



# Namespace com.absence.dialoguesystem

## Classes

### [Dialogue](#)

The scriptable object derived type that holds all of the data which is essential for a dialogue.

### [DialogueAnimationsPlayer](#)

A small component which is responsible for playing the animations (if there is any) of the dialogue instance attached to the same game object.

### [DialogueDisplayer](#)

A singleton with the duty of displaying the current dialogue context. Written for the Unity UI package. Not compatible with the UI Toolkit.

### [DialogueExtensionBase](#)

This is the base class to derive from in order to handle some custom logic over the system.

### [DialogueInputHandler\\_Legacy](#)

### [DialogueInstance](#)

Lets you manage a single `DialoguePlayer` in the scene easily.

### [DialogueOptionText](#)

A small component that manages the functionality of an option's drawing and input.

### [DialoguePlayer](#)

Lets you progress in a dialogue easily.

### [DialogueSoundsPlayer](#)

A small component which is responsible for playing the sounds (if there is any) of the `DialogueInstance` attached to the same gameobject.

## Enums

### [DialoguePlayer.PlayerState](#)

# Delegates

## [DialogueInstance.SpeechEventHandler](#)

The delegate responsible for handling events directly about speech.



# Class Dialogue

The scriptable object derived type that holds all of the data which is essential for a dialogue.

## Inheritance

- ↳ [object](#)
- ↳ [Object](#)
- ↳ [ScriptableObject](#)
- ↳ [Dialogue](#)

## Inherited Members

[ScriptableObject.SetDirty\(\)](#)  
[ScriptableObject.CreateInstance\(string\)](#)  
[ScriptableObject.CreateInstance\(Type\)](#)  
[ScriptableObject.CreateInstance<T>\(\)](#)

Namespace: [com.absence.dialoguesystem](#)

Assembly: Assembly-CSharp-firstpass.dll

## Syntax

```
public class Dialogue : ScriptableObject
```

# Fields

## AllNodes

---

A list of all of the nodes that are in this dialogue.

### Declaration

```
[HideInInspector]  
public List<Node> AllNodes
```

### Field Value

## TYPE

---

List<Node>

# Blackboard

---

The `Blackboard` of this dialogue.

## Declaration

```
[HideInInspector]  
public Blackboard Blackboard
```

## Field Value

### TYPE

---

Blackboard

# LastOrCurrentNode

---

The current node reached while progressing in this dialogue. Or the last one reached before exiting the dialogue.

## Declaration

```
[HideInInspector]  
public Node LastOrCurrentNode
```

## Field Value

### TYPE

---

Node

# RootNode

---

The `RootNode` of this dialogue.

## Declaration

```
[HideInInspector]
```

```
public RootNode RootNode
```

## Field Value

### TYPE

---

RootNode

# Properties

## ClonedFrom

---

The original dialogue which is used to create this cloned one. Returns null if this dialogue is not a clone.

### Declaration

```
public Dialogue ClonedFrom { get; }
```

## Property Value

### TYPE

---

Dialogue

## IsClone

---

Use to check if this dialogue is a clone.

### Declaration

```
public bool IsClone { get; }
```

## Property Value

### TYPE

---

bool

## People

---

People in this dialogue (might be overridden on clones).

## Declaration

```
public List<Person> People { get; }
```

## Property Value

### TYPE

---

List<Person>

## Methods

### Clone()

---

Use to clone the dialogue scriptable object. Useful to progress in a copy while keeping the original unchanged.

## Declaration

```
public Dialogue Clone()
```

## Returns

### TYPE

---

Dialogue

### CreateNode(Type)

---

Use to create new nodes. Using runtime is not recommended.

## Declaration

```
public Node CreateNode(Type type)
```

## Parameters

### TYPE NAME

---

Type type

## Returns

**TYPE****Node**

## DeleteNode(Node)

---

Use to delete existing nodes. Using runtime is not recommended.

**Declaration**

```
public void DeleteNode(Node node)
```

**Parameters****TYPE NAME**

Node node

## GetAllDialogueParts()

---

Use to get a list of all `DialoguePartNode`s in this dialogue.

**Declaration**

```
public List<DialoguePartNode> GetAllDialogueParts()
```

**Returns****TYPE DESCRIPTION**List<DialoguePartNode> The entire list of `DialoguePartNode`s in the current dialogue.

## GetDialoguePartNodesWithName(string)

---

Use to find `DialoguePartNode`s with a specific name.

**Declaration**

```
public List<DialoguePartNode> GetDialoguePartNodesWithName(string targetName)
```

**Parameters**

**TYPE**    **NAME**

string    targetName

**Returns**

TYPE	DESCRIPTION
List<DialoguePartNode>	A list of <b>DialoguePartNode</b> s with that specific name. Throws an exception nothing's found.

## Initialize()

---

It teleports the flow back to the root node.

**Declaration**

```
public void Initialize()
```

## OverridePeople(List<Person>)

---

Use to override the people in this dialogue. Keeping person count the same is highly recommended. The original scriptable object's people list won't be affected by this.

**CAUTION!** The recommended way is to use this function on clones only.

**Declaration**

```
public void OverridePeople(List<Person> overridePeople)
```

**Parameters****TYPE**    **NAME**

List<Person>    overridePeople

## Pass(params object[])

---

Use to progress to the next node in the dialogue. Using this method directly is not recommended if you're not adding an extra functionality. You can consider using **DialoguePlayer** instead.

**Declaration**

```
public void Pass(params object[] passData)
```

## Parameters

TYPE	NAME
object[]	passData

## PerformEditorRefresh()

---

Invokes the `OnEditorRefresh`.

## Declaration

```
public void PerformEditorRefresh()
```

## Events

### OnEditorRefresh

---

Action which will get invoked if any value gets changed in the inspector when this dialogue is selected.

## Declaration

```
public event Action OnEditorRefresh
```

## Event Type

TYPE
Action



# Class DialogueAnimationsPlayer

A small component which is responsible for playing the animations (if there is any) of the dialogue instance attached to the same game object.

## Inheritance

- ↳ [object](#)
- ↳ [Object](#)
- ↳ [Component](#)
- ↳ [Behaviour](#)
- ↳ [MonoBehaviour](#)
- ↳ [DialogueExtensionBase](#)

[DialogueAnimationsPlayer](#)

## Inherited Members

[DialogueExtensionBase.m\\_instance](#)  
[DialogueExtensionBase.OnBeforeSpeech\(ref Person, ref string, ref List<Option>\)](#)  
[DialogueExtensionBase.OnAfterCloning\(\)](#)

Namespace: [com.absence.dialoguesystem](#)

Assembly: Assembly-CSharp-firstpass.dll

## Syntax

```
[RequireComponent(typeof(DialogueInstance))]  
[AddComponentMenu("absencee_absent-dialogues/Dialogue Animations Player")]  
public class DialogueAnimationsPlayer : DialogueExtensionBase
```

## Methods

### OnHandleAdditionalData(AdditionalSpeechData)

Use to define what to do with the current [AdditionalSpeechData](#). Gets called when the [m\\_instance](#) progresses.

#### Declaration

```
protected override void OnHandleAdditionalData(AdditionalSpeechData data)
```

## Parameters

TYPE	NAME
AdditionalSpeechData	data

## Overrides

[DialogueExtensionBase.OnHandleAdditionalData\(AdditionalSpeechData\)](#)



# Class DialogueDisplayer

A singleton with the duty of displaying the current dialogue context. Written for the Unity UI package. Not compatible with the UI Toolkit.

## Inheritance

- ↳ [object](#)
- ↳ [Object](#)
- ↳ [Component](#)
- ↳ [Behaviour](#)
- ↳ [MonoBehaviour](#)
- ↳ [StaticInstance<DialogueDisplayer>](#)
- [Singleton<DialogueDisplayer>](#)
- [DialogueDisplayer](#)

## Inherited Members

- [Singleton<DialogueDisplayer>.Awake\(\)](#)
- [StaticInstance<DialogueDisplayer>.OnApplicationQuit\(\)](#)
- [StaticInstance<DialogueDisplayer>.Instance](#)

Namespace: [com.absence.dialoguesystem](#)  
Assembly: Assembly-CSharp-firstpass.dll

## Syntax

```
[AddComponentMenu("absencee_absent-dialogues/Dialogue Displayer")]
public class DialogueDisplayer : Singleton<DialogueDisplayer>
```

## Methods

### Display(Person, string)

Displays a speech with no options.

#### Declaration

```
public void Display(Person speaker, string speech)
```

## Parameters

TYPE	NAME
Person	speaker
string	speech

## Display(Person, string, List<Option>, Action<int>)

---

Displays a speech with options.

### Declaration

```
public void Display(Person speaker, string speech, List<Option> options, Action<int> optionPressAction)
```

## Parameters

TYPE	NAME
Person	speaker
string	speech
List<Option>	options
Action<int>	optionPressAction

## Occupy()

---

Let's you occupy the singleton. If it is occupied by any other scripts about dialogues, you can't occupy.

### Declaration

```
public bool Occupy()
```

## Returns

TYPE	DESCRIPTION
bool	Returns false if the display is already occupied. Returns true otherwise.

## Release()

---

Removes the occupancy of the display. CAUTION! `DialogueDisplayer` does not hold a reference to the current occupier. Because of that, be careful calling this function.

## Declaration

```
public void Release()
```



# Class DialogueExtensionBase

This is the base class to derive from in order to handle some custom logic over the system.

## Inheritance

- ↳ [object](#)
- ↳ [Object](#)
- ↳ [Component](#)
- ↳ [Behaviour](#)
- ↳ [MonoBehaviour](#)
- ↳ [DialogueExtensionBase](#)

[DialogueAnimationsPlayer](#)

[DialogueInputHandler\\_Legacy](#)

[DialogueSoundsPlayer](#)

Namespace: [com.absence.dialoguesystem](#)

Assembly: Assembly-CSharp-firstpass.dll

## Syntax

```
[RequireComponent(typeof(DialogueInstance))]  
public abstract class DialogueExtensionBase : MonoBehaviour
```

# Fields

## m\_instance

---

[DialogueInstance](#) component attached to the current gameobject.

## Declaration

```
[SerializeField]  
[Readonly]  
protected DialogueInstance m_instance
```

## Field Value

## TYPE

---

DialogueInstance

# Methods

## OnAfterCloning()

---

Use to define what to do right after the target instance clones it's `ReferencedDialogue`.

### Declaration

```
protected virtual void OnAfterCloning()
```

## OnBeforeSpeech(ref Person, ref string, ref List<Option>)

---

Use to define what to do with the original speech data right before displaying it.

### Declaration

```
protected virtual void OnBeforeSpeech(ref Person speaker, ref string speech, ref List<Option> options)
```

### Parameters

TYPE	NAME	DESCRIPTION
Person	speaker	Speaker of this speech.
string	speech	Speech in context.
List<Option>	options	Option of this speech.

## OnHandleAdditionalData(AdditionalSpeechData)

---

Use to define what to do with the current `AdditionalSpeechData`. Gets called when the `m_instance` progresses.

### Declaration

```
protected virtual void OnHandleAdditionalData(AdditionalSpeechData data)
```

## Parameters

TYPE	NAME
AdditionalSpeechData	data



# Class DialogueInputHandler\_Legacy

## Inheritance

```
↳ object
  ↳ Object
    ↳ Component
      ↳ Behaviour
        ↳ MonoBehaviour
          ↳ DialogueExtensionBase
```

DialogueInputHandler\_Legacy

## Inherited Members

[DialogueExtensionBase.m\\_instance](#)  
[DialogueExtensionBase.OnBeforeSpeech\(ref Person, ref string, ref List<Option>\)](#)  
[DialogueExtensionBase.OnAfterCloning\(\)](#)

Namespace: [com.absence.dialoguesystem](#)

Assembly: Assembly-CSharp-firstpass.dll

## Syntax

```
public class DialogueInputHandler_Legacy : DialogueExtensionBase
```

## Methods

### OnHandleAdditionalData(AdditionalSpeechData)

Use to define what to do with the current `AdditionalSpeechData`. Gets called when the `m_instance` progresses.

#### Declaration

```
protected override void OnHandleAdditionalData(AdditionalSpeechData data)
```

#### Parameters

TYPE

NAME

---

AdditionalSpeechData data

## Overrides

[DialogueExtensionBase.OnHandleAdditionalData\(AdditionalSpeechData\)](#)



# Class DialogueInstance

Lets you manage a single [DialoguePlayer](#) in the scene easily.

## Inheritance

- ↳ [object](#)
- ↳ [Object](#)
- ↳ [Component](#)
- ↳ [Behaviour](#)
- ↳ [MonoBehaviour](#)
- ↳ [DialogueInstance](#)

Namespace: [com.absence.dialoguesystem](#)

Assembly: Assembly-CSharp-firstpass.dll

## Syntax

```
[AddComponentMenu("absencee_absent-dialogues/Dialogue Instance")]
[DisallowMultipleComponent]
public class DialogueInstance : MonoBehaviour
```

# Properties

## ClonedDialogue

---

Use to get the cloned dialogue which the [DialoguePlayer](#) of this instance uses.

### Declaration

```
public Dialogue ClonedDialogue { get; }
```

### Property Value

#### TYPE

---

[Dialogue](#)

# Player

---

[DialoguePlayer](#) of this instance.

