# 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

A small component with the responsibility of using the input comes from player (uses legacy input system of unity) on the dialogue.

### 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.

# Interfaces

### IUseDialogueInScene

Any game object with a script that implements this interface attached will display it's dialogue when gets selected.

# Enums

### DialoguePlayer.PlayerState

Shows what state the dialogue player is in.

# 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

```
[HelpURL("https://b1lodhand.github.io/absent-dialogues/api/com.absence.dialoguesystem.Dialogue.html")]
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 `DialoguePartNode` 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 |

# Events

## OnValidateAction

Action which will get invoked if `OnValidate()` gets called in the editor.

### Declaration

```
public event Action OnValidateAction
```

### 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.Order
DialogueExtensionBase.OnBeforeSpeech(ref Person, ref string, ref List<Option>)
DialogueExtensionBase.OnAfterCloning()
DialogueExtensionBase.OnDialogueUpdate()
DialogueExtensionBase.FindInstance()

**Namespace:** com.absence.dialoguesystem

**Assembly:** Assembly-CSharp-firstpass.dll

### Syntax

```
[RequireComponent(typeof(DialogueInstance))]
[AddComponentMenu("absencee_/absent-dialogues/Dialogue Instance Extensions/Dialogue Animations Player")
[DisallowMultipleComponent]
[HelpURL("https://b1lodhand.github.io/absent-dialogues/api/com.absence.dialoguesystem.DialogueAnimatior
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

```
public 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/UI/Dialogue Displayer")]
[HelpURL("https://b1lodhand.github.io/absent-dialogues/api/com.absence.dialoguesystem.DialogueDisplayer
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 sinleton. If it is occuppied by any other scripts about dialogues, you can't occupy.

### Declaration

```
public bool Occupy()
```

### Returns

| TYPE | DESCRIPTION |
| --- | --- |
| bool | Returns false if the displayer is already occupied. Returns true otherwise. |

# Release()

Removes the occupience of the displayer. 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
Demo_GUI

**Namespace:** com.absence.dialoguesystem

**Assembly:** Assembly-CSharp-firstpass.dll

### Syntax

```
[RequireComponent(typeof(DialogueInstance))]
[HelpURL("https://b1lodhand.github.io/absent-dialogues/api/com.absence.dialoguesystem.DialogueExtensior
public abstract class DialogueExtensionBase : MonoBehaviour
```

### Remarks

Execution order goes like:

```
OnHandleAdditionalData(...);
OnBeforeSpeech(...);
```

## Fields

### m_instance

#### Declaration

```
[SerializeField]
[Readonly]
[Tooltip("Dialogue instance component attached to the current gameobject.")]
protected DialogueInstance m_instance
```

### Field Value

| TYPE |
| --- |
| DialogueInstance |

# Properties

## Order

### Declaration

```
public sbyte Order { get; }
```

### Property Value

| TYPE |
| --- |
| sbyte |

# Methods

## FindInstance()

### Declaration

```
public void FindInstance()
```

## OnAfterCloning()

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

### Declaration

```
public 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

```
public 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. |

## OnDialogueUpdate()

Use to define what to do on each frame when the target instance is `InDialogue`

### Declaration

```
public virtual void OnDialogueUpdate()
```

## OnHandleAdditionalData(AdditionalSpeechData)

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

### Declaration

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

### Parameters

| TYPE | NAME |
| --- | --- |
| AdditionalSpeechData | `data` |

# Class DialogueInputHandler_Legacy

A small component with the responsibility of using the input comes from player (uses legacy input system of unity) on the dialogue.

### Inheritance

↳ object
  ↳ Object
    ↳ Component
      ↳ Behaviour
        ↳ MonoBehaviour
          ↳ DialogueExtensionBase
DialogueInputHandler_Legacy

### Inherited Members

DialogueExtensionBase.m_instance
DialogueExtensionBase.Order
DialogueExtensionBase.OnHandleAdditionalData(AdditionalSpeechData)
DialogueExtensionBase.OnBeforeSpeech(ref Person, ref string, ref List<Option>)
DialogueExtensionBase.OnAfterCloning()
DialogueExtensionBase.FindInstance()

**Namespace:** com.absence.dialoguesystem

**Assembly:** Assembly-CSharp-firstpass.dll

### Syntax

```
[RequireComponent(typeof(DialogueInstance))]
[AddComponentMenu("absencee_/absent-dialogues/Dialogue Instance Extensions/Dialogue Input Handler (Lega
[DisallowMultipleComponent]
[HelpURL("https://b1lodhand.github.io/absent-dialogues/api/com.absence.dialoguesystem.DialogueInputHanc
public class DialogueInputHandler_Legacy : DialogueExtensionBase
```

## Methods

### OnDialogueUpdate()

Use to define what to do on each frame when the target instance is `InDialogue`

## Declaration

