DIALOGUE SYSTEM 2.0

by Indie Devs

For the most up-to-date docs, follow this link.

Please rate the asset so Unity recommends it.

Video Tutorials

- Unity Dialogue System 2.0: Upgrade from older version
- Unity Dialogue System 2.0: Quick Start
- Unity Dialogue System 2.0: Custom Fields
- Unity Dialogue System 2.0: Localization
- Unity Dialogue System 2.0: Variable Injection
- Unity Dialogue System: Integration with Text Animator for Unity
- Unity Dialogue System: Integration with Easy Save
- Unity Dialogue System: Integration with Playmaker - Start Dialogue

Contact

In case you have some questions, suggestions, or need help getting the dialogue system set up, feel free to contact us at:
Email: jozefgocik03@gmail.com
Discord: https://discord.gg/swadkVj2sY

Setup

1. Create a new C# script that inherits from DialogueUIManager.

```
using UnityEngine;
using DialogueSystem;

public class DialogueUIManagerExample : DialogueUIManager
{

}
```

2. In the C# script editor, click “Implement abstract class” or manually add all override methods. Fill in the override methods with corresponding logic e.g. Put all the logic that should happen when choice node is the current node in the OnChoiceNode(ChoiceNode choiceNode) method etc.

```
using UnityEngine;
using DialogueSystem;

public class DialogueUIManagerExample : DialogueUIManager
{
    public override void OnDialogueNode(DialogueNode dialogueNode)
    {
        // TODO: ADD HERE logic for DialogueNode
    }

    public override void OnChoiceNode(ChoiceNode choiceNode)
    {
        // TODO: ADD HERE logic for ChoiceNode
    }

    public override void OnEventNode(EventNode eventNode,
DialogueEventManager dialogueEventManager)
    {
        // TODO: ADD HERE logic for EventNode
    }

    public override void OnIfNode(IfNode ifNode)
    {
        // TODO: ADD HERE logic for IfNode
    }

    public override void OnEndNode(EndNode endNode)
    {
        // TODO: ADD HERE logic for EndNode
    }
}
```

3. (Optional) You can use the following sample logic for the script as a starting point. Several other sample scripts are provided within the Sample Scenes. You can find this script on Github as well.

```
using TMPro;
using UnityEngine;
using DialogueSystem;
using UnityEngine.UI;

public class DialogueUIManagerExample : DialogueUIManager
{
    public DialogueTreeSO currentDialogue; // Reference to the current
dialogue being executed.

    [Header("Dialogue UI")]
    public GameObject dialogueUI; // The UI element that contains all
dialogue components.
    public TextMeshProUGUI dialogueText; // The TextMeshProUGUI element used
to display the dialogue text.


    [Header("Choice Node UI")]
    public GameObject choicesContainer; // The container GameObject for
displaying choice buttons.
    public GameObject choicePrefab; // The prefab for each choice button.
Will be instantiated dynamically.

    private string language = "English"; // The language currently used to
retrieve dialogue text. Defaults to English.

    public void StartDialogue() {
        dialogueManager.StartDialogue(currentDialogue);
    }

    public override void OnDialogueNode(DialogueNode dialogueNode)
    {
        dialogueText.text = dialogueNode.GetMessage(language); // Display
message in the selected language.
    }

    public override void OnChoiceNode(ChoiceNode choiceNode)
    {
        dialogueText.text = choiceNode.GetMessage(language); // Display the
main message of the choice node.

        // Loop through each choice in the node to create corresponding UI
buttons.
        for (int i = 0; i < choiceNode.Choices.Count; i++) {
            GameObject choice = Instantiate(choicePrefab); // Create a new
choice button.
            choice.transform.SetParent(choicesContainer.transform, false);
// Set the parent of the button to the choices container.

            int index = i; // Store the index of this choice for the
button's callback.

            // Add a listener to handle when the choice is selected.
            Button button = choice.GetComponent<Button>();
            button.onClick.AddListener(() => OnChoiceClick(index));

            // Set the text of the choice button.
            TextMeshProUGUI textMeshPro =
choice.GetComponentInChildren<TextMeshProUGUI>();
            textMeshPro.text =
choiceNode.Choices[index].GetMessage(language);
        }
    }

    private void OnChoiceClick(int index)
    {
        dialogueManager.NextNode(index); // Move to the next node using the
choice index.

        // Clear all choice buttons from the choices container.
        foreach (Transform child in choicesContainer.transform) {
            Destroy(child.gameObject);
        }
    }

    public override void OnEventNode(EventNode eventNode,
DialogueEventManager dialogueEventManager)
    {
        dialogueEventManager.Invoke(eventNode.DialogueEvent); // Trigger the
dialogue event.
        dialogueManager.NextNode(); // Move to the next node.
    }

    public override void OnIfNode(IfNode ifNode)
    {
        dialogueManager.NextNode(); // Moving to the next node will evaluate
the If/Else If conditions automatically
    }

    public override void OnEndNode(EndNode endNode)
    {
        currentDialogue = endNode.NextDialogue; // Set the next dialogue, of
the end node to be the currentDialogue.
```

```
        dialogueUI.SetActive(false); // Hide the dialogue UI as the dialogue
session ends.
    }
}
```

4. In the Unity Editor select a game object that should trigger a dialogue and add your newly created script to it as a component by clicking the “Add Component” button.

Dialogue Tree Editor

To open a “Dialogue Tree Editor” go to Window/Dialogue System/Dialogue Tree Editor. Alternatively, double-click on any Dialogue Tree to bring up the editor.