## Declaration

```
public DialoguePlayer Player { get; }
```

## Property Value

### TYPE

---

[DialoguePlayer](#)

# ReferencedDialogue

---

Use to get the original (not the cloned one) dialogue of this instance.

## Declaration

```
public Dialogue ReferencedDialogue { get; }
```

## Property Value

### TYPE

---

[Dialogue](#)

# Methods

## AddExtension<T>()

---

Adds a [DialogueExtensionBase](#) to the target dialogue instance. **Does not work runtime.**

## Declaration

```
public void AddExtension<T>() where T : DialogueExtensionBase
```

## Type Parameters

## EnterDialogue()

---

Use to enter dialogue.

### Declaration

```
public bool EnterDialogue()
```

### Returns

TYPE	DESCRIPTION
------	-------------

---

bool	<b>False</b> if the <a href="#">DialogueDisplayer</a> is already occupied by any other script. Returns <b>true</b> otherwise.
------	---

## ExitDialogue()

---

Use to exit current dialogue.

### Declaration

```
public void ExitDialogue()
```

## Events

### OnAfterCloning

---

Action which will get invoked right after this instance clons it's [ReferencedDialogue](#).

### Declaration

```
public event Action OnAfterCloning
```

### Event Type

## OnBeforeSpeech

---

Subscribe to this delegate to override any data will get displayed.

### Declaration

```
public event DialogueInstance.SpeechEventHandler OnBeforeSpeech
```

### Event Type

---

[DialogueInstance.SpeechEventHandler](#)

## OnHandleAdditionalData

---

The Action which will get invoked when [HandleAdditionalData\(\)](#) gets called.

### Declaration

```
public event Action<AdditionalSpeechData> OnHandleAdditionalData
```

### Event Type

---

[Action<AdditionalSpeechData>](#)



# Delegate DialogueInstance.SpeechEvent Handler

The delegate responsible for handling events directly about speech.

Namespace: [com.absence.dialoguesystem](#)

Assembly: Assembly-CSharp-firstpass.dll

## Syntax

```
public delegate void DialogueInstance.SpeechEventHandler(ref Person speaker, ref string speech, ref List<Option> options);
```

## Parameters

TYPE	NAME	DESCRIPTION
Person	speaker	Speaker of this speech.
string	speech	Speech in context.
List<Option>	options	Options of this speech (null if there is no options).



# Class DialogueOptionText

A small component that manages the functionality of an option's drawing and input.

## Inheritance

- ↳ **object**
- ↳ Object
- ↳ Component
- ↳ Behaviour
- ↳ MonoBehaviour
- ↳ DialogueOptionText

Namespace: [com.absence.dialoguesystem](#)

Assembly: Assembly-CSharp-firstpass.dll

## Syntax

```
[AddComponentMenu("absencee_absent-dialogues/Option Text")]
public class DialogueOptionText : MonoBehaviour
```

## Methods

### Initialize(int, string)

---

Sets the index and the text of this option.

#### Declaration

```
public void Initialize(int optionIndex, string text)
```

#### Parameters

TYPE	NAME
------	------

int	optionIndex
-----	-------------

string	text
--------	------

# OnClick()

---

Calls `OnClickAction`.

## Declaration

```
public void OnClick()
```

# Events

## OnClickAction

---

### Declaration

```
public event Action<int> OnClickAction
```

### Event Type

#### TYPE

---

`Action<int>`



# Class DialoguePlayer

Lets you progress in a dialogue easily.

## Inheritance

↳ [object](#)  
↳ [DialoguePlayer](#)  
Namespace: [com.absence.dialoguesystem](#)  
Assembly: Assembly-CSharp-firstpass.dll

## Syntax

```
[Serializable]  
public class DialoguePlayer
```

## Constructors

### DialoguePlayer(Dialogue)

---

Use to create a new [DialoguePlayer](#).

#### Declaration

```
public DialoguePlayer(Dialogue dialogue)
```

#### Parameters

TYPE	NAME	DESCRIPTION
<a href="#">Dialogue</a>	dialogue	The original dialogue to clone from.

### DialoguePlayer(Dialogue, List<Person>)

---

Use to create a new [DialoguePlayer](#) with an overridden people list.

## Declaration

```
public DialoguePlayer(Dialogue dialogue, List<Person> overridePeople)
```

## Parameters

TYPE	NAME	DESCRIPTION
Dialogue	dialogue	The original dialogue to clone from.
List<Person>	overridePeople	The list of new people.

## Properties

### AdditionalSpeechData

Additional data of the current node.

#### Declaration

```
public AdditionalSpeechData AdditionalSpeechData { get; }
```

#### Property Value

##### TYPE

AdditionalSpeechData

## ClonedDialogue

The dialogue cloned from the original one from constructor.

#### Declaration

```
public Dialogue ClonedDialogue { get; }
```

#### Property Value

##### TYPE

Dialogue

# HasOptions

---

Use to check if current node is a `FastSpeechNode` or not.

## Declaration

```
public bool HasOptions { get; }
```

## Property Value

### TYPE

---

bool

# HasPerson

---

Use to check if current node `PersonDependent` or not.

## Declaration

```
public bool HasPerson { get; }
```

## Property Value

### TYPE

---

bool

# HasSpeech

---

Use to check if current node is a `IContainSpeech` or not.

## Declaration

```
public bool HasSpeech { get; }
```

## Property Value

### TYPE

---

bool

# Options

---

Options of the current node, if there is any.

## Declaration

```
public List<Option> Options { get; }
```

## Property Value

### TYPE

---

List<Option>

# Speaker

---

Person who speaks.

## Declaration

```
public Person Speaker { get; }
```

## Property Value

### TYPE

---

Person

# Speech

---

Speech of the current node.

## Declaration

```
public string Speech { get; }
```

## Property Value

### TYPE

---

string

# State

---

Current state of the player.

## Declaration

```
public DialoguePlayer.PlayerState State { get; }
```

## Property Value

### TYPE

---

DialoguePlayer.PlayerState

# Methods

## Continue(params object[])

---

Use to progress in the target dialogue with some optional data.

## Declaration

```
public void Continue(params object[] passData)
```

## Parameters

TYPE	NAME	DESCRIPTION
object[]	passData	Anything that you want to pass as data. (e.g. <a href="#">DecisionSpeechNode</a> uses the [0] element to get the selected option index.)

## TeleportToRoot()

---

Teleports the flow to the [RootNode](#) of the dialogue clone.

## Declaration

```
public void TeleportToRoot()
```

## OnContinue

---

Action which will get invoked when `Continue(params object[])` gets called.

### Declaration

```
public event Action<DialoguePlayer.PlayerState> OnContinue
```

### Event Type

TYPE

---

`Action<DialoguePlayer.PlayerState>`



# Enum DialoguePlayer.PlayerState

Shows what state the dialogue player is in.

Namespace: [com.absence.dialoguesystem](#)

Assembly: Assembly-CSharp-firstpass.dll

## Syntax

```
public enum DialoguePlayer.PlayerState
```

## Fields

### NAME

---

NoSpeech

WaitingForOption

WaitingForSkip

WillExit



# Class DialogueSoundsPlayer

A small component which is responsible for playing the sounds (if there is any) of the `DialogueInstance` attached to the same gameobject.

## Inheritance

- ↳ `object`
- ↳ `Object`
- ↳ `Component`
- ↳ `Behaviour`
- ↳ `MonoBehaviour`
- ↳ `DialogueExtensionBase`

`DialogueSoundsPlayer`

## Inherited Members

`DialogueExtensionBase.m_instance`  
`DialogueExtensionBase.OnBeforeSpeech(ref Person, ref string, ref List<Option>)`  
`DialogueExtensionBase.OnAfterCloning()`

Namespace: `com.absence.dialoguesystem`

Assembly: Assembly-CSharp-firstpass.dll

## Syntax

```
[RequireComponent(typeof(DialogueInstance))]  
[AddComponentMenu("absencee_absent-dialogues/Dialogue Sounds Player")]  
public class DialogueSoundsPlayer : DialogueExtensionBase
```

## Methods

### OnHandleAdditionalData(AdditionalSpeechData)

Use to define what to do with the current `AdditionalSpeechData`. Gets called when the `m_instance` progresses.

#### Declaration

```
protected override void OnHandleAdditionalData(AdditionalSpeechData data)
```

## Parameters

TYPE	NAME
AdditionalSpeechData	data

## Overrides

[DialogueExtensionBase.OnHandleAdditionalData\(AdditionalSpeechData\)](#)



# Namespace com.absence.dialoguesystem.editor

## Classes

[DialogueCustomEditor](#)[DialogueEditorWindow](#)[DialogueGraphView](#)[DialogueGraphView.UxmlFactory](#)[InspectorView](#)[InspectorView.UxmlFactory](#)[NodeCustomEditor](#)[NodeView](#)[RuntimeSelectionHandler](#)[VariableBankCreationHandler](#)



# Class DialogueCustomEditor

## Inheritance

```
↳ object
  ↳ Object
    ↳ ScriptableObject
      ↳ Editor
        ↳ DialogueCustomEditor
```

## Inherited Members

Editor.SaveChanges()  
Editor.DiscardChanges()  
**Editor.CreateEditorWithContext(Object[], Object, Type)**  
Editor.CreateEditorWithContext(Object[], Object)  
**Editor.CreateCachedEditorWithContext(Object, Object, Type, ref Editor)**  
**Editor.CreateCachedEditorWithContext(Object[], Object, Type, ref Editor)**  
**Editor.CreateCachedEditor(Object, Type, ref Editor)**  
**Editor.CreateCachedEditor(Object[], Type, ref Editor)**  
Editor.CreateEditor(Object)  
**Editor.CreateEditor(Object, Type)**  
Editor.CreateEditor(Object[])  
**Editor.CreateEditor(Object[], Type)**  
**Editor.DrawPropertiesExcluding(SerializedObject, params string[])**  
Editor.DrawDefaultInspector()  
Editor.Repaint()  
Editor.CreateInspectorGUI()  
Editor.RequiresConstantRepaint()  
Editor.DrawHeader()  
Editor.OnHeaderGUI()  
Editor.ShouldHideOpenButton()  
Editor.DrawFoldoutInspector(Object, ref Editor)  
Editor.HasPreviewGUI()  
Editor.GetPreviewTitle()  
**Editor.RenderStaticPreview(string, Object[], int, int)**  
Editor.OnPreviewGUI(Rect, GUIStyle)  
Editor.OnInteractivePreviewGUI(Rect, GUIStyle)  
Editor.OnPreviewSettings()  
Editor.GetInfoString()  
Editor.DrawPreview(Rect)  
Editor.ReloadPreviewInstances()  
Editor.UseDefaultMargins()  
Editor.MoveNextTarget()

Editor.ResetTarget()  
Editor.hasUnsavedChanges  
Editor.saveChangesMessage  
Editor.target  
Editor.targets  
Editor.serializedObject  
Editor.finishedDefaultHeaderGUI  
ScriptableObject.SetDirty()  
**ScriptableObject.CreateInstance(string)**  
**ScriptableObject.CreateInstance(Type)**  
ScriptableObject.CreateInstance<T>()  
Namespace: [com.absence.dialoguesystem.editor](#)  
Assembly: Assembly-CSharp-Editor-firstpass.dll

## Syntax

```
[CustomEditor(typeof(Dialogue), false)]  
public class DialogueCustomEditor : Editor
```

# Methods

## OnInspectorGUI()

---

Implement this function to make a custom inspector.

### Declaration

```
public override void OnInspectorGUI()
```

### Overrides

UnityEditor.Editor.OnInspectorGUI()



# Class DialogueEditorWindow

## Inheritance

```
↳ object
  ↳ Object
    ↳ ScriptableObject
      ↳ EditorWindow
        ↳ DialogueEditorWindow
```

## Inherited Members

[ScriptableObject.SetDirty\(\)](#)  
[ScriptableObject.CreateInstance\(string\)](#)  
[ScriptableObject.CreateInstance\(Type\)](#)  
[ScriptableObject.CreateInstance<T>\(\)](#)

Namespace: [com.absence.dialoguesystem.editor](#)  
Assembly: Assembly-CSharp-Editor-firstpass.dll

## Syntax

```
public class DialogueEditorWindow : EditorWindow
```

## Methods

### CreateGUI()

---

#### Declaration

```
public void CreateGUI()
```

### FrameToNode(Node)

---

#### Declaration

```
public void FrameToNode(Node node)
```

## Parameters

TYPE	NAME
------	------

Node	node
------	------

## LoadLastDialogue()

---

### Declaration

```
public static void LoadLastDialogue()
```

## OnOpenAsset(int, int)

---

### Declaration

```
[OnOpenAsset]  
public static bool OnOpenAsset(int instanceId, int line)
```

## Parameters

TYPE	NAME
------	------

int	instanceId
-----	------------

int	line
-----	------

## Returns

TYPE
------

bool
------

## OpenWindow()

---

### Declaration

```
[MenuItem("absencee/_absent-dialogues/Open Dialogue Graph Window")]
public static void OpenWindow()
```

## PopulateDialogueView(Dialogue)

---

### Declaration

```
public static bool PopulateDialogueView(Dialogue dialogue)
```

### Parameters

TYPE	NAME
------	------

Dialogue	dialogue
----------	----------

### Returns

TYPE
------

bool
------

## SaveLastDialogue()

---

### Declaration

```
public static void SaveLastDialogue()
```

## SelectNode(Node)

---

### Declaration

```
public void SelectNode(Node node)
```

### Parameters

TYPE	NAME
------	------

Node	node
------	------

## OnGUIDelayCall

---

### Declaration

```
public static event Action OnGUIDelayCall
```

### Event Type

TYPE

---

Action



# Class DialogueGraphView

## Inheritance

- ↳ [object](#)
- ↳ [CallbackEventHandler](#)
- ↳ [Focusable](#)
- ↳ [VisualElement](#)
- ↳ [GraphView](#)
- ↳ [DialogueGraphView](#)

## Implements

- [IEventHandler](#)
- [IResolvedStyle](#)
- [ITransform](#)
- [ITransitionAnimations](#)
- [IExperimentalFeatures](#)
- [IVisualElementScheduler](#)

Namespace: [com.absence.dialoguesystem.editor](#)

Assembly: Assembly-CSharp-Editor-firstpass.dll

## Syntax

```
public class DialogueGraphView : GraphView, IEventHandler, IResolvedStyle, ITransform, ITransitionAnim
```

## Constructors

### DialogueGraphView()

#### Declaration

```
public DialogueGraphView()
```

## Methods

# BuildContextualMenu(ContextualMenuPopulateEvent)

---

Add menu items to the contextual menu.

