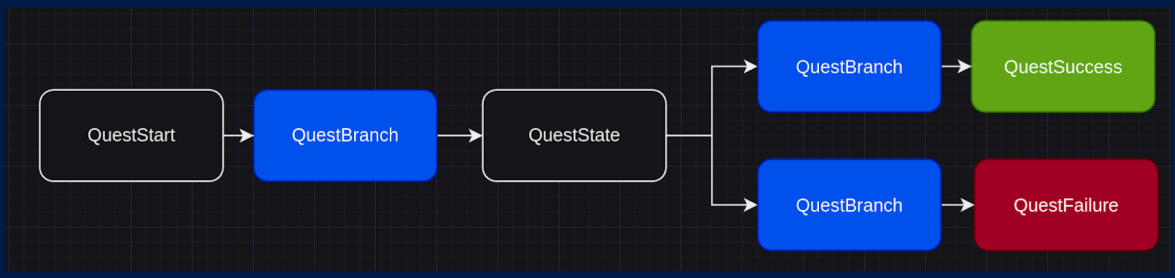
In Narrative, quests are made up of branches and states.

States are points within your quest where you are waiting for the player to complete the next task.

Branches are the tasks within your quest that you need to complete. A quest can have multiple branches connected to a single state.

Each quest will start with a State (*QuestStart*), a series of branches and connected states, and then it will end with a State (*QuestSuccess* / *QuestFailure*)

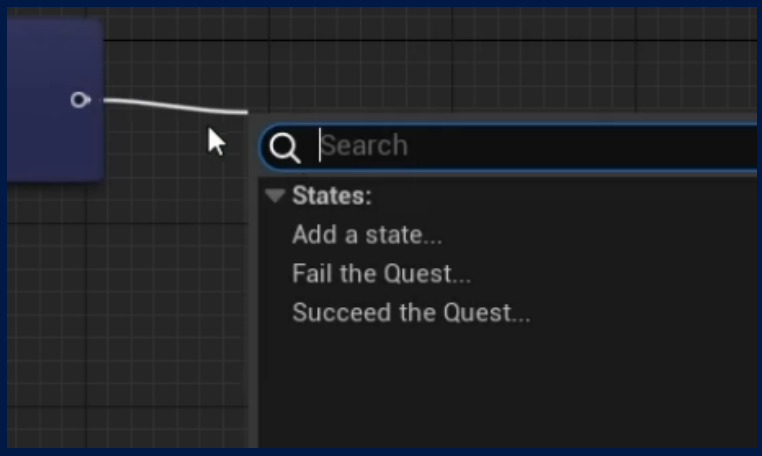
A state can also have multiple branches to create different paths in the quest. We will explore this more later.



States

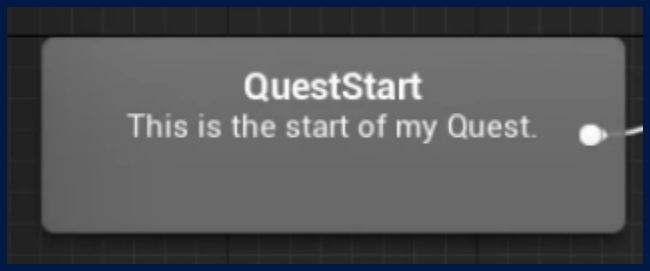
Narrative has 4 states you can use to create quests. These states are positions within the quest where you are waiting for the player to action the next task. The player might be presented with 1 or more tasks but at least one must be completed to move to the next state.

States must be connected to a branch and cannot be connected to other states.



Quest Start

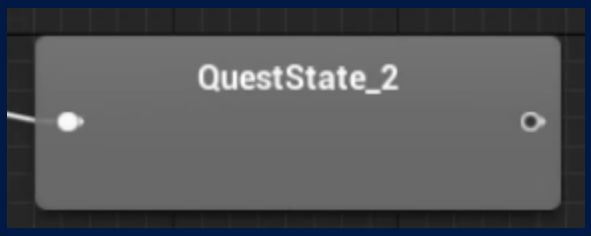
This state does not have any input nodes and is typically the starting point for a quest. It always exists and cannot be deleted. Only one can exist per quest.



Quest State

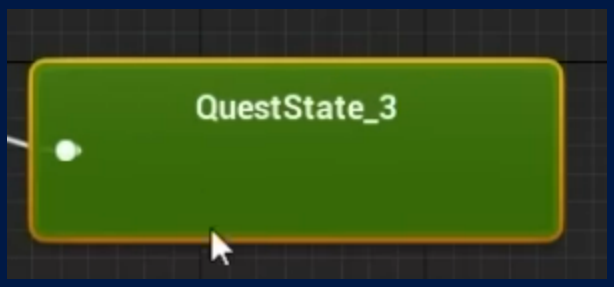
This state is used after every branch. This is the position within the quest that the player currently sits. Used often in conjunction with the condition **IsQuestAtState** to conditionally use a dialogue node if the player has reached a certain point. It has 1 input node to come from a branch and 1 output node to connect to one or more branches.

This state is how you create branching quests.



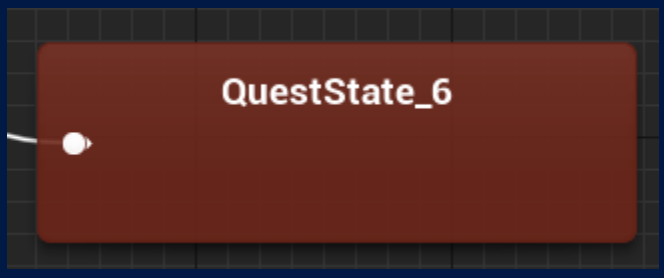
Quest Success

This state is used when you want to mark a quest as successfully complete. Once this state has been reached, the quest will no longer action any other tasks without you forgetting the quest. This can be used in conjunction with the condition **HasQuestSucceeded**



Quest Fail

This state is used when you want to mark a quest as failed. Any task can connect to the failed state whether they have passed the task or not. This means you can create tasks to fail such as a task to **Reach the castle before Reubs** does which connects to the success state and then a duplicate task with NOT ticked of **Reubs reaches the castle first** which then connects to the fail state.



Branches

Default Properties

Quest Tasks

The Quest Tasks property is an array of tasks you want to complete in this branch. Every task added in here must be complete (unless optional) in order to allow the branch to continue to the next state. The tasks are treated as an AND logic operator and can be any mixture of tasks you need. An example could be 3 tasks added which are:

- PickupItem - collect the Blue key
- PickupItem - collect the Red key
- PickupItem - collect the Green key

ID

This is the ID of the branch. Often automatically generated by Narrative, but sometimes replacing the ID with something memorable can help with debugging and any custom game logic you require.

Description

This is the description of the branch not to be confused with the tasks description. In the above example for finding 3 keys, the branch description would be “Find the 3 keys”. It is then up to each task to provide the description “The red key is in the Orc camp”.

Hidden

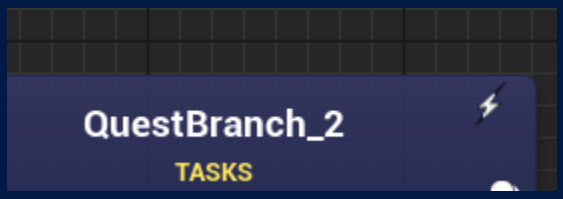
This boolean is used in the UI to display or hide the quest branch. If hidden is ticked, it will not update the UI to inform the player. Highly useful for hidden objectives that are used to control the quest without confusing the player. For example, the description might be - go to the final location but is actually the first door. Upon reaching the first door, the objective is hidden and just has an event to open the door and do other logic.

On Entered Func Name

Narrative Events will be discussed latest, allowing you to run generic code against a node. However, a generic event is not always useful when you might want to run some code only once or very specific to a quest. You can double-click a branch node to add custom code to that single branch.

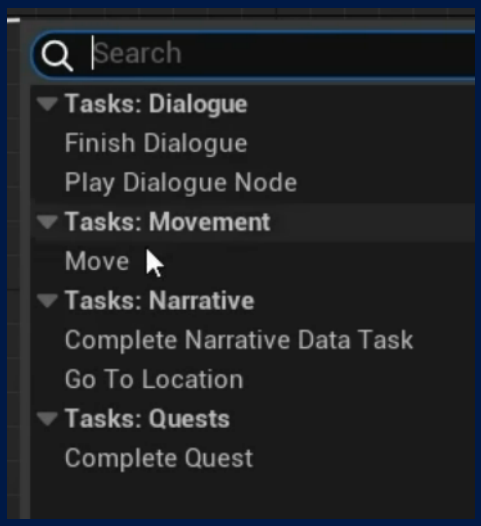
This variable stores the function name of the OnEntered function assigned to this

branch but this is also indicated by the lightning icon on the node when an entered function has been added.



Tasks

Narrative comes with some basic tasks pre-defined to help you get started. These tasks can be common ones that Narrative can use or ones that are not game-dependent. Other tasks that are considered common often rely on game logic that is specific to each game meaning Narrative cannot include it by default.



Default Properties

All tasks inherit from NarrativeTask which gives them some default properties they can use.

Required Quantity

This property represents how many times the task must be completed before it can move on to the next task and be marked as successful. A task will typically use the **Add Progress** node to add to the current quantity. Upon reaching the required quantity the task will be complete. You can also skip this and just set the **Add Progress** to the Required Quantity if you require.

Description Override

By default, task descriptions such as **Go to the farmers market** will be auto generated if the **GetTaskDescription** function is overridden within the task. However, in some cases, this will not work for your situation. So instead of changing the function to handle all cases, you can use this field to override the description.

Optional

This boolean field sets the task to optional. Any optional task will be skipped when all the non-optional tasks have been completed.

Hidden

Compared to the hidden property on Branches, this hidden flag only hides the task instead making it not show on the UI.

Tick Interval

If the task overrides the function **Tick Task**, this interval is how often the tick will be called. By default, it is set to 0 and will be ignored. However, if you set it to anything more than 0, it will be called by an Unreal timer on loop.

Default Functions

Each task has functions it can override by default. These functions are called by Narrative at specific intervals to make the task work.

BeginTask

This function is the first function that is called on a task. This is the most common place you will add logic to your task.

EndTask

This function is the last function that is called on a task. It is useful for cleaning up anything the task did such as removing event dispatcher binds. This is called when the task is finished; whether it completed or skipped (the player went down another branch)TaskTick
This function is a replacement for Unreals default Tick event and can be used if a task needs to frequently check values that cannot be event-driven such as arriving at a vector. The iteration is set in the class defaults or overridden from the **Quest Graph**.

OnTaskCompleted

This function is called when a task has been completed. It differs from the EndTask as this will not be called if the task is skipped. Only if it completes its required quantity.

GetTaskDescription

Override this function if you want your tasks to automatically generate objectives. For example, the GoToLocation task overrides this and returns **Go to the {location}**.

GetTaskProgressText

This function handles how the progress update message is shown to the user. Typically used when a quantity more than 1 is used. The default will be the TaskDescription with (currentQuanty / requiredQuantity). For example, **Find the apples (5/10)**

GetTaskNodeDescription

This function can be overridden if you want the Quest Graph UI in the editor to display something different. Useful for helping devs quickly understand what a task does.

Default Tasks

Finish Dialogue

This allows you to create objectives such as “Go and talk to Reubs” and upon exiting the dialogue with Reubs, this will trigger this objective to finish.

Properties

Dialogue

This is the dialogue which the task needs to check to complete the task.

Add Waypoint To Avatar?

Whether or not Narratives built in Waypoint be added to the actor's location so the player can track them on screen.

Play Dialogue Node

This allows you to create objectives with dialogue, but unlike the Finish Dialogue task, this task will be completed when a single dialogue node is played. Consider RPG games such as Skyrim or Witcher where you can complete quests in the middle

of dialogue without leaving it. Objectives such as “Inform Reubs that you found the item”

Properties

Dialogue Node ID

This is the dialogue node ID that the task needs to wait for to complete the task.