Controls

The Dialogue Tree Editor includes several intuitive controls to make editing Dialogue trees quick and efficient.

Panning

To navigate around the node view, press and hold the **middle mouse button**, then drag the mouse to pan the view.

Zooming

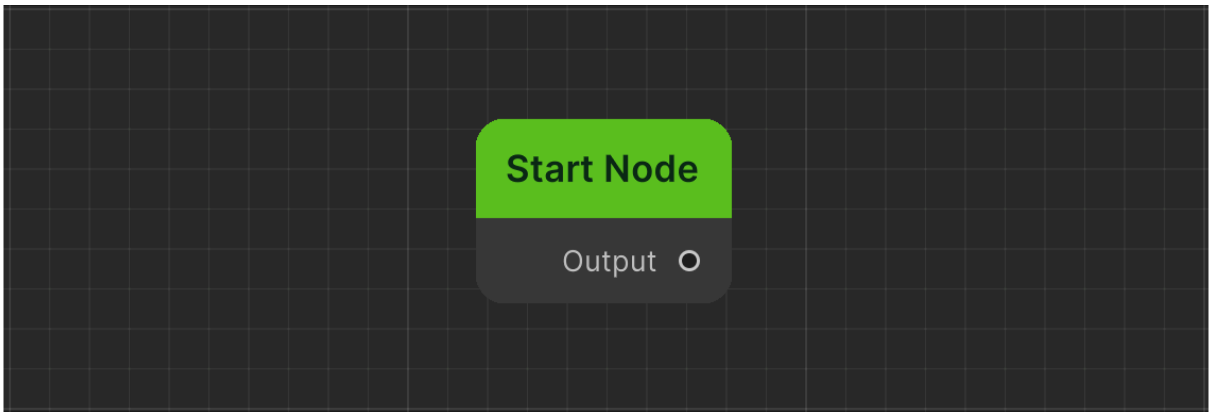
To adjust your view, scroll the **middle mouse wheel** to zoom in or out.

Moving Nodes

To reposition a node, **click and hold** on the title or an empty area within the node, then drag the mouse to move the node to a new position.

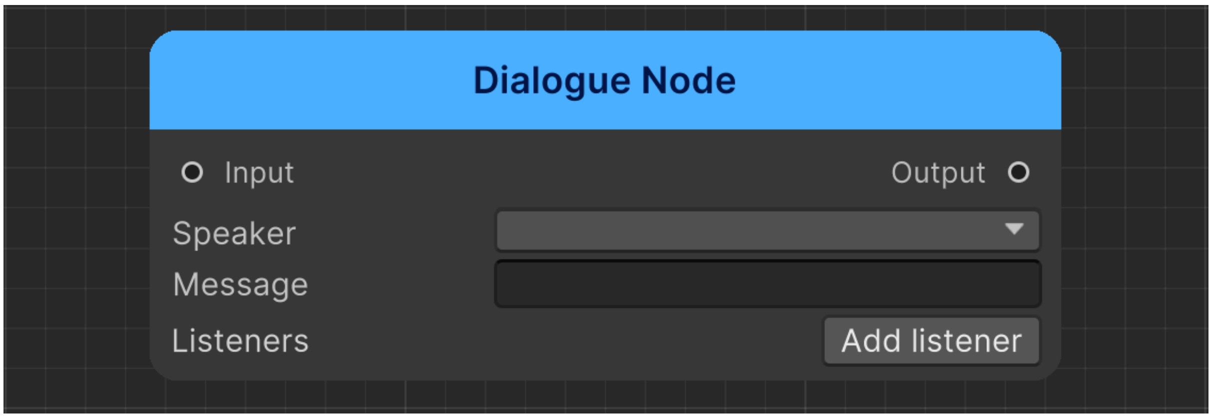
Nodes

Start Node



The Start Node is created automatically for each Dialogue Tree. Every Dialogue Tree can have only one Start Node. If you accidentally delete the Start Node from the Dialogue Tree, it will be recreated automatically when you reopen that tree in the Dialogue Tree Editor.

Dialogue Node

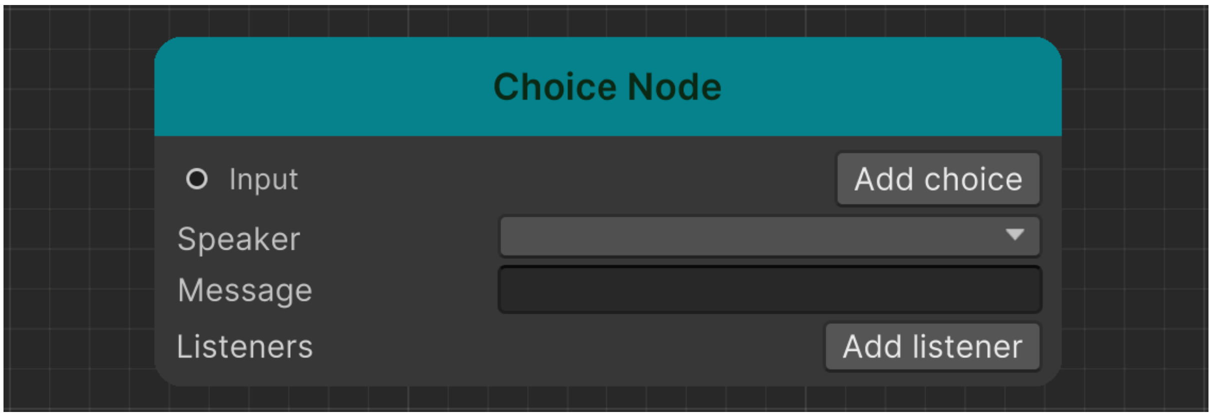


To create a Dialogue Node in your Dialogue Tree, right-click in the Dialogue Tree Editor window and select the "Dialogue Node" option from the pop-up menu.