## Declaration

```
public override void BuildContextualMenu(ContextualMenuPopulateEvent evt)
```

## Parameters

TYPE	NAME	DESCRIPTION
ContextMenuPopulateEvent	evt	The event holding the menu to populate.

## Overrides

UnityEditor.Experimental.GraphView.GraphView.BuildContextualMenu(UnityEngine.UIElements.ContextualMenuPopulateEvent)

# FindNodeView(Node)

---

## Declaration

```
public NodeView FindNodeView(Node node)
```

## Parameters

TYPE	NAME
Node	node

## Returns

TYPE
NodeView

# GetCompatiblePorts(Port, NodeAdapter)

---

Get all ports compatible with given port.

## Declaration

```
public override List<Port> GetCompatiblePorts(Port startPort, NodeAdapter nodeAdapter)
```

## Parameters

TYPE	NAME	DESCRIPTION
Port	startPort	Start port to validate against.
NodeAdapter	nodeAdapter	Node adapter.

## Returns

TYPE	DESCRIPTION
List<Port>	List of compatible ports.

## Overrides

UnityEditor.Experimental.GraphView.GraphView.GetCompatiblePorts(UnityEditor.Experimental.GraphView.Port, UnityEditor.Experimental.GraphView.NodeAdapter)

## Refresh()

---

### Declaration

```
public void Refresh()
```

## Events

### OnNodeSelected

---

#### Declaration

```
public event Action<NodeView> OnNodeSelected
```

#### Event Type

TYPE
Action<NodeView>

## OnPopulateView

---

### Declaration

```
public event Action OnPopulateView
```

## Event Type

TYPE

---

Action

# Implements

UnityEngine.UIElements.IEventHandler

UnityEngine.UIElements.IResolvedStyle

UnityEngine.UIElements.ITransform

UnityEngine.UIElements.Experimental.ITransitionAnimations

UnityEngine.UIElements.IExperimentalFeatures

UnityEngine.UIElements.IVisualElementScheduler



# Class DialogueGraphView.UxmlFactory

## Inheritance

↳ [object](#)

↳ [BaseUxmlFactory<DialogueGraphView, VisualElement.UxmlTraits>](#)

↳ [UxmlFactory<DialogueGraphView, VisualElement.UxmlTraits>](#)

↳ [DialogueGraphView.UxmlFactory](#)

## Implements

[IUxmlFactory](#)

[IBaseUxmlFactory](#)

Namespace: [com.absence.dialoguesystem.editor](#)

Assembly: Assembly-CSharp-Editor-firstpass.dll

## Syntax

```
public class DialogueGraphView.UxmlFactory : UxmlFactory<DialogueGraphView, VisualElement.UxmlTraits>,
```

## Implements

[UnityEngine.UIElements.IUxmlFactory](#)

[UnityEngine.UIElements.IBaseUxmlFactory](#)



# Class InspectorView

## Inheritance

- ↳ [object](#)
- ↳ [CallbackEventHandler](#)
- ↳ [Focusable](#)
- ↳ [VisualElement](#)
- ↳ [InspectorView](#)

## Implements

[IEventHandler](#)

[IResolvedStyle](#)

[ITransform](#)

[ITransitionAnimations](#)

[IExperimentalFeatures](#)

[IVisualElementScheduler](#)

Namespace: [com.absence.dialoguesystem.editor](#)

Assembly: Assembly-CSharp-Editor-firstpass.dll

## Syntax

```
public class InspectorView : VisualElement, IEventHandler, IResolvedStyle, ITransform, ITransitionAnim
```

## Constructors

### InspectorView()

#### Declaration

```
public InspectorView()
```

## Implements

UnityEngine.UIElements.IEventHandler  
UnityEngine.UIElements.IResolvedStyle  
UnityEngine.UIElements.ITransform  
UnityEngine.UIElements.Experimental.ITransitionAnimations  
UnityEngine.UIElements.IExperimentalFeatures  
UnityEngine.UIElements.IVisualElementScheduler



# Class InspectorView.UxmlFactory

## Inheritance

- ↳ [object](#)
- ↳ [BaseUxmlFactory<InspectorView, VisualElement.UxmlTraits>](#)
- ↳ [UxmlFactory<InspectorView, VisualElement.UxmlTraits>](#)
- ↳ [InspectorView.UxmlFactory](#)

## Implements

[IUxmlFactory](#)  
[IBaseUxmlFactory](#)

Namespace: [com.absence.dialoguesystem.editor](#)  
Assembly: Assembly-CSharp-Editor-firstpass.dll

## Syntax

```
public class InspectorView.UxmlFactory : UxmlFactory<InspectorView, VisualElement.UxmlTraits>, IUxmlFac
```

## Implements

[UnityEngine.UIElements.IUxmlFactory](#)  
[UnityEngine.UIElements.IBaseUxmlFactory](#)



# Class NodeCustomEditor

## Inheritance

```
↳ object
  ↳ Object
    ↳ ScriptableObject
      ↳ Editor
        ↳ NodeCustomEditor
```

## Inherited Members

Editor.SaveChanges()  
Editor.DiscardChanges()  
**Editor.CreateEditorWithContext(Object[], Object, Type)**  
Editor.CreateEditorWithContext(Object[], Object)  
**Editor.CreateCachedEditorWithContext(Object, Object, Type, ref Editor)**  
**Editor.CreateCachedEditorWithContext(Object[], Object, Type, ref Editor)**  
**Editor.CreateCachedEditor(Object, Type, ref Editor)**  
**Editor.CreateCachedEditor(Object[], Type, ref Editor)**  
Editor.CreateEditor(Object)  
**Editor.CreateEditor(Object, Type)**  
Editor.CreateEditor(Object[])  
**Editor.CreateEditor(Object[], Type)**  
**Editor.DrawPropertiesExcluding(SerializedObject, params string[])**  
Editor.DrawDefaultInspector()  
Editor.Repaint()  
Editor.CreateInspectorGUI()  
Editor.RequiresConstantRepaint()  
Editor.DrawHeader()  
Editor.OnHeaderGUI()  
Editor.ShouldHideOpenButton()  
Editor.DrawFoldoutInspector(Object, ref Editor)  
Editor.HasPreviewGUI()  
Editor.GetPreviewTitle()  
**Editor.RenderStaticPreview(string, Object[], int, int)**  
Editor.OnPreviewGUI(Rect, GUIStyle)  
Editor.OnInteractivePreviewGUI(Rect, GUIStyle)  
Editor.OnPreviewSettings()  
Editor.GetInfoString()  
Editor.DrawPreview(Rect)  
Editor.ReloadPreviewInstances()  
Editor.UseDefaultMargins()  
Editor.MoveNextTarget()

Editor.ResetTarget()  
Editor.hasUnsavedChanges  
Editor.saveChangesMessage  
Editor.target  
Editor.targets  
Editor.serializedObject  
Editor.finishedDefaultHeaderGUI  
ScriptableObject.SetDirty()  
**ScriptableObject.CreateInstance(string)**  
**ScriptableObject.CreateInstance(Type)**  
ScriptableObject.CreateInstance<T>()  
Namespace: [com.absence.dialoguesystem.editor](#)  
Assembly: Assembly-CSharp-Editor-firstpass.dll

## Syntax

```
[CustomEditor(typeof(Node), true)]  
public class NodeCustomEditor : Editor
```

# Methods

## OnInspectorGUI()

---

Implement this function to make a custom inspector.

### Declaration

```
public override void OnInspectorGUI()
```

### Overrides

UnityEditor.Editor.OnInspectorGUI()



# Class NodeView

## Inheritance

```
↳ object
  ↳ CallbackEventHandler
    ↳ Focusable
      ↳ VisualElement
        ↳ GraphElement
          ↳ Node
```

NodeView

## Implements

```
IEventHandler
IResolvedStyle
ITransform
ITransitionAnimations
IExperimentalFeatures
IVisualElementScheduler
```

Namespace: [com.absence.dialoguesystem.editor](#)

Assembly: Assembly-CSharp-Editor-firstpass.dll

## Syntax

```
public class NodeView : Node, IEventHandler, IResolvedStyle, ITransform, ITransitionAnimations, IExperi
```

## Constructors

### NodeView(Node)

#### Declaration

```
public NodeView(Node node)
```

#### Parameters

**TYPE** NAME

---

Node node

## Fields

### Input

---

#### Declaration

```
public Port Input
```

#### Field Value

**TYPE**

---

Port

## K\_PERSONDEPENDENT\_CLASSNAME

---

#### Declaration

```
public static string K_PERSONDEPENDENT_CLASSNAME
```

#### Field Value

**TYPE**

---

string

## Node

---

#### Declaration

```
public Node Node
```

#### Field Value

**TYPE**

---

Node

## OnNodeSelected

---

### Declaration

```
public Action<NodeView> OnNodeSelected
```

### Field Value

**TYPE**

---

Action<NodeView>

## Outputs

---

### Declaration

```
public List<Port> Outputs
```

### Field Value

**TYPE**

---

List<Port>

## m\_serializedNode

---

### Declaration

```
protected SerializedObject m_serializedNode
```

### Field Value

**TYPE**

---

SerializedObject

# Properties

## Master

---

### Declaration

```
public DialogueGraphView Master { get; }
```

### Property Value

#### TYPE

---

DialogueGraphView

## Methods

### OnSelected()

---

Called when the GraphElement is selected.

### Declaration

```
public override void OnSelected()
```

### Overrides

UnityEditor.Experimental.GraphView.GraphElement.OnSelected()

### OnUnselected()

---

Called when the GraphElement is unselected.

### Declaration

```
public override void OnUnselected()
```

### Overrides

UnityEditor.Experimental.GraphView.GraphElement.OnUnselected()

# SetPosition(Rect)

---

Set node position.

## Declaration

```
public override void SetPosition(Rect newPos)
```

## Parameters

TYPE	NAME	DESCRIPTION
------	------	-------------

Rect	newPos	New position.
------	--------	---------------

## Overrides

UnityEditor.Experimental.GraphView.Node.SetPosition(UnityEngine.Rect)

## Implements

UnityEngine.UIElements.IEventHandler

UnityEngine.UIElements.IResolvedStyle

UnityEngine.UIElements.ITransform

UnityEngine.UIElements.Experimental.ITransitionAnimations

UnityEngine.UIElements.IExperimentalFeatures

UnityEngine.UIElements.IVisualElementScheduler



# Class RuntimeSelectionHandler

## Inheritance

↳ [object](#)  
↳ [RuntimeSelectionHandler](#)

Namespace: [com.absence.dialoguesystem.editor](#)

Assembly: Assembly-CSharp-Editor-firstpass.dll

## Syntax

```
[InitializeOnLoad]  
public static class RuntimeSelectionHandler
```



# Class VariableBankCreationHandler

## Inheritance

↳ [object](#)  
↳ [VariableBankCreationHandler](#)

Namespace: [com.absence.dialoguesystem.editor](#)

Assembly: Assembly-CSharp-Editor-firstpass.dll

## Syntax

```
public class VariableBankCreationHandler
```



# Namespace com.absence.dialoguesystem.internals

## Classes

### ActionNode

Node which invokes some actions on the flow.

### AdditionalSpeechData

### Blackboard

This is a class for holding any variables in the dialogues. It also contains a `com.absence.variablesystem.VariableBank`.

### ConditionNode

Node which re-routes the flow under some conditions.

### DecisionSpeechNode

Node which displays a speech with options.

### DialoguePartNode

Node which let's you create more and separate routes.

### FastSpeechNode

Node which displays a speech without options.

### GotoNode

Node which teleports the flow to a specific `DialoguePartNode`.

### Node

This is the base abstract class to derive from for any new node subtypes.

### Option

The type to hold references to dialogue options.

### Option.ShowIf

A class specifically designed for calculating an option's visibility.

### RootNode

Node which is essential if you want to have a dialogue graph.

## StickyNoteNode

Node which contains a user defined string.

## TitleNode

Node which is simply `StickyNoteNode` but bigger.