Retroactive?

Whether or not the task should look at previously completed nodes before completing this. Useful if you want to skip a task if the player has already done it.

Add Waypoint To Avatar?

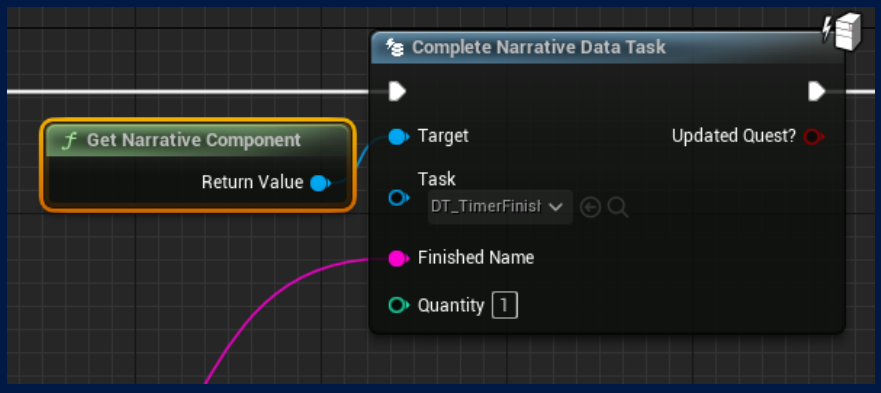
Whether or not Narratives built in Waypoint be added to the actor's location so the player can track them on screen. This will only work if the dialogue node ID matches the {speaker ID}_{NodeID}.

Move

This allows you to create objectives that make the player move a certain distance. For example, telling the player to move 100 meters.

Complete Narrative Data Task

One of the most common tasks you may use. Narrative Tasks are blueprints that allow you to combine logic into a single task which can connect via Event Dispatchers or other means to actors and complete automatically. However, sometimes this might not be feasible and this is where Data Tasks come in. Data Tasks can be completed from any blueprint that has access to the NarrativeComponent and does not require a task to be created. Data Tasks can then be queried to get details about it. An example would be every time you defeat an enemy, a data task is marked as complete using the node **CompleteNarrativeDataTask**. You can then query this in your quest to detect if a certain number of enemies have been defeated.



Properties

Data Task

This is which data task the task needs to check for before completing itself. For example, the data task could be **PickupItem**.

Argument

This is which argument when the data task is completed that the task needs to be checked for. For example, the data task could be **PickupItem** and the argument would be **Apple**.

Retroactive?

Whether or not the task should look at previously completed nodes before completing this. Useful if you want to skip a task if the player has already done it.

Go To Location

A very common node which allows you to complete an objective when the player arrives at a destination using the location vector. You can customize how often it checks the distance to save performance.

Properties

Goal Location

This is the vector location that the player needs to reach to complete the task.

Distance Tolerance

This is how close or far away the player needs to get to the goal location.

Friendly Location Name

The location name that will display on the UI for the player. For example, **The House** would display on the UI as **Go to The House**.

Goal Actor Class

If you don't want to use the fixed goal location, you can provide a goal actor class and a **goal actor tag** (see below) to track instead. Useful for moving targets.

Goal Actor Tag

The tag of the actor you are trying to track. Useful if you have a generic class that applies to multiple instances.

Invert

By default, the task will wait for the player to arrive at the location. However, you may have scenarios where the player needs to get away from a location. You can use this to change this setting.

Add Waypoint?

Whether or not Narratives built in Waypoint be added to the goal location or goal actor's location so the player can track them on screen.

Complete Quest

A unique one that has its place, this task allows you to complete an objective when you complete another quest. An example could be a door quest which requires you to get 3 keys. To get each key, requires you to do a quest. The door quest would be made up of 3 Complete Quest tasks.

Properties

Quest

This is the quest in which the task needs to be checked to complete the task.

Custom Tasks common in the Community

The below are tasks that are common to add among games but require your specific game logic to be used.

Go To Location Trigger

The default GoToLocation task is often enough for an objective to be met. However, sometimes you may need the player to reach a very specific location such as a door

or location which has a custom shape. The GoToLocation task measures the player's distance meaning being next to the door would also complete it.

The GoToLocation Trigger is a custom task you can create simply by adding an Event Dispatcher to your trigger when it overlaps then your new Narrative task, connected to this event dispatcher to complete the task. Gives you more fine-grain control over specifically where the player has to go.
You can even build in the actual location trigger spawning to spawn the locations when the task begins.

Defeat NPC

It is common for games to have some sort of combat system and objectives which require you to take out a specific NPC. This task will find the NPC and connect to an event dispatcher “OnDefeat” which is called by the NPC when the NPC’s health reaches 0.

Pickup Items

A common trope of gaming is loot. Picking up items from around the map. This task can work in two methods depending on which works best for you.

The easiest method is to find the item in the world and connect it to its event dispatcher of OnPickedUp. Once the item is picked up, it completes the objective.

The second method which is more involved is to connect to your games inventory system and bind to the event dispatcher OnItemRecieved. When the event triggers you can check which item was picked up and if it matches the item the task requires, then you can complete this task.

Quest Timer Finished

Commonly used to give the player a sense of pressure, quest timers are common when an objective must be completed within a given time such as reaching a location before the timer runs out or saving an NPC before the timer ends. Upon this task starting it would start an Unreal Timer. When this Timer finishes or reaches 0, you can then mark the task as complete. This would then connect to either the next state of your quest or fail the quest.

Using tasks

To use a task in Narrative, enter the **Quest Graph** and right-click. This will open Narratives list of tasks and your custom tasks. Click on the option you want to add

and Narrative will add the required Branch for your task and the quest state afterwards.

Alternatively, you can click on an existing branch and add the task you want to the **Quest Tasks property**.

Self-contained quests vs decentralized quests

Narrative lets you freely create quests however you like and what suits you best. However, some design methodologies can be used to help keep your quests maintained.

There is no right or wrong way to create quests. It really comes down to how you want to work.

Decentralized Quests

Decentralized quests are the most common way and often faster to create. You spawn each item, or NPC, into the world by manually placing them and making them destroy themselves if the quest is finished or at a certain state. It's quicker to get up and running and gives you a better visualization of how the quest is laid out.

However, the disadvantages of this method are repeatability and load times. If you fail at a quest or wish to repeat the quest from a quest selection screen, you will have to reload or respawn the actors in the world which will add more loading time. The load times disadvantage is when your level first loads every actor that is quest-based must check its own state causing delays with the more actors that need to check.

Self-contained quests

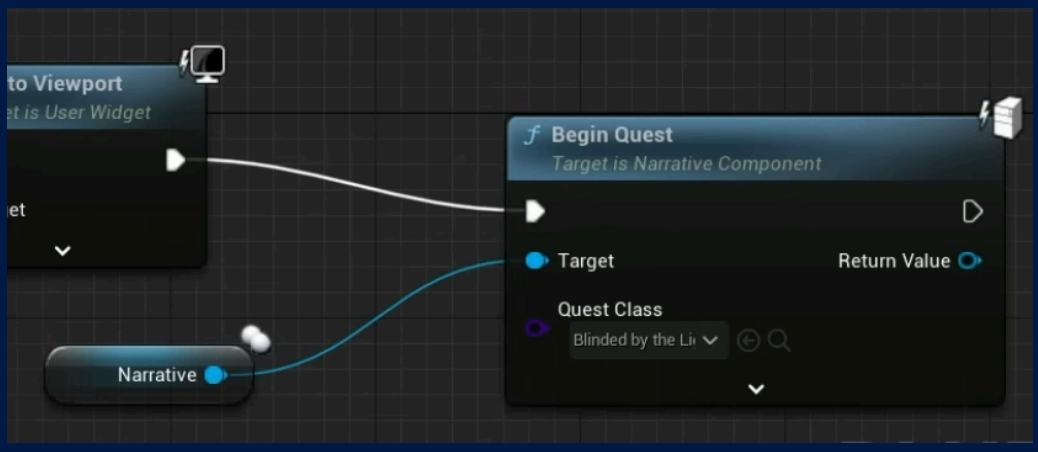
The second methodology is self-contained quests where your quest will handle the spawn and despawning of actors required. Thinking of games such as Grand Theft Auto or Skyrim, often you can arrive at a location and nothing will exist or it will be locked until you return during an active quest. This methodology removes the disadvantages of decentralized quests since each quest will only spawn/despawn items when it starts and ends decreasing load times.

Since the quests control the lifespan of each actor, this also means quest selection screens are super easy because you can just call **Begin Quest** and have everything ready.

However, it is less artistically friendly to do so. Since Narrative's Quest Graph will now control the actor's lifespan, you cannot simply browse the world to place actors. The process would be placing the actor in the world, copying its location, rotation and other details then adding the corresponding **Narrative Event** with the details required. For example, placing the pick-up in the world, adding a custom **Narrative Event** to spawn a pickup and adding the pickup class, location, rotation, name, tag etc...

Starting Quests

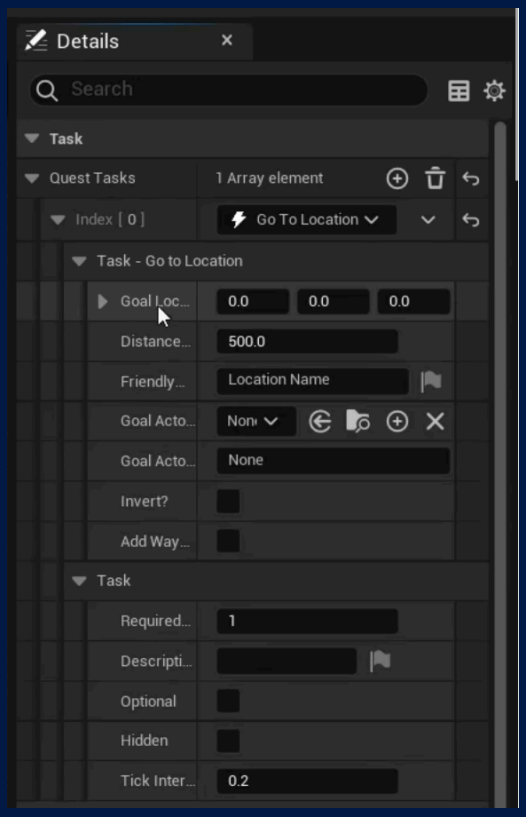
Starting a quest in Narrative is super easy. All you have to do is call BeginQuest from the Narrative component. See the section about **Getting the Narrative Component** for ways to do this.



Creating Quests

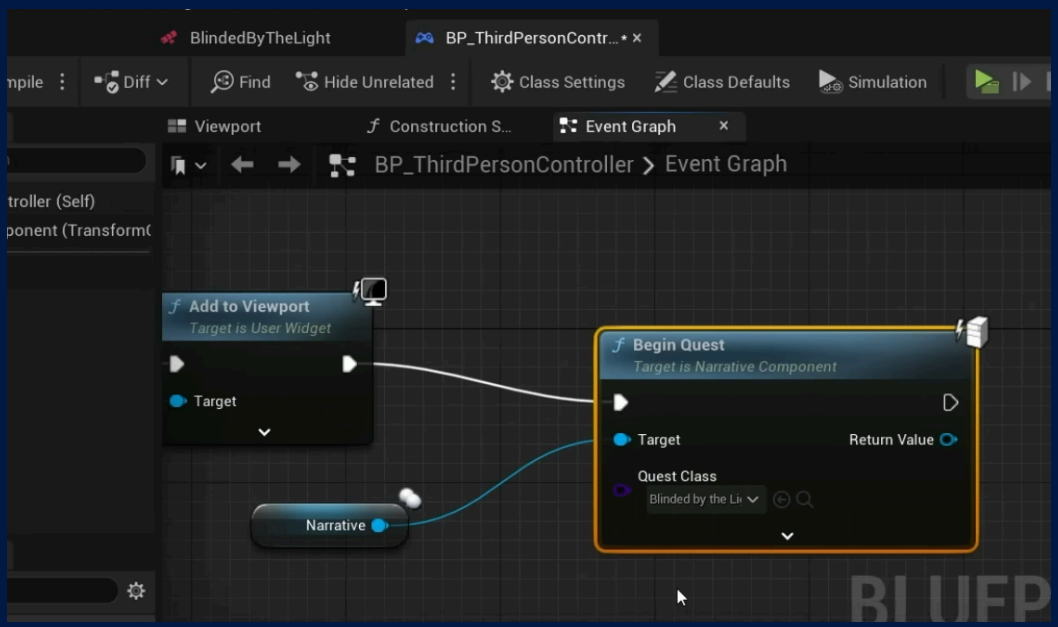
In this example, we are going to add the Go To Location task.