Field Name	Property Name	Property Type	Property Description
Speaker	Speaker	Character	The character that is speaking.
Message	Accessible through a GetMessage Public Method	Public Method	The description can be found in the Public Methods of this node.
Listeners	Listeners	List<Character>	List of Characters that are listening to the conversation.

Public Methods		
Method Name	**Return Type**	**Method Description**
GetMessage(Language language)	string	Returns a message delivered by the Speaker. The language of the message should be passed as a parameter through the method.

Choice Node



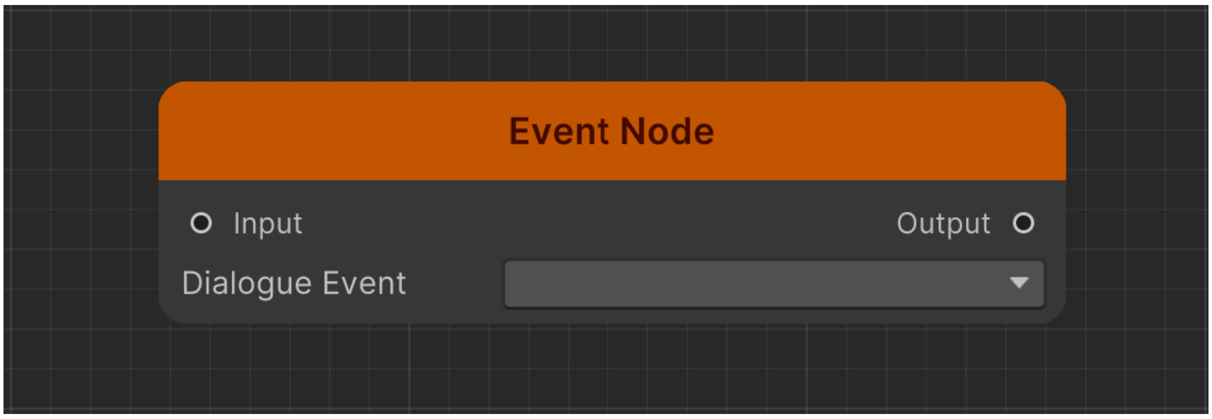
To create a Choice Node in your Dialogue Tree, right-click in the Dialogue Tree Editor window and select the “Choice Node” option from the pop-up menu.

Field Name	**Property Name**	**Property Type**	**Property Description**
Speaker	Speaker	Character	The character that is speaking.
Message	Accessible through a GetMessage Public Method	Public Method	The description can be found in the Public Methods of this node.
Choices	Choices	List<Choice>	List of all Choices of the Choice Node.
Default Choice	DefaultChoice	int	An index representing a default choice from the Choices list. There can be only one default choice.
Listeners	Listeners	List<Character>	List of Scriptable Objects representing characters that are listening.

Public Methods		
Method Name	**Return Type**	**Method Description**

GetMessage(Language language)	string	Returns a message delivered by the Speaker. The language of the message should be passed as a parameter through the method.

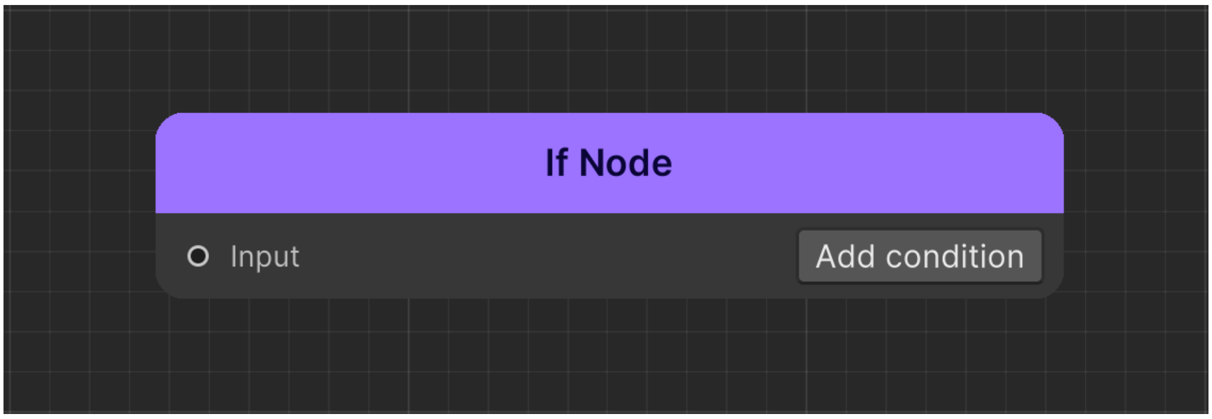
Event Node



To create an Event Node in your Dialogue Tree right-click in the Dialogue Tree Editor window and select the “Event Node” option from the pop-up menu.

Field Name	**Property Name**	**Property Type**	**Property Description**
Dialogue Event	DialogueEvent	DialogueEvent	A Dialogue Event enum that can be mapped with UnityEvent in the Dialogue Event Manager.