```
public override void OnDialogueUpdate()
```

## Overrides

DialogueExtensionBase.OnDialogueUpdate()

# Class DialogueInstance

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

### Inheritance

↳ object
  ↳ Object
    ↳ Component
      ↳ Behaviour
        ↳ MonoBehaviour
          ↳ DialogueInstance

### Implements

IUseDialogueInScene

**Namespace:** com.absence.dialoguesystem

**Assembly:** Assembly-CSharp-firstpass.dll

### Syntax

```
[AddComponentMenu("absencee_/absent-dialogues/Dialogue Instance")]
[DisallowMultipleComponent]
[HelpURL("https://b1lodhand.github.io/absent-dialogues/api/com.absence.dialoguesystem.DialogueInstance.
public class DialogueInstance : MonoBehaviour, IUseDialogueInScene
```

# Properties

## ClonedDialogue

The dialogue cloned and in-use.

### Declaration

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

### Property Value

| TYPE |
| --- |
| Dialogue |

## InDialogue

Use to check if this instance is in progress right now.

### Declaration

```
public bool InDialogue { get; }
```

### Property Value

| TYPE |
| --- |
| bool |

## Player

`DialoguePlayer` of this instance.

### Declaration

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

### Property Value

| TYPE |
| --- |
| DialoguePlayer |

## ReferencedDialogue

The original dialogue provided for the script (not the cloned one).

### 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

| NAME |
| --- |
| T |

## EnterDialogue()

Use to enter dialogue.

### Declaration

```
public bool EnterDialogue()
```

### Returns

| TYPE | DESCRIPTION |
| --- | --- |
| bool | **False** if the `DialogueDisplayer` is already occupied by any other script. Returns **true** 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

| TYPE |
| --- |
| Action |

## OnBeforeSpeech

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

### Declaration

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

### Event Type

| TYPE |
| --- |
| DialogueInstance.SpeechEventHandler |

## OnHandleAdditionalData

The Action which will get invoked when `HandleAdditionalData()` gets called.

### Declaration

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

### Event Type

| TYPE |
| --- |
| Action<AdditionalSpeechData> |

## Implements

IUseDialogueInScene

# Delegate DialogueInstance.SpeechEventHandler

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 Lis
```

## 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/UI/Option Text")]
[HelpURL("https://b1lodhand.github.io/absent-dialogues/api/com.absence.dialoguesystem.DialogueOptionTex
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]
[HelpURL("https://b1lodhand.github.io/absent-dialogues/api/com.absence.dialoguesystem.DialoguePlayer.ht
public class DialoguePlayer
```

## Constructors

### DialoguePlayer(Dialogue)

Use to create a new `DialoguePlayer`.

#### Declaration

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

#### Parameters

| TYPE | NAME | DESCRIPTION |
|---|---|---|
| Dialogue | `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 wih 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. `DecisionSpeechNode` 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()
```

# Events

## 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> |

Edit this page

# 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 | DESCRIPTION |
| --- | --- |
| NoSpeech | The player is not displaying any dialogue or the current node is not `IContainSpeech`. |
| WaitingForOption | The player is displaying a speech which has some options and waiting for player to pick an option. |
| WaitingForSkip | The player is displaying a speech without any options and waiting for the player to skip it. |
| WillExit | The player's last node was a `ExitDialogAfterwards`. |

# 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.Order
DialogueExtensionBase.OnBeforeSpeech(ref Person, ref string, ref List<Option>)
DialogueExtensionBase.OnAfterCloning()
DialogueExtensionBase.OnDialogueUpdate()
DialogueExtensionBase.FindInstance()

**Namespace: com.absence.dialoguesystem**

**Assembly: Assembly-CSharp-firstpass.dll**

### Syntax

```
[RequireComponent(typeof(DialogueInstance))]
[AddComponentMenu("absencee_/absent-dialogues/Dialogue Instance Extensions/Dialogue Sounds Player")]
[DisallowMultipleComponent]
[HelpURL("https://b1lodhand.github.io/absent-dialogues/api/com.absence.dialoguesystem.DialogueSoundsPla
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

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

### Parameters

| TYPE | NAME |
| --- | --- |
| AdditionalSpeechData | data |

### Overrides

DialogueExtensionBase.OnHandleAdditionalData(AdditionalSpeechData)

# Interface IUseDialogueInScene

Any game object with a script that implements this interface attached will display it's dialogue when gets selected.

**Namespace:** com.absence.dialoguesystem

**Assembly:** Assembly-CSharp-firstpass.dll

### Syntax