# Interfaces

## IContainSpeech

Interface to use if any of your dialogue elements has a speech, has options or has `AdditionalSpeechData`.

## IContainVariableManipulators

Any node subtype with this interface implemented will refresh its `com.absence.variablesystem.VariableComparer`s and `com.absence.variablesystem.VariableSetter`s to have the correct reference to the `Bank` of the current `Dialogue` everytime the editor window refreshes.

## IPerformEditorRefresh

Any node subtype with this interface implemented will get a callback when you change any value in the inspector.

# Enums

## Node.NodeState

Describes the node's state on the flow. While progressing in the dialogue.

## VBProcessType



# Class ActionNode

Node which invokes some actions on the flow.

## Inheritance

- ↳ [object](#)
- ↳ [Object](#)
- ↳ [ScriptableObject](#)
- ↳ [Node](#)
- ↳ [ActionNode](#)

## Implements

- [IContainVariableManipulators](#)
- [IPerformEditorRefresh](#)

## Inherited Members

- [Node.Guid](#)
- [Node.Position](#)
- [Node.MasterDialogue](#)
- [Node.Blackboard](#)
- [Node.State](#)
- [Node.ExitDialogAfterwards](#)
- [Node.OnSetState](#)
- [Node.OnRemove](#)
- [Node.OnValidation](#)
- [Node.OnReach](#)
- [Node.OnPass](#)
- [Node.PersonIndex](#)
- [Node.Person](#)
- [Node.DisplayState](#)
- [Node.ShowInMinimap](#)
- [Node.PersonDependent](#)
- [Node.AddNextNode\(Node, int\)](#)
- [Node.RemoveNextNode\(int\)](#)
- [Node.GetNextNodes\(\)](#)
- [Node.Pass\(params object\[\]\)](#)
- [Node.Reach\(\)](#)
- [Node.OnRemoval\(\)](#)
- [Node.GetInputPortNameForCreation\(\)](#)
- [Node.GetOutputPortNamesForCreation\(\)](#)
- [Node.SetState\(Node.NodeState\)](#)
- [Node.Clone\(\)](#)

[Node.OnValidate\(\)](#)  
[ScriptableObject.SetDirty\(\)](#)  
[ScriptableObject.CreateInstance\(string\)](#)  
[ScriptableObject.CreateInstance\(Type\)](#)  
[ScriptableObject.CreateInstance<T>\(\)](#)  
Namespace: [com.absence.dialoguesystem.internals](#)  
Assembly: Assembly-CSharp-firstpass.dll

## Syntax

```
public class ActionNode : Node, IContainVariableManipulators, IPerformEditorRefresh
```

## Fields

### Next

---

#### Declaration

```
[HideInInspector]  
public Node Next
```

#### Field Value

##### TYPE

---

Node

## UnityEvents

---

#### Declaration

```
public UnityEvent UnityEvents
```

#### Field Value

##### TYPE

---

UnityEvent

# VBActions

---

## Declaration

```
public List<FixedVariableSetter> VBActions
```

## Field Value

### TYPE

---

[List<FixedVariableSetter>](#)

## Methods

### AddNextNode\_Inline(Node, int)

---

Use to write the functionality of connecting a node to any port of this node.

## Declaration

```
protected override void AddNextNode_Inline(Node nextWillBeAdded, int atPort)
```

## Parameters

### TYPE NAME

---

[Node](#) nextWillBeAdded

[int](#) atPort

## Overrides

[Node.AddNextNode\\_Inline\(Node, int\)](#)

### CustomAction()

---

## Declaration

```
protected virtual void CustomAction()
```

# DelayedClone(Dialogue)

---

This method will get called right after the dialogue gets cloned.

## Declaration

```
public void DelayedClone(Dialogue originalDialogue)
```

## Parameters

TYPE	NAME	DESCRIPTION
Dialogue	originalDialogue	This is the dialogue the cloned dialogue had cloned from.

# GetClassName()

---

Use if you have a special USS class for this node. If you don't have any, return null.

## Declaration

```
public override string GetClassName()
```

## Returns

TYPE	DESCRIPTION
string	Returns the USS class name of this node type as a string.

## Overrides

[Node.GetClassName\(\)](#)

# GetComparers()

---

A list of comparers which you want to restrict in terms of `com.absence.variablesystem.VariableBank` selection

## Declaration

```
public List<FixedVariableComparer> GetComparers()
```

## Returns

## TYPE

---

`List<FixedVariableComparer>`

# GetNextNodes\_Inline(ref List<(int portIndex, Node node)>)

---

Use to describe the editor which nodes are the next nodes of this one in the chain by modifying the list.

## Declaration

```
protected override void GetNextNodes_Inline(ref List<(int portIndex, Node node)> result)
```

## Parameters

TYPE	NAME
<code>List&lt;(int portIndex, Node node)&gt;</code>	<code>result</code>

## Overrides

`Node.GetNextNodes_Inline(ref List<(int portIndex, Node node)>)`

# GetSetters()

---

A list of comparers which you want to restrict in terms of `com.absence.variablesystem.VariableBank` selection

## Declaration

```
public List<FixedVariableSetter> GetSetters()
```

## Returns

### TYPE

---

`List<FixedVariableSetter>`

# GetTitle()

---

Use to set the title of this node type in the graph view.

## Declaration

```
public override string GetTitle()
```

## Returns

TYPE	DESCRIPTION
------	-------------

string	The title as a string.
--------	------------------------

## Overrides

[Node.GetTitle\(\)](#)

## Pass\_Inline(params object[])

---

Use to write what happens when the dialogue passes this node.

## Declaration

```
protected override void Pass_Inline(params object[] passData)
```

## Parameters

TYPE	NAME
------	------

object[]	passData
----------	----------

## Overrides

[Node.Pass\\_Inline\(params object\[\]\)](#)

## PerformEditorRefresh()

---

Use to declare what to do when any value gets changed in the inspector/

## Declaration

```
public void PerformEditorRefresh()
```

## Reach\_Inline()

---

Use to write what happens when the dialogue reaches this node.

## Declaration

```
protected override void Reach_Inline()
```

## Overrides

[Node.Reach\\_Inline\(\)](#)

# RemoveNextNode\_Inline(int)

---

Use to write the functionality of removing the next node of this one.

## Declaration

```
protected override void RemoveNextNode_Inline(int atPort)
```

## Parameters

TYPE	NAME
int	atPort

## Overrides

[Node.RemoveNextNode\\_Inline\(int\)](#)

# Traverse(Action<Node>)

---

Use to traverse any action on a node chain. Nodes not connected directly won't transmit the action to another.

## Declaration

```
public override void Traverse(Action<Node> action)
```

## Parameters

TYPE	NAME
Action<Node>	action

## Overrides

[Node.Traverse\(Action<Node>\)](#)

# Implements

IContainVariableManipulators

IPerformEditorRefresh



# Class AdditionalSpeechData

## Inheritance

↳ [object](#)  
↳ [AdditionalSpeechData](#)

Namespace: [com.absence.dialoguesystem.internals](#)

Assembly: Assembly-CSharp-firstpass.dll

## Syntax

```
[Serializable]  
public class AdditionalSpeechData
```

# Properties

## AnimatorMemberName

---

### Declaration

```
public string AnimatorMemberName { get; }
```

### Property Value

#### TYPE

---

[string](#)

## AudioClip

---

### Declaration

```
public AudioClip AudioClip { get; }
```

## Property Value

TYPE

---

AudioClip

## CustomInfo

---

### Declaration

```
public string[] CustomInfo { get; }
```

## Property Value

TYPE

---

string[]

## Sprite

---

### Declaration

```
public Sprite Sprite { get; }
```

## Property Value

TYPE

---

Sprite



# Class Blackboard

This is a class for holding any variables in the dialogues. It also contains a `com.absence.variablesystem.VariableBank`.

## Inheritance

↳ `object`

↳ `Blackboard`

Namespace: `com.absence.dialoguesystem.internals`

Assembly: Assembly-CSharp-firstpass.dll

## Syntax

```
[Serializable]  
public class Blackboard
```

# Fields

## Bank

---

Bank of this blackboard.

## Declaration

```
[HideInInspector]  
public VariableBank Bank
```

## Field Value

### TYPE

---

VariableBank

# Methods

# Clone()

---

Use to clone this blackboard.

## Declaration

```
public Blackboard Clone()
```

## Returns

### TYPE

---

[Blackboard](#)



# Class ConditionNode

Node which re-routes the flow under some conditions.

## Inheritance

- ↳ [object](#)
- ↳ [Object](#)
- ↳ [ScriptableObject](#)
- ↳ [Node](#)
- ↳ [ConditionNode](#)

## Implements

- [IContainVariableManipulators](#)
- [IPerformEditorRefresh](#)

## Inherited Members

- [Node.Guid](#)
- [Node.Position](#)
- [Node.MasterDialogue](#)
- [Node.Blackboard](#)
- [Node.State](#)
- [Node.ExitDialogAfterwards](#)
- [Node.OnSetState](#)
- [Node.OnRemove](#)
- [Node.OnValidation](#)
- [Node.OnReach](#)
- [Node.OnPass](#)
- [Node.PersonIndex](#)
- [Node.Person](#)
- [Node.DisplayState](#)
- [Node.ShowInMinimap](#)
- [Node.PersonDependent](#)
- [Node.AddNextNode\(Node, int\)](#)
- [Node.RemoveNextNode\(int\)](#)
- [Node.GetNextNodes\(\)](#)
- [Node.Pass\(params object\[\]\)](#)
- [Node.Reach\(\)](#)
- [Node.OnRemoval\(\)](#)
- [Node.GetInputPortNameForCreation\(\)](#)
- [Node.SetState\(Node.NodeState\)](#)
- [Node.Clone\(\)](#)
- [Node.OnValidate\(\)](#)

ScriptableObject.SetDirty()  
ScriptableObject.CreateInstance(string)  
ScriptableObject.CreateInstance(Type)  
ScriptableObject.CreateInstance<T>()  
Namespace: [com.absence.dialoguesystem.internals](#)  
Assembly: Assembly-CSharp-firstpass.dll

## Syntax

```
public class ConditionNode : Node, IContainVariableManipulators, IPerformEditorRefresh
```

# Fields

## Comparers

---

### Declaration

```
public List<FixedVariableComparer> Comparers
```

### Field Value

#### TYPE

---

[List<FixedVariableComparer>](#)

## FalseNext

---

### Declaration

```
[HideInInspector]  
public Node FalseNext
```

### Field Value

#### TYPE

---

[Node](#)

# Processor

---

## Declaration

```
public VBProcessType Processor
```

## Field Value

### TYPE

---

VBProcessType

## TrueNext

---

## Declaration

```
[HideInInspector]  
public Node TrueNext
```

## Field Value

### TYPE

---

Node

## Methods

### AddNextNode\_Inline(Node, int)

---

Use to write the functionality of connecting a node to any port of this node.

## Declaration

```
protected override void AddNextNode_Inline(Node nextWillBeAdded, int atPort)
```

## Parameters

### TYPE NAME

---

Node nextWillBeAdded

int atPort

## Overrides

## DelayedClone(Dialogue)

---

This method will get called right after the dialogue gets cloned.

### Declaration

```
public void DelayedClone(Dialogue originalDialogue)
```

### Parameters

TYPE	NAME	DESCRIPTION
Dialogue	originalDialogue	This is the dialogue the cloned dialogue had cloned from.

## GetClassName()

---

Use if you have a special USS class for this node. If you don't have any, return null.

### Declaration

```
public override string GetClassName()
```

### Returns

TYPE	DESCRIPTION
string	Returns the USS class name of this node type as a string.

### Overrides

[Node.GetClassName\(\)](#)

## GetComparers()

---

A list of comparers which you want to restrict in terms of `com.absence.variablesystem.VariableBank` selection

### Declaration

```
public List<FixedVariableComparer> GetComparers()
```

## Returns

### TYPE

`List<FixedVariableComparer>`

## GetNextNodes\_Inline(ref List<(int portIndex, Node node)>)

Use to describe the editor which nodes are the next nodes of this one in the chain by modifying the list.

### Declaration

```
protected override void GetNextNodes_Inline(ref List<(int portIndex, Node node)> result)
```

### Parameters

TYPE	NAME
<code>List&lt;(int portIndex, Node node)&gt;</code>	<code>result</code>

### Overrides

[Node.GetNextNodes\\_Inline\(ref List<\(int portIndex, Node node\)>\)](#)

## GetOutputPortNamesForCreation()

Use to describe the dialogue editor how many output ports this node has and what are their names.

### Declaration

```
public override List<string> GetOutputPortNamesForCreation()
```

## Returns

TYPE	DESCRIPTION
<code>List&lt;string&gt;</code>	Returns the port names as a list of strings. Return an empty list if you want no output ports.

### Overrides

[Node.GetOutputPortNamesForCreation\(\)](#)

## GetSetters()

## Declaration

```
public List<FixedVariableSetter> GetSetters()
```

## Returns

### TYPE

---

`List<FixedVariableSetter>`

## GetTitle()

---

Use to set the title of this node type in the graph view.

## Declaration

```
public override string GetTitle()
```

## Returns

### TYPE DESCRIPTION

---

`string` The title as a string.

## Overrides

`Node.GetTitle()`

## Pass\_Inline(params object[])

---

Use to write what happens when the dialogue passes this node.