If Node

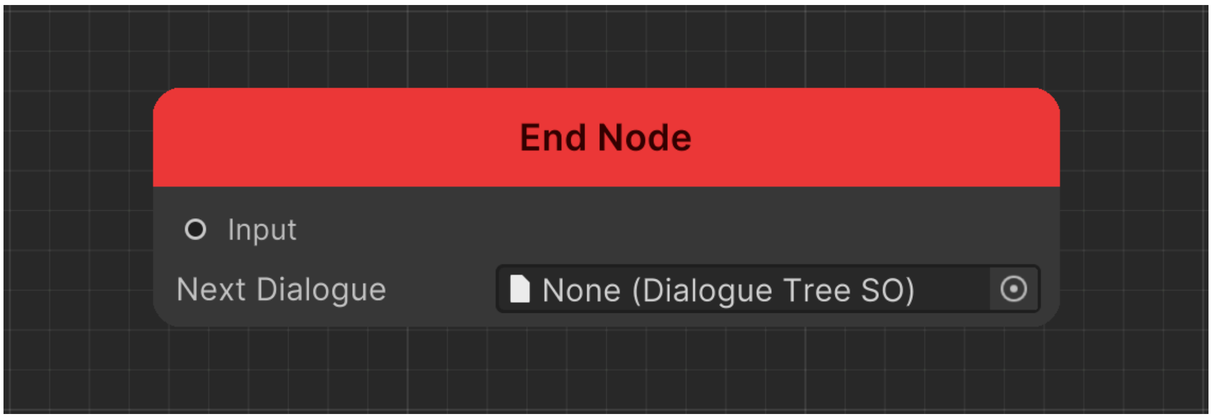


To create an If Node in your Dialogue Tree right-click in the Dialogue Tree Editor window and select the “If Node” option from the pop-up menu.

Field Name	**Property Name**	**Property Type**	**Property Description**

Variable Type	NOT ACCESSIBLE	string	A type of variable that should be checked in the condition.
If/Else if	NOT ACCESSIBLE	DialogueStringVariable / DialogueIntVariable / DialogueFloatVariable / DialogueBoolVariable	An enum representing a variable that should be checked.
Operator	NOT ACCESSIBLE	string	An operator is used to check the value of the current statement.
Value	NOT ACCESSIBLE	string/float/int/bool	Value of the variable that is being checked.

End Node

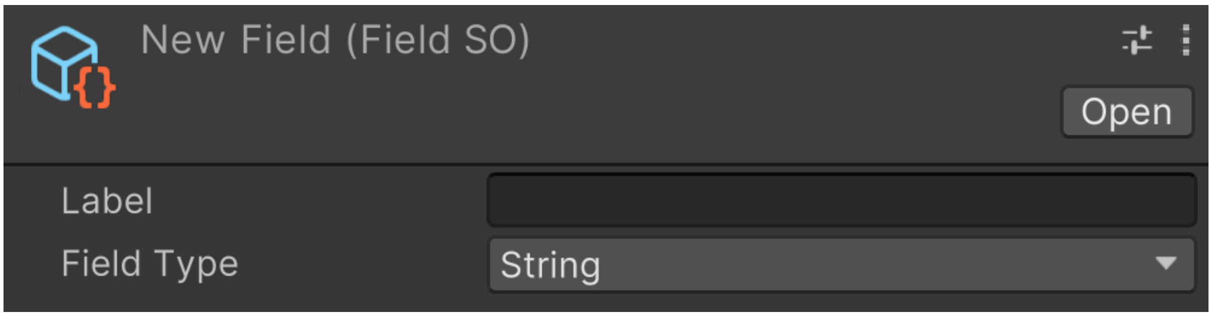


To create an End Node in your Dialogue Tree, right-click in the Dialogue Tree Editor window and select the “End Node” option from the pop-up menu.

Field Name	**Property Name**	**Property Type**	**Property Description**
Next Dialogue	NextDialogue	DialogueTreeSO	Scriptable Object representing a Dialogue Tree that should be set as the new current dialogue.

Extension of Nodes via Custom Fields

You can create custom Fields to extend the properties of each node. To create a new Field, open Unity's "Project" window, right-click, and select Create/Dialogue System/Field.



New Field (Field SO)