If you click the QuestBranch node, in the details panel you can see the properties for this node.



Within the Goal Location, we will add the location vector of a random cube in our scene. The Friendly Location Name we will set to red cube and we will also tick Add Waypoint to add a 3d waypoint to show the players where to go.

Finally, we will add BeginQuest to our PlayerController to start this quest.



When you play the game now, your quest should begin and ask you to go to the red cube. Since we ticked the waypoint option, it will also display on your UI.

Narrative Events

Narrative events are a powerful tool that truly makes Narrative suit your game's requirements. Narrative is a framework that lets you expand upon its features and Events are the way to do it.

Narrative events are small portions of code (mainly generic and reusable, but they can also be specific) that allow you to run code from any node across Narrative; be it Quests or Dialogue.

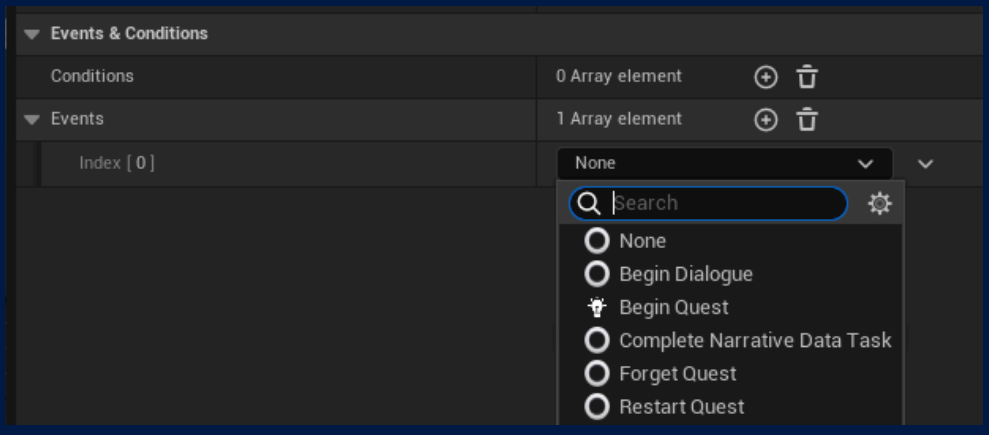
Need a quest to start when you select a dialogue option? Add an event.
Need to give an item to your player after finishing a quest? Add an event.
Need to spawn an NPC, set their AI behavior, or give them a weapon? Add 3 events!

A key thing to remember with Events is to **always** make it return. If you don't it can cause issues with Narrative waiting for the event to finish.

You also have the option to select whether an event runs at the start, end or both (start and end) of a node. Useful for running the event before a dialogue or task or after. For example, a dialogue node saying "Thank you for helping me. The priceless red jewel is in the red bandit camp" - then adding the quest instead of adding the quest before they have even spoken.

Adding Events

To add an event to a node, simply select the node (quest or dialogue) and in the details screen, add as many events to your Events list as you require.



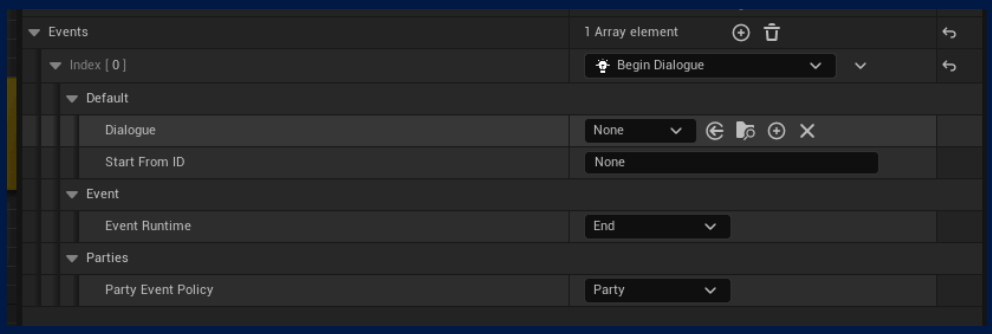
Default Events

Narrative comes with a bunch of default events to help kickstart your game.

Begin Dialogue

This event allows you to start dialogue directly from a node. Whether this be during a quest or another dialogue (it will simply end the current dialogue and move to the new one - it will not return. To make it return, call BeginDialogue and specify a Start From ID)

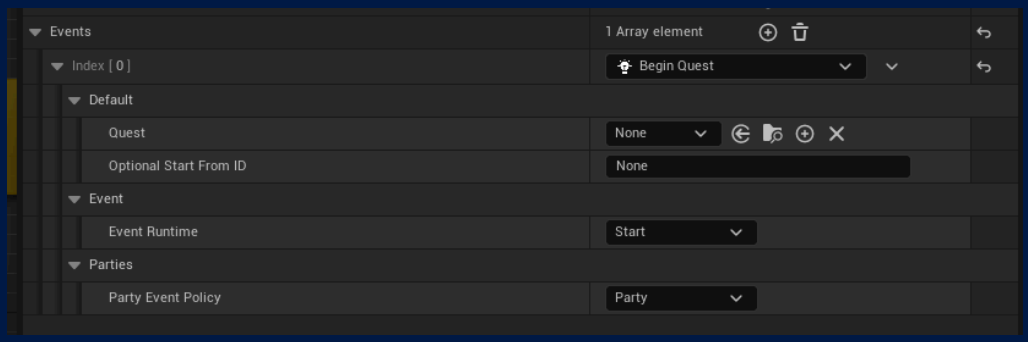
Simply provide the Dialogue and the **ID of the node you wish to start from (optional)**.



Begin Quest

This event allows you to start a quest directly from a node. Whether this be during a quest to spawn another quest or during dialogue.

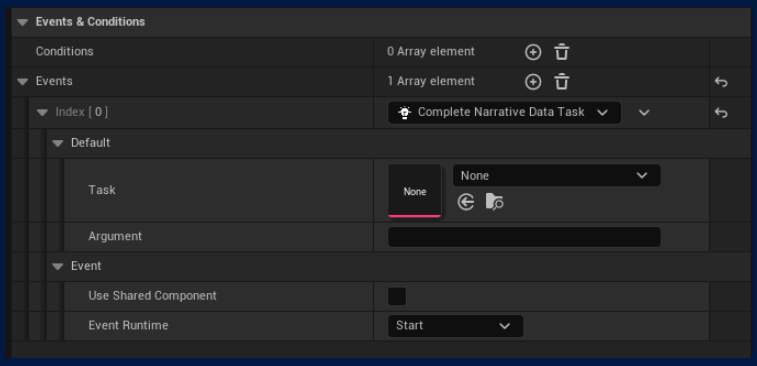
Simply provide the Quest and the ID of the node you wish to start from.



Complete Narrative Data Task

It is common that you may want to complete a task within Narrative that doesn't specifically match up with a task the player has to action. This could be as simple as finding an item location via dialogue but not having it, or completing a task to talk to X amount of NPCs.

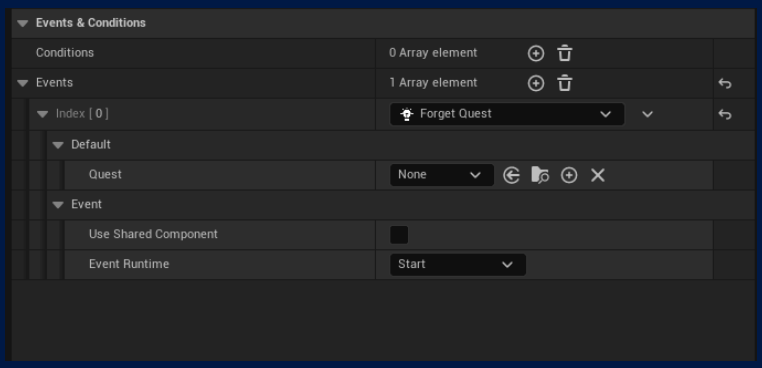
Using **Narrative's Data Tasks** you can complete them simply by providing the task and the argument.



Forget Quest

Often you may need to remove progress from a quest. This may be if you return to the NPC and ask to repeat it or the user may have failed the quest.

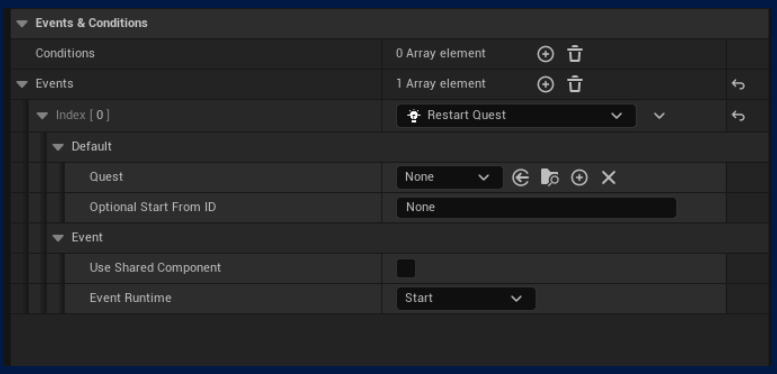
Simply provide the Quest.



Restart Quest

This event allows you to restart a quest directly from a node. Removing all progress and starting again from the beginning.

Simply provide the Quest and the ID of the node you wish to start from.



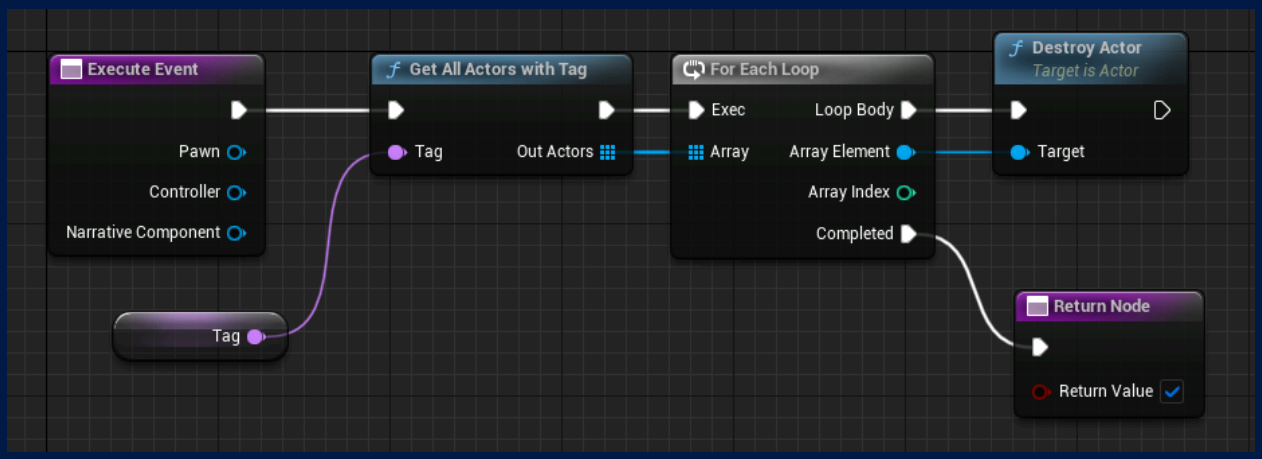
Custom Events common in the Community

The below are events that are common to add among games but require your specific game logic to be used.