## Declaration

```
protected override void Pass_Inline(params object[] passData)
```

## Parameters

### TYPE NAME

---

`object[]` `passData`

## Overrides

## PerformEditorRefresh()

---

Use to declare what to do when any value gets changed in the inspector/

### Declaration

```
public void PerformEditorRefresh()
```

## Process()

---

### Declaration

```
protected virtual bool Process()
```

### Returns

#### TYPE

---

bool

## Reach\_Inline()

---

Use to write what happens when the dialogue reaches this node.

### Declaration

```
protected override void Reach_Inline()
```

### Overrides

[Node.Reach\\_Inline\(\)](#)

## RemoveNextNode\_Inline(int)

---

Use to write the functionality of removing the next node of this one.

### Declaration

```
protected override void RemoveNextNode_Inline(int atPort)
```

## Parameters

TYPE	NAME
int	atPort

## Overrides

[Node.RemoveNextNode\\_Inline\(int\)](#)

# Traverse(Action<Node>)

---

Use to traverse any action on a node chain. Nodes not connected directly won't transmit the action to another.

## Declaration

```
public override void Traverse(Action<Node> action)
```

## Parameters

TYPE	NAME
Action<Node>	action

## Overrides

[Node.Traverse\(Action<Node>\)](#)

# Implements

[IContainVariableManipulators](#)

[IPerformEditorRefresh](#)



# Class DecisionSpeechNode

Node which displays a speech with options.

## Inheritance

- ↳ [object](#)
- ↳ [Object](#)
- ↳ [ScriptableObject](#)
- ↳ [Node](#)
- ↳ [DecisionSpeechNode](#)

## Implements

- [IContainSpeech](#)
- [IContainVariableManipulators](#)
- [IPerformEditorRefresh](#)

## Inherited Members

- [Node.Guid](#)
- [Node.Position](#)
- [Node.MasterDialogue](#)
- [Node.Blackboard](#)
- [Node.State](#)
- [Node.ExitDialogAfterwards](#)
- [Node.OnSetState](#)
- [Node.OnRemove](#)
- [Node.OnValidation](#)
- [Node.OnReach](#)
- [Node.OnPass](#)
- [Node.PersonIndex](#)
- [Node.Person](#)
- [Node.DisplayState](#)
- [Node.ShowInMinimap](#)
- [Node.AddNextNode\(Node, int\)](#)
- [Node.RemoveNextNode\(int\)](#)
- [Node.GetNextNodes\(\)](#)
- [Node.Pass\(params object\[\]\)](#)
- [Node.Reach\(\)](#)
- [Node.OnRemoval\(\)](#)
- [Node.GetInputPortNameForCreation\(\)](#)
- [Node.SetState\(Node.NodeState\)](#)
- [Node.Clone\(\)](#)
- [ScriptableObject.SetDirty\(\)](#)

ScriptableObject.CreateInstance(string)

ScriptableObject.CreateInstance(Type)

ScriptableObject.CreateInstance<T>()

Namespace: [com.absence.dialoguesystem.internals](#)

Assembly: Assembly-CSharp-firstpass.dll

## Syntax

```
public sealed class DecisionSpeechNode : Node, IContainSpeech, IContainVariableManipulators, IPerformEc
```

## Fields

### Options

---

#### Declaration

```
[Space(10)]  
public List<Option> Options
```

#### Field Value

##### TYPE

---

[List<Option>](#)

## Speech

---

#### Declaration

```
[HideInInspector]  
public string Speech
```

#### Field Value

##### TYPE

---

[string](#)

## Properties

# PersonDependent

---

Is this node person dependent.

## Declaration

```
public override bool PersonDependent { get; }
```

## Property Value

### TYPE

---

bool

## Overrides

[Node.PersonDependent](#)

# Methods

## AddNextNode\_Inline(Node, int)

---

Use to write the functionality of connecting a node to any port of this node.

## Declaration

```
protected override void AddNextNode_Inline(Node nextWillBeAdded, int atPort)
```

## Parameters

TYPE	NAME
------	------

---

Node	nextWillBeAdded
------	-----------------

int	atPort
-----	--------

## Overrides

[Node.AddNextNode\\_Inline\(Node, int\)](#)

## DelayedClone(Dialogue)

---

This method will get called right after the dialogue gets cloned.

## Declaration

```
public void DelayedClone(Dialogue originalDialogue)
```

### Parameters

TYPE	NAME	DESCRIPTION
Dialogue	originalDialogue	This is the dialogue the cloned dialogue had cloned from.

## GetAdditionalSpeechData()

---

### Declaration

```
public AdditionalSpeechData GetAdditionalSpeechData()
```

### Returns

#### TYPE

---

AdditionalSpeechData

## GetClassName()

---

Use if you have a special USS class for this node. If you don't have any, return null.

### Declaration

```
public override string GetClassName()
```

### Returns

#### TYPE DESCRIPTION

---

string Returns the USS class name of this node type as a string.

### Overrides

[Node.GetClassName\(\)](#)

## GetComparers()

---

A list of comparers which you want to restrict in terms of `com.absence.variablesystem.VariableBank` selection

## Declaration

```
public List<FixedVariableComparer> GetComparers()
```

## Returns

### TYPE

---

`List<FixedVariableComparer>`

## GetNextNodes\_Inline(ref List<(int portIndex, Node node)>)

---

Use to describe the editor which nodes are the next nodes of this one in the chain by modifying the list.

## Declaration

```
protected override void GetNextNodes_Inline(ref List<(int portIndex, Node node)> result)
```

## Parameters

TYPE	NAME
<code>List&lt;(int portIndex, Node node)&gt;</code>	<code>result</code>

## Overrides

`Node.GetNextNodes_Inline(ref List<(int portIndex, Node node)>)`

## GetOptions()

---

## Declaration

```
public List<Option> GetOptions()
```

## Returns

### TYPE

---

`List<Option>`

# GetOutputPortNamesForCreation()

---

Use to describe the dialogue editor how many output ports this node has and what are their names.

## Declaration

```
public override List<string> GetOutputPortNamesForCreation()
```

## Returns

TYPE	DESCRIPTION
List<string>	Returns the port names as a list of strings. Return an empty list if you want no output ports.

## Overrides

[Node.GetOutputPortNamesForCreation\(\)](#)

# GetSetters()

---

A list of comparers which you want to restrict in terms of `com.absence.variablesystem.VariableBank` selection

## Declaration

```
public List<FixedVariableSetter> GetSetters()
```

## Returns

TYPE
List<FixedVariableSetter>

# GetSpeech()

---

## Declaration

```
public string GetSpeech()
```

## Returns

TYPE
string

# GetTitle()

---

Use to set the title of this node type in the graph view.

## Declaration

```
public override string GetTitle()
```

## Returns

TYPE	DESCRIPTION
------	-------------

string	The title as a string.
--------	------------------------

## Overrides

[Node.GetTitle\(\)](#)

# Pass\_Inline(params object[])

---

Use to write what happens when the dialogue passes this node.

## Declaration

```
protected override void Pass_Inline(params object[] passData)
```

## Parameters

TYPE	NAME
------	------

object[]	passData
----------	----------

## Overrides

[Node.Pass\\_Inline\(params object\[\]\)](#)

# PerformEditorRefresh()

---

Use to declare what to do when any value gets changed in the inspector/

## Declaration

```
public void PerformEditorRefresh()
```

# Reach\_Inline()

---

Use to write what happens when the dialogue reaches this node.

## Declaration

```
protected override void Reach_Inline()
```

## Overrides

[Node.Reach\\_Inline\(\)](#)

# RemoveNextNode\_Inline(int)

---

Use to write the functionality of removing the next node of this one.

## Declaration

```
protected override void RemoveNextNode_Inline(int atPort)
```

## Parameters

TYPE	NAME
int	atPort

## Overrides

[Node.RemoveNextNode\\_Inline\(int\)](#)

# Traverse(Action<Node>)

---

Use to traverse any action on a node chain. Nodes not connected directly won't transmit the action to another.

## Declaration

```
public override void Traverse(Action<Node> action)
```

## Parameters

TYPE	NAME
Action<Node>	action

## Overrides

# Implements

IContainSpeech

IContainVariableManipulators

IPerformEditorRefresh



# Class DialoguePartNode

Node which let's you create more and separate routes.

## Inheritance

- ↳ [object](#)
- ↳ [Object](#)
- ↳ [ScriptableObject](#)
- ↳ [Node](#)
- ↳ [DialoguePartNode](#)

## Inherited Members

- [Node.Guid](#)
- [Node.Position](#)
- [Node.MasterDialogue](#)
- [Node.Blackboard](#)
- [Node.State](#)
- [Node.ExitDialogAfterwards](#)
- [Node.OnSetState](#)
- [Node.OnRemove](#)
- [Node.OnValidation](#)
- [Node.OnReach](#)
- [Node.OnPass](#)
- [Node.PersonIndex](#)
- [Node.Person](#)
- [Node.ShowInMinimap](#)
- [Node.PersonDependent](#)
- [Node.AddNextNode\(Node, int\)](#)
- [Node.RemoveNextNode\(int\)](#)
- [Node.GetNextNodes\(\)](#)
- [Node.Pass\(params object\[\]\)](#)
- [Node.Reach\(\)](#)
- [Node.OnRemoval\(\)](#)
- [Node.GetOutputPortNamesForCreation\(\)](#)
- [Node.SetState\(Node.NodeState\)](#)
- [Node.Clone\(\)](#)
- [ScriptableObject.SetDirty\(\)](#)
- [ScriptableObject.CreateInstance\(string\)](#)
- [ScriptableObject.CreateInstance\(Type\)](#)
- [ScriptableObject.CreateInstance<T>\(\)](#)

Namespace: [com.absence.dialoguesystem.internals](#)

Assembly: Assembly-CSharp-firstpass.dll

## Syntax

```
public sealed class DialoguePartNode : Node
```

## Fields

### DialoguePartName

---

#### Declaration

```
public string DialoguePartName
```

#### Field Value

##### TYPE

---

string

## Next

---

#### Declaration

```
[HideInInspector]
public Node Next
```

#### Field Value

##### TYPE

---

Node

## Properties

### DisplayState

---

Will this node display it's state in editor on the flow.

## Declaration

```
public override bool DisplayState { get; }
```

## Property Value

### TYPE

bool

## Overrides

[Node.DisplayState](#)

## Methods

### AddNextNode\_Inline(Node, int)

---

Use to write the functionality of connecting a node to any port of this node.

## Declaration

```
protected override void AddNextNode_Inline(Node nextWillBeAdded, int atPort)
```

## Parameters

### TYPE NAME

Node nextWillBeAdded

int atPort

## Overrides

[Node.AddNextNode\\_Inline\(Node, int\)](#)

### DelayedClone(Dialogue)

---

This method will get called right after the dialogue gets cloned.

## Declaration

```
public void DelayedClone(Dialogue originalDialogue)
```

## Parameters

TYPE	NAME	DESCRIPTION
Dialogue	originalDialogue	This is the dialogue the cloned dialogue had cloned from.

## GetClassName()

Use if you have a special USS class for this node. If you don't have any, return null.

### Declaration

```
public override string GetClassName()
```

### Returns

TYPE	DESCRIPTION
string	Returns the USS class name of this node type as a string.

### Overrides

[Node.GetClassName\(\)](#)

## GetInputPortNameForCreation()

Use to describe the name of the input port of this node.

### Declaration

```
public override string GetInputPortNameForCreation()
```

### Returns

TYPE	DESCRIPTION
string	Returns the name as a string. Return null if you don't want any input ports.

### Overrides

[Node.GetInputPortNameForCreation\(\)](#)

## GetNextNodes\_Inline(ref List<(int portIndex, Node node)>)

Use to describe the editor which nodes are the next nodes of this one in the chain by modifying the list.

## Declaration

```
protected override void GetNextNodes_Inline(ref List<(int portIndex, Node node)> result)
```

## Parameters

TYPE	NAME
List<(int portIndex, Node node)>	result

## Overrides

[Node.GetNextNodes\\_Inline\(ref List<\(int portIndex, Node node\)>\)](#)

## GetTitle()

---

Use to set the title of this node type in the graph view.

## Declaration

```
public override string GetTitle()
```

## Returns

TYPE	DESCRIPTION
string	The title as a string.

## Overrides

[Node.GetTitle\(\)](#)

## Pass\_Inline(params object[])

---

Use to write what happens when the dialogue passes this node.

## Declaration

```
protected override void Pass_Inline(params object[] passData)
```

## Parameters

TYPE NAME

---

object[] passData

## Overrides

[Node.Pass\\_Inline\(params object\[\]\)](#)

# Reach\_Inline()

---

Use to write what happens when the dialogue reaches this node.

## Declaration

```
protected override void Reach_Inline()
```

## Overrides

[Node.Reach\\_Inline\(\)](#)

# RemoveNextNode\_Inline(int)

---

Use to write the functionality of removing the next node of this one.

## Declaration

```
protected override void RemoveNextNode_Inline(int atPort)
```

## Parameters

TYPE NAME

---

int atPort

## Overrides

[Node.RemoveNextNode\\_Inline\(int\)](#)

# Traverse(Action<Node>)

---

Use to traverse any action on a node chain. Nodes not connected directly won't transmit the action to another.