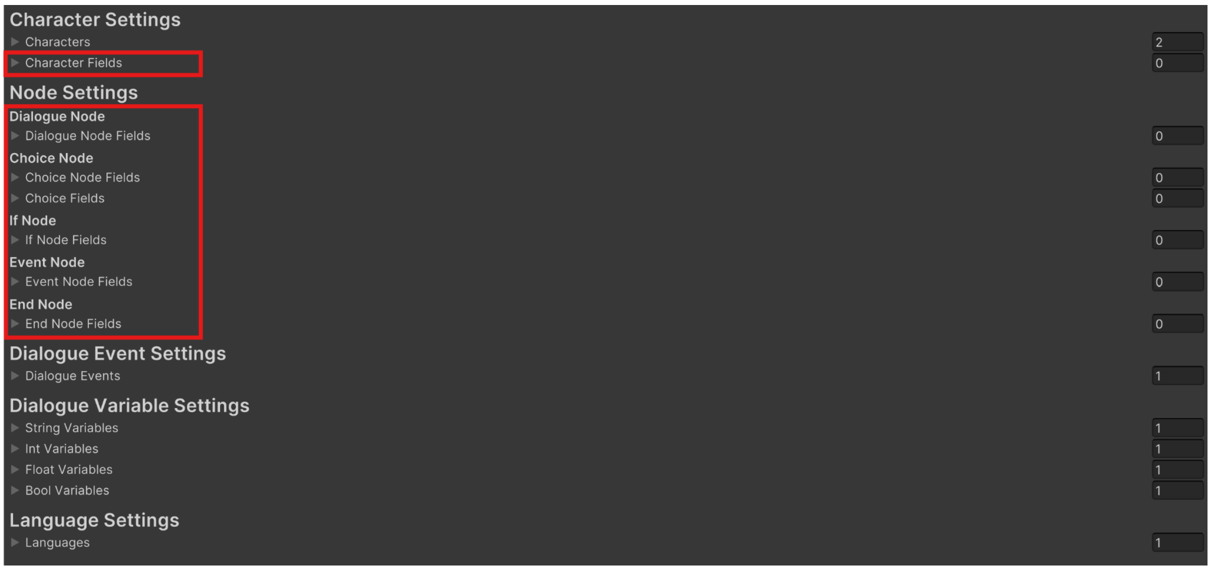
Open

Label	
Field Type	String

Each field has a couple of properties that need to be set.

1. The label property is the name of the field that will show in the Dialogue Tree Editor, and it's also the name you will use in code to get the value of that property.
2. The Field Type can be set to the following values: String, Int, Float, Bool, Enum, Vector2, Vector3, Sprite, GameObject, AudioClip, DialogueTree

Once the Field is created, you can add it to the Fields section for any of the nodes. Each section can have the same Field at most once. You can find the Fields sections under Edit/Project Settings…/Dialogue System.



Character Settings

Characters	2
Character Fields	0

Node Settings

Dialogue Node	
Dialogue Node Fields	0
Choice Node	
Choice Node Fields	0
Choice Fields	0
If Node	
If Node Fields	0
Event Node	
Event Node Fields	0
End Node	
End Node Fields	0

Dialogue Event Settings

Dialogue Events	1

Dialogue Variable Settings

String Variables	1
Int Variables	1
Float Variables	1
Bool Variables	1

Language Settings

Languages	1

Once you make any changes to the Dialogue System Settings, you will need to regenerate the code. To do that, close the Dialogue System Settings and in the upper left corner, navigate to Tools/Dialogue System and click on Regenerate Code.

Scripts

DialogueUIManager

DialogueUIManager is an abstract class designed to manage Dialogue Trees within a Unity scene. It acts as a bridge between GameObjects, UI elements, and dialogue logic, allowing you to implement custom behaviors for dialogue interactions.

This class provides useful properties for accessing dialogue-related data and overrideable methods for handling different types of dialogue nodes.

Property Name	**Property Type**	**Property Description**
dialogueManager	DialogueManager	Provides access to other properties and functions you can use to manage dialogues.

Public Override Methods		
Method Name	**Return Type**	**Method Description**
OnDialogueNode(DialogueNode)	void	Called when the dialogue system processes a DialogueNode. Use this method to handle UI changes based on the properties of a current DialogueNode.
OnChoiceNode(ChoiceNode)	void	Called when processing a ChoiceNode. You can use this to display dialogue choices to the player.
OnEventNode(EventNode, DialogueEventManager)	void	Called when processing an EventNode. Use this method to trigger game events during dialogue.
OnIfNode(IfNode)	void	Called when processing an IfNode. Calling dialogueManager.NextNode() in this function will handle the If/Else conditional logic automatically.
OnEndNode(EndNode)	void	Called when the dialogue system reaches an EndNode. Use this to clean up UI or trigger post-dialogue actions.

DialogueManager

The DialogueManager class handles dialogue flow within the Unity scene. It provides properties to track the current state of dialogue and methods to control dialogue progression.

Property Name	**Property Type**	**Property Description**
CurrentNode	Node	Returns the current node being processed by the DialogueManager.
DialogueStarted	bool	Returns true if a dialogue is currently active, otherwise false.

Method Name	**Return Type**	**Method Description**
StartDialogue(DialogueTreeSO)	void	Starts the dialogue tree that is provided as an argument.
NextNode(int choice = -1)	void	Call this function to move the currently processed dialogue to the next node. If the current node is a choice node, include the index of the picked choice as an argument to this function.
ExitDialogue()	void	Exits the current dialogue, stopping all further dialogue processing.

DialogueEventManager

DialogueEventManager allows you to trigger a Unity event that is mapped to DialogueEvent in the Dialogue Event Manager prefab.

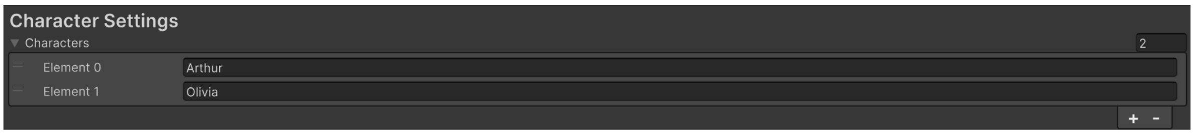
Method Name	**Return Type**	**Method Description**
Invoke(DialogueEvent)	void	Invokes a Unity event mapped to the DialogueEvent passed in the function.

DialogueVariableManager

DialogueVariableManager is attached to the Dialogue Variable Manager prefab, which maps DialogueVariables (DialogueStringVariable, DialogueIntVariable, DialogueFloatVariable, DialogueBoolVariable) with the corresponding variables from the Scene.

Character

To create a new Character, navigate to Edit/Project Settings…/Dialogue System. Under Character Settings, you can create a new Character simply by writing its name in the Characters list.