```
public interface IUseDialogueInScene
```

## Properties

## ClonedDialogue

The dialogue cloned and in-use.

### Declaration

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

### Property Value

| TYPE |
| --- |
| Dialogue |

## ReferencedDialogue

The original dialogue provided for the script (not the cloned one).

### Declaration

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

Property Value

| TYPE |
| --- |
| Dialogue |

# Namespace com.absence.dialoguesystem.editor

## Classes

### BlackboardView

A visual element subtype which is responsible for displaying a `Blackboard`.

### BlackboardView.UxmlFactory

### DialogueCreationHandler

A script responsible for handling the creation of a dialogue.

### DialogueEditorWindow

The dialogue editor window responsible for letting you open, edit and save a dialogue.

### DialogueGraphView

The graph view responsible for rendering a dialogue's graph elements.

### DialogueGraphView.UxmlFactory

### InspectorView

A visual element subtype which is responsible for rendering a node's inspector properties when selected.

### InspectorView.UxmlFactory

### NodeView

The view class responsible for rendering a node's data in the graph.

### RuntimeSelectionHandler

It handles the selection events of `IUseDialogueInScene` game objects.

### SplitView

### SplitView.UxmlFactory

# Class BlackboardView

A visual element subtype which is responsible for displaying a `Blackboard`.

### Inheritance

↳ object
  ↳ CallbackEventHandler
    ↳ Focusable
      ↳ VisualElement
        ↳ BlackboardView

### Implements

IEventHandler
IResolvedStyle
ITransform
ITransitionAnimations
IExperimentalFeatures
IVisualElementScheduler

**Namespace:** com.absence.dialoguesystem.editor

**Assembly:** Assembly-CSharp-Editor-firstpass.dll

### Syntax

```
[HelpURL("https://b1lodhand.github.io/absent-dialogues/api/com.absence.dialoguesystem.editor.Blackboarc
public class BlackboardView : VisualElement, IEventHandler, IResolvedStyle, ITransform, ITransitionAnim
```

## Constructors

### BlackboardView()

### Declaration

```
public BlackboardView()
```

# Implements

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

Edit this page

# Class BlackboardView.UxmlFactory

## Inheritance

↳ object
  ↳ BaseUxmlFactory<BlackboardView, VisualElement.UxmlTraits>
    ↳ UxmlFactory<BlackboardView, VisualElement.UxmlTraits>
      ↳ BlackboardView.UxmlFactory

## Implements

IUxmlFactory
IBaseUxmlFactory

**Namespace: com.absence.dialoguesystem.editor**

**Assembly: Assembly-CSharp-Editor-firstpass.dll**

## Syntax

```
public class BlackboardView.UxmlFactory : UxmlFactory<BlackboardView, VisualElement.UxmlTraits>, IUxmlF
```

# Implements

UnityEngine.UIElements.IUxmlFactory
UnityEngine.UIElements.IBaseUxmlFactory

Edit this page

# Class DialogueCreationHandler

A script responsible for handling the creation of a dialogue.

## Inheritance

↳ object
  ↳ DialogueCreationHandler

**Namespace:** com.absence.dialoguesystem.editor

**Assembly:** Assembly-CSharp-Editor-firstpass.dll

## Syntax

```
[HelpURL("https://b1lodhand.github.io/absent-dialogues/api/com.absence.dialoguesystem.editor.DialogueCr
public static class DialogueCreationHandler
```

# Class DialogueEditorWindow

The dialogue editor window responsible for letting you open, edit and save a dialogue.

### 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

```
[HelpURL("https://b1lodhand.github.io/absent-dialogues/api/com.absence.dialoguesystem.editor.DialogueEd
public sealed class DialogueEditorWindow : EditorWindow
```

## Methods

### CreateGUI()

#### Declaration

```
public void CreateGUI()
```

### FrameToNode(Node)

Teleports the view to the target node and selects it.

### Declaration

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

### Parameters

| TYPE | NAME | DESCRIPTION |
| --- | --- | --- |
| Node | node | Target node. |

## LoadLastDialogue()

Use to load the last dialogue displayed in the editor.

### Declaration

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

## OnOpenAsset(int, int)

The method that handles the asset selection events.

### Declaration

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

### Parameters

| TYPE | NAME |
| --- | --- |
| int | instanceId |
| int | line |

### Returns

| TYPE |
| --- |
| bool |

# OpenWindow()

Use to open the dialogue editor window.

### Declaration

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

# PopulateDialogueView(Dialogue)

Use to display a dialogue in the graph.

### Declaration

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

### Parameters

| TYPE | NAME | DESCRIPTION |
| --- | --- | --- |
| Dialogue | `dialogue` | Target dialogue. |

### Returns

| TYPE |
| --- |
| bool |

# SaveLastDialogue()

Use to save the dialogue displayed currently in the editor.

### Declaration

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

# SelectNode(Node)

Selects the target node.

### Declaration

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

### Parameters

| TYPE | NAME | DESCRIPTION |
| --- | --- | --- |
| Node | node | Target node. |

# Events

## OnGUIDelayCall

Gets invoked when `CreateGUI()` gets called. **Clears itself everytime it gets invoked.**

### Declaration