Destroy actors with tag

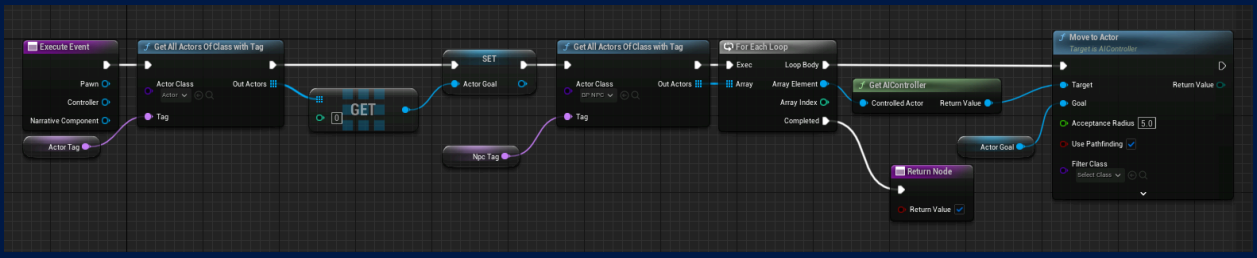
This event is useful for cleaning up or affecting the world. Simply specify a tag on the actors you want to remove and call this event. It will loop through and delete all actors with the same tag cleaning up your world.

It is highly useful for keeping quests self-contained. Spawn the enemies, and quest items, on failure or success - clean up.



Move NPC to Actor

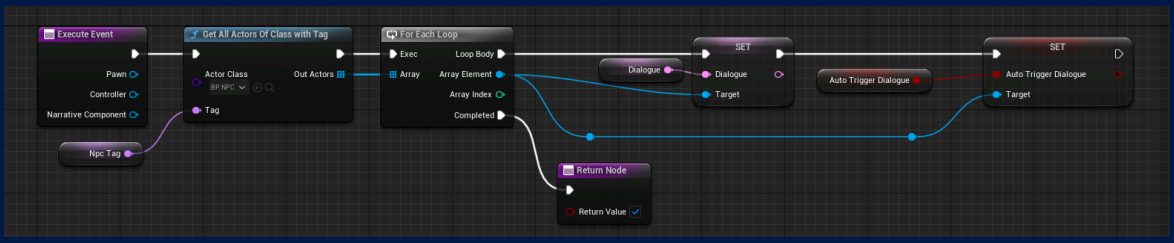
Using Unreals default AI behavior, this event tells the NPC found by its tag to go to a specific Actor. Useful for making an NPC go to another NPC, door, trigger or even the player. For example, after speaking to an NPC and accepting a certain job, make them run to the starting location to help the player.



Set NPC Dialogue

It is common to have an NPC's dialogue stored as a **soft reference** directly on the NPC. Walking up to the NPC and interacting will start the dialogue. However, putting all of an NPC's dialogue in a single file can often become unmanageable. So it's common to split an NPC's dialogue into sections that suit your game. Some games may have all dialogue in a single file split by comments, others may split by quest. It depends on how you prefer to work.

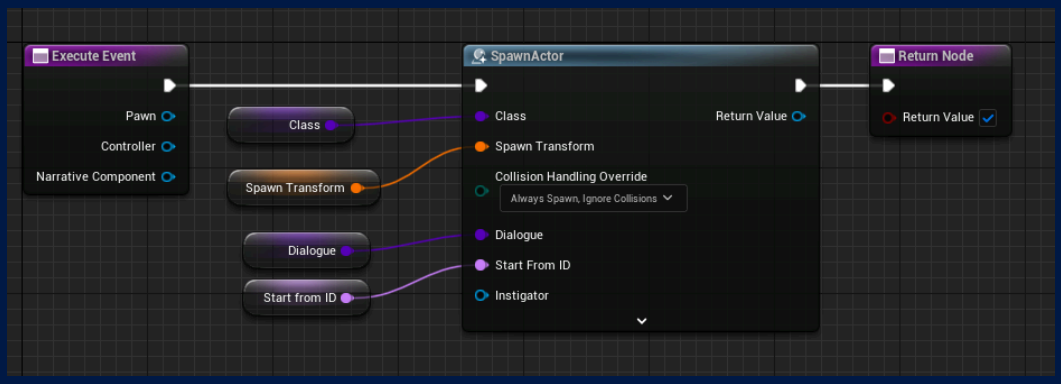
However, if you work in a state where a single NPC can have multiple dialogues, it's common that during a quest you very well may want to change their dialogue during the quest. This event gives you the ability to select an NPC by its tag and then the new dialogue you wish to assign.



Spawn Pickup

A common event to spawn pickups throughout your game. Simply specify the Class (based on a parent blueprint class for the Pickup) and a transform (location, rotation, scale). In this example, a dialogue is also added to play upon pickup.

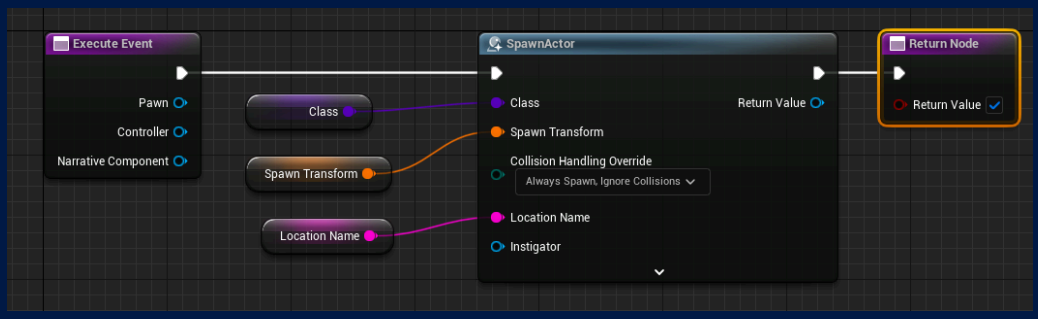
This will then spawn the item in the location with the right settings.



Spawn GoToLocation Trigger

Related to the community-based **GoToLocation Trigger task**, it is common to spawn in the triggers during the quest to guide the player into the right areas. For example, starting a quest then it will spawn a location trigger at the castle door and the objective tells you to go there.

Simply provide the go-to location class (cube, sphere, any shape you have created based on the blueprint parent class), the transform (location, rotation, scale) and the location name to update Narrative.



Give Item

An event that plugs into your custom inventory solution (a custom-designed one, Narrative Inventory) to give the item to the actor with the tag. Useful for giving the player quest items or allowing an NPC to have them.

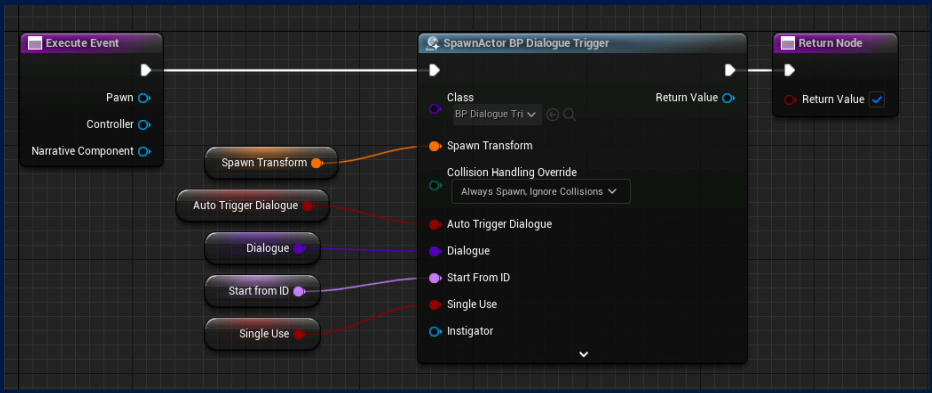
Note: since every inventory system will be different, a screenshot cannot be provided. The rough plan would be:

Execute Event - GetAllActorsOfClassWithTag - get inventory - GiveItem.

Spawn Dialogue Trigger

Sometimes you may require dialogue to run without any NPC being near. Some examples are barks or simply the player talking to themselves, a phone call etc...

An invisible dialogue trigger can be spawned by using this event to trigger cutscenes or dialogue when the player reaches a location.
Provide the transform (location, rotation, scale) of the dialogue trigger, and specify whether it's an auto-starting dialogue or the dialogue. Optionally you can also provide the start from ID of the dialogue and whether it's single use (destroys after triggered) or repeatable.



Narrative Conditions

Narrative conditions are another powerful tool that allows you to control the flow of dialogue based on any condition you require.

Narrative conditions are small portions of code (mainly generic and reusable, but they can also be specific) that allow you to check a value and have it act as the gate to the next dialogue node to enter.

Do you require a dialogue to only be used if the player has a specific item? Add a condition.
Do you want to only go down a dialogue route if the player has completed a quest? Add a condition.
Want to only play a dialogue node a single time? Add a condition.

A key thing to remember with Conditions is to **always** make it return true or false. If you don't it can cause issues with Narrative waiting for the event to finish. If you

return true, it will allow the node to be used. If you return false, it will look for another node, otherwise end the dialogue.

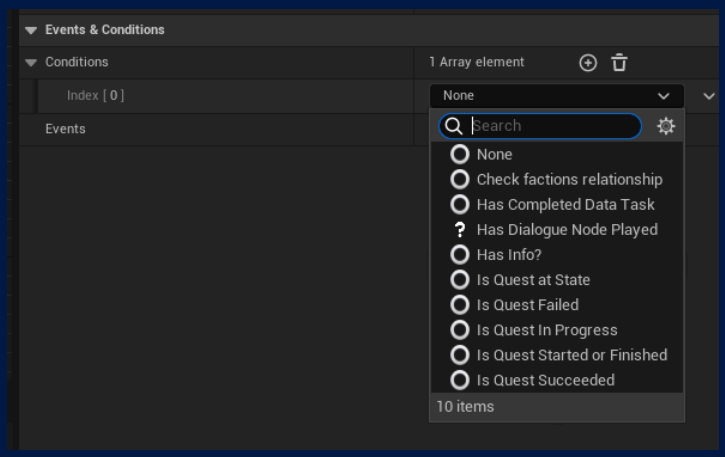
Narrative Conditions allow you to add multiple conditions to a node. All of these conditions must be met for Narrative to proceed. If even one is not met, the node will be rejected for use. If you require this or that, use two nodes instead.

Adding Conditions

To add a condition to a node, simply select the node (quest or dialogue) and in the details screen, add as many conditions to your Conditions list as you require.

Default Conditions

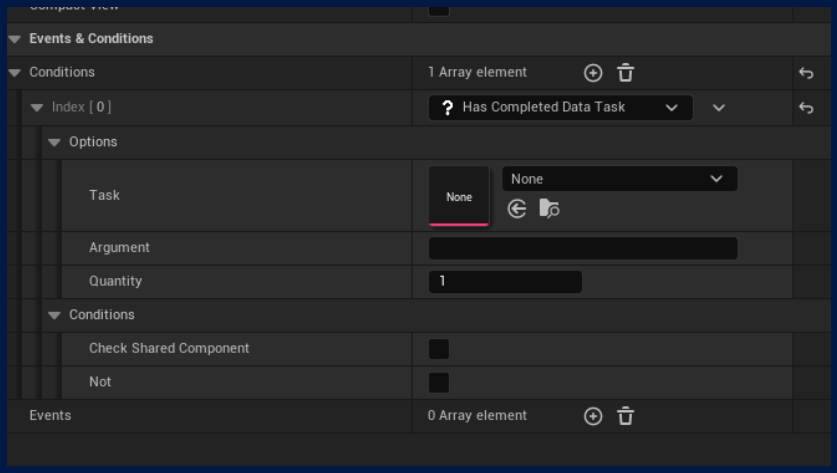
Narrative comes with a bunch of default conditions to help kickstart your game.