Element 0	Arthur
Element 1	Olivia

Property Name	Property Type	Property Description
CharacterName	string	

Choice

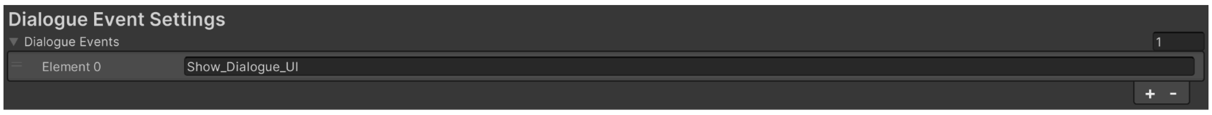
Property Name	Property Type	Property Description
IsDefaultChoice	bool	If true, the choice is a default one.

Public Methods		
Method Name	**Return Type**	**Method Description**
GetMessage(Language language)	string	Returns a message of the current Choice. The language of the message should be passed as a parameter through the method.

Enums

DialogueEvent

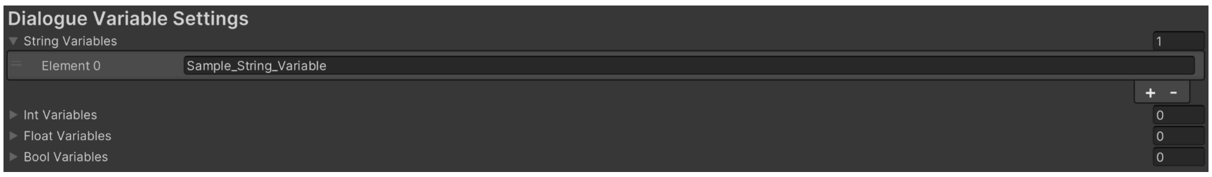
To create a new DialogueEvent, navigate to Edit/Project Settings…/Dialogue System. Under Dialogue Event Settings, you can create a new Dialogue Event simply by writing its name in the Dialogue Events list.



Dialogue Event Settings
Dialogue Events 1
Element 0 Show_Dialogue_UI

DialogueStringVariable

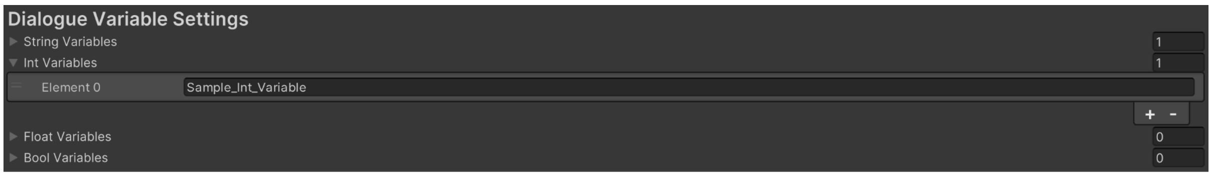
To create a new DialogueStringVariable, navigate to Edit/Project Settings…/Dialogue System. Under Dialogue Variable Settings, you can create a new DialogueStringVariable simply by writing its name in the String Variables list.



Dialogue Variable Settings
String Variables 1
Element 0 Sample_String_Variable
Int Variables 0
Float Variables 0
Bool Variables 0

DialogueIntVariable

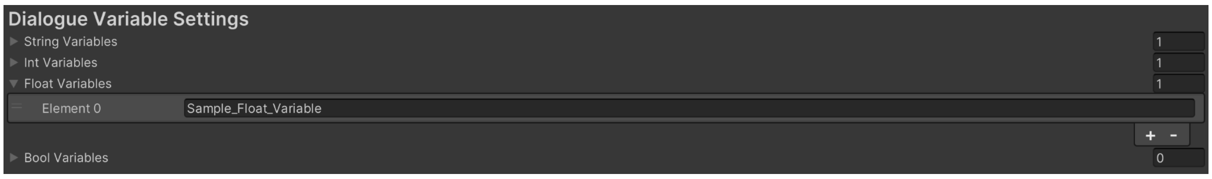
To create a new DialogueIntVariable, navigate to Edit/Project Settings…/Dialogue System. Under Dialogue Variable Settings, you can create a new DialogueIntVariable simply by writing its name in the Int Variables list.



Dialogue Variable Settings
String Variables 1
Int Variables 1
Element 0 Sample_Int_Variable
Float Variables 0
Bool Variables 0

DialogueFloatVariable

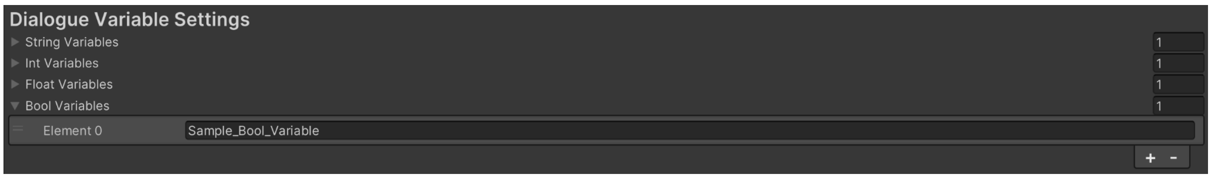
To create a new DialogueFloatVariable, navigate to Edit/Project Settings…/Dialogue System. Under Dialogue Variable Settings, you can create a new DialogueFloatVariable simply by writing its name in the Float Variables list.