## Declaration

```
public override void Traverse(Action<Node> action)
```

## Parameters

TYPE	NAME
Action<Node>	action

## Overrides

[Node.Traverse\(Action<Node>\)](#)



# Class FastSpeechNode

Node which displays a speech without options.

## Inheritance

- ↳ [object](#)
- ↳ [Object](#)
- ↳ [ScriptableObject](#)
- ↳ [Node](#)
- ↳ [FastSpeechNode](#)

## Implements

[IContainSpeech](#)

## Inherited Members

- [Node.Guid](#)
- [Node.Position](#)
- [Node.MasterDialogue](#)
- [Node.Blackboard](#)
- [Node.State](#)
- [Node.ExitDialogAfterwards](#)
- [Node.OnSetState](#)
- [Node.OnRemove](#)
- [Node.OnValidation](#)
- [Node.OnReach](#)
- [Node.OnPass](#)
- [Node.PersonIndex](#)
- [Node.Person](#)
- [Node.DisplayState](#)
- [Node.ShowInMinimap](#)
- [Node.AddNextNode\(Node, int\)](#)
- [Node.RemoveNextNode\(int\)](#)
- [Node.GetNextNodes\(\)](#)
- [Node.Pass\(params object\[\]\)](#)
- [Node.Reach\(\)](#)
- [Node.OnRemoval\(\)](#)
- [Node.GetInputPortNameForCreation\(\)](#)
- [Node.GetOutputPortNamesForCreation\(\)](#)
- [Node.SetState\(Node.NodeState\)](#)
- [Node.Clone\(\)](#)
- [ScriptableObject.SetDirty\(\)](#)
- [ScriptableObject.CreateInstance\(string\)](#)

[ScriptableObject.CreateInstance\(Type\)](#)

[ScriptableObject.CreateInstance<T>\(\)](#)

Namespace: [com.absence.dialoguesystem.internals](#)

Assembly: Assembly-CSharp-firstpass.dll

## Syntax

```
public sealed class FastSpeechNode : Node, IContainSpeech
```

## Fields

### Next

---

#### Declaration

```
[HideInInspector]  
public Node Next
```

#### Field Value

TYPE

---

[Node](#)

## Speech

---

#### Declaration

```
[HideInInspector]  
public string Speech
```

#### Field Value

TYPE

---

[string](#)

## Properties

# PersonDependent

---

Is this node person dependent.

## Declaration

```
public override bool PersonDependent { get; }
```

## Property Value

### TYPE

---

bool

## Overrides

[Node.PersonDependent](#)

# Methods

## AddNextNode\_Inline(Node, int)

---

Use to write the functionality of connecting a node to any port of this node.

## Declaration

```
protected override void AddNextNode_Inline(Node nextWillBeAdded, int atPort)
```

## Parameters

TYPE	NAME
------	------

---

Node	nextWillBeAdded
------	-----------------

int	atPort
-----	--------

## Overrides

[Node.AddNextNode\\_Inline\(Node, int\)](#)

## DelayedClone(Dialogue)

---

This method will get called right after the dialogue gets cloned.

## Declaration

```
public void DelayedClone(Dialogue originalDialogue)
```

## Parameters

TYPE	NAME	DESCRIPTION
Dialogue	originalDialogue	This is the dialogue the cloned dialogue had cloned from.

## GetAdditionalSpeechData()

---

## Declaration

```
public AdditionalSpeechData GetAdditionalSpeechData()
```

## Returns

### TYPE

---

AdditionalSpeechData

## GetClassName()

---

Use if you have a special USS class for this node. If you don't have any, return null.

## Declaration

```
public override string GetClassName()
```

## Returns

### TYPE DESCRIPTION

---

string Returns the USS class name of this node type as a string.

## Overrides

[Node.GetClassName\(\)](#)

## GetNextNodes\_Inline(ref List<(int portIndex, Node node)>)

---

Use to describe the editor which nodes are the next nodes of this one in the chain by modifying the list.

## Declaration

```
protected override void GetNextNodes_Inline(ref List<(int portIndex, Node node)> result)
```

## Parameters

TYPE	NAME
List<(int portIndex, Node node)>	result

## Overrides

```
Node.GetNextNodes_Inline(ref List<(int portIndex, Node node)>)
```

# GetOptions()

---

## Declaration

```
public List<Option> GetOptions()
```

## Returns

TYPE
List<Option>

# GetSpeech()

---

## Declaration

```
public string GetSpeech()
```

## Returns

TYPE
string

# GetTitle()

---

Use to set the title of this node type in the graph view.

## Declaration

```
public override string GetTitle()
```

## Returns

TYPE	DESCRIPTION
------	-------------

string	The title as a string.
--------	------------------------

## Overrides

[Node.GetTitle\(\)](#)

# Pass\_Inline(params object[])

---

Use to write what happens when the dialogue passes this node.

## Declaration

```
protected override void Pass_Inline(params object[] passData)
```

## Parameters

TYPE	NAME
------	------

object[]	passData
----------	----------

## Overrides

[Node.Pass\\_Inline\(params object\[\]\)](#)

# Reach\_Inline()

---

Use to write what happens when the dialogue reaches this node.

## Declaration

```
protected override void Reach_Inline()
```

## Overrides

[Node.Reach\\_Inline\(\)](#)

# RemoveNextNode\_Inline(int)

---

Use to write the functionality of removing the next node of this one.

## Declaration

```
protected override void RemoveNextNode_Inline(int atPort)
```

## Parameters

TYPE	NAME
------	------

int	atPort
-----	--------

## Overrides

[Node.RemoveNextNode\\_Inline\(int\)](#)

# Traverse(Action<Node>)

---

Use to traverse any action on a node chain. Nodes not connected directly won't transmit the action to another.

## Declaration

```
public override void Traverse(Action<Node> action)
```

## Parameters

TYPE	NAME
------	------

Action<Node>	action
--------------	--------

## Overrides

[Node.Traverse\(Action<Node>\)](#)

# Implements

[IContainSpeech](#)



# Class GotoNode

Node which teleports the flow to a specific [DialoguePartNode](#).

## Inheritance

- ↳ [object](#)
- ↳ [Object](#)
- ↳ [ScriptableObject](#)
- ↳ [Node](#)
- ↳ [GotoNode](#)

## Inherited Members

- [Node.Guid](#)
- [Node.Position](#)
- [Node.MasterDialogue](#)
- [Node.Blackboard](#)
- [Node.State](#)
- [Node.ExitDialogAfterwards](#)
- [Node.OnSetState](#)
- [Node.OnRemove](#)
- [Node.OnValidation](#)
- [Node.OnReach](#)
- [Node.OnPass](#)
- [Node.PersonIndex](#)
- [Node.Person](#)
- [Node.DisplayState](#)
- [Node.ShowInMinimap](#)
- [Node.PersonDependent](#)
- [Node.AddNextNode\(Node, int\)](#)
- [Node.RemoveNextNode\(int\)](#)
- [Node.GetNextNodes\(\)](#)
- [Node.Pass\(params object\[\]\)](#)
- [Node.Reach\(\)](#)
- [Node.OnRemoval\(\)](#)
- [Node.GetInputPortNameForCreation\(\)](#)
- [Node.SetState\(Node.NodeState\)](#)
- [Node.Clone\(\)](#)
- [Node.Traverse\(Action<Node>\)](#)
- [ScriptableObject.SetDirty\(\)](#)
- [ScriptableObject.CreateInstance\(string\)](#)
- [ScriptableObject.CreateInstance\(Type\)](#)
- [ScriptableObject.CreateInstance<T>\(\)](#)

## Syntax

```
public sealed class GotoNode : Node
```

## Fields

### TargetDialoguePartName

---

#### Declaration

```
public string TargetDialoguePartName
```

#### Field Value

TYPE

---

string

## Methods

### AddNextNode\_Inline(Node, int)

---

Use to write the functionality of connecting a node to any port of this node.

#### Declaration

```
protected override void AddNextNode_Inline(Node nextWillBeAdded, int atPort)
```

#### Parameters

TYPE NAME

---

Node nextWillBeAdded

int atPort

#### Overrides

## GetClassName()

---

Use if you have a special USS class for this node. If you don't have any, return null.

### Declaration

```
public override string GetClassName()
```

### Returns

TYPE	DESCRIPTION
------	-------------

---

string	Returns the USS class name of this node type as a string.
--------	---

### Overrides

[Node.GetClassName\(\)](#)

## GetNextNodes\_Inline(ref List<(int portIndex, Node node)>)

---

Use to describe the editor which nodes are the next nodes of this one in the chain by modifying the list.

### Declaration

```
protected override void GetNextNodes_Inline(ref List<(int portIndex, Node node)> result)
```

### Parameters

TYPE	NAME
List<(int portIndex, Node node)>	result

### Overrides

[Node.GetNextNodes\\_Inline\(ref List<\(int portIndex, Node node\)>\)](#)

## GetOutputPortNamesForCreation()

---

Use to describe the dialogue editor how many output ports this node has and what are their names.

### Declaration

```
public override List<string> GetOutputPortNamesForCreation()
```

## Returns

TYPE	DESCRIPTION
List<string>	Returns the port names as a list of strings. Return an empty list if you want no output ports.

## Overrides

[Node.GetOutputPortNamesForCreation\(\)](#)

## GetTitle()

---

Use to set the title of this node type in the graph view.

## Declaration

```
public override string GetTitle()
```

## Returns

TYPE	DESCRIPTION
string	The title as a string.

## Overrides

[Node.GetTitle\(\)](#)

## Pass\_Inline(params object[])

---

Use to write what happens when the dialogue passes this node.

## Declaration

```
protected override void Pass_Inline(params object[] passData)
```

## Parameters

TYPE	NAME
object[]	passData

## Overrides

## Reach\_Inline()

---

Use to write what happens when the dialogue reaches this node.

### Declaration

```
protected override void Reach_Inline()
```

### Overrides

[Node.Reach\\_Inline\(\)](#)

## RemoveNextNode\_Inline(int)

---

Use to write the functionality of removing the next node of this one.

### Declaration

```
protected override void RemoveNextNode_Inline(int atPort)
```

### Parameters

TYPE	NAME
------	------

int	atPort
-----	--------

### Overrides

[Node.RemoveNextNode\\_Inline\(int\)](#)



# Interface IContainSpeech

Interface to use if any of your dialogue elements has a speech, has options or has [AdditionalSpeechData](#).

Namespace: [com.absence.dialoguesystem.internals](#)

Assembly: Assembly-CSharp-firstpass.dll

## Syntax

```
public interface IContainSpeech
```

## Methods

### GetAdditionalSpeechData()

---

#### Declaration

```
AdditionalSpeechData GetAdditionalSpeechData()
```

#### Returns

TYPE

---

[AdditionalSpeechData](#)

### GetOptions()

---

#### Declaration

```
List<Option> GetOptions()
```

#### Returns

TYPE

---

List<Option>

## GetSpeech()

---

### Declaration

```
string GetSpeech()
```

### Returns

TYPE

---

string



# Interface IContainVariableManipulators

Any node subtype with this interface implemented will refresh its `com.absence.variablesystem.VariableComparer`s and `com.absence.variablesystem.VariableSetter`s to have the correct reference to the `Bank` of the current `Dialogue` everytime the editor window refreshes.

Namespace: `com.absence.dialoguesystem.internals`

Assembly: Assembly-CSharp-firstpass.dll

## Syntax

```
public interface IContainVariableManipulators
```

## Methods

### GetComparers()

A list of comparers which you want to restrict in terms of `com.absence.variablesystem.VariableBank` selection

#### Declaration

```
List<FixedVariableComparer> GetComparers()
```

#### Returns

##### TYPE

---

```
List<FixedVariableComparer>
```

### GetSetters()

A list of comparers which you want to restrict in terms of `com.absence.variablesystem.VariableBank` selection

#### Declaration

```
List<FixedVariableSetter> GetSetters()
```

## Returns

### TYPE

---

[List<FixedVariableSetter>](#)



# Interface IPerformEditorRefresh

Any node subtype with this interface implemented will get a callback when you change any value in the inspector.

Namespace: [com.absence.dialoguesystem.internals](#)

Assembly: Assembly-CSharp-firstpass.dll

## Syntax

```
public interface IPerformEditorRefresh
```

## Methods

### PerformEditorRefresh()

---

Use to declare what to do when any value gets changed in the inspector/

#### Declaration

```
void PerformEditorRefresh()
```



# Class Node

This is the base abstract class to derive from for any new node subtypes.