Has Completed Data Task

This condition allows you to check if a **Narrative Data Task** has been completed to enter this node.

Using **Narrative's Data Tasks** you can complete them simply by providing the task and the argument. You also have the option to provide a quantity if you want to only check if the data task has been completed a number of times.

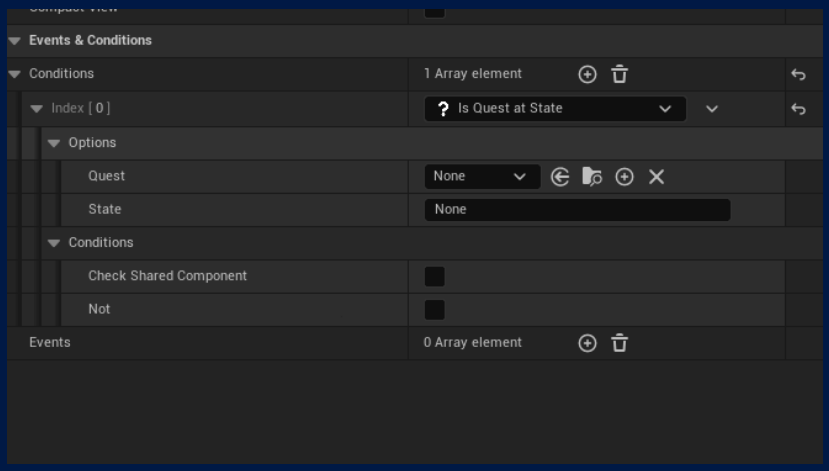


Is Quest At State

This condition allows you to check if a specified quest has arrived at a specific **state**.

Simply provide the Quest you want to check and the state ID in which it must have arrived.

Note, that this condition checks if the quest is currently AT a current state. Not if a past state has been completed.

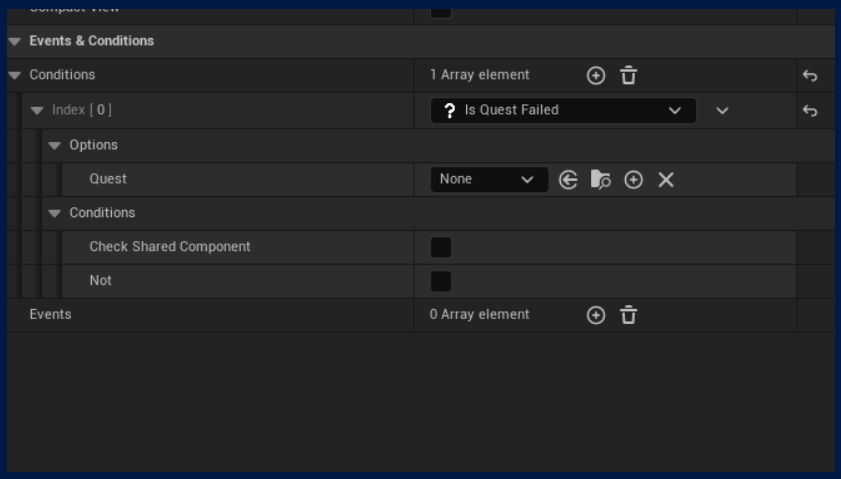


Is Quest Failed

This condition checks if a quest has ever reached the **failed state**.

Simply provide the Quest you want to check.

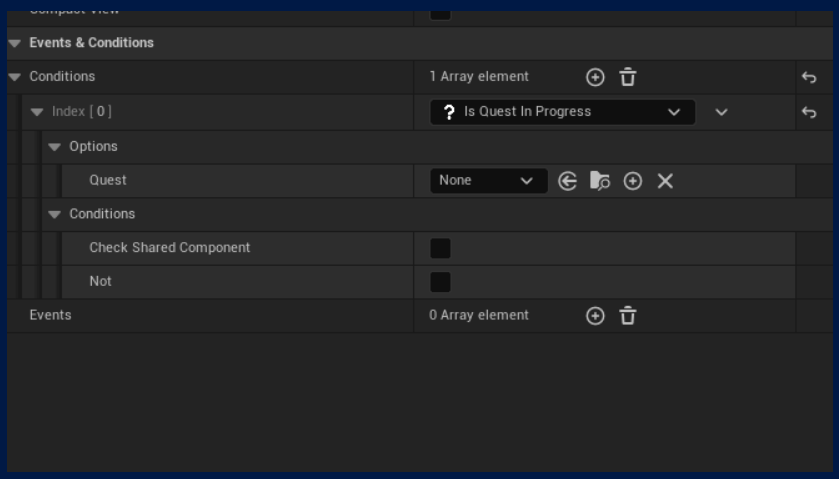
Note, that a quest will only retain a failed state whilst it has not been forgotten or restarted.



Is Quest In Progress

This condition checks if a quest is currently in progress. This means it has not reached a **success** or **failed** state.

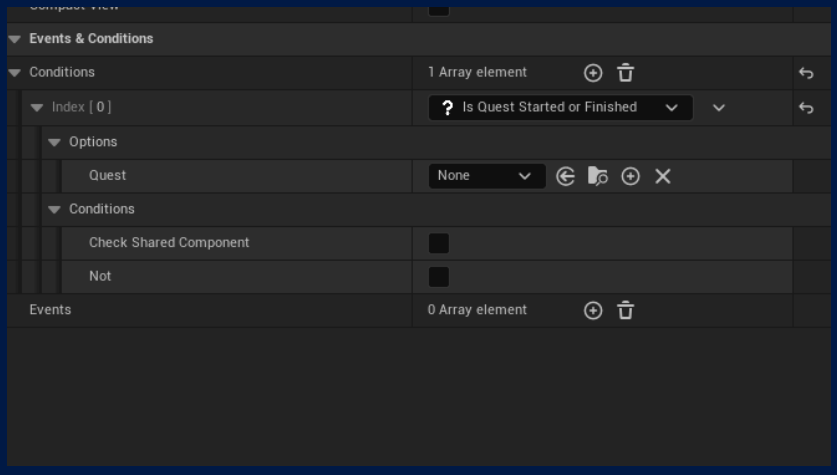
Simply provide the Quest you want to check.



Is Quest Started or Finished

This condition checks if a quest has currently started or finished. This means it has been encountered and started at least once at any state.

Simply provide the Quest you want to check.

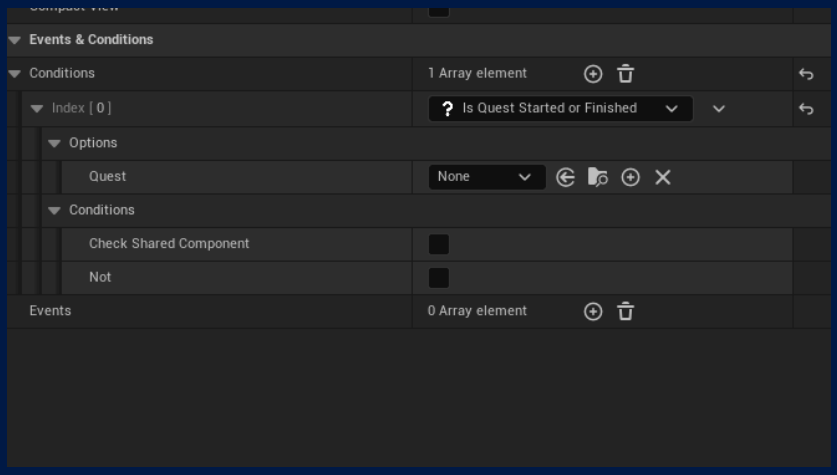


Is Quest Succeeded

This condition checks if a quest has ever reached the **success state**.

Simply provide the Quest you want to check.

Note, that a quest will only retain a success state whilst it has not been forgotten or restarted.



Custom Conditions common in the Community

The below are conditions that are common to add among games but require your specific game logic to be used.

Has Item

This condition is used to check an actor's inventory to give access to a specific dialogue node. An example could be asking a character where they received an item, but only if they have an item.

You can also add an optional quantity field to check if they have a specific amount.

Note: This Has Item condition is specific to your inventory implementation. The basic design is to get all actors of class with a tag - get inventory - check if they have an item.

Is Level

This condition is used to check if a character has a specific XP level. Useful in games which contain an XP / leveling system, you can prevent dialogue based on the level of the character.

Saving and loading

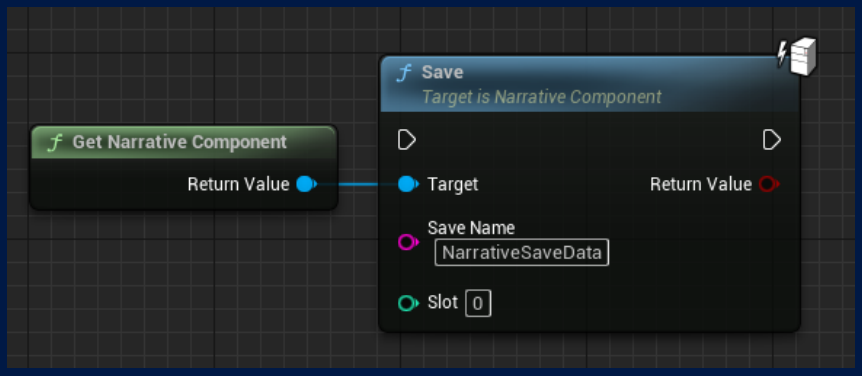
Narrative features a fully functional save and load system. Built around Unreal's save and loading functionality, it can save all dialogue, quests and tasks that the player has achieved to make loading it all back in easy.

Saving

To save your game, you simply **get the narrative component** and call the Save node.

You are presented with 2 options. The save Name and the Slot.

These are the default Unreal options for saving.



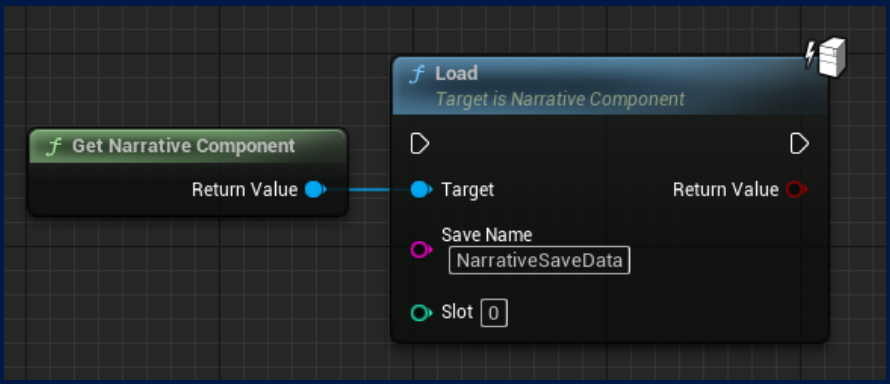
Loading

To load your game, you simply **get the narrative component** and call the Load node.

You are presented with 2 options. The save Name and the Slot to load.

These are the default Unreal options for loading the correct save file.