Dialogue Variable Settings
String Variables 1
Int Variables 1
Float Variables 1
Element 0 Sample_Float_Variable
Bool Variables 0

DialogueBoolVariable

To create a new DialogueBoolVariable, navigate to Edit/Project Settings…/Dialogue System. Under Dialogue Variable Settings, you can create a new DialogueBoolVariable simply by writing its name in the Bool Variables list.

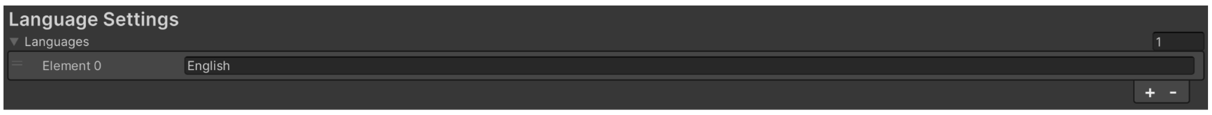


Dialogue Variable Settings

String Variables	1
Int Variables	1
Float Variables	1
Bool Variables	1
Element 0	Sample_Bool_Variable

Language

To create a new Language, navigate to Edit/Project Settings…/Dialogue System. Under Language Settings, you can create a new Language simply by writing its name in the Languages list. There has to be at least one Language in the Languages list at all times. If you clear the list, the system will recreate the English language automatically.



Language Settings

Languages	1
Element 0	English

Scriptable Objects

DialogueTreeSO

To create a new “Dialogue Tree”, open Unity’s “Project” window, right-click, and select Create/Dialogue System/Dialogue Tree. To open the “Dialogue Tree” in the Dialogue Tree Editor window, double-click on the “Dialogue Tree” asset.

Public Properties		
Property Name	**Property Type**	**Property Description**
DialogueID	string	GUID of the Dialogue Tree.

DialogueRegistrySO

The DialogueRegistrySO is a centralized ScriptableObject used to store and access multiple DialogueTreeSO assets. It serves as a runtime dictionary for fast lookup of dialogue trees by ID.

To create a new DialogueRegistrySO, right-click in Unity’s “Project” window and select Create/Dialogue System/Dialogue Registry.

Public Methods		
Method Name	**Return Type**	**Method Description**
GetDialogueByID(string id)	DialogueTreeSO	Returns a Dialogue Tree with the given ID.
ContainsID(string id)	bool	Returns true if the Dialogue Registry contains a dialogue tree with the given ID.

FieldSO

To create a new Field, open Unity’s “Project” window, right-click, and select Create/Dialogue System/Field.

Public Properties		
Property Name	**Property Type**	**Property Description**
FieldID	string	GUID of the Field.
Label	string	Name of the field that will show in the Dialogue Tree Editor. It is also used in the code to access the value of that field.

FieldType	CustomFieldType	Type of the field.
Choices	List<string>	A list of values used as enum values, if the field type is set to Enum.

Prefabs

Dialogue Event Manager prefab

The DialogueEventManager allows you to trigger Unity events from your dialogues.

The Dialogue Event Manager follows a Singleton pattern, meaning **only one instance can be active at a time**. If multiple Dialogue Event Managers are placed in a scene, only one will remain active, while the others will be automatically destroyed. It is recommended to use one Dialogue Event Manager per scene to avoid conflicts.

How to Use the Dialogue Event Manager

1. Drag and drop the Dialogue Event Manager prefab (located at ../Dialogue System/Prefabs/Managers) into the scene where you want to trigger Unity events.
2. In the Dialogue Event Manager, you'll find a field called Events, which is a dictionary. Use this dictionary to map DialogueEvent enums to the corresponding Unity events you want to trigger.

Dialogue Variable Manager prefab

The Dialogue Variable Manager allows you to check the value of any variable (of type int, float, string, and bool) within your scene and evaluate the logic in the If Node based on its value.

The Dialogue Variable Manager follows a Singleton pattern, meaning **only one instance can be active at a time**. If multiple Dialogue Variable Managers are placed in a scene, only one will remain active, while the others will be automatically destroyed. It is recommended to use one Dialogue Variable Manager per scene to avoid conflicts.

How to Use the Dialogue Variable Manager

1. Drag and drop the Dialogue Variable Manager prefab (located at ../Dialogue System/Prefabs/Managers) into the scene where you need variable support for the dialogue.
2. Inside the Dialogue Variable Manager game object, there are four dictionaries, one for each variable type: int, float, string, and bool. For each dictionary, map a DialogueVariable (DialogueStringVariable, DialogueIntVariable, etc.) to the corresponding variable you want to use in your dialogue:
    - Drag a GameObject with the target variable into the dictionary field.
    - Use the dropdown menu that appears to select the appropriate script attached to the GameObject.
    - Once the script is selected, pick the specific variable from the dropdown.

