

# Namespace com.absence.dialoguesystem

## Classes

### [Dialogue](#)

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

### [DialogueAnimationsPlayer](#)

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

### [DialogueDisplayer](#)

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

### [DialogueExtensionBase](#)

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

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

### [DialogueInstance](#)

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

### [DialogueOptionText](#)

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

### [DialoguePlayer](#)

Lets you progress in a dialogue easily.

### [DialogueSoundsPlayer](#)

A small component which is responsible for playing the sounds (if there is any) of the [Dialogue Instance](#) attached to the same gameobject.

## Enums

### [DialogueAnimationsPlayer.WorkMode](#)

Lets you select the way this extension uses the [AnimatorMemberName](#).

### [DialoguePlayer.PlayerState](#)

Shows what state the dialogue player is in.

## Delegates

### [DialogueInstance.BeforeSpeechEventHandler](#)

# Class Dialogue

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

Assembly: Assembly-CSharp-firstpass.dll

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

```
public class Dialogue : ScriptableObject
```

## Inheritance

[object](#) ← Object ← ScriptableObject ← Dialogue

## Inherited Members

ScriptableObject.SetDirty() , [ScriptableObject.CreateInstance\(string\)](#) ,  
[ScriptableObject.CreateInstance\(Type\)](#) , ScriptableObject.CreateInstance<T>() , Object.GetInstanceID() ,  
Object.GetHashCode() , [Object.Equals\(object\)](#) , Object.Instantiate(Object, Vector3, Quaternion) ,  
Object.Instantiate(Object, Vector3, Quaternion, Transform) , Object.Instantiate(Object) ,  
Object.Instantiate(Object, Transform) , [Object.Instantiate\(Object, Transform, bool\)](#) ,  
Object.Instantiate<T>(T) , Object.Instantiate<T>(T, Vector3, Quaternion) ,  
Object.Instantiate<T>(T, Vector3, Quaternion, Transform) , Object.Instantiate<T>(T, Transform) ,  
[Object.Instantiate<T>\(T, Transform, bool\)](#) , [Object.Destroy\(Object, float\)](#) , Object.Destroy(Object) ,  
[Object.DestroyImmediate\(Object, bool\)](#) , Object.DestroyImmediate(Object) ,  
[Object.FindObjectsOfType\(Type\)](#) , [Object.FindObjectsOfType\(Type, bool\)](#) ,  
[Object.FindObjectsByType\(Type, FindObjectsSortMode\)](#) ,  
[Object.FindObjectsByType\(Type, FindObjectsInactive, FindObjectsSortMode\)](#) ,  
Object.DontDestroyOnLoad(Object) , [Object.DestroyObject\(Object, float\)](#) ,  
Object.DestroyObject(Object) , [Object.FindSceneObjectsOfType\(Type\)](#) ,  
[Object.FindObjectsOfTypeIncludingAssets\(Type\)](#) , Object.FindObjectsOfType<T>() ,  
Object.FindObjectsByType<T>(FindObjectsSortMode) , [Object.FindObjectsOfType<T>\(bool\)](#) ,  
Object.FindObjectsByType<T>(FindObjectsInactive, FindObjectsSortMode) ,  
Object.FindObjectOfType<T>() , [Object.FindObjectOfType<T>\(bool\)](#) ,  
Object.FindFirstObjectByType<T>() , Object.FindAnyObjectByType<T>() ,  
Object.FindFirstObjectByType<T>(FindObjectsInactive) ,  
Object.FindAnyObjectByType<T>(FindObjectsInactive) , [Object.FindObjectsOfTypeAll\(Type\)](#) ,  
[Object.FindObjectOfType\(Type\)](#) , [Object.FindFirstObjectByType\(Type\)](#) ,  
[Object.FindAnyObjectByType\(Type\)](#) , [Object.FindObjectOfType\(Type, bool\)](#) ,  
[Object.FindFirstObjectByType\(Type, FindObjectsInactive\)](#) ,  
[Object.FindAnyObjectByType\(Type, FindObjectsInactive\)](#) , Object.ToString() , Object.name ,

`Object.hideFlags` , [object.Equals\(object, object\)](#) , [object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

## Fields

### AllNodes

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

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

### Field Value

[List](#)<[Node](#)>

### Blackboard

The [Blackboard](#) of this dialogue.

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

### Field Value

[Blackboard](#)

### LastOrCurrentNode

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

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

### Field Value

## RootNode

The [RootNode](#) of this dialogue.

```
[HideInInspector]  
public RootNode RootNode
```

### Field Value

[RootNode](#)

## Properties

### People

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

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

### Property Value

[List](#) <Person>

## Methods

### Clone()

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

```
public Dialogue Clone()
```

### Returns

## [Dialogue](#)

### CreateNode(Type)

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

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

Parameters

type [Type](#)

Returns

[Node](#)

### DeleteNode(Node)

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

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

Parameters

node [Node](#)

### GetAllDialogParts()

Use to get a list of all [DialoguePartNodes](#)s in this dialogue.

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

Returns

[List](#) <[DialoguePartNode](#)>

The entire list of [DialoguePartNodes](#) in the current dialogue.

## GetDialogPartNodesWithName(string)

Use to find [DialoguePartNodes](#) with a specific name.

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

Parameters

targetName [string](#)

Returns

[List](#) <[DialoguePartNode](#)>

A list of [DialoguePartNodes](#) with that specific name. Throws an exception nothing's found.