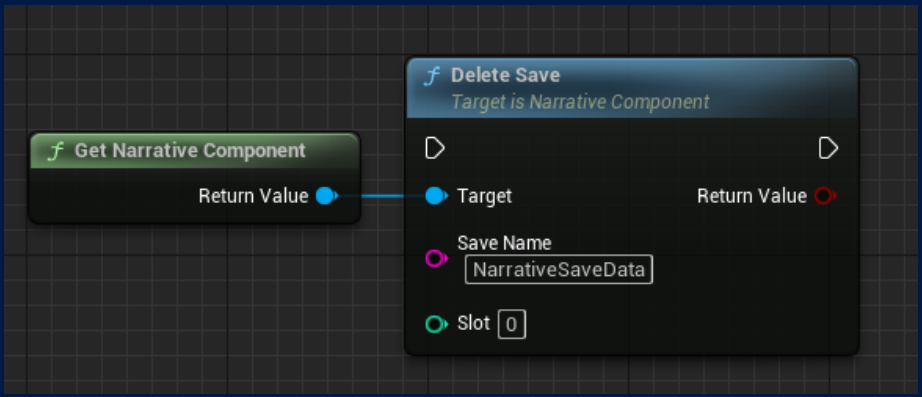
Note: the Loading node is soft failing meaning it does not error or stop the game. You can check the return value to see if the game has loaded correctly. This allows you to call load without a save file existing without any errors.



Deleting a save

To delete your save file, you simply **get the narrative component** and call the Delete Save node.

You are presented with 2 options. The save Name and the Slot to delete.



Retaining data across levels

Since the Narrative component is typically stored on actors that do not persist across levels, it is recommended that if you wish to retain data across levels, to simply call Save before you call OpenLevel and Load just after you have loaded a level. This will ensure Narrative always retains the data and in case of a crash, the data will be saved for the user.

Multiplayer

Narrative has out-of-the-box support for multiple versions of multiplayer depending on your use case. Whether you need couch co-op, MMO support with or without parties or server-driven quest & dialogue; Narrative supports it all.

Parties

Introduction

Narrative Parties is a feature added in 3.4 which gives you the ability to dynamically add clients (players of the game) into shared Narrative components which will distribute quests and dialogue across them. Perfect for clan or guild-based MMO games.

When **Dialogue**, **Quests** for **QuestTasks** are completed, Narrative will automatically replicate this across every member of the party. Meaning 1 player could start the dialogue, the second player could complete the tasks and the third player could take the info back to the start.

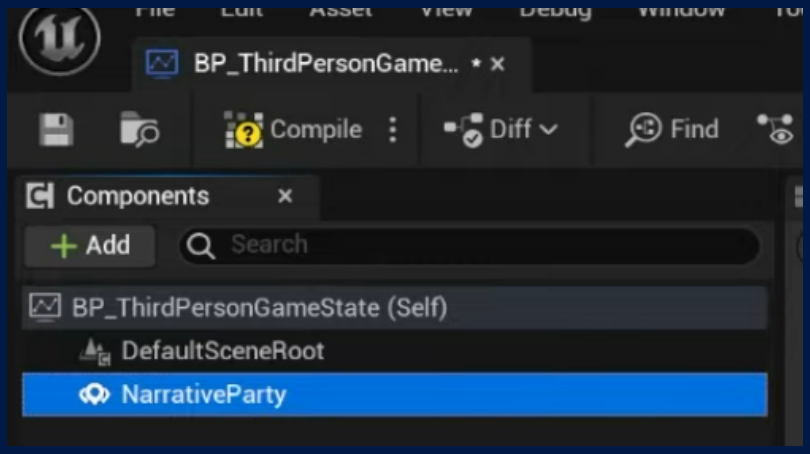
Note: If a player is added to a party, it does not remove the functionality for them to do non-party-based quests or dialogue.

Setup

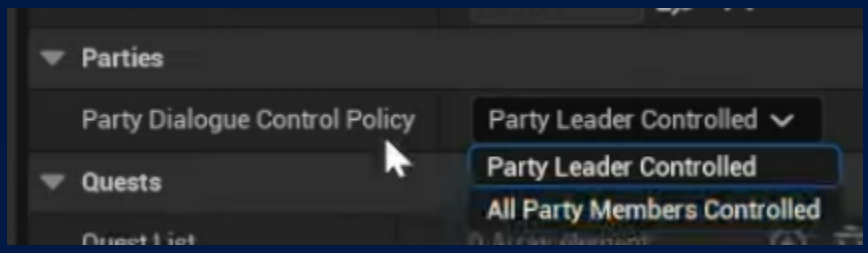
To get started setting up Narrative Parties, firstly you need to **install Narrative** and have it fully set up and working.

Narrative Party Component

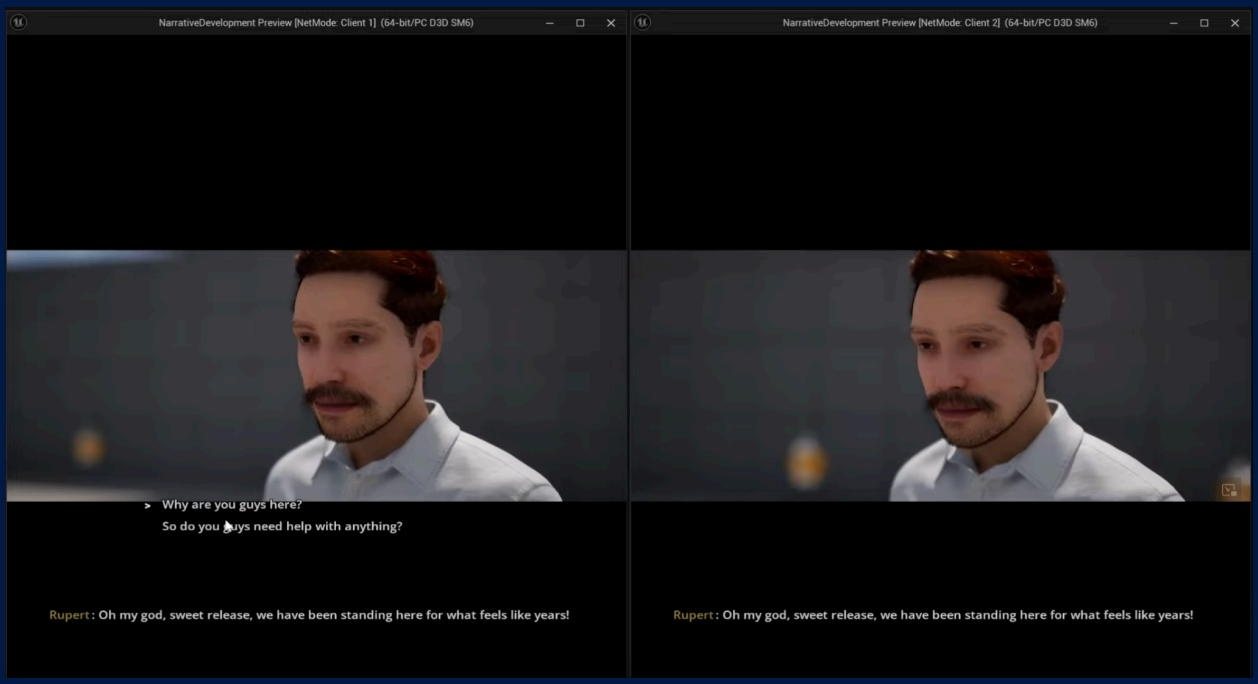
Next, you need to add the new **Narrative Party component** to your GameState if you want a single party in your game or another replicated actor that will be shared across each member to support your game. In this example, we have set it on the GameState.



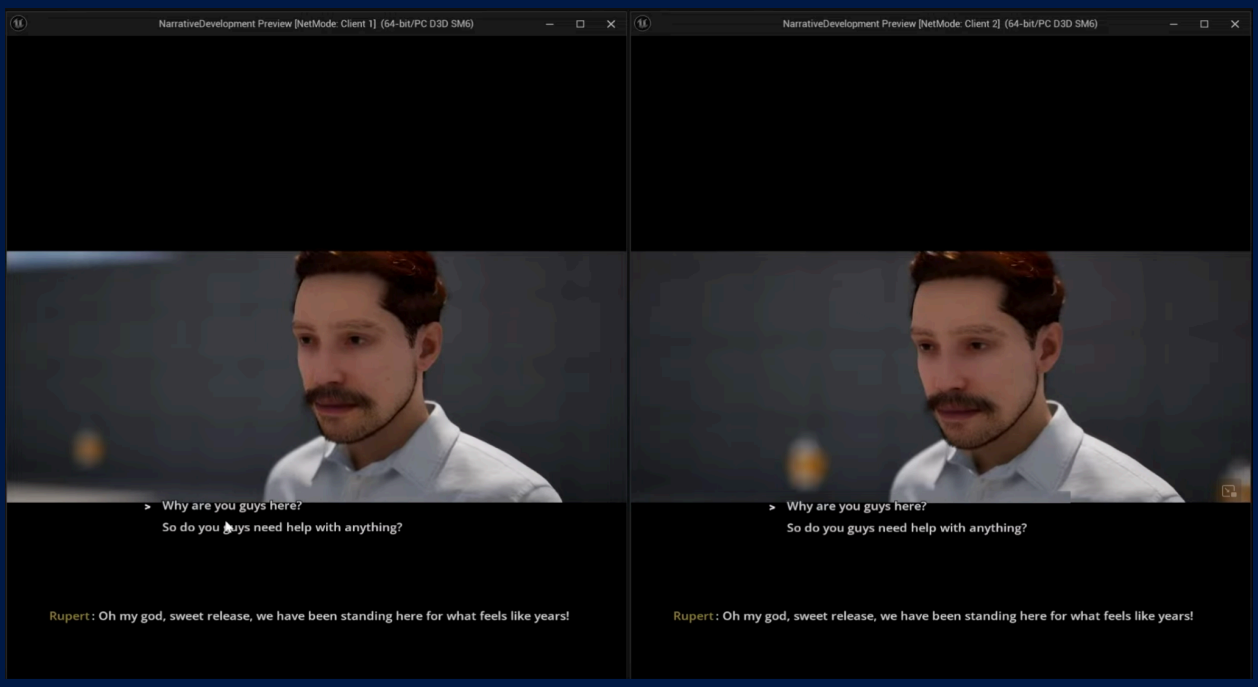
Upon clicking the Narrative Party Component you can set who is allowed to control Dialogue.



The Party Leader - is the first client added to the party. Often this will be the user who initiates the party and everyone else joins into it.



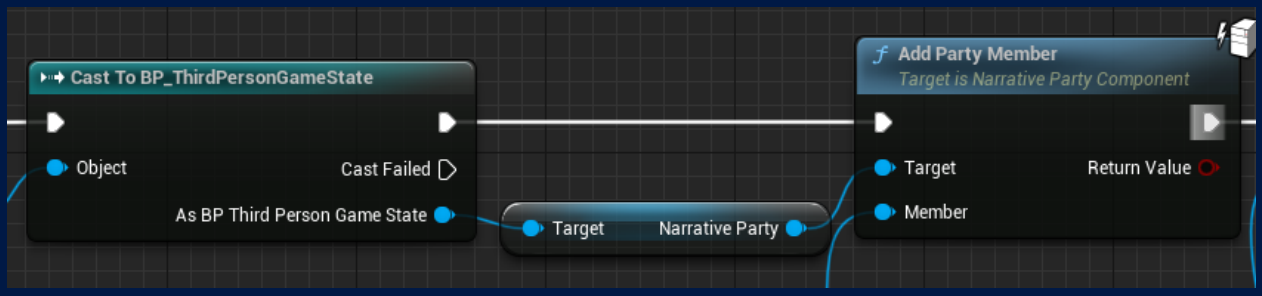
All Party Members Controlled - Will allow every party member to select a dialogue option on behalf of all members.