```
public static event Action OnGUIDelayCall
```

### Event Type

| TYPE |
| --- |
| Action |

# Class DialogueGraphView

The graph view responsible for rendering a dialogue's graph elements.

### 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

```
[HelpURL("https://b1lodhand.github.io/absent-dialogues/api/com.absence.dialoguesystem.editor.DialogueGr
public sealed class DialogueGraphView : GraphView, IEventHandler, IResolvedStyle, ITransform, ITransiti
```

## Constructors

### DialogueGraphView()

Default constructor.

### Declaration

```
public DialogueGraphView()
```

# Methods

## BuildContextualMenu(ContextualMenuPopulateEvent)

Add menu items to the contextual menu.

### Declaration

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

### Parameters

| TYPE | NAME | DESCRIPTION |
| --- | --- | --- |
| ContextualMenuPopulateEvent | `evt` | The event holding the menu to populate. |

### Overrides

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

## FindNodeView(Node)

Use to find the view of a node.

### Declaration

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

### Parameters

| TYPE | NAME | DESCRIPTION |
| --- | --- | --- |
| Node | `node` | Target node. |

### Returns

| TYPE | DESCRIPTION |
| --- | --- |
| NodeView | Returns the view of the target node. |

## 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()

Use to refresh the current graph view.

### Declaration

```
public void Refresh()
```

# Events

## OnBeforeNodeDeleted

### Declaration

```
public event Action<Node> OnBeforeNodeDeleted
```

### Event Type

| TYPE |
| --- |
| Action<Node> |

## OnNodeCreated

### Declaration

```
public event Action<Node> OnNodeCreated
```

### Event Type

| TYPE |
| --- |
| Action<Node> |

## OnNodeSelected

Gets invoked when a node gets selected.

### Declaration

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

### Event Type

| TYPE |
| --- |
| Action<NodeView> |

## OnPopulateView

Gets invoked when a dialogue gets displayed.

### 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

Edit this page

# 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

A visual element subtype which is responsible for rendering a node's inspector properties when selected.

### 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

```
[HelpURL("https://b1lodhand.github.io/absent-dialogues/api/com.absence.dialoguesystem.editor.InspectorV
public class InspectorView : VisualElement, IEventHandler, IResolvedStyle, ITransform, ITransitionAnima
```

## Constructors

### InspectorView()

Default constructor.

### Declaration

```
public InspectorView()
```

# Events

## OnNodeValidation

### Declaration

```
public event Action OnNodeValidation
```

### Event Type

| TYPE |
| --- |
| Action |

# Implements

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

Edit this page

# 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

Edit this page

# Class NodeView

The view class responsible for rendering a node's data in the graph.

### 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

```
[HelpURL("https://b1lodhand.github.io/absent-dialogues/api/com.absence.dialoguesystem.editor.NodeView.h
public class NodeView : Node, IEventHandler, IResolvedStyle, ITransform, ITransitionAnimations, IExperi
```

## Constructors

### NodeView(Node)

Use to construct a node view from a node.

### Declaration

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

### Parameters

| TYPE | NAME | DESCRIPTION |
| --- | --- | --- |
| Node | node | Target node. |

# Fields

## Input

The left-hand side port.

### Declaration

```
public Port Input
```

### Field Value

| TYPE |
| --- |
| Port |

## K_PERSONDEPENDENT_CLASSNAME

The USS class name for person dependent nodes.

### Declaration

```
public static string K_PERSONDEPENDENT_CLASSNAME
```

### Field Value

| TYPE |
| --- |
| string |

## Node

The node this view displays.

### Declaration

```
public Node Node
```

### Field Value

| TYPE |
| --- |
| Node |

## OnNodeSelected

Action gets invoked when this node gets selected or unselected.

### Declaration

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

### Field Value

| TYPE |
| --- |
| Action<NodeView> |

## Outputs

A list of right-hand side ports.

### Declaration

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

### Field Value

| TYPE |
| --- |
| List<Port> |

# Properties

## Master

The graph we're in.

### 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

It handles the selection events of `IUseDialogueInScene` game objects.

## Inheritance

↳ object
  ↳ RuntimeSelectionHandler

**Namespace:** com.absence.dialoguesystem.editor

**Assembly:** Assembly-CSharp-Editor-firstpass.dll

## Syntax