## Initialize()

It teleports the flow back to the root node.

```
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.

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

Parameters

overridePeople [List](#) <Person>

## 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.

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

## Parameters

passData [object](#)[]

# Class DialogueAnimationsPlayer

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

Assembly: Assembly-CSharp-firstpass.dll

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

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

## Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← [DialogueExtensionBase](#) ← DialogueAnimationsPlayer

## Inherited Members

[DialogueExtensionBase.m\\_instance](#) , MonoBehaviour.IsInvoking() , MonoBehaviour.CancelInvoke() ,  
[MonoBehaviour.Invoke\(string, float\)](#) , [MonoBehaviour.InvokeRepeating\(string, float, float\)](#) ,  
[MonoBehaviour.CancelInvoke\(string\)](#) , [MonoBehaviour.IsInvoking\(string\)](#) ,  
[MonoBehaviour.StartCoroutine\(string\)](#) , [MonoBehaviour.StartCoroutine\(string, object\)](#) ,  
[MonoBehaviour.StartCoroutine\(IEnumerator\)](#) , [MonoBehaviour.StartCoroutine\\_Auto\(IEnumerator\)](#) ,  
[MonoBehaviour.StopCoroutine\(IEnumerator\)](#) , MonoBehaviour.StopCoroutine(Coroutine) ,  
[MonoBehaviour.StopCoroutine\(string\)](#) , MonoBehaviour.StopAllCoroutines() ,  
[MonoBehaviour.print\(object\)](#) , MonoBehaviour.destroyCancellationToken ,  
MonoBehaviour.useGUILayout , MonoBehaviour.runInEditMode , Behaviour.enabled ,  
Behaviour.isActiveAndEnabled , [Component.GetComponent\(Type\)](#) , Component.GetComponent<T>() ,  
[Component.TryGetComponent\(Type, out Component\)](#) , Component.TryGetComponent<T>(out T) ,  
[Component.GetComponent\(string\)](#) , [Component.GetComponentInChildren\(Type, bool\)](#) ,  
[Component.GetComponentInChildren\(Type\)](#) , [Component.GetComponentInChildren<T>\(bool\)](#) ,  
Component.GetComponentInChildren<T>() , [Component.GetComponentsInChildren\(Type, bool\)](#) ,  
[Component.GetComponentsInChildren\(Type\)](#) , [Component.GetComponentsInChildren<T>\(bool\)](#) ,  
[Component.GetComponentsInChildren<T>\(bool, List<T>\)](#) ,  
Component.GetComponentsInChildren<T>() , [Component.GetComponentsInChildren<T>\(List<T>\)](#) ,  
[Component.GetComponentInParent\(Type, bool\)](#) , [Component.GetComponentInParent\(Type\)](#) ,  
[Component.GetComponentInParent<T>\(bool\)](#) , Component.GetComponentInParent<T>() ,  
[Component.GetComponentsInParent\(Type, bool\)](#) , [Component.GetComponentsInParent\(Type\)](#) ,  
[Component.GetComponentsInParent<T>\(bool\)](#) ,  
Component.GetComponentsInParent<T>(bool, List<T>) , Component.GetComponentInParent<T>()

[Component.GetComponents\(Type\)](#) , [Component.GetComponents\(Type, List<Component>\)](#) ,  
[Component.GetComponents<T>\(List<T>\)](#) , Component.GetComponents<T>() ,  
[Component.CompareTag\(string\)](#) ,  
[Component.SendMessageUpwards\(string, object, SendMessageOptions\)](#) ,  
[Component.SendMessageUpwards\(string, object\)](#) , [Component.SendMessageUpwards\(string\)](#) ,  
[Component.SendMessageUpwards\(string, SendMessageOptions\)](#) ,  
[Component.SendMessage\(string, object\)](#) , [Component.SendMessage\(string\)](#) ,  
[Component.SendMessage\(string, object, SendMessageOptions\)](#) ,  
[Component.SendMessage\(string, SendMessageOptions\)](#) ,  
[Component.BroadcastMessage\(string, object, SendMessageOptions\)](#) ,  
[Component.BroadcastMessage\(string, object\)](#) , [Component.BroadcastMessage\(string\)](#) ,  
[Component.BroadcastMessage\(string, SendMessageOptions\)](#) , Component.transform ,  
Component.gameObject , Component.tag , Object.GetInstanceID() , Object.GetHashCode()  
[Object.Equals\(object\)](#) , Object.Instantiate(Object, Vector3, Quaternion) ,  
Object.Instantiate(Object, Vector3, Quaternion, Transform) , Object.Instantiate(Object) ,  
Object.Instantiate(Object, Transform) , [Object.Instantiate\(Object, Transform, bool\)](#) ,  
Object.Instantiate<T>(T) , Object.Instantiate<T>(T, Vector3, Quaternion) ,  
Object.Instantiate<T>(T, Vector3, Quaternion, Transform) , Object.Instantiate<T>(T, Transform) ,  
[Object.Instantiate<T>\(T, Transform, bool\)](#) , [Object.Destroy\(Object, float\)](#) , Object.Destroy(Object) ,  
[Object.DestroyImmediate\(Object, bool\)](#) , Object.DestroyImmediate(Object) ,  
[Object.FindObjectsOfType\(Type\)](#) , [Object.FindObjectsOfType\(Type, bool\)](#) ,  
[Object.FindObjectsByType\(Type, FindObjectsSortMode\)](#) ,  
[Object.FindObjectsByType\(Type, FindObjectsInactive, FindObjectsSortMode\)](#) ,  
Object.DontDestroyOnLoad(Object) , [Object.DestroyObject\(Object, float\)](#) ,  
Object.DestroyObject(Object) , [Object.FindSceneObjectsOfType\(Type\)](#) ,  
[Object.FindObjectsOfTypeIncludingAssets\(Type\)](#) , Object.FindObjectsOfType<T>() ,  
Object.FindObjectsByType<T>(FindObjectsSortMode) , [Object.FindObjectsOfType<T>\(bool\)](#) ,  
Object.FindObjectsByType<T>(FindObjectInactive, FindObjectsSortMode) ,  
Object.FindObjectOfType<T>() , [Object.FindObjectOfType<T>\(bool\)](#) ,  
Object.FindFirstObjectOfType<T>() , Object.FindAnyObjectOfType<T>() ,  
Object.FindFirstObjectOfType<T>(FindObjectInactive) ,  
Object.FindAnyObjectOfType<T>(FindObjectInactive) , [Object.FindObjectsOfTypeAll\(Type\)](#) ,  
[Object.FindObjectOfType\(Type\)](#) , [Object.FindFirstObjectOfType\(Type\)](#) ,  
[Object.FindAnyObjectOfType\(Type\)](#) , [Object.FindObjectOfType\(Type, bool\)](#) ,  
[Object.FindFirstObjectOfType\(Type, FindObjectsInactive\)](#) ,  
[Object.FindAnyObjectOfType\(Type, FindObjectsInactive\)](#) , Object.ToString() , Object.name ,  
Object.hideFlags , [object.Equals\(object, object\)](#) , [object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) ,  
[object.ReferenceEquals\(object, object\)](#)

# Methods

## OnHandleAdditionalData(AdditionalSpeechData)

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

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

### Parameters

data [AdditionalSpeechData](#)

# Enum DialogueAnimationsPlayer.WorkMode

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

Assembly: Assembly-CSharp-firstpass.dll

Lets you select the way this extension uses the [AnimatorMemberName](#).

```
public enum DialogueAnimationsPlayer.WorkMode
```

## Fields

CrossFade = 0

SetTrigger = 1

# Class DialogueDisplay

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

Assembly: Assembly-CSharp-firstpass.dll

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

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

## Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← StaticInstance<[DialogueDisplay](#)> ← Singleton<[DialogueDisplay](#)> ← DialogueDisplay

## Inherited Members

Singleton<DialogueDisplay>.Awake() , StaticInstance<DialogueDisplay>.OnApplicationQuit() ,  
StaticInstance<DialogueDisplay>.Instance , MonoBehaviour.IsInvoking() ,  
MonoBehaviour.CancelInvoke() , [MonoBehaviour.Invoke\(string, float\)](#) ,  
[MonoBehaviour.InvokeRepeating\(string, float, float\)](#) , [MonoBehaviour.CancelInvoke\(string\)](#) ,  
[MonoBehaviour.IsInvoking\(string\)](#) , [MonoBehaviour.StartCoroutine\(string\)](#) ,  
[MonoBehaviour.StartCoroutine\(string, object\)](#) , [MonoBehaviour.StartCoroutine\(IEnumerator\)](#) ,  
[MonoBehaviour.StartCoroutine\\_Auto\(IEnumerator\)](#) , [MonoBehaviour.StopCoroutine\(IEnumerator\)](#) ,  
MonoBehaviour.StopCoroutine(Coroutine) , [MonoBehaviour.StopCoroutine\(string\)](#) ,  
MonoBehaviour.StopAllCoroutines() , [MonoBehaviour.print\(object\)](#) ,  
MonoBehaviour.destroyCancellationToken , MonoBehaviour.useGUILayout ,  
MonoBehaviour.runInEditMode , Behaviour.enabled , Behaviour.isActiveAndEnabled ,  
[Component.GetComponent\(Type\)](#) , Component.GetComponent<T>() ,  
[Component.TryGetComponent\(Type, out Component\)](#) , Component.TryGetComponent<T>(out T) ,  
[Component.GetComponent\(string\)](#) , [Component.GetComponentInChildren\(Type, bool\)](#) ,  
[Component.GetComponentInChildren\(Type\)](#) , [Component.GetComponentInChildren<T>\(bool\)](#) ,  
Component.GetComponentInChildren<T>() , [Component.GetComponentsInChildren\(Type, bool\)](#) ,  
[Component.GetComponentsInChildren\(Type\)](#) , [Component.GetComponentsInChildren<T>\(bool\)](#) ,  
[Component.GetComponentsInChildren<T>\(bool, List<T>\)](#) ,  
Component.GetComponentsInChildren<T>() , [Component.GetComponentsInChildren<T>\(List<T>\)](#) ,  
[Component.GetComponentInParent\(Type, bool\)](#) , [Component.GetComponentInParent\(Type\)](#) ,  
[Component.GetComponentInParent<T>\(bool\)](#) , Component.GetComponentInParent<T>() ,  
[Component.GetComponentsInParent\(Type, bool\)](#) , [Component.GetComponentsInParent\(Type\)](#) ,  
[Component.GetComponentsInParent<T>\(bool\)](#) ,

[Component.GetComponentInParent<T>\(bool, List<T>\)](#) , Component.GetComponentInParent<T>() ,  
[Component.GetComponents\(Type\)](#) , [Component.GetComponents\(Type, List<Component>\)](#) ,  
[Component.GetComponents<T>\(List<T>\)](#) , Component.GetComponents<T>() ,  
[Component.CompareTag\(string\)](#) ,  
[Component.SendMessageUpwards\(string, object, SendMessageOptions\)](#) ,  
[Component.SendMessageUpwards\(string, object\)](#) , [Component.SendMessageUpwards\(string\)](#) ,  
[Component.SendMessageUpwards\(string, SendMessageOptions\)](#) ,  
[Component.SendMessage\(string, object\)](#) , [Component.SendMessage\(string\)](#) ,  
[Component.SendMessage\(string, object, SendMessageOptions\)](#) ,  
[Component.SendMessage\(string, SendMessageOptions\)](#) ,  
[Component.BroadcastMessage\(string, object, SendMessageOptions\)](#) ,  
[Component.BroadcastMessage\(string, object\)](#) , [Component.BroadcastMessage\(string\)](#) ,  
[Component.BroadcastMessage\(string, SendMessageOptions\)](#) , Component.transform ,  
Component.gameObject , Component.tag , Object.GetInstanceID() , Object.GetHashCode() ,  
[Object.Equals\(object\)](#) , Object.Instantiate(Object, Vector3, Quaternion) ,  
Object.Instantiate(Object, Vector3, Quaternion, Transform) , Object.Instantiate(Object) ,  
Object.Instantiate(Object, Transform) , [Object.Instantiate\(Object, Transform, bool\)](#) ,  
Object.Instantiate<T>(T) , Object.Instantiate<T>(T, Vector3, Quaternion) ,  
Object.Instantiate<T>(T, Vector3, Quaternion, Transform) , Object.Instantiate<T>(T, Transform) ,  
[Object.Instantiate<T>\(T, Transform, bool\)](#) , [Object.Destroy\(Object, float\)](#) , Object.Destroy(Object) ,  
[Object.DestroyImmediate\(Object, bool\)](#) , Object.DestroyImmediate(Object) ,  
[Object.FindObjectsOfType\(Type\)](#) , [Object.FindObjectsOfType\(Type, bool\)](#) ,  
[Object.FindObjectsByType\(Type, FindObjectsSortMode\)](#) ,  
[Object.FindObjectsByType\(Type, FindObjectsInactive, FindObjectsSortMode\)](#) ,  
Object.DontDestroyOnLoad(Object) , [Object.DestroyObject\(Object, float\)](#) ,  
Object.DestroyObject(Object) , [Object.FindSceneObjectsOfType\(Type\)](#) ,  
[Object.FindObjectsOfTypeIncludingAssets\(Type\)](#) , Object.FindObjectOfType<T>() ,  
Object.FindObjectsByType<T>(FindObjectsSortMode) , [Object.FindObjectsOfType<T>\(bool\)](#) ,  
Object.FindObjectsByType<T>(FindObjectsInactive, FindObjectsSortMode) ,  
Object.FindObjectOfType<T>() , [Object.FindObjectOfType<T>\(bool\)](#) ,  
Object.FindFirstObjectByType<T>() , Object.FindAnyObjectByType<T>() ,  
Object.FindFirstObjectByType<T>(FindObjectsInactive) ,  
Object.FindAnyObjectByType<T>(FindObjectsInactive) , [Object.FindObjectsOfTypeAll\(Type\)](#) ,  
[Object.FindObjectOfType\(Type\)](#) , [Object.FindFirstObjectByType\(Type\)](#) ,  
[Object.FindAnyObjectByType\(Type\)](#) , [Object.FindObjectOfType\(Type, bool\)](#) ,  
[Object.FindFirstObjectByType\(Type, FindObjectsInactive\)](#) ,  
[Object.FindAnyObjectByType\(Type, FindObjectsInactive\)](#) , Object.ToString() , Object.name ,  
Object.hideFlags , [object.Equals\(object, object\)](#) , [object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) ,  
[object.ReferenceEquals\(object, object\)](#)

# Methods

## Display(Person, string)

Displays a speech with no options.

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

Parameters

**speaker** Person

**speech** [string](#)

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

Displays a speech with options.

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

Parameters

**speaker** Person

**speech** [string](#)

**options** [List](#)<[Option](#)>

**optionPressAction** [Action](#)<[int](#)>

## Occupy()

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

```
public bool Occupy()
```

Returns

bool ↗

Returns false if the display is already occupied. Returns true otherwise.

## Release()

Removes the occupancy of the display. CAUTION! [DialogueDisplay](#) does not hold a reference to the current occupier. Because of that, be careful calling this function.

```
public void Release()
```

# Class DialogueExtensionBase

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

Assembly: Assembly-CSharp-firstpass.dll

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

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

## Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← DialogueExtensionBase

## Derived

[DialogueAnimationsPlayer](#), [DialogueInputHandler\\_Legacy](#), [DialogueSoundsPlayer](#)

## Inherited Members

MonoBehaviour.IsInvoking() , MonoBehaviour.CancelInvoke() , [MonoBehaviour.Invoke\(string, float\)](#) ,  
[MonoBehaviour.InvokeRepeating\(string, float, float\)](#) , [MonoBehaviour.CancelInvoke\(string\)](#) ,  
[MonoBehaviour.IsInvoking\(string\)](#) , [MonoBehaviour.StartCoroutine\(string\)](#) ,  
[MonoBehaviour.StartCoroutine\(string, object\)](#) , [MonoBehaviour.StartCoroutine\(IEnumerator\)](#) ,  
[MonoBehaviour.StartCoroutine\\_Auto\(IEnumerator\)](#) , [MonoBehaviour.StopCoroutine\(IEnumerator\)](#) ,  
MonoBehaviour.StopCoroutine(Coroutine) , [MonoBehaviour.StopCoroutine\(string\)](#) ,  
MonoBehaviour.StopAllCoroutines() , [MonoBehaviour.print\(object\)](#) ,  
MonoBehaviour.destroyCancellationToken , MonoBehaviour.useGUILayout ,  
MonoBehaviour.runInEditMode , Behaviour.enabled , Behaviour.isActiveAndEnabled ,  
[Component.GetComponent\(Type\)](#) , Component.GetComponent<T>() ,  
[Component.TryGetComponent\(Type, out Component\)](#) , Component.TryGetComponent<T>(out T) ,  
[Component.GetComponent\(string\)](#) , [Component.GetComponentInChildren\(Type, bool\)](#) ,  
[Component.GetComponentInChildren\(Type\)](#) , [Component.GetComponentInChildren<T>\(bool\)](#) ,  
Component.GetComponentInChildren<T>() , [Component.GetComponentsInChildren\(Type, bool\)](#) ,  
[Component.GetComponentsInChildren\(Type\)](#) , [Component.GetComponentsInChildren<T>\(bool\)](#) ,  
[Component.GetComponentsInChildren<T>\(bool, List<T>\)](#) ,  
Component.GetComponentsInChildren<T>() , [Component.GetComponentsInChildren<T>\(List<T>\)](#) ,  
[Component.GetComponentInParent\(Type, bool\)](#) , [Component.GetComponentInParent\(Type\)](#) ,  
[Component.GetComponentInParent<T>\(bool\)](#) , Component.GetComponentInParent<T>() ,  
[Component.GetComponentsInParent\(Type, bool\)](#) , [Component.GetComponentsInParent\(Type\)](#) ,  
[Component.GetComponentsInParent<T>\(bool\)](#) ,  
[Component.GetComponentsInParent<T>\(bool, List<T>\)](#) , Component.GetComponentsInParent<T>() ,  
[Component.GetComponents\(Type\)](#) , [Component.GetComponents\(Type, List<Component>\)](#) ,

[Component.GetComponents<T>\(List<T>\)](#) , Component.GetComponents<T>() ,  
[Component.CompareTag\(string\)](#) ,  
[Component.SendMessageUpwards\(string, object, SendMessageOptions\)](#) ,  
[Component.SendMessageUpwards\(string, object\)](#) , [Component.SendMessageUpwards\(string\)](#) ,  
[Component.SendMessageUpwards\(string, SendMessageOptions\)](#) ,  
[Component.SendMessage\(string, object\)](#) , [Component.SendMessage\(string\)](#) ,  
[Component.SendMessage\(string, object, SendMessageOptions\)](#) ,  
[Component.SendMessage\(string, SendMessageOptions\)](#) ,  
[Component.BroadcastMessage\(string, object, SendMessageOptions\)](#) ,  
[Component.BroadcastMessage\(string, object\)](#) , [Component.BroadcastMessage\(string\)](#) ,  
[Component.BroadcastMessage\(string, SendMessageOptions\)](#) , Component.transform ,  
Component.gameObject , Component.tag , Object.GetInstanceID() , Object.GetHashCode() ,  
[Object.Equals\(object\)](#) , Object.Instantiate(Object, Vector3, Quaternion) ,  
Object.Instantiate(Object, Vector3, Quaternion, Transform) , Object.Instantiate(Object) ,  
Object.Instantiate(Object, Transform) , [Object.Instantiate\(Object, Transform, bool\)](#) ,  
Object.Instantiate<T>(T) , Object.Instantiate<T>(T, Vector3, Quaternion) ,  
Object.Instantiate<T>(T, Vector3, Quaternion, Transform) , Object.Instantiate<T>(T, Transform) ,  
[Object.Instantiate<T>\(T, Transform, bool\)](#) , [Object.Destroy\(Object, float\)](#) , Object.Destroy(Object) ,  
[Object.DestroyImmediate\(Object, bool\)](#) , Object.DestroyImmediate(Object) ,  
[Object.FindObjectsOfType\(Type\)](#) , [Object.FindObjectsOfType\(Type, bool\)](#) ,  
[Object.FindObjectsByType\(Type, FindObjectsSortMode\)](#) ,  
[Object.FindObjectsByType\(Type, FindObjectsInactive, FindObjectsSortMode\)](#) ,  
Object.DontDestroyOnLoad(Object) , [Object.DestroyObject\(Object, float\)](#) ,  
Object.DestroyObject(Object) , [Object.FindSceneObjectsOfType\(Type\)](#) ,  
[Object.FindObjectsOfTypeIncludingAssets\(Type\)](#) , Object.FindObjectsOfType<T>() ,  
Object.FindObjectsByType<T>(FindObjectsSortMode) , [Object.FindObjectsOfType<T>\(bool\)](#) ,  
Object.FindObjectsByType<T>(FindObjectInactive, FindObjectsSortMode) ,  
Object.FindObjectOfType<T>() , [Object.FindObjectOfType<T>\(bool\)](#) ,  
Object.FindFirstObjectOfType<T>() , Object.FindAnyObjectOfType<T>() ,  
Object.FindFirstObjectOfType<T>(FindObjectInactive) ,  
Object.FindAnyObjectOfType<T>(FindObjectInactive) , [Object.FindObjectsOfTypeAll\(Type\)](#) ,  
[Object.FindObjectOfType\(Type\)](#) , [Object.FindFirstObjectOfType\(Type\)](#) ,  
[Object.FindAnyObjectOfType\(Type\)](#) , [Object.FindObjectOfType\(Type, bool\)](#) ,  
[Object.FindFirstObjectOfType\(Type, FindObjectsInactive\)](#) ,  
[Object.FindAnyObjectOfType\(Type, FindObjectsInactive\)](#) , Object.ToString() , Object.name ,  
Object.hideFlags , [object.Equals\(object, object\)](#) , [object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) ,  
[object.ReferenceEquals\(object, object\)](#)

## Fields

## m\_instance

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

```
[SerializeField]  
[ReadOnly]  
protected DialogueInstance m_instance
```

### Field Value

[DialogueInstance](#)

## Methods

### OnHandleAdditionalData(AdditionalSpeechData)

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

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

### Parameters

data [AdditionalSpeechData](#)

# Class DialogueInputHandler\_Legacy

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

Assembly: Assembly-CSharp-firstpass.dll

```
public class DialogueInputHandler_Legacy : DialogueExtensionBase
```

## Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← [DialogueExtensionBase](#) ← DialogueInputHandler\_Legacy

## Inherited Members

[DialogueExtensionBase.m\\_instance](#) , MonoBehaviour.IsInvoking() , MonoBehaviour.CancelInvoke() ,  
[MonoBehaviour.Invoke\(string, float\)](#) , [MonoBehaviour.InvokeRepeating\(string, float, float\)](#) ,  
[MonoBehaviour.CancelInvoke\(string\)](#) , [MonoBehaviour.IsInvoking\(string\)](#) ,  
[MonoBehaviour.StartCoroutine\(string\)](#) , [MonoBehaviour.StartCoroutine\(string, object\)](#) ,  
[MonoBehaviour.StartCoroutine\(IEnumerator\)](#) , [MonoBehaviour.StartCoroutine\\_Auto\(IEnumerator\)](#) ,  
[MonoBehaviour.StopCoroutine\(IEnumerator\)](#) , MonoBehaviour.StopCoroutine(Coroutine) ,  
[MonoBehaviour.StopCoroutine\(string\)](#) , MonoBehaviour.StopAllCoroutines() ,  
[MonoBehaviour.print\(object\)](#) , MonoBehaviour.destroyCancellationToken ,  
MonoBehaviour.useGUILayout , MonoBehaviour.runInEditMode , Behaviour.enabled ,  
Behaviour.isActiveAndEnabled , [Component.GetComponent\(Type\)](#) , Component.GetComponent<T>() ,  
[Component.TryGetComponent\(Type, out Component\)](#) , Component.TryGetComponent<T>(out T) ,  
[Component.GetComponent\(string\)](#) , [Component.GetComponentInChildren\(Type, bool\)](#) ,  
[Component.GetComponentInChildren\(Type\)](#) , [Component.GetComponentInChildren<T>\(bool\)](#) ,  
Component.GetComponentInChildren<T>() , [Component.GetComponentsInChildren\(Type, bool\)](#) ,  
[Component.GetComponentsInChildren\(Type\)](#) , [Component.GetComponentsInChildren<T>\(bool\)](#) ,  
[Component.GetComponentsInChildren<T>\(bool, List<T>\)](#) ,  
Component.GetComponentsInChildren<T>() , [Component.GetComponentsInChildren<T>\(List<T>\)](#) ,  
[Component.GetComponentInParent\(Type, bool\)](#) , [Component.GetComponentInParent\(Type\)](#) ,  
[Component.GetComponentInParent<T>\(bool\)](#) , Component.GetComponentInParent<T>() ,  
[Component.GetComponentsInParent\(Type, bool\)](#) , [Component.GetComponentsInParent\(Type\)](#) ,  
[Component.GetComponentsInParent<T>\(bool\)](#) ,  
[Component.GetComponentsInParent<T>\(bool, List<T>\)](#) , Component.GetComponentsInParent<T>() ,  
[Component.GetComponents\(Type\)](#) , [Component.GetComponents\(Type, List<Component>\)](#) ,  
[Component.GetComponents<T>\(List<T>\)](#) , Component.GetComponents<T>() ,  
[Component.CompareTag\(string\)](#) ,  
[Component.SendMessageUpwards\(string, object, SendMessageOptions\)](#) ,  
[Component.SendMessageUpwards\(string, object\)](#) , [Component.SendMessageUpwards\(string\)](#) ,

[Component.SendMessageUpwards\(string, SendMessageOptions\)](#) ,  
[Component.SendMessage\(string, object\)](#) , [Component.SendMessage\(string\)](#) ,  
[Component.SendMessage\(string, object, SendMessageOptions\)](#) ,  
[Component.SendMessage\(string, SendMessageOptions\)](#) ,  
[Component.BroadcastMessage\(string, object, SendMessageOptions\)](#) ,  
[Component.BroadcastMessage\(string, object\)](#) , [Component.BroadcastMessage\(string\)](#) ,  
[Component.BroadcastMessage\(string, SendMessageOptions\)](#) , Component.transform ,  
Component.gameObject , Component.tag , Object.GetInstanceID() , Object.GetHashCode() ,  
[Object.Equals\(object\)](#) , Object.Instantiate(Object, Vector3, Quaternion) ,  
Object.Instantiate(Object, Vector3, Quaternion, Transform) , Object.Instantiate(Object) ,  
Object.Instantiate(Object, Transform) , [Object.Instantiate\(Object, Transform, bool\)](#) ,  
Object.Instantiate<T>(T) , Object.Instantiate<T>(T, Vector3, Quaternion) ,  
Object.Instantiate<T>(T, Vector3, Quaternion, Transform) , Object.Instantiate<T>(T, Transform) ,  
[Object.Instantiate<T>\(T, Transform, bool\)](#) , [Object.Destroy\(Object, float\)](#) , Object.Destroy(Object) ,  
[Object.DestroyImmediate\(Object, bool\)](#) , Object.DestroyImmediate(Object) ,  
[Object.FindObjectsOfType\(Type\)](#) , [Object.FindObjectsOfType\(Type, bool\)](#) ,  
[Object.FindObjectsByType\(Type, FindObjectsSortMode\)](#) ,  
[Object.FindObjectsByType\(Type, FindObjectsInactive, FindObjectsSortMode\)](#) ,  
Object.DontDestroyOnLoad(Object) , [Object.DestroyObject\(Object, float\)](#) ,  
Object.DestroyObject(Object) , [Object.FindSceneObjectsOfType\(Type\)](#) ,  
[Object.FindObjectsOfTypeIncludingAssets\(Type\)](#) , Object.FindObjectsOfType<T>() ,  
Object.FindObjectsByType<T>(FindObjectsSortMode) , [Object.FindObjectsOfType<T>\(bool\)](#) ,  
Object.FindObjectsByType<T>(FindObjectsInactive, FindObjectsSortMode) ,  
Object.FindObjectOfType<T>() , [Object.FindObjectOfType<T>\(bool\)](#) ,  
Object.FindFirstObjectByType<T>() , Object.FindAnyObjectByType<T>() ,  
Object.FindFirstObjectByType<T>(FindObjectsInactive) ,  
Object.FindAnyObjectByType<T>(FindObjectsInactive) , [Object.FindObjectsOfTypeAll\(Type\)](#) ,  
[Object.FindObjectOfType\(Type\)](#) , [Object.FindFirstObjectByType\(Type\)](#) ,  
[Object.FindAnyObjectByType\(Type\)](#) , [Object.FindObjectOfType\(Type, bool\)](#) ,  
[Object.FindFirstObjectByType\(Type, FindObjectsInactive\)](#) ,  
[Object.FindAnyObjectByType\(Type, FindObjectsInactive\)](#) , Object.ToString() , Object.name ,  
Object.hideFlags , [object.Equals\(object, object\)](#) , [object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) ,  
[object.ReferenceEquals\(object, object\)](#)

## Methods

### OnHandleAdditionalData(AdditionalSpeechData)

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

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

## Parameters

data [AdditionalSpeechData](#)

# Class DialogueInstance

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

Assembly: Assembly-CSharp-firstpass.dll

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

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

## Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← DialogueInstance

## Inherited Members

MonoBehaviour.IsInvoking() , MonoBehaviour.CancelInvoke() , [MonoBehaviour.Invoke\(string, float\)](#) ,  
[MonoBehaviour.InvokeRepeating\(string, float, float\)](#) , [MonoBehaviour.CancelInvoke\(string\)](#) ,  
[MonoBehaviour.IsInvoking\(string\)](#) , [MonoBehaviour.StartCoroutine\(string\)](#) ,  
[MonoBehaviour.StartCoroutine\(string, object\)](#) , [MonoBehaviour.StartCoroutine\(IEnumerator\)](#) ,  
[MonoBehaviour.StartCoroutine\\_Auto\(IEnumerator\)](#) , [MonoBehaviour.StopCoroutine\(IEnumerator\)](#) ,  
MonoBehaviour.StopCoroutine(Coroutine) , [MonoBehaviour.StopCoroutine\(string\)](#) ,  
MonoBehaviour.StopAllCoroutines() , [MonoBehaviour.print\(object\)](#) ,  
MonoBehaviour.destroyCancellationToken , MonoBehaviour.useGUILayout ,  
MonoBehaviour.RunWithEditMode , Behaviour.enabled , Behaviour.isActiveAndEnabled ,  
[Component.GetComponent\(Type\)](#) , Component.GetComponent<T>() ,  
[Component.TryGetComponent\(Type, out Component\)](#) , Component.TryGetComponent<T>(out T) ,  
[Component.GetComponent\(string\)](#) , [Component.GetComponentInChildren\(Type, bool\)](#) ,  
[Component.GetComponentInChildren\(Type\)](#) , [Component.GetComponentInChildren<T>\(bool\)](#) ,  
Component.GetComponentInChildren<T>() , [Component.GetComponentsInChildren\(Type, bool\)](#) ,  
[Component.GetComponentsInChildren\(Type\)](#) , [Component.GetComponentsInChildren<T>\(bool\)](#) ,  
[Component.GetComponentsInChildren<T>\(bool, List<T>\)](#) ,  
Component.GetComponentsInChildren<T>() , [Component.GetComponentsInChildren<T>\(List<T>\)](#) ,  
[Component.GetComponentInParent\(Type, bool\)](#) , [Component.GetComponentInParent\(Type\)](#) ,  
[Component.GetComponentInParent<T>\(bool\)](#) , Component.GetComponentInParent<T>() ,  
[Component.GetComponentsInParent\(Type, bool\)](#) , [Component.GetComponentsInParent\(Type\)](#) ,  
[Component.GetComponentsInParent<T>\(bool\)](#) ,  
[Component.GetComponentsInParent<T>\(bool, List<T>\)](#) , Component.GetComponentsInParent<T>() ,  
[Component.GetComponents\(Type\)](#) , [Component.GetComponents\(Type, List<Component>\)](#) ,  
[Component.GetComponents<T>\(List<T>\)](#) , Component.GetComponents<T>() ,  
[Component.CompareTag\(string\)](#) ,

[Component.SendMessageUpwards\(string, object, SendMessageOptions\)](#) ,  
[Component.SendMessageUpwards\(string, object\)](#) , [Component.SendMessageUpwards\(string\)](#) ,  
[Component.SendMessageUpwards\(string, SendMessageOptions\)](#) ,  
[Component.SendMessage\(string, object\)](#) , [Component.SendMessage\(string\)](#) ,  
[Component.SendMessage\(string, object, SendMessageOptions\)](#) ,  
[Component.SendMessage\(string, SendMessageOptions\)](#) ,  
[Component.BroadcastMessage\(string, object, SendMessageOptions\)](#) ,  
[Component.BroadcastMessage\(string, object\)](#) , [Component.BroadcastMessage\(string\)](#) ,  
[Component.BroadcastMessage\(string, SendMessageOptions\)](#) , Component.transform ,  
Component.gameObject , Component.tag , Object.GetInstanceID() , Object.GetHashCode() ,  
[Object.Equals\(object\)](#) , Object.Instantiate(Object, Vector3, Quaternion) ,  
Object.Instantiate(Object, Vector3, Quaternion, Transform) , Object.Instantiate(Object) ,  
Object.Instantiate(Object, Transform) , [Object.Instantiate\(Object, Transform, bool\)](#) ,  
Object.Instantiate<T>(T) , Object.Instantiate<T>(T, Vector3, Quaternion) ,  
Object.Instantiate<T>(T, Vector3, Quaternion, Transform) , Object.Instantiate<T>(T, Transform) ,  
[Object.Instantiate<T>\(T, Transform, bool\)](#) , [Object.Destroy\(Object, float\)](#) , Object.Destroy(Object) ,  
[Object.DestroyImmediate\(Object, bool\)](#) , Object.DestroyImmediate(Object) ,  
[Object.FindObjectsOfType\(Type\)](#) , [Object.FindObjectsOfType\(Type, bool\)](#) ,  
[Object.FindObjectsByType\(Type, FindObjectsSortMode\)](#) ,  
[Object.FindObjectsByType\(Type, FindObjectsInactive, FindObjectsSortMode\)](#) ,  
Object.DontDestroyOnLoad(Object) , [Object.DestroyObject\(Object, float\)](#) ,  
Object.DestroyObject(Object) , [Object.FindSceneObjectsOfType\(Type\)](#) ,  
[Object.FindObjectsOfTypeIncludingAssets\(Type\)](#) , Object.FindObjectsOfType<T>() ,  
Object.FindObjectsByType<T>(FindObjectsSortMode) , [Object.FindObjectsOfType<T>\(bool\)](#) ,  
Object.FindObjectsByType<T>(FindObjectsInactive, FindObjectsSortMode) ,  
Object.FindObjectOfType<T>() , [Object.FindObjectOfType<T>\(bool\)](#) ,  
Object.FindFirstObjectByType<T>() , Object.FindAnyObjectByType<T>() ,  
Object.FindFirstObjectByType<T>(FindObjectsInactive) ,  
Object.FindAnyObjectByType<T>(FindObjectsInactive) , [Object.FindObjectsOfTypeAll\(Type\)](#) ,  
[Object.FindObjectOfType\(Type\)](#) , [Object.FindFirstObjectByType\(Type\)](#) ,  
[Object.FindAnyObjectByType\(Type\)](#) , [Object.FindObjectOfType\(Type, bool\)](#) ,  
[Object.FindFirstObjectByType\(Type, FindObjectsInactive\)](#) ,  
[Object.FindAnyObjectByType\(Type, FindObjectsInactive\)](#) , Object.ToString() , Object.name ,  
Object.hideFlags , [object.Equals\(object, object\)](#) , [object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) ,  
[object.ReferenceEquals\(object, object\)](#)

## Properties

### Player

[DialoguePlayer](#) of this instance.

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

Property Value

[DialoguePlayer](#)

## Methods

### AddExtension<T>()

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

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

Type Parameters

T

### EnterDialogue()

Use to enter dialogue.

```
public bool EnterDialogue()
```

Returns

[bool](#) ↗

**False** if the [DialogueDisplayer](#) is already occupied by any other script. Returns **true** otherwise.

### ExitDialogue()

Use to exit current dialogue.

```
public void ExitDialogue()
```

## Events

### OnBeforeSpeech

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

```
public event DialogueInstance.BeforeSpeechEventHandler OnBeforeSpeech
```

#### Event Type

[DialogueInstance.BeforeSpeechEventHandler](#)

### OnHandleAdditionalData

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

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

#### Event Type

[Action<AdditionalSpeechData>](#)

# Delegate DialogueInstance.BeforeSpeechEventHandler

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

Assembly: Assembly-CSharp-firstpass.dll

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

## Parameters

**speaker** Person

**speech** [string](#)

**options** [List](#)<Option>

# Class DialogueOptionText

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

Assembly: Assembly-CSharp-firstpass.dll

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

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

## Inheritance

[object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← DialogueOptionText

## Inherited Members

MonoBehaviour.IsInvoking() , MonoBehaviour.CancelInvoke() , [MonoBehaviour.Invoke\(string, float\)](#) ,  
[MonoBehaviour.InvokeRepeating\(string, float, float\)](#) , [MonoBehaviour.CancelInvoke\(string\)](#) ,  
[MonoBehaviour.IsInvoking\(string\)](#) , [MonoBehaviour.StartCoroutine\(string\)](#) ,  
[MonoBehaviour.StartCoroutine\(string, object\)](#) , [MonoBehaviour.StartCoroutine\(IEnumerator\)](#) ,  
[MonoBehaviour.StartCoroutine\\_Auto\(IEnumerator\)](#) , [MonoBehaviour.StopCoroutine\(IEnumerator\)](#) ,  
MonoBehaviour.StopCoroutine(Coroutine) , [MonoBehaviour.StopCoroutine\(string\)](#) ,  
MonoBehaviour.StopAllCoroutines() , [MonoBehaviour.print\(object\)](#) ,  
MonoBehaviour.destroyCancellationToken , MonoBehaviour.useGUILayout ,  
MonoBehaviour.RunWithEditMode , Behaviour.enabled , Behaviour.isActiveAndEnabled ,  
[Component.GetComponent\(Type\)](#) , Component.GetComponent<T>() ,  
[Component.TryGetComponent\(Type, out Component\)](#) , Component.TryGetComponent<T>(out T) ,  
[Component.GetComponent\(string\)](#) , [Component.GetComponentInChildren\(Type, bool\)](#) ,  
[Component.GetComponentInChildren\(Type\)](#) , [Component.GetComponentInChildren<T>\(bool\)](#) ,  
Component.GetComponentInChildren<T>() , [Component.GetComponentsInChildren\(Type, bool\)](#) ,  
[Component.GetComponentsInChildren\(Type\)](#) , [Component.GetComponentsInChildren<T>\(bool\)](#) ,  
[Component.GetComponentsInChildren<T>\(bool, List<T>\)](#) ,  
Component.GetComponentsInChildren<T>() , [Component.GetComponentsInChildren<T>\(List<T>\)](#) ,  
[Component.GetComponentInParent\(Type, bool\)](#) , [Component.GetComponentInParent\(Type\)](#) ,  
[Component.GetComponentInParent<T>\(bool\)](#) , Component.GetComponentInParent<T>() ,  
[Component.GetComponentsInParent\(Type, bool\)](#) , [Component.GetComponentsInParent\(Type\)](#) ,  
[Component.GetComponentsInParent<T>\(bool\)](#) ,  
[Component.GetComponentsInParent<T>\(bool, List<T>\)](#) , Component.GetComponentsInParent<T>() ,  
[Component.GetComponents\(Type\)](#) , [Component.GetComponents\(Type, List<Component>\)](#) ,  
[Component.GetComponents<T>\(List<T>\)](#) , Component.GetComponents<T>() ,  
[Component.CompareTag\(string\)](#) ,

[Component.SendMessageUpwards\(string, object, SendMessageOptions\)](#) ,  
[Component.SendMessageUpwards\(string, object\)](#) , [Component.SendMessageUpwards\(string\)](#) ,  
[Component.SendMessageUpwards\(string, SendMessageOptions\)](#) ,  
[Component.SendMessage\(string, object\)](#) , [Component.SendMessage\(string\)](#) ,  
[Component.SendMessage\(string, object, SendMessageOptions\)](#) ,  
[Component.SendMessage\(string, SendMessageOptions\)](#) ,  
[Component.BroadcastMessage\(string, object, SendMessageOptions\)](#) ,  
[Component.BroadcastMessage\(string, object\)](#) , [Component.BroadcastMessage\(string\)](#) ,  
[Component.BroadcastMessage\(string, SendMessageOptions\)](#) , Component.transform ,  
Component.gameObject , Component.tag , Object.GetInstanceID() , Object.GetHashCode() ,  
[Object.Equals\(object\)](#) , Object.Instantiate(Object, Vector3, Quaternion) ,  
Object.Instantiate(Object, Vector3, Quaternion, Transform) , Object.Instantiate(Object) ,  
Object.Instantiate(Object, Transform) , [Object.Instantiate\(Object, Transform, bool\)](#) ,  
Object.Instantiate<T>(T) , Object.Instantiate<T>(T, Vector3, Quaternion) ,  
Object.Instantiate<T>(T, Vector3, Quaternion, Transform) , Object.Instantiate<T>(T, Transform) ,  
[Object.Instantiate<T>\(T, Transform, bool\)](#) , [Object.Destroy\(Object, float\)](#) , Object.Destroy(Object) ,  
[Object.DestroyImmediate\(Object, bool\)](#) , Object.DestroyImmediate(Object) ,  
[Object.FindObjectsOfType\(Type\)](#) , [Object.FindObjectsOfType\(Type, bool\)](#) ,  
[Object.FindObjectsByType\(Type, FindObjectsSortMode\)](#) ,  
[Object.FindObjectsByType\(Type, FindObjectsInactive, FindObjectsSortMode\)](#) ,  
Object.DontDestroyOnLoad(Object) , [Object.DestroyObject\(Object, float\)](#) ,  
Object.DestroyObject(Object) , [Object.FindSceneObjectsOfType\(Type\)](#) ,  
[Object.FindObjectsOfTypeIncludingAssets\(Type\)](#) , Object.FindObjectsOfType<T>() ,  
Object.FindObjectsByType<T>(FindObjectsSortMode) , [Object.FindObjectsOfType<T>\(bool\)](#) ,  
Object.FindObjectsByType<T>(FindObjectsInactive, FindObjectsSortMode) ,  
Object.FindObjectOfType<T>() , [Object.FindObjectOfType<T>\(bool\)](#) ,  
Object.FindFirstObjectByType<T>() , Object.FindAnyObjectByType<T>() ,  
Object.FindFirstObjectByType<T>(FindObjectsInactive) ,  
Object.FindAnyObjectByType<T>(FindObjectsInactive) , [Object.FindObjectsOfTypeAll\(Type\)](#) ,  
[Object.FindObjectOfType\(Type\)](#) , [Object.FindFirstObjectByType\(Type\)](#) ,  
[Object.FindAnyObjectByType\(Type\)](#) , [Object.FindObjectOfType\(Type, bool\)](#) ,  
[Object.FindFirstObjectByType\(Type, FindObjectsInactive\)](#) ,  
[Object.FindAnyObjectByType\(Type, FindObjectsInactive\)](#) , Object.ToString() , Object.name ,  
Object.hideFlags , [object.Equals\(object, object\)](#) , [object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) ,  
[object.ReferenceEquals\(object, object\)](#)

## Methods

Initialize(int, string)

Sets the index and the text of this option.

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

Parameters

optionIndex [int](#)

text [string](#)

## OnClick()

Calls [OnClickAction](#).

```
public void OnClick()
```

## Events

### OnClickAction

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

Event Type

[Action](#)<[int](#)>

# Class DialoguePlayer

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

Assembly: Assembly-CSharp-firstpass.dll

Lets you progress in a dialogue easily.

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

Inheritance

[object](#) ← DialoguePlayer

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

## Constructors

### DialoguePlayer(Dialogue)

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

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

Parameters

[dialogue](#) [Dialogue](#)

The original dialogue to clone from.

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

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

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

## Parameters

### dialogue [Dialogue](#)

The original dialogue to clone from.

### overridePeople [List](#) <Person>

The list of new people.

## Properties

### AdditionalSpeechData

Additional data of the current node.

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

### Property Value

#### [AdditionalSpeechData](#)

### ClonedDialogue

The dialogue cloned from the original one from constructor.

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

### Property Value

#### [Dialogue](#)

### HasOptions

Use to check if current node is a [FastSpeechNode](#) or not.

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

Property Value

[bool](#)

## HasPerson

Use to check if current node [PersonDependent](#) or not.

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

Property Value

[bool](#)

## HasSpeech

Use to check if current node is a [IContainSpeech](#) or not.

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

Property Value

[bool](#)

## Options

Options of the current node, if there is any.

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

Property Value

[List](#)<[Option](#)>

## Speaker

Person who speaks.

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

Property Value

Person

## Speech

Speech of the current node.

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

Property Value

[string](#) ↗

## State

Current state of the player.

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

Property Value

[DialoguePlayer.PlayerState](#)

## Methods

### Continue(params object[])

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

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

## Parameters

`passData object[]`

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.

```
public void TeleportToRoot()
```

## Events

### OnContinue

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

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

### Event Type

[Action](#)<[DialoguePlayer](#).[PlayerState](#)>

# Enum DialoguePlayer.PlayerState

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

Assembly: Assembly-CSharp-firstpass.dll

Shows what state the dialogue player is in.

```
public enum DialoguePlayer.PlayerState
```

## Fields

NoSpeech = 0

WaitingForOption = 1

WaitingForSkip = 2

WillExit = 3

# Class DialogueSoundsPlayer

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

Assembly: Assembly-CSharp-firstpass.dll

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

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

## Inheritance

[Object](#) ← Object ← Component ← Behaviour ← MonoBehaviour ← [DialogueExtensionBase](#) ← DialogueSoundsPlayer

## Inherited Members

[DialogueExtensionBase.m\\_instance](#) , MonoBehaviour.IsInvoking() , MonoBehaviour.CancelInvoke() ,  
[MonoBehaviour.Invoke\(string, float\)](#) , [MonoBehaviour.InvokeRepeating\(string, float, float\)](#) ,  
[MonoBehaviour.CancelInvoke\(string\)](#) , [MonoBehaviour.IsInvoking\(string\)](#) ,  
[MonoBehaviour.StartCoroutine\(string\)](#) , [MonoBehaviour.StartCoroutine\(string, object\)](#) ,  
[MonoBehaviour.StartCoroutine\(IEnumerator\)](#) , [MonoBehaviour.StartCoroutine\\_Auto\(IEnumerator\)](#) ,  
[MonoBehaviour.StopCoroutine\(IEnumerator\)](#) , MonoBehaviour.StopCoroutine(Coroutine) ,  
[MonoBehaviour.StopCoroutine\(string\)](#) , MonoBehaviour.StopAllCoroutines() ,  
[MonoBehaviour.print\(object\)](#) , MonoBehaviour.destroyCancellationToken ,  
MonoBehaviour.useGUILayout , MonoBehaviour.runInEditMode , Behaviour.enabled ,  
Behaviour.isActiveAndEnabled , [Component.GetComponent\(Type\)](#) , Component.GetComponent<T>() ,  
[Component.TryGetComponent\(Type, out Component\)](#) , Component.TryGetComponent<T>(out T) ,  
[Component.GetComponent\(string\)](#) , [Component.GetComponentInChildren\(Type, bool\)](#) ,  
[Component.GetComponentInChildren\(Type\)](#) , [Component.GetComponentInChildren<T>\(bool\)](#) ,  
Component.GetComponentInChildren<T>() , [Component.GetComponentsInChildren\(Type, bool\)](#) ,  
[Component.GetComponentsInChildren\(Type\)](#) , [Component.GetComponentsInChildren<T>\(bool\)](#) ,  
[Component.GetComponentsInChildren<T>\(bool, List<T>\)](#) ,  
Component.GetComponentsInChildren<T>() , [Component.GetComponentsInChildren<T>\(List<T>\)](#) ,  
[Component.GetComponentInParent\(Type, bool\)](#) , [Component.GetComponentInParent\(Type\)](#) ,  
[Component.GetComponentInParent<T>\(bool\)](#) , Component.GetComponentInParent<T>() ,  
[Component.GetComponentsInParent\(Type, bool\)](#) , [Component.GetComponentsInParent\(Type\)](#) ,  
[Component.GetComponentsInParent<T>\(bool\)](#) ,  
Component.GetComponentsInParent<T>(bool, List<T>) , Component.GetComponentInParent<T>()

[Component.GetComponents\(Type\)](#) , [Component.GetComponents\(Type, List<Component>\)](#) ,  
[Component.GetComponents<T>\(List<T>\)](#) , Component.GetComponents<T>() ,  
[Component.CompareTag\(string\)](#) ,  
[Component.SendMessageUpwards\(string, object, SendMessageOptions\)](#) ,  
[Component.SendMessageUpwards\(string, object\)](#) , [Component.SendMessageUpwards\(string\)](#) ,  
[Component.SendMessageUpwards\(string, SendMessageOptions\)](#) ,  
[Component.SendMessage\(string, object\)](#) , [Component.SendMessage\(string\)](#) ,  
[Component.SendMessage\(string, object, SendMessageOptions\)](#) ,  
[Component.SendMessage\(string, SendMessageOptions\)](#) ,  
[Component.BroadcastMessage\(string, object, SendMessageOptions\)](#) ,  
[Component.BroadcastMessage\(string, object\)](#) , [Component.BroadcastMessage\(string\)](#) ,  
[Component.BroadcastMessage\(string, SendMessageOptions\)](#) , Component.transform ,  
Component.gameObject , Component.tag , Object.GetInstanceID() , Object.GetHashCode()  
[Object.Equals\(object\)](#) , Object.Instantiate(Object, Vector3, Quaternion) ,  
Object.Instantiate(Object, Vector3, Quaternion, Transform) , Object.Instantiate(Object) ,  
Object.Instantiate(Object, Transform) , [Object.Instantiate\(Object, Transform, bool\)](#) ,  
Object.Instantiate<T>(T) , Object.Instantiate<T>(T, Vector3, Quaternion) ,  
Object.Instantiate<T>(T, Vector3, Quaternion, Transform) , Object.Instantiate<T>(T, Transform) ,  
[Object.Instantiate<T>\(T, Transform, bool\)](#) , [Object.Destroy\(Object, float\)](#) , Object.Destroy(Object) ,  
[Object.DestroyImmediate\(Object, bool\)](#) , Object.DestroyImmediate(Object) ,  
[Object.FindObjectsOfType\(Type\)](#) , [Object.FindObjectsOfType\(Type, bool\)](#) ,  
[Object.FindObjectsByType\(Type, FindObjectsSortMode\)](#) ,  
[Object.FindObjectsByType\(Type, FindObjectsInactive, FindObjectsSortMode\)](#) ,  
Object.DontDestroyOnLoad(Object) , [Object.DestroyObject\(Object, float\)](#) ,  
Object.DestroyObject(Object) , [Object.FindSceneObjectsOfType\(Type\)](#) ,  
[Object.FindObjectsOfTypeIncludingAssets\(Type\)](#) , Object.FindObjectsOfType<T>() ,  
Object.FindObjectsByType<T>(FindObjectsSortMode) , [Object.FindObjectsOfType<T>\(bool\)](#) ,  
Object.FindObjectsByType<T>(FindObjectInactive, FindObjectsSortMode) ,  
Object.FindObjectOfType<T>() , [Object.FindObjectOfType<T>\(bool\)](#) ,  
Object.FindFirstObjectOfType<T>() , Object.FindAnyObjectOfType<T>() ,  
Object.FindFirstObjectOfType<T>(FindObjectInactive) ,  
Object.FindAnyObjectOfType<T>(FindObjectInactive) , [Object.FindObjectsOfTypeAll\(Type\)](#) ,  
[Object.FindObjectOfType\(Type\)](#) , [Object.FindFirstObjectOfType\(Type\)](#) ,  
[Object.FindAnyObjectOfType\(Type\)](#) , [Object.FindObjectOfType\(Type, bool\)](#) ,  
[Object.FindFirstObjectOfType\(Type, FindObjectsInactive\)](#) ,  
[Object.FindAnyObjectOfType\(Type, FindObjectsInactive\)](#) , Object.ToString() , Object.name ,  
Object.hideFlags , [object.Equals\(object, object\)](#) , [object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) ,  
[object.ReferenceEquals\(object, object\)](#)

# Methods

## OnHandleAdditionalData(AdditionalSpeechData)

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

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

### Parameters

data [AdditionalSpeechData](#)

# Namespace com.absence.dialoguesystem.editor

## Classes

[DialogueEditorWindow](#)

[DialogueGraphView](#)

[DialogueGraphView.UxmlFactory](#)

[InspectorView](#)

[InspectorView.UxmlFactory](#)

[NodeView](#)

[VariableBankCreationHandler](#)

# Class DialogueEditorWindow

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

Assembly: Assembly-CSharp-Editor-firstpass.dll

```
public class DialogueEditorWindow : EditorWindow
```

## Inheritance

[Object](#) ← Object ← ScriptableObject ← EditorWindow ← DialogueEditorWindow

## Inherited Members

EditorWindow.BeginWindows() , EditorWindow.EndWindows() ,  
EditorWindow.ShowNotification(GUIContent) , [EditorWindow.ShowNotification\(GUIContent, double\)](#) ,  
EditorWindow.RemoveNotification() , EditorWindow.ShowTab() , EditorWindow.Focus() ,  
EditorWindow.ShowUtility() , EditorWindow.ShowPopup() , EditorWindow.ShowModalUtility() ,  
EditorWindow.ShowAsDropDown(Rect, Vector2) , EditorWindow.Show() , [EditorWindow.Show\(bool\)](#) ,  
EditorWindow.ShowAuxWindow() , EditorWindow.ShowModal() ,  
[EditorWindow.GetWindow\(Type, bool, string, bool\)](#) , [EditorWindow.GetWindow\(Type, bool, string\)](#) ,  
[EditorWindow.GetWindow\(Type, bool\)](#) , [EditorWindow.GetWindow\(Type\)](#) ,  
[EditorWindow.GetWindowWithRect\(Type, Rect, bool, string\)](#) ,  
[EditorWindow.GetWindowWithRect\(Type, Rect, bool\)](#) ,  
[EditorWindow.GetWindowWithRect\(Type, Rect\)](#) , EditorWindow.GetWindow<T>() ,  
[EditorWindow.GetWindow<T>\(bool\)](#) , [EditorWindow.GetWindow<T>\(bool, string\)](#) ,  
[EditorWindow.GetWindow<T>\(string\)](#) , [EditorWindow.GetWindow<T>\(string, bool\)](#) ,  
[EditorWindow.GetWindow<T>\(bool, string, bool\)](#) , [EditorWindow.GetWindow<T>\(params Type\[\]\)](#) ,  
[EditorWindow.GetWindow<T>\(string, params Type\[\]\)](#) ,  
[EditorWindow.GetWindow<T>\(string, bool, params Type\[\]\)](#) ,  
[EditorWindow.CreateWindow<T>\(params Type\[\]\)](#) ,  
[EditorWindow.CreateWindow<T>\(string, params Type\[\]\)](#) , EditorWindow.HasOpenInstances<T>() ,  
[EditorWindow.FocusWindowIfItsOpen\(Type\)](#) , EditorWindow.FocusWindowIfItsOpen<T>() ,  
EditorWindow.GetWindowWithRect<T>(Rect) , [EditorWindow.GetWindowWithRect<T>\(Rect, bool\)](#) ,  
[EditorWindow.GetWindowWithRect<T>\(Rect, bool, string\)](#) ,  
[EditorWindow.GetWindowWithRect<T>\(Rect, bool, string, bool\)](#) , EditorWindow.SaveChanges() ,  
EditorWindow.DiscardChanges() , EditorWindow.Close() , EditorWindow.Repaint() ,  
EditorWindow.SendEvent(Event) , EditorWindow.GetExtraPaneTypes() ,  
[EditorWindow.TryGetOverlay\(string, out Overlay\)](#) , EditorWindow.OnBackingScaleFactorChanged() ,  
EditorWindow.dataModeController , EditorWindow.rootVisualElement , EditorWindow.overlayCanvas ,  
EditorWindow.wantsMouseMove , EditorWindow.wantsMouseEnterLeaveWindow ,  
EditorWindow.wantsLessLayoutEvents , EditorWindow.autoRepaintOnSceneChange ,

EditorWindow.maximized , EditorWindow.hasFocus , EditorWindow.docked ,  
EditorWindow.focusedWindow , EditorWindow.mouseOverWindow ,  
EditorWindow.hasUnsavedChanges , EditorWindow.saveChangesMessage , EditorWindow.minSize ,  
EditorWindow.maxSize , EditorWindow.title , EditorWindow.titleContent , EditorWindow.depthBufferBits ,  
EditorWindow.antiAlias , EditorWindow.position , ScriptableObject.SetDirty() ,  
[ScriptableObject.CreateInstance\(string\)](#) , [ScriptableObject.CreateInstance\(Type\)](#) ,  
ScriptableObject.CreateInstance<T>() , Object.GetInstanceId() , Object.GetHashCode() ,  
[Object.Equals\(object\)](#) , Object.Instantiate(Object, Vector3, Quaternion) ,  
Object.Instantiate(Object, Vector3, Quaternion, Transform) , Object.Instantiate(Object) ,  
Object.Instantiate(Object, Transform) , [Object.Instantiate\(Object, Transform, bool\)](#) ,  
Object.Instantiate<T>(T) , Object.Instantiate<T>(T, Vector3, Quaternion) ,  
Object.Instantiate<T>(T, Vector3, Quaternion, Transform) , Object.Instantiate<T>(T, Transform) ,  
[Object.Instantiate<T>\(T, Transform, bool\)](#) , [Object.Destroy\(Object, float\)](#) , Object.Destroy(Object) ,  
[Object.DestroyImmediate\(Object, bool\)](#) , Object.DestroyImmediate(Object) ,  
[Object.FindObjectsOfType\(Type\)](#) , [Object.FindObjectsOfType\(Type, bool\)](#) ,  
[Object.FindObjectsByType\(Type, FindObjectsSortMode\)](#) ,  
[Object.FindObjectsByType\(Type, FindObjectsInactive, FindObjectsSortMode\)](#) ,  
Object.DontDestroyOnLoad(Object) , [Object.DestroyObject\(Object, float\)](#) ,  
Object.DestroyObject(Object) , [Object.FindSceneObjectsOfType\(Type\)](#) ,  
[Object.FindObjectsOfTypeIncludingAssets\(Type\)](#) , Object.FindObjectsOfType<T>() ,  
Object.FindObjectsByType<