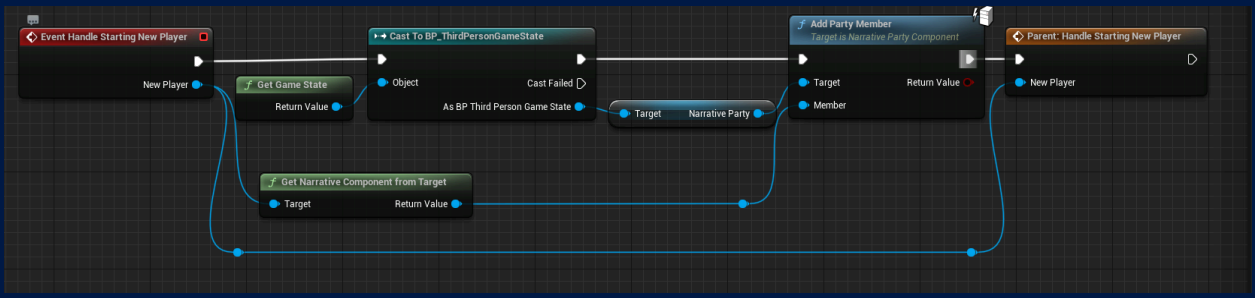
Adding Party Members

To add a member to a Party, you need to call the Narrative Party Component's Add Party Member function. This can be actioned from a UI if you select a party to join, or when the player first joins the game.

The Add Party Member function takes 2 input parameters; the Party you are adding the player to, and the player's Narrative Component.



In the example below, we have overridden the HandleStartingNewPlayer event on our custom **Game Mode**, got the Narrative Component from the new player and added it to the Game States Party component. This will add every new player that starts into the same Narrative party.



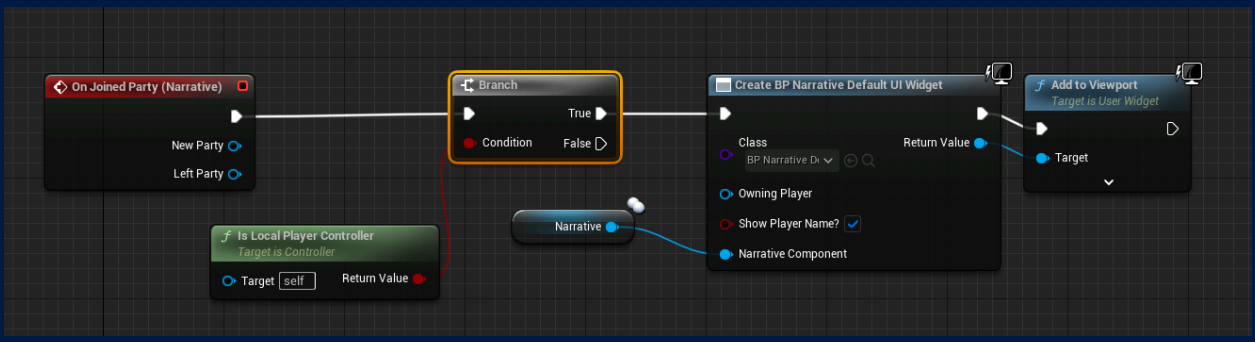
On Joined Party Event

Depending on your game, you might want to restrict players from using Narrative until they have joined a Party. This could be an entire server or some specialist rule / restricted quest.

The Narrative Parties update adds a new event called OnJoinedParty which has 2 output parameters. **New Party** is the party the individual is joining and **Left Party** is the Party the individual has just left.

This event is available from the **Narrative Component**.

In this example, we want the players to only be able to start quests and dialogue when they have joined a party, so we spawn the UI from this event instead of BeginPlay.

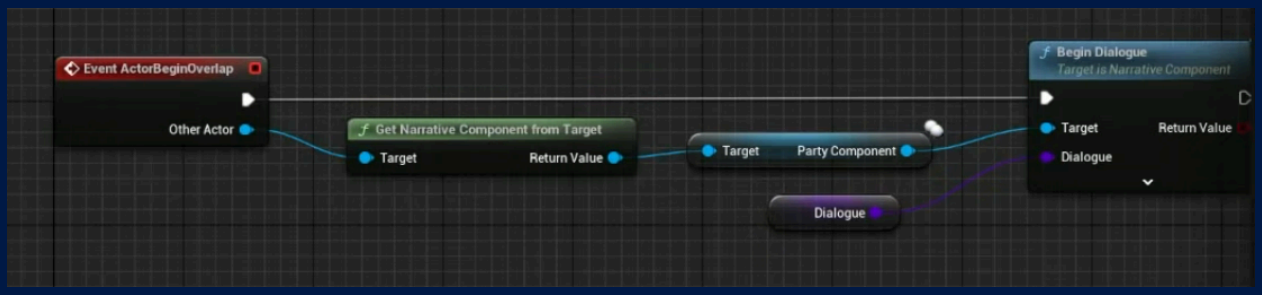


Starting Party Dialogue

The main difference between starting Dialogue on a party vs single-player games is that you need to start the dialogue on the Party component instead of the player's Narrative component. This will tell Narrative to start the dialogue for everyone in this Party.

You can get a reference to a player's Party by getting the PartyComponent from the player's Narrative component. From here, you can now call BeginDialogue.

In this example, we have a trigger which starts Dialogue. We have modified it to start from the PartyComponent instead.



Starting Quests

Narrative quests can be started in many ways. With Narrative Parties, there are minor tweaks that have to be made to make quests and tasks work for parties.

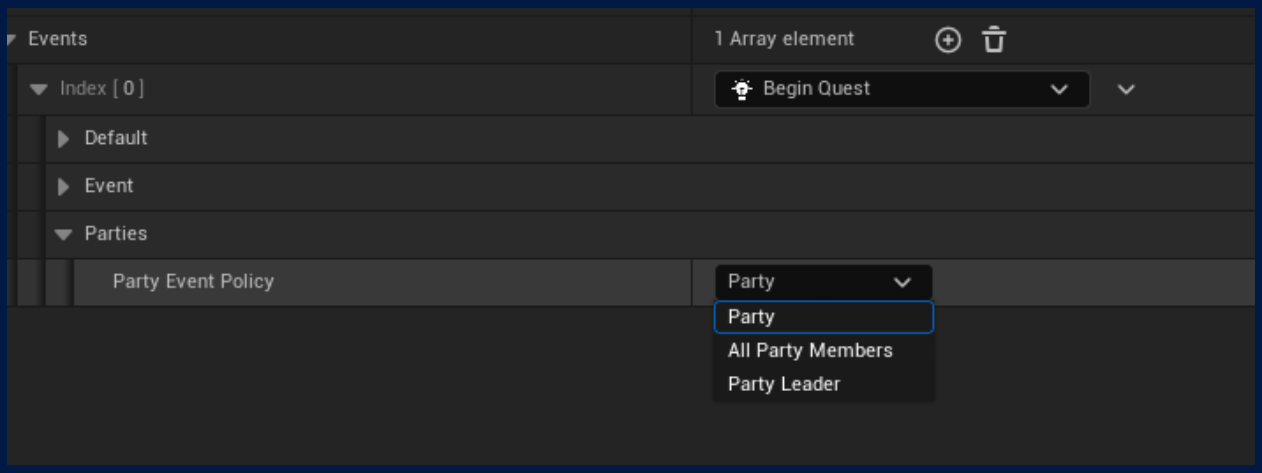
BeginQuest event

If you use the **BeginQuest event** within Narrative Dialogue that is started via a Narrative Party Component, the quest will automatically be started for the party.

Narrative Events

Narrative Events have now been updated to feature controls for Parties. When running an event, you can choose who the event runs for. By default, it will run for all Party Members which will work for each individual player within the party. It will not be a shared event.

However, you have other options to select Party, which will run the event as a shared event across the party or Party Leader which will also run it for the **Party Leader**.



A common example where you would change an event's Party Event Policy would be throughout a quest. The quest may start from dialogue which you want to start for the Party as a shared quest but the reward you may want to give each player. So for granting rewards, you would set this event to All Party Members.

Narrative Conditions

Narrative Conditions have also been updated to feature controls for Parties. When running conditions in Dialogue, you can now choose who the condition will run for. Unlike single-player Narrative which will only run for the player who started the quest, Narrative provides 4 Policies to check.

Any Party Member Passes

This condition will check the condition against each member. If there are 5 members and only 1 pass, the condition will succeed since at least 1 member has passed.

Party Passes

This condition will check the condition against the party component that has started the quest. This is useful for shared quests where the Party will be updated.

All Party Members Pass

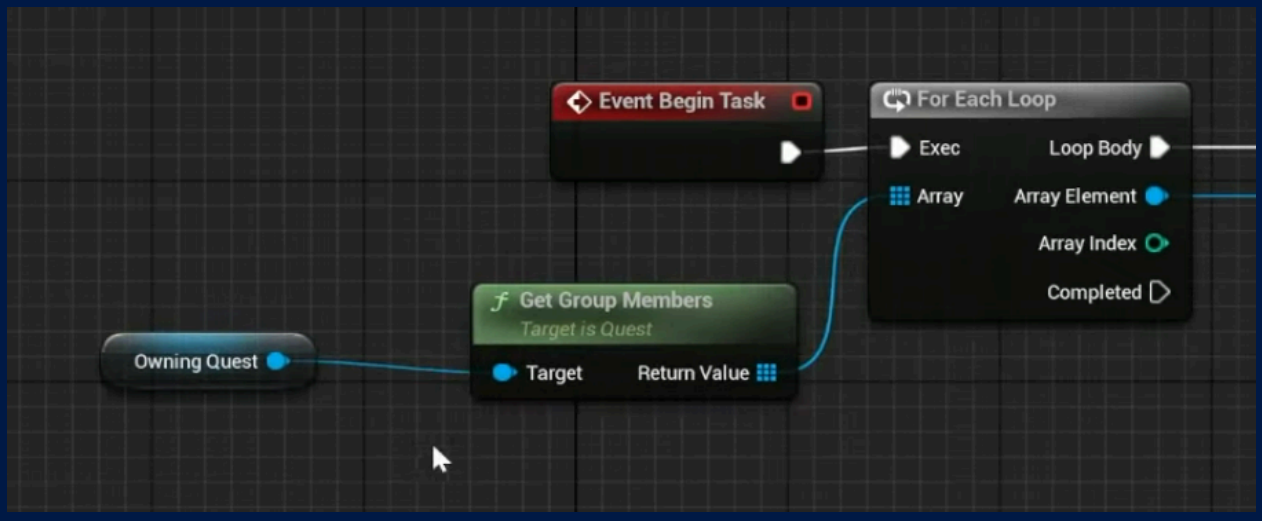
This condition will run against each member in the party. Every single member must pass for this condition to return true.

Party Leader Passes

This condition runs solely on the Party Leader. If they do not meet the condition, no one in the party will succeed in the condition.

Narrative Tasks

Narrative tasks will currently only function for single-player games. To make them work with Narrative parties you need to make 1 small change. Instead of making any binds or checks to the OwningController or GetPlayerPawn, you need to loop over the GroupMembers of the Owning Quest and run the logic for every member.

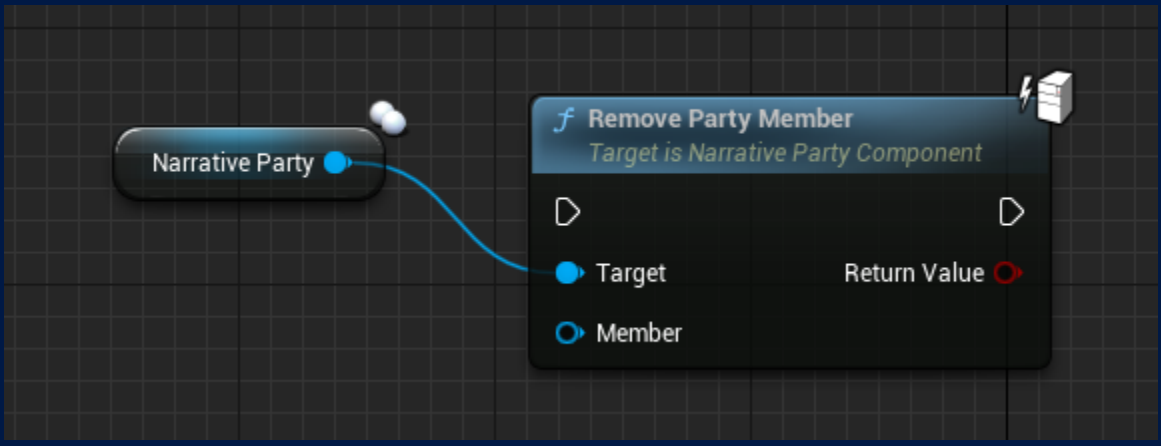


Note, any task built into Narrative has already had these changes applied.

Removing Party Members

When building your game, is it common to remove members from a party. This could be for a multitude of reasons such as disbanding the party.

Narrative comes with a RemovePartyMember function that allows you to remove a NarrativeComponent from the Party.



UI - Narrative 3.4

Narrative is shipped with a fully usable UI. The UI is written in standard Unreal UMG widgets and is fully customisable for your needs. Anything Narrative puts on the UI you have full control over.

The UI is stored within the plugins folder and is broken into several components.

The Narrative component does not communicate with the UI in any way. Instead, it broadcasts (calls) an event dispatcher (delegate) which the UI binds to inside the event graph.

BP_NarrativeDefaultUI

This widget is the main widget for Narrative inside Narrative **3.4**. Everything in the Narrative UI is controlled here. It is broken down into several sub categories for Dialogue and Quests.

In the graph, the event graph is split into 3.

General Stuff event graph

This event graph is the first and main opening to Narrative UI. It connects to each Narrative event dispatcher and assigns the correct event. For example, OnDialogueBegin connects to the OnDialogueBegin event within the **Dialogue event graph**.

It also contains the **waypoint** functionality.

Dialogue event graph

This event graph controls the dialogue. Each implemented event from the **General Stuff graph**, is stored here. Each event provides the dialogue in question and any other details required.

Quest event graph

This event graph controls the quest. Each implemented event from the **General Stuff graph**, is stored here. Each event provides the question question and any other details required.

BP_DialogueOption

This widget is the dialogue options the player can control on screen. It stores the text and the player option for Narrative to control. It is generated from the **Dialogue Event Graph's** **OnDialogueRepliesAvailable** Event.

BP_QuestBranch

This widget is the **quest branch** widget. The **quest branch** surrounds all **tasks** in the **branch**. Multiple branches can be rendered each with their own set of tasks as defined by the current active quest. This is rendered inside the **Quest Event Graph**.

BP_QuestTask

This widget is each individual task within a **quest branch**. This is rendered inside the **BP_QuestBranch** widget.

BP_QuestJournal

Narrative features a fully functional quest journal. It allows you to see your current and past quests with branches and tasks being displayed. This quest journal is stored in its own component but is spawned from the **Narrative Default UI**.

BP_QuestJournalQuest

This widget controls the quests displayed within the **Quest Journal**.

UI - Narrative 3.5

Narrative 3.5 is shipped with a fully usable UI re-written to be used with Common UI providing fantastic cross platform functionality. The UI is still written in standard Unreal UMG widgets and is fully customisable for your needs. Anything Narrative puts on the UI you have full control over.

The UI is stored within the plugins folder and is broken into several components.

The Narrative component does not communicate with the UI in any way. Instead, it broadcasts (calls) an event dispatcher (delegate) which the UI binds to inside the event graph.

BP_Narrative3Overlay

This widget is the main widget for Narrative that controls the quests and dialogue text. It does not control the players dialogue options or the quest journal. It is broken down into several sub categories for Dialogue and Quests.

W_NarrativeMenu_Dialogue

This widget handles the players dialogue options. It was split off in order for CommonUI to take over the input. It is opened via the OpenMenu node provided by NarrativeCommonUI.

WBP_NarrativeButton_DialogueOption

This widget handles each dialogue option created from the **W_NarrativeMenu_Dialogue** widget.

BP_QuestBranch

This widget is the quest branch widget. The quest branch surrounds all tasks in the branch. Multiple branches can be rendered each with their own set of tasks as defined by the current active quest. This is rendered inside the Quest Event Graph.

BP_QuestTask

This widget is each individual task within a **quest branch**. This is rendered inside the **BP_QuestBranch** widget.

W_NarrativeMenu_QuestJournal

Narrative features a fully functional quest journal. It allows you to see your current and past quests with branches and tasks being displayed.

BP_QuestJournalQuest

This widget controls the quests displayed within the **Quest Journal**.

F.A.Q's

Does Narrative support JRPG style dialogue?

Yes, you can easily achieve this in Narrative. Simply set your dialogue durations to **never**, to stop them from automatically completing, then modify **BP_NarrativeDefaultUI** to add a continue button. Make the continue button visible on BeginDialogue (or on NPC / Player DialogueFinished if you only want it at the end) then hide it on DialogueEnd.

The click event of the button should call the SkipCurrentLine event on the **Dialogue Event Graph** of **BP_NarrativeDefaultUI**.

Does Narrative support static images during dialogue?

Yes, you can easily achieve this in Narrative. You need to modify **BP_NarrativeDefaultUI** so that when displaying dialogue in OnNPCDialogueBegin and OnPlayerDialogueBegin the correct images are rendered where they need to be. This will be on the **Dialogue Event Graph**.

Does Narrative support random quests?

In a sense, yes. Narrative allows you to randomize quest task's quantities and within the tasks you can easily randomize the actual task. For example, you can create a task to fetch an item, but the task can randomize which item needs to be fetched.

Narrative does not support total task randomisation where the actual nodes on a quest can be randomly picked from a list. You will have to add this yourself since Narrative works based on fixed quests in the **Quest Graph**.

Does Narrative allow you to create quests/dialogue from external sources?

Yes and no. You can create Dialogue simply by copying the speaker ID and text in the following format.

SpeakerID: Text

For example
Reubs: Hey! Welcome to Narrative

RandomJoe: Thanks!

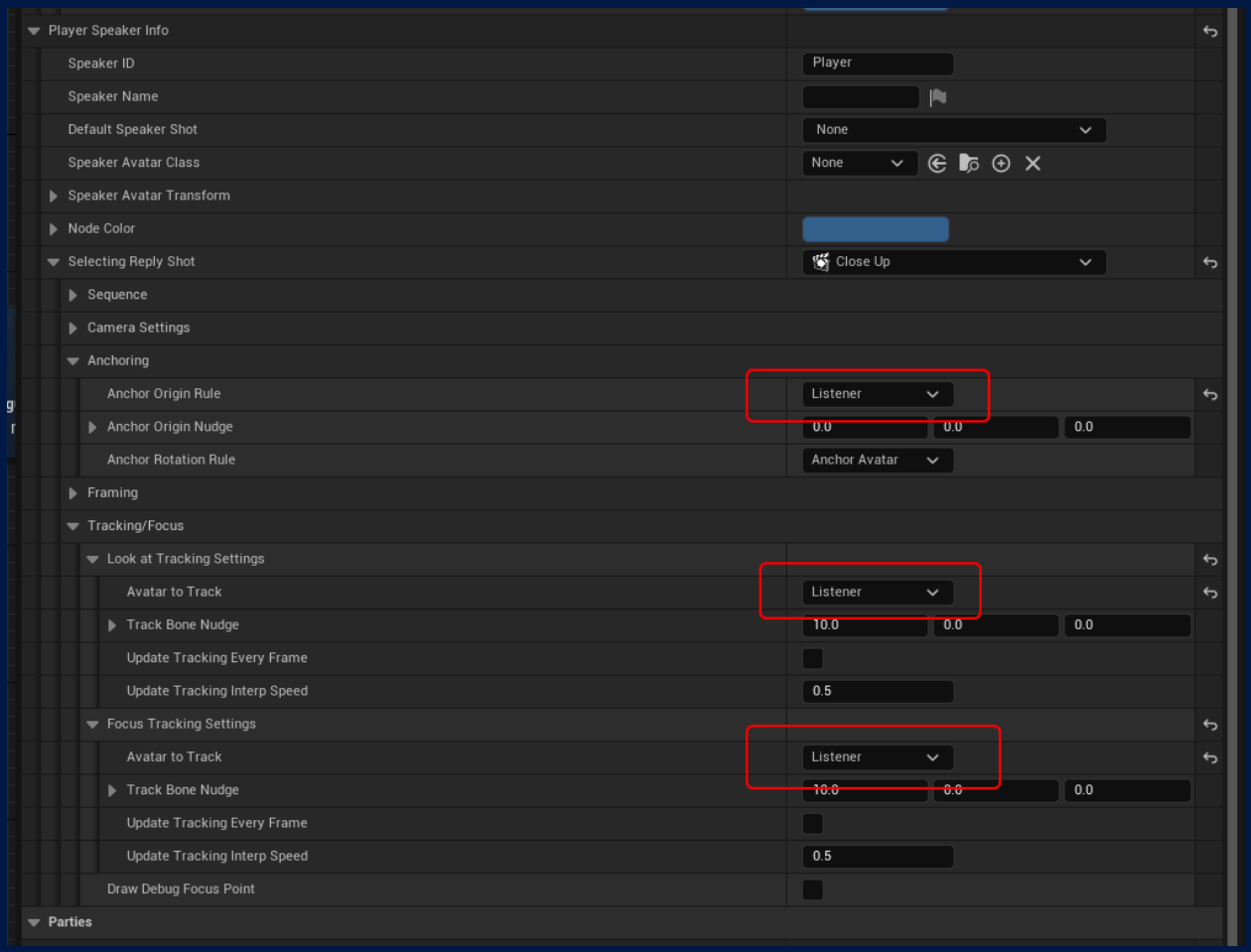
Narrative does not let you import from Excel files, CSV's or any other media.

NPC only cameras?

Yes you can set Narrative up to only show the NPC's during dialogue.

Apply a dialogue shot to your Players Selecting Reply Shot option then open it up.

Modify the Anchoring -> Origin Rule and set it to be **Listener.**
Modify the Tracking / Focus -> Look at Tracking Settings -> Avatar to Track and set it to be **Listener**.
Modify the Tracking / Focus -> Focus Tracking Settings -> Avatar to Track and set it to be **Listener**.



Other Narrative plugins

Narrative started as a simple but flexible Dialogue and Quest system which expanded into a flexible fully integrated system to support all types of dialogue with easy modification. All features added are reusable in most games and if not, are not too intrusive to become bloatware.

This fundamental principle has been expanded into other new Narrative plugins to help you create the game of your dreams easier and faster.

Narrative Inventory

Adds an entire inventory system to your Blueprint/C++ Project in seconds. Lightweight, extensible, networked.

https://www.unrealengine.com/marketplace/en-US/product/narrative-inventory-networked-lightweight-inventory-system

Narrative Common UI

Incredible Multi-platform UI for Every Narrative Tool - Install this if you want to use the UI that comes with any of the Narrative tools. It's free!

https://www.unrealengine.com/marketplace/en-US/product/narrative-inventory-networked-lightweight-inventory-system

Narrative Interaction

Adds an amazing, multiplayer-ready interaction system to your game in seconds. Lightweight, Networked, Full C++ Source Code provided.

https://www.unrealengine.com/marketplace/en-US/product/narrative-interaction-rich-multiplayer-ready-interaction/reviews

Narrative Navigator

Minimaps, World Maps, Compasses, Waypoints - A Powerful, Flexible, Navigation System for your game.

https://www.unrealengine.com/marketplace/en-US/product/narrative-navigator-maps-compasses-waypoints/questions