## Inheritance

- ↳ [object](#)
- ↳ [Object](#)
- ↳ [ScriptableObject](#)
- ↳ [Node](#)
  - ↳ [ActionNode](#)
  - ↳ [ConditionNode](#)
  - ↳ [DecisionSpeechNode](#)
  - ↳ [DialoguePartNode](#)
  - ↳ [FastSpeechNode](#)
  - ↳ [GotoNode](#)
  - ↳ [RootNode](#)
  - ↳ [StickyNoteNode](#)
  - ↳ [TitleNode](#)

## Inherited Members

[ScriptableObject.SetDirty\(\)](#)

[ScriptableObject.CreateInstance\(string\)](#)

[ScriptableObject.CreateInstance\(Type\)](#)

[ScriptableObject.CreateInstance<T>\(\)](#)

Namespace: [com.absence.dialoguesystem.internals](#)

Assembly: Assembly-CSharp-firstpass.dll

## Syntax

```
public abstract class Node : ScriptableObject
```

## Fields

## Blackboard

```
[HideInInspector]  
public Blackboard Blackboard
```

## Field Value

### TYPE

---

Blackboard

## ExitDialogAfterwards

---

### Declaration

```
[Tooltip("Toggling this on will make the dialogue exit right after this node getting passed.")]  
public bool ExitDialogAfterwards
```

## Field Value

### TYPE

---

bool

## Guid

---

### Declaration

```
[HideInInspector]  
public string Guid
```

## Field Value

### TYPE

---

string

## MasterDialogue

---

### Declaration

```
[ Readonly ]  
public Dialogue MasterDialogue
```

## Field Value

### TYPE

---

Dialogue

## PersonIndex

---

Index of the person this node depends on (if it is `PersonDependent` ) on the person list of the `MasterDialogue` .

## Declaration

```
[ HideInInspector ]  
public int PersonIndex
```

## Field Value

### TYPE

---

int

## Position

---

## Declaration

```
[ HideInInspector ]  
public Vector2 Position
```

## Field Value

### TYPE

---

Vector2

## State

---

## Declaration

```
[HideInInspector]  
public Node.NodeState State
```

## Field Value

### TYPE

---

Node.NodeState

# Properties

## DisplayState

---

Will this node display it's state in editor on the flow.

### Declaration

```
public virtual bool DisplayState { get; }
```

## Property Value

### TYPE

---

bool

# Person

---

Property which returns the person with the index of `PersonIndex` from the person list.

### Declaration

```
[HideInInspector]  
public Person Person { get; }
```

## Property Value

### TYPE

---

Person

# PersonDependent

---

Is this node person dependent.

## Declaration

```
public virtual bool PersonDependent { get; }
```

## Property Value

### TYPE

---

bool

# ShowInMinimap

---

Will this node be visible on the minimap.

## Declaration

```
public virtual bool ShowInMinimap { get; }
```

## Property Value

### TYPE

---

bool

# Methods

## AddNextNode(Node, int)

---

Use when you connect a new node to a right-side port of this node.

## Declaration

```
public void AddNextNode(Node nextWillBeAdded, int atPort)
```

## Parameters

TYPE	NAME	DESCRIPTION
Node	nextWillBeAdded	The reference value of the node connected.
int	atPort	The port which hold the connection.

## AddNextNode\_Inline(Node, int)

---

Use to write the functionality of connecting a node to any port of this node.

### Declaration

```
protected abstract void AddNextNode_Inline(Node nextWillBeAdded, int atPort)
```

### Parameters

TYPE	NAME
Node	nextWillBeAdded
int	atPort

## Clone()

---

Use to clone this node.

**CAUTION!** It works as a traverse function. If you clone any node, it will automatically clone any node connected to it (forward-only). But the `GotoNode` won't clone the `DialoguePartNode` referenced to it. Simply because they are not connected directly.

### Declaration

```
public virtual Node Clone()
```

### Returns

TYPE
Node

## GetClassName()

---

Use if you have a special USS class for this node. If you don't have any, return null.

### Declaration

```
public abstract string GetClassName()
```

## Returns

TYPE	DESCRIPTION
------	-------------

---

string	Returns the USS class name of this node type as a string.
--------	---

## GetInputPortNameForCreation()

---

Use to describe the name of the input port of this node.

## Declaration

```
public virtual string GetInputPortNameForCreation()
```

## Returns

TYPE	DESCRIPTION
------	-------------

---

string	Returns the name as a string. Return null if you don't want any input ports.
--------	--

## GetNextNodes()

---

Use to get all of the nodes which are **directly** connected to this node (**only the right-side ones**).

## Declaration

```
public List<(int portIndex, Node node)> GetNextNodes()
```

## Returns

TYPE
------

---

List<(int portIndex, Node node)>
----------------------------------

## GetNextNodes\_Inline(ref List<(int portIndex, Node node)>)

---

Use to describe the editor which nodes are the next nodes of this one in the chain by modifying the list.

## Declaration

```
protected abstract void GetNextNodes_Inline(ref List<(int portIndex, Node node)> result)
```

## Parameters

TYPE	NAME
List<(int portIndex, Node node)>	result

## GetOutputPortNamesForCreation()

---

Use to describe the dialogue editor how many output ports this node has and what are their names.

### Declaration

```
public virtual List<string> GetOutputPortNamesForCreation()
```

### Returns

TYPE	DESCRIPTION
List<string>	Returns the port names as a list of strings. Return an empty list if you want no output ports.

## GetTitle()

---

Use to set the title of this node type in the graph view.

### Declaration

```
public abstract string GetTitle()
```

### Returns

TYPE	DESCRIPTION
string	The title as a string.

## OnRemoval()

---

### Declaration

```
public void OnRemoval()
```

## OnValidate()

---

### Declaration

```
protected virtual void OnValidate()
```

## Pass(params object[])

---

### Declaration

```
public void Pass(params object[] passData)
```

### Parameters

TYPE	NAME
object[]	passData

## Pass\_Inline(params object[])

---

Use to write what happens when the dialogue passes this node.

### Declaration

```
protected abstract void Pass_Inline(params object[] passData)
```

### Parameters

TYPE	NAME
object[]	passData

## Reach()

---

### Declaration

```
public void Reach()
```

## Reach\_Inline()

---

Use to write what happens when the dialogue reaches this node.

### Declaration

```
protected abstract void Reach_Inline()
```

## RemoveNextNode(int)

---

Use when you disconnect a node from a right-side port of this node.

### Declaration

```
public void RemoveNextNode(int atPort)
```

### Parameters

TYPE	NAME	DESCRIPTION
------	------	-------------

---

int	atPort	The port which handled the disconnection event.
-----	--------	---

## RemoveNextNode\_Inline(int)

---

Use to write the functionality of removing the next node of this one.

### Declaration

```
protected abstract void RemoveNextNode_Inline(int atPort)
```

### Parameters

TYPE	NAME
------	------

---

int	atPort
-----	--------

# SetState(NodeState)

---

Use to set the flow state of this node.

## Declaration

```
public virtual void SetState(Node.NodeState newState)
```

## Parameters

TYPE	NAME
Node.NodeState	newState

# Traverse(Action<Node>)

---

Use to traverse any action on a node chain. Nodes not connected directly won't transmit the action to another.

## Declaration

```
public virtual void Traverse(Action<Node> action)
```

## Parameters

TYPE	NAME
Action<Node>	action

# Events

## OnPass

---

Action which will get invoked when this node get passed on the flow.

## Declaration

```
public event Action OnPass
```

## Event Type

**TYPE**

---

[Action](#)

## OnReach

---

Action which will get invoked when this node gets reached on the flow.

### Declaration

```
public event Action OnReach
```

### Event Type

**TYPE**

---

[Action](#)

## OnRemove

---

Action which will get invoked when this node gets removed from the dialogue.

### Declaration

```
public event Action OnRemove
```

### Event Type

**TYPE**

---

[Action](#)

## OnSetState

---

Action which will get invoked when the state of this node gets changed.

### Declaration

```
public event Action<Node.NodeState> OnSetState
```

### Event Type

---

Action<Node.NodeState>

## OnValidation

---

Action which will get invoked when `OnValidate()` function gets called.

### Declaration

```
public event Action OnValidation
```

### Event Type

---

Action



# Enum Node.NodeState

Describes the node's state on the flow. While progressing in the dialogue.

Namespace: [com.absence.dialoguesystem.internals](#)

Assembly: Assembly-CSharp-firstpass.dll

## Syntax

```
public enum Node.NodeState
```

## Fields

NAME
Current
Past
Unreached



# Class Option

The type to hold references to dialogue options.

## Inheritance

↳ [object](#)

↳ [Option](#)

Namespace: [com.absence.dialoguesystem.internals](#)

Assembly: Assembly-CSharp-firstpass.dll

## Syntax

```
[Serializable]  
public class Option
```

# Fields

## AdditionalData

---

Additional speech data this option contains.

## Declaration

```
public AdditionalSpeechData AdditionalData
```

## Field Value

### TYPE

---

[AdditionalSpeechData](#)

## LeadsTo

---

The node this option leads to.

## Declaration

```
[HideInInspector]  
public Node LeadsTo
```

### Field Value

#### TYPE

---

Node

## Speech

---

Speech of this option.

### Declaration

```
[HideInInspector]  
public string Speech
```

### Field Value

#### TYPE

---

string

## Visibility

---

### Declaration

```
[SerializeField]  
[ShowIf("m_useShowIf")]  
public Option.ShowIf Visibility
```

### Field Value

#### TYPE

---

Option.ShowIf

## Properties

# UseShowIf

---

## Declaration

```
public bool UseShowIf { get; }
```

## Property Value

### TYPE

---

bool

## Methods

### Clone(VariableBank)

---

Use to get a clone of this option.

## Declaration

```
public Option Clone(VariableBank overrideBank)
```

## Parameters

TYPE	NAME
VariableBank	overrideBank

## Returns

TYPE
Option

### IsVisible()

---

Calculates the visibility of this option.

## Declaration

```
public bool IsVisible()
```

## Returns

TYPE	DESCRIPTION
------	-------------

bool	Returns true if the option is visible, returns false otherwise.
------	---



# Class Option.ShowIf

A class specifically designed for calculating an option's visibility.

## Inheritance

↳ [object](#)  
↳ [Option.ShowIf](#)

Namespace: [com.absence.dialoguesystem.internals](#)

Assembly: Assembly-CSharp-firstpass.dll

## Syntax

```
[Serializable]  
public class Option.ShowIf
```

## Fields

## Processor

---

An enum which defines what to do with multiple comparers in conclusion.

### Declaration

```
public VBProcessType Processor
```

### Field Value

#### TYPE

---

[VBProcessType](#)

## ShowIfList

---

A list of all `com.absence.variablesystem.VariableComparer`s which has a role on determining this option's visibility on display.

## Declaration

```
public List<FixedVariableComparer> ShowIfList
```

## Field Value

### TYPE

---

`List<FixedVariableComparer>`

## Methods

### Clone(VariableBank)

---

Use to clone this instance.

## Declaration

```
public Option.ShowIf Clone(VariableBank overrideBank)
```

## Parameters

TYPE	NAME
VariableBank	overrideBank

## Returns

TYPE
<code>Option.ShowIf</code>

### GetResult()

---

Use to get the composite result of all of the comparers of this instance.

## Declaration

```
public bool GetResult()
```

## Returns

### TYPE

---

bool



# Class RootNode

Node which is essential if you want to have a dialogue graph.

## Inheritance

- ↳ [object](#)
- ↳ [Object](#)
- ↳ [ScriptableObject](#)
- ↳ [Node](#)
- ↳ [RootNode](#)

## Inherited Members

- [Node.Guid](#)
- [Node.Position](#)
- [Node.MasterDialogue](#)
- [Node.Blackboard](#)
- [Node.State](#)
- [Node.ExitDialogAfterwards](#)
- [Node.OnSetState](#)
- [Node.OnRemove](#)
- [Node.OnValidation](#)
- [Node.OnReach](#)
- [Node.OnPass](#)
- [Node.PersonIndex](#)
- [Node.Person](#)
- [Node.ShowInMinimap](#)
- [Node.PersonDependent](#)
- [Node.AddNextNode\(Node, int\)](#)
- [Node.RemoveNextNode\(int\)](#)
- [Node.GetNextNodes\(\)](#)
- [Node.Pass\(params object\[\]\)](#)
- [Node.Reach\(\)](#)
- [Node.OnRemoval\(\)](#)

- [Node.SetState\(Node.NodeState\)](#)
- [Node.Clone\(\)](#)

- [ScriptableObject.SetDirty\(\)](#)
- [ScriptableObject.CreateInstance\(string\)](#)
- [ScriptableObject.CreateInstance\(Type\)](#)
- [ScriptableObject.CreateInstance<T>\(\)](#)

Namespace: [com.absence.dialoguesystem.internals](#)

Assembly: Assembly-CSharp-firstpass.dll

## Syntax

```
public sealed class RootNode : Node
```

## Fields

### Next

---

#### Declaration

```
[HideInInspector]  
public Node Next
```

#### Field Value

##### TYPE

---

Node

## Properties

### DisplayState

---

Will this node display it's state in editor on the flow.

#### Declaration

```
public override bool DisplayState { get; }
```

#### Property Value

##### TYPE

---

bool

#### Overrides

[Node.DisplayState](#)

# Methods

## AddNextNode\_Inline(Node, int)

---

Use to write the functionality of connecting a node to any port of this node.

### Declaration

```
protected override void AddNextNode_Inline(Node nextWillBeAdded, int atPort)
```

### Parameters

TYPE	NAME
------	------

Node	nextWillBeAdded
------	-----------------

int	atPort
-----	--------

### Overrides

[Node.AddNextNode\\_Inline\(Node, int\)](#)

## DelayedClone(Dialogue)

---

This method will get called right after the dialogue gets cloned.

### Declaration

```
public void DelayedClone(Dialogue originalDialogue)
```

### Parameters

TYPE	NAME	DESCRIPTION
------	------	-------------

Dialogue	originalDialogue	This is the dialogue the cloned dialogue had cloned from.
----------	------------------	---

## GetClassName()

---

Use if you have a special USS class for this node. If you don't have any, return null.