```
[InitializeOnLoad]
[HelpURL("https://b1lodhand.github.io/absent-dialogues/api/com.absence.dialoguesystem.editor.RuntimeSel
public static class RuntimeSelectionHandler
```

Edit this page

# Class SplitView

### Inheritance

↳ object
  ↳ CallbackEventHandler
    ↳ Focusable
      ↳ VisualElement
        ↳ TwoPaneSplitView
          ↳ SplitView

### Implements

IEventHandler
IResolvedStyle
ITransform
ITransitionAnimations
IExperimentalFeatures
IVisualElementScheduler

### Inherited Members

TwoPaneSplitView.CollapseChild(int)
TwoPaneSplitView.UnCollapse()
TwoPaneSplitView.fixedPane
TwoPaneSplitView.flexedPane
TwoPaneSplitView.fixedPaneIndex
TwoPaneSplitView.fixedPaneInitialDimension
TwoPaneSplitView.orientation
TwoPaneSplitView.contentContainer

**Namespace: com.absence.dialoguesystem.editor**

**Assembly: Assembly-CSharp-Editor-firstpass.dll**

### Syntax

```
public class SplitView : TwoPaneSplitView, IEventHandler, IResolvedStyle, ITransform, ITransitionAnimat
```

# Implements

UnityEngine.UIElements.IEventHandler
UnityEngine.UIElements.IResolvedStyle

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

# Class SplitView.UxmlFactory

## Inheritance

↳ object
  ↳ BaseUxmlFactory<SplitView, TwoPaneSplitView.UxmlTraits>
    ↳ UxmlFactory<SplitView, TwoPaneSplitView.UxmlTraits>
      ↳ SplitView.UxmlFactory

## Implements

IUxmlFactory
IBaseUxmlFactory

**Namespace: com.absence.dialoguesystem.editor**

**Assembly: Assembly-CSharp-Editor-firstpass.dll**

## Syntax

```
public class SplitView.UxmlFactory : UxmlFactory<SplitView, TwoPaneSplitView.UxmlTraits>, IUxmlFactory,
```

# Implements

UnityEngine.UIElements.IUxmlFactory
UnityEngine.UIElements.IBaseUxmlFactory

# Namespace com.absence.dialoguesystem.examples

## Classes

### Demo_GUI

A simple component to show how to use the system in the recommended way.

# Class Demo_GUI

A simple component to show how to use the system in the recommended way.

### Inheritance

↳ object
  ↳ Object
    ↳ Component
      ↳ Behaviour
        ↳ MonoBehaviour
          ↳ DialogueExtensionBase
Demo_GUI

### Inherited Members

DialogueExtensionBase.m_instance
DialogueExtensionBase.Order
DialogueExtensionBase.OnHandleAdditionalData(AdditionalSpeechData)
DialogueExtensionBase.OnBeforeSpeech(ref Person, ref string, ref List<Option>)
DialogueExtensionBase.OnDialogueUpdate()
DialogueExtensionBase.FindInstance()

**Namespace: com.absence.dialoguesystem.examples**

**Assembly: Assembly-CSharp-firstpass.dll**

### Syntax

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

# Methods

## OnAfterCloning()

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

### Declaration

```
public override void OnAfterCloning()
```

### Overrides

DialogueExtensionBase.OnAfterCloning()

# Namespace com.absence.dialoguesystem.internals

## Classes

### ActionNode

Node which invokes some actions on the flow.

### AdditionalSpeechData

Holds some extra data which you can use on the flow.

### 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 seperate 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.

### NodeVariableComparer

The comparer specifically designed for working with dialogue nodes.

### NodeVariableSetter

The setter specifically designed for working with dialogue nodes.

### 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 VariableComparers and VariableSetters to have the correct reference to the `Bank` of the current `Dialogue` everytime the editor window refreshes.

## Enums

### Node.NodeState

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

### VBProcessType

An enum used to define the way to handle multiple variable manipulators at once.

Edit this page

# Class ActionNode

Node which invokes some actions on the flow.

### Inheritance

↳ object
  ↳ Object
    ↳ ScriptableObject
      ↳ Node
        ↳ ActionNode

### Implements

IContainVariableManipulators

### 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()
ScriptableObject.SetDirty()

ScriptableObject.CreateInstance(string)
ScriptableObject.CreateInstance(Type)
ScriptableObject.CreateInstance<T>()

**Namespace:** com.absence.dialoguesystem.internals

**Assembly:** Assembly-CSharp-firstpass.dll

### Syntax

```
[HelpURL("https://b1lodhand.github.io/absent-dialogues/api/com.absence.dialoguesystem.internals.ActionN
public class ActionNode : Node, IContainVariableManipulators
```

### Remarks

Execution order goes like:

```
VBActions.ForEach(action => action.Perform());
UnityEvents?.Invoke();
CustomAction();
```

# Fields

## Next

### Declaration

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

### Field Value

| TYPE |
| --- |
| Node |

## UnityEvents

### Declaration

```
[Tooltip("All of the unity based events of this action node.")]
public UnityEvent UnityEvents
```

### Field Value

| TYPE |
| --- |
| UnityEvent |

## VBActions

### Declaration

```
[Tooltip("All of the 'VariableBank' based actions of this action node.")]
public List<NodeVariableSetter> VBActions
```

### Field Value

| TYPE |
| --- |
| List<NodeVariableSetter> |

# 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()

Use to define what to do when this action node gets passed on the flow.

### 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 VariableBank selection

### Declaration

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

### Returns

| TYPE |
| --- |
| List<NodeVariableComparer> |

## 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)>)

## GetSetters()

A list of comparers which you want to restrict in terms of VariableBank selection

### Declaration

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

### Returns

| TYPE |
| --- |
| List<NodeVariableSetter> |

# 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()

# OnValidate()

### Declaration

```
protected override void OnValidate()
```

### Overrides

Node.OnValidate()

# Pass_Inline(params object[])

Use to write what happenswhen 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 transmitthe action to another.

### Declaration

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

### Parameters

| TYPE | NAME |
|---|---|
| Action<Node> | action |

### Overrides

Node.Traverse(Action<Node>)

# Implements

IContainVariableManipulators

# Class AdditionalSpeechData

Holds some extra data which you can use on the flow.

### Inheritance

↳ object
  ↳ AdditionalSpeechData

**Namespace:** com.absence.dialoguesystem.internals

**Assembly:** Assembly-CSharp-firstpass.dll

### Syntax

```
[Serializable]
[HelpURL("https://b1lodhand.github.io/absent-dialogues/api/com.absence.dialoguesystem.internals.Additic
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 |

Edit this page

# 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]
[HelpURL("https://b1lodhand.github.io/absent-dialogues/api/com.absence.dialoguesystem.internals.Blackbc
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 |

Edit this page

# Class ConditionNode

Node which re-routes the flow under some conditions.

### Inheritance

↳ object
  ↳ Object
    ↳ ScriptableObject
      ↳ Node
        ↳ ConditionNode

### Implements

IContainVariableManipulators

### 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()
ScriptableObject.SetDirty()
ScriptableObject.CreateInstance(string)

ScriptableObject.CreateInstance(Type)
ScriptableObject.CreateInstance<T>()

**Namespace:** com.absence.dialoguesystem.internals

**Assembly:** Assembly-CSharp-firstpass.dll

### Syntax

```
[HelpURL("https://b1lodhand.github.io/absent-dialogues/api/com.absence.dialoguesystem.internals.Conditi
public class ConditionNode : Node, IContainVariableManipulators
```

# Fields

## Comparers

### Declaration

```
[Tooltip("All of the comparers this node relies on.")]
public List<NodeVariableComparer> Comparers
```

### Field Value

| TYPE |
| --- |
| List<NodeVariableComparer> |

## FalseNext

### Declaration

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

### Field Value

| TYPE |
| --- |
| Node |

## Processor

### Declaration

```
[Tooltip("Use to declare what to do with the sum of the results of comparers.")]
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

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()

## GetComparers()

A list of comparers which you want to restrict in terms of VariableBank selection

### Declaration

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

### Returns

| TYPE |
| --- |
| List<NodeVariableComparer> |

## 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()

## GetSetters()

A list of comparers which you want to restrict in terms of VariableBank selection

### Declaration

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

### Returns

| TYPE |
| --- |
| List<NodeVariableSetter> |

## 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()

## OnValidate()

### Declaration

```
protected override void OnValidate()
```

### Overrides

Node.OnValidate()

## Pass_Inline(params object[])

Use to write what happenswhen 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[])

## Process()

Use this to override (if you need) the checking result of this node.

### Declaration

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

### Returns

| TYPE | DESCRIPTION |
| --- | --- |
| bool | Normally returns the sum of the results of node's comparer list in a way declared by `Processor` |

## 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 transmitthe action to another.

### Declaration

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

### Parameters

| TYPE | NAME |
| --- | --- |
| Action<Node> | action |

### Overrides

Node.Traverse(Action<Node>)

# Implements

IContainVariableManipulators

# Class DecisionSpeechNode

Node which displays a speech with options.

## Inheritance

↳ object
  ↳ Object
    ↳ ScriptableObject
      ↳ Node
        ↳ DecisionSpeechNode

## Implements

IContainSpeech
IContainVariableManipulators

## 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