Variable Injection

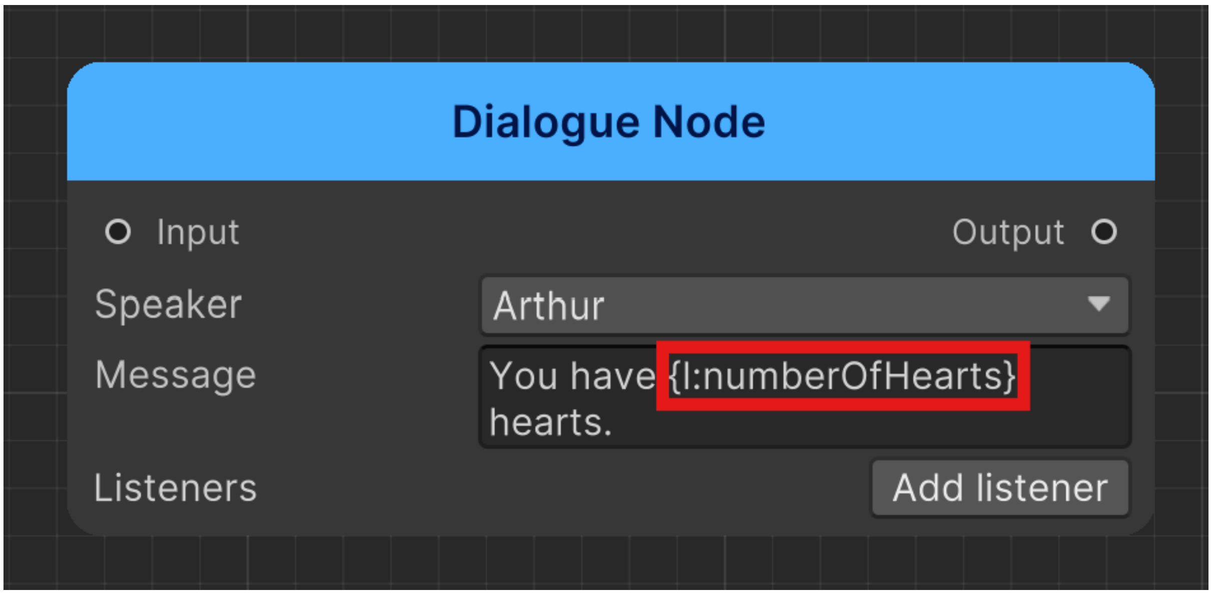
Variable injection allows you to use values of any dialogue variables dynamically within your dialogue text. It is supported **anywhere text is defined** in the Dialogue Tree Editor:

- Dialogue Node and Choice Node Message field.
- Message field in the individual Choice Node choices

To use variable injection:

1. Add and set up the Dialogue Variable Manager prefab in your scene.
2. Use the following syntax to inject variables into the Message field:
    a. {S:stringVariableName} for **String** variables
    b. {I:intVariableName} for **Int** variables
    c. {F:floatVariableName} for **Float** variables
    d. {B:boolVariableName} for **Bool** variables

Example of injecting an Int variable called "numberOfHearts" into DialogueNode:



Dialogue Node

Speaker: Arthur

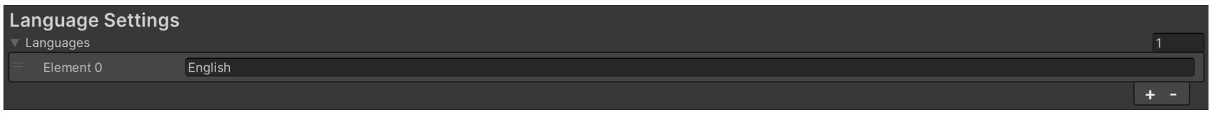
Message: You have {I:numberOfHearts} hearts.

WARNING: If a variable doesn't exist, the system will leave the placeholder as-is.

Localization

Create a new Language

To create a new language, navigate to Edit/Project Settings…/Dialogue System in the upper left corner. In the Language Settings section, you can create a new language simply by writing its name in the Languages list.


Language Settings
Languages 1
Element 0 English

Export Dialogue Tree as CSV

Exporting a single Dialogue Tree

To export a dialogue tree in CSV format, follow these steps:

1. Open the desired dialogue tree in the Dialogue Tree Editor.
2. Click the **"Export CSV"** button located in the editor's toolbar.
3. Choose your preferred location to save the file and confirm the action.

Your dialogue tree will now be saved as a CSV file in the specified location.

Exporting Multiple Dialogue Trees

To export multiple dialogue trees to CSV files, follow these steps:

1. In the **Unity Editor**, navigate to the **Project** window.
2. Select the dialogue tree assets you want to export. You can do this by holding **Ctrl** (Windows) or **Cmd** (Mac) while clicking to select multiple items.
3. Right-click on any of the selected dialogue tree assets.
4. From the context menu, choose **"Export to CSV"**.

Each selected dialogue tree will be exported as a separate CSV file and saved to the corresponding file locations or your specified export directory. Ensure all selected assets are valid dialogue trees to avoid problems during the export process.

Import Dialogue Tree from CSV

To import a CSV file into a dialogue tree, follow these steps:

1. Open the desired dialogue tree in the Dialogue Tree Editor.
2. Click the **"Import CSV"** button located in the editor's toolbar.
3. Select the desired CSV file from your system and confirm the selection.

The dialogue tree will be populated with the data from the imported CSV file. Ensure the CSV follows the required format for a seamless import.

Setting the Dialogue language

The Dialogue Tree Editor supports editing dialogue trees in multiple languages, making localization very straightforward. You can select a language from the dropdown menu in the **editor's upper-left corner**.

How to Use the Language Selection:

1. **Open the Dialogue Tree Editor**: Begin by opening the desired dialogue tree.
2. **Locate the Language Dropdown Menu**: In the upper-left corner of the editor, you'll find a dropdown menu listing all supported languages.
3. **Select a Language**: Click on the dropdown and choose your preferred language from the list. The selected language becomes the active one for editing.
4. **Edit Content**: After selecting a language, all dialogue nodes will be editable in the context of that specific language. Changes made will be saved as part of the localized data for the dialogue tree.