### Declaration

```
public override string GetClassName()
```

## Returns

TYPE	DESCRIPTION
------	-------------

string	Returns the USS class name of this node type as a string.
--------	---

## Overrides

[Node.GetClassName\(\)](#)

## GetInputPortNameForCreation()

---

Use to describe the name of the input port of this node.

## Declaration

```
public override string GetInputPortNameForCreation()
```

## Returns

TYPE	DESCRIPTION
------	-------------

string	Returns the name as a string. Return null if you don't want any input ports.
--------	--

## Overrides

[Node.GetInputPortNameForCreation\(\)](#)

## GetNextNodes\_Inline(ref List<(int portIndex, Node node)>)

---

Use to describe the editor which nodes are the next nodes of this one in the chain by modifying the list.

## Declaration

```
protected override void GetNextNodes_Inline(ref List<(int portIndex, Node node)> result)
```

## Parameters

TYPE	NAME
List<(int portIndex, Node node)>	result

## Overrides

## GetOutputPortNamesForCreation()

---

Use to describe the dialogue editor how many output ports this node has and what are their names.

### Declaration

```
public override List<string> GetOutputPortNamesForCreation()
```

### Returns

TYPE	DESCRIPTION
List<string>	Returns the port names as a list of strings. Return an empty list if you want no output ports.

### Overrides

[Node.GetOutputPortNamesForCreation\(\)](#)

## GetTitle()

---

Use to set the title of this node type in the graph view.

### Declaration

```
public override string GetTitle()
```

### Returns

TYPE	DESCRIPTION
string	The title as a string.

### Overrides

[Node.GetTitle\(\)](#)

## Pass\_Inline(params object[])

---

Use to write what happens when the dialogue passes this node.

### Declaration

```
protected override void Pass_Inline(params object[] passData)
```

## Parameters

TYPE	NAME
object[]	passData

## Overrides

[Node.Pass\\_Inline\(params object\[\]\)](#)

## Reach\_Inline()

---

Use to write what happens when the dialogue reaches this node.

## Declaration

```
protected override void Reach_Inline()
```

## Overrides

[Node.Reach\\_Inline\(\)](#)

## RemoveNextNode\_Inline(int)

---

Use to write the functionality of removing the next node of this one.

## Declaration

```
protected override void RemoveNextNode_Inline(int atPort)
```

## Parameters

TYPE	NAME
int	atPort

## Overrides

[Node.RemoveNextNode\\_Inline\(int\)](#)

## Traverse(Action<Node>)

---

Use to traverse any action on a node chain. Nodes not connected directly won't transmit the action to another.

## Declaration

```
public override void Traverse(Action<Node> action)
```

## Parameters

TYPE	NAME
Action<Node>	action

## Overrides

[Node.Traverse\(Action<Node>\)](#)



# Class StickyNoteNode

Node which contains a user defined string.

## Inheritance

- ↳ [object](#)
- ↳ [Object](#)
- ↳ [ScriptableObject](#)
- ↳ [Node](#)
- ↳ [StickyNoteNode](#)

## Inherited Members

- [Node.Guid](#)
- [Node.Position](#)
- [Node.MasterDialogue](#)
- [Node.Blackboard](#)
- [Node.State](#)
- [Node.ExitDialogAfterwards](#)
- [Node.OnSetState](#)
- [Node.OnRemove](#)
- [Node.OnValidation](#)
- [Node.OnReach](#)
- [Node.OnPass](#)
- [Node.PersonIndex](#)
- [Node.Person](#)
- [Node.PersonDependent](#)
- [Node.AddNextNode\(Node, int\)](#)
- [Node.RemoveNextNode\(int\)](#)
- [Node.GetNextNodes\(\)](#)
- [Node.Pass\(params object\[\]\)](#)
- [Node.Reach\(\)](#)
- [Node.OnRemoval\(\)](#)
- [Node.SetState\(Node.NodeState\)](#)
- [Node.Clone\(\)](#)
- [Node.Traverse\(Action<Node>\)](#)
- [ScriptableObject.SetDirty\(\)](#)
- [ScriptableObject.CreateInstance\(string\)](#)
- [ScriptableObject.CreateInstance\(Type\)](#)
- [ScriptableObject.CreateInstance<T>\(\)](#)

Namespace: [com.absence.dialoguesystem.internals](#)  
Assembly: Assembly-CSharp-firstpass.dll

## Syntax

```
public sealed class StickyNoteNode : Node
```

## Fields

### Speech

---

#### Declaration

```
[HideInInspector]  
public string Speech
```

#### Field Value

##### TYPE

---

string

## Properties

### DisplayState

---

Will this node display it's state in editor on the flow.

#### Declaration

```
public override bool DisplayState { get; }
```

#### Property Value

##### TYPE

---

bool

#### Overrides

[Node.DisplayState](#)

# ShowInMinimap

---

Will this node be visible on the minimap.

## Declaration

```
public override bool ShowInMinimap { get; }
```

## Property Value

### TYPE

---

bool

## Overrides

[Node.ShowInMinimap](#)

# Methods

## AddNextNode\_Inline(Node, int)

---

Use to write the functionality of connecting a node to any port of this node.

## Declaration

```
protected override void AddNextNode_Inline(Node nextWillBeAdded, int atPort)
```

## Parameters

TYPE	NAME
------	------

---

Node	nextWillBeAdded
------	-----------------

int	atPort
-----	--------

## Overrides

[Node.AddNextNode\\_Inline\(Node, int\)](#)

## GetClassName()

---

Use if you have a special USS class for this node. If you don't have any, return null.

## Declaration

```
public override string GetClassName()
```

## Returns

TYPE	DESCRIPTION
------	-------------

**string** Returns the USS class name of this node type as a string.

## Overrides

[Node.GetClassName\(\)](#)

## GetInputPortNameForCreation()

---

Use to describe the name of the input port of this node.

## Declaration

```
public override string GetInputPortNameForCreation()
```

## Returns

TYPE	DESCRIPTION
------	-------------

**string** Returns the name as a string. Return null if you don't want any input ports.

## Overrides

[Node.GetInputPortNameForCreation\(\)](#)

## GetNextNodes\_Inline(ref List<(int portIndex, Node node)>)

---

Use to describe the editor which nodes are the next nodes of this one in the chain by modifying the list.

## Declaration

```
protected override void GetNextNodes_Inline(ref List<(int portIndex, Node node)> result)
```

## Parameters

TYPE	NAME
------	------

**List<(int portIndex, Node node)>** result

## Overrides

[Node.GetNextNodes\\_Inline\(ref List<\(int portIndex, Node node\)>\)](#)

# GetOutputPortNamesForCreation()

---

Use to describe the dialogue editor how many output ports this node has and what are their names.

## Declaration

```
public override List<string> GetOutputPortNamesForCreation()
```

## Returns

TYPE	DESCRIPTION
List<string>	Returns the port names as a list of strings. Return an empty list if you want no output ports.

## Overrides

[Node.GetOutputPortNamesForCreation\(\)](#)

# GetTitle()

---

Use to set the title of this node type in the graph view.

## Declaration

```
public override string GetTitle()
```

## Returns

TYPE	DESCRIPTION
string	The title as a string.

## Overrides

[Node.GetTitle\(\)](#)

# Pass\_Inline(params object[])

---

Use to write what happens when the dialogue passes this node.

## Declaration

```
protected override void Pass_Inline(params object[] passData)
```

## Parameters

TYPE	NAME
object[]	passData

## Overrides

[Node.Pass\\_Inline\(params object\[\]\)](#)

## Reach\_Inline()

---

Use to write what happens when the dialogue reaches this node.

## Declaration

```
protected override void Reach_Inline()
```

## Overrides

[Node.Reach\\_Inline\(\)](#)

## RemoveNextNode\_Inline(int)

---

Use to write the functionality of removing the next node of this one.

## Declaration

```
protected override void RemoveNextNode_Inline(int atPort)
```

## Parameters

TYPE	NAME
int	atPort

## Overrides

[Node.RemoveNextNode\\_Inline\(int\)](#)



# Class TitleNode

Node which is simply `StickyNoteNode` but bigger.

## Inheritance

- ↳ `object`
- ↳ `Object`
- ↳ `ScriptableObject`
- ↳ `Node`
- ↳ `TitleNode`

## Inherited Members

- `Node.Guid`
- `Node.Position`
- `Node.MasterDialogue`
- `Node.Blackboard`
- `Node.State`
- `Node.ExitDialogAfterwards`
- `Node.OnSetState`
- `Node.OnRemove`
- `Node.OnValidation`
- `Node.OnReach`
- `Node.OnPass`
- `Node.PersonIndex`
- `Node.Person`
- `Node.PersonDependent`
- `Node.AddNextNode(Node, int)`
- `Node.RemoveNextNode(int)`
- `Node.GetNextNodes()`
- `Node.Pass(params object[])`
- `Node.Reach()`
- `Node.OnRemoval()`
- `Node.SetState(Node.NodeState)`
- `Node.Clone()`
- `Node.Traverse(Action<Node>)`
- `ScriptableObject.SetDirty()`
- `ScriptableObject.CreateInstance(string)`
- `ScriptableObject.CreateInstance(Type)`
- `ScriptableObject.CreateInstance<T>()`

Namespace: [com.absence.dialoguesystem.internals](#)  
Assembly: Assembly-CSharp-firstpass.dll

## Syntax

```
public sealed class TitleNode : Node
```

## Fields

### Speech

---

#### Declaration

```
[HideInInspector]  
public string Speech
```

#### Field Value

##### TYPE

---

string

## Properties

### DisplayState

---

Will this node display it's state in editor on the flow.

#### Declaration

```
public override bool DisplayState { get; }
```

#### Property Value

##### TYPE

---

bool

#### Overrides

[Node.DisplayState](#)

# ShowInMinimap

---

Will this node be visible on the minimap.

## Declaration

```
public override bool ShowInMinimap { get; }
```

## Property Value

### TYPE

---

bool

## Overrides

[Node.ShowInMinimap](#)

# Methods

## AddNextNode\_Inline(Node, int)

---

Use to write the functionality of connecting a node to any port of this node.

## Declaration

```
protected override void AddNextNode_Inline(Node nextWillBeAdded, int atPort)
```

## Parameters

TYPE	NAME
------	------

---

Node	nextWillBeAdded
------	-----------------

int	atPort
-----	--------

## Overrides

[Node.AddNextNode\\_Inline\(Node, int\)](#)

## GetClassName()

---

Use if you have a special USS class for this node. If you don't have any, return null.

## Declaration

```
public override string GetClassName()
```

## Returns

TYPE	DESCRIPTION
------	-------------

string	Returns the USS class name of this node type as a string.
--------	---

## Overrides

[Node.GetClassName\(\)](#)

## GetInputPortNameForCreation()

---

Use to describe the name of the input port of this node.

## Declaration

```
public override string GetInputPortNameForCreation()
```

## Returns

TYPE	DESCRIPTION
------	-------------

string	Returns the name as a string. Return null if you don't want any input ports.
--------	--

## Overrides

[Node.GetInputPortNameForCreation\(\)](#)

## GetNextNodes\_Inline(ref List<(int portIndex, Node node)>)

---

Use to describe the editor which nodes are the next nodes of this one in the chain by modifying the list.

## Declaration

```
protected override void GetNextNodes_Inline(ref List<(int portIndex, Node node)> result)
```

## Parameters

TYPE	NAME
------	------

List<(int portIndex, Node node)>	result
----------------------------------	--------

## Overrides

[Node.GetNextNodes\\_Inline\(ref List<\(int portIndex, Node node\)>\)](#)

# GetOutputPortNamesForCreation()

---

Use to describe the dialogue editor how many output ports this node has and what are their names.

## Declaration

```
public override List<string> GetOutputPortNamesForCreation()
```

## Returns

TYPE	DESCRIPTION
List<string>	Returns the port names as a list of strings. Return an empty list if you want no output ports.

## Overrides

[Node.GetOutputPortNamesForCreation\(\)](#)

# GetTitle()

---

Use to set the title of this node type in the graph view.

## Declaration

```
public override string GetTitle()
```

## Returns

TYPE	DESCRIPTION
string	The title as a string.

## Overrides

[Node.GetTitle\(\)](#)

# Pass\_Inline(params object[])

---

Use to write what happens when the dialogue passes this node.

## Declaration

```
protected override void Pass_Inline(params object[] passData)
```

## Parameters

TYPE	NAME
object[]	passData

## Overrides

[Node.Pass\\_Inline\(params object\[\]\)](#)

## Reach\_Inline()

---

Use to write what happens when the dialogue reaches this node.

## Declaration

```
protected override void Reach_Inline()
```

## Overrides

[Node.Reach\\_Inline\(\)](#)

## RemoveNextNode\_Inline(int)

---

Use to write the functionality of removing the next node of this one.

## Declaration

```
protected override void RemoveNextNode_Inline(int atPort)
```

## Parameters

TYPE	NAME
int	atPort

## Overrides

[Node.RemoveNextNode\\_Inline\(int\)](#)



# Enum VBProcessType

Namespace: [com.absence.dialoguesystem.internals](#)

Assembly: Assembly-CSharp-firstpass.dll

## Syntax

```
public enum VBProcessType
```

## Fields

### NAME

---

All

Any