```
[HelpURL("https://b1lodhand.github.io/absent-dialogues/api/com.absence.dialoguesystem.internals.Decisic
public sealed class DecisionSpeechNode : Node, IContainSpeech, IContainVariableManipulators
```

# Fields

## Options

### Declaration

```
[Space(10)]
[Tooltip("All of the options of this node.")]
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 VariableBank selection

### Declaration

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

### Returns

| TYPE |
| --- |
| List<NodeVariableComparer> |

# 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> |

# 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 VariableBank selection

### Declaration

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

### Returns

| TYPE |
| --- |
| List<NodeVariableSetter> |

# 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()

# OnValidate()

### Declaration

```
protected override void OnValidate()
```

### Overrides

Node.OnValidate()

# Pass_Inline(params object[])

Use to write what happenswhen 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 transmitthe action to another.

### Declaration

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

### Parameters

| TYPE | NAME |
|---|---|
| Action<Node> | action |

### Overrides

Node.Traverse(Action<Node>)

# Implements

IContainSpeech
IContainVariableManipulators

# Class DialoguePartNode

Node which let's you create more and seperate 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

```
[HelpURL("https://b1lodhand.github.io/absent-dialogues/api/com.absence.dialoguesystem.internals.Dialogu
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 happenswhen 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 transmitthe 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

```
[HelpURL("https://b1lodhand.github.io/absent-dialogues/api/com.absence.dialoguesystem.internals.FastSpe
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 happenswhen 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 transmitthe 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>()

Namespace: com.absence.dialoguesystem.internals

Assembly: Assembly-CSharp-firstpass.dll

### Syntax

```
[HelpURL("https://b1lodhand.github.io/absent-dialogues/api/com.absence.dialoguesystem.internals.GotoNoc
public sealed class GotoNode : Node
```

# Fields

## TargetNode

The node which will get reached when this goto node gets passed.

### Declaration

```
[HideInInspector]
public DialoguePartNode TargetNode
```

### Field Value

| TYPE |
| --- |
| DialoguePartNode |

# 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 |

| TYPE | NAME |
| --- | --- |
| 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()

## 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 happenswhen 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)

# 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 VariableComparers and VariableSetters 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 VariableBank selection

### Declaration

```
List<NodeVariableComparer> GetComparers()
```

### Returns

| TYPE |
| --- |
| List<NodeVariableComparer> |

## GetSetters()

A list of comparers which you want to restrict in terms of VariableBank selection

### Declaration

```
List<NodeVariableSetter> GetSetters()
```

### Returns

| TYPE |
| --- |
| List<NodeVariableSetter> |

# 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

```
[HelpURL("https://b1lodhand.github.io/absent-dialogues/api/com.absence.dialoguesystem.internals.Node.ht
public abstract class Node : ScriptableObject
```

# Fields

## Blackboard

### Declaration

```
[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.

### Declaration

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

### Returns

| TYPE | DESCRIPTION |
| --- | --- |
| Node | The clone. |

## 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 happenswhen 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 riht-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 transmitthe 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

| TYPE |
| --- |
| Action<Node.NodeState> |

## OnValidation

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

### Declaration

```
public event Action OnValidation
```

### Event Type

| TYPE |
| --- |
| Action |

Edit this page

# 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 NodeVariableComparer

The comparer specifically designed for working with dialogue nodes.

### Inheritance

↳ object
  ↳ BaseVariableComparer
    ↳ NodeVariableComparer

### Inherited Members

BaseVariableComparer.m_comparisonType
BaseVariableComparer.m_targetBankGuid
BaseVariableComparer.m_targetVariableName
BaseVariableComparer.m_intValue
BaseVariableComparer.m_floatValue
BaseVariableComparer.m_stringValue
BaseVariableComparer.m_boolValue
BaseVariableComparer.GetResult()

**Namespace: com.absence.dialoguesystem.internals**

**Assembly: Assembly-CSharp-firstpass.dll**

### Syntax

```
[Serializable]
public class NodeVariableComparer : BaseVariableComparer
```

# Properties

## BlackboardBank

Bank of the blackboard in context.

### Declaration

```
public VariableBank BlackboardBank { get; set; }
```

### Property Value

| TYPE |
| --- |
| VariableBank |

## HasFixedBank

### Declaration

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

### Property Value

| TYPE |
| --- |
| bool |

### Overrides

com.absence.variablesystem.BaseVariableComparer.HasFixedBank

# Methods

## Clone(VariableBank)

Use to copy this comparer.

### Declaration

```
public NodeVariableComparer Clone(VariableBank clonedBlackboardBank)
```

### Parameters

| TYPE | NAME | DESCRIPTION |
| --- | --- | --- |
| VariableBank | `clonedBlackboardBank` | Cloned blackboard bank. |

### Returns

| TYPE | DESCRIPTION |
| --- | --- |
| NodeVariableComparer | The clone. |

# GetRuntimeBank()

### Declaration

```
protected override VariableBank GetRuntimeBank()
```

### Returns

| TYPE |
| --- |
| VariableBank |

### Overrides

com.absence.variablesystem.BaseVariableComparer.GetRuntimeBank()

# SetBlackboardBank(VariableBank)

Use to set the blackboard bank of this comparer.

### Declaration

```
public void SetBlackboardBank(VariableBank originalBlackboardBank)
```

### Parameters

| TYPE | NAME | DESCRIPTION |
| --- | --- | --- |
| VariableBank | `originalBlackboardBank` | Target bank. |

# Class NodeVariableSetter

The setter specifically designed for working with dialogue nodes.

### Inheritance

↳ object
  ↳ BaseVariableSetter
    ↳ NodeVariableSetter

### Inherited Members

BaseVariableSetter.m_setType
BaseVariableSetter.m_targetBankGuid
BaseVariableSetter.m_targetVariableName
BaseVariableSetter.m_intValue
BaseVariableSetter.m_floatValue
BaseVariableSetter.m_stringValue
BaseVariableSetter.m_boolValue
BaseVariableSetter.Perform()
BaseVariableSetter.Perform_Boolean(VariableBank)
BaseVariableSetter.Perform_String(VariableBank)
BaseVariableSetter.Perform_Float(VariableBank)
BaseVariableSetter.Perform_Int(VariableBank)

**Namespace:** com.absence.dialoguesystem.internals

**Assembly:** Assembly-CSharp-firstpass.dll

### Syntax

```
[Serializable]
public class NodeVariableSetter : BaseVariableSetter
```

# Properties

## BlackboardBank

Bank of the blackboard in context.

### Declaration

```
public VariableBank BlackboardBank { get; set; }
```

### Property Value

| TYPE |
| --- |
| VariableBank |

## HasFixedBank

### Declaration

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

### Property Value

| TYPE |
| --- |
| bool |

### Overrides

com.absence.variablesystem.BaseVariableSetter.HasFixedBank

# Methods

## Clone(VariableBank)

Use to copy this setter.

### Declaration

```
public NodeVariableSetter Clone(VariableBank clonedVariableBank)
```

### Parameters

| TYPE | NAME |
| --- | --- |
| VariableBank | clonedVariableBank |

### Returns

| TYPE | DESCRIPTION |
| --- | --- |
| NodeVariableSetter | The clone. |

## GetRuntimeBank()

### Declaration

```
protected override VariableBank GetRuntimeBank()
```

### Returns

| TYPE |
| --- |
| VariableBank |

### Overrides

com.absence.variablesystem.BaseVariableSetter.GetRuntimeBank()

## SetBlackboardBank(VariableBank)

Use to set the blackboard bank of this setter.

### Declaration

```
public void SetBlackboardBank(VariableBank originalBlackboardBank)
```

### Parameters

| TYPE | NAME | DESCRIPTION |
| --- | --- | --- |
| VariableBank | `originalBlackboardBank` | Target bank. |

# 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]
[HelpURL("https://b1lodhand.github.io/absent-dialogues/api/com.absence.dialoguesystem.internals.Option.
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]
[HelpURL("https://b1lodhand.github.io/absent-dialogues/api/com.absence.dialoguesystem.internals.Option.
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 VariableComparers which has a role on determining this option's visibility on display.

### Declaration

```
public List<NodeVariableComparer> ShowIfList
```

### Field Value

| TYPE |
| --- |
| List<NodeVariableComparer> |

# Methods

## Clone(VariableBank)

Use to clone this instance.

### Declaration

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

### Parameters

| TYPE | NAME |
| --- | --- |
| VariableBank | overrideBank |

### Returns

| TYPE |
| --- |
| Option.ShowIf |

## 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

```
[HelpURL("https://b1lodhand.github.io/absent-dialogues/api/com.absence.dialoguesystem.internals.RootNoc
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

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 happenswhen 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 transmitthe 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

```
[HelpURL("https://b1lodhand.github.io/absent-dialogues/api/com.absence.dialoguesystem.internals.StickyN
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 happenswhen 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

```
[HelpURL("https://b1lodhand.github.io/absent-dialogues/api/com.absence.dialoguesystem.internals.TitleNc
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 happenswhen 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)

Edit this page

# Enum VBProcessType

An enum used to define the way to handle multiple variable manipulators at once.

**Namespace:** com.absence.dialoguesystem.internals

**Assembly:** Assembly-CSharp-firstpass.dll

## Syntax

```
public enum VBProcessType
```

## Fields

| NAME |
| --- |
| All |
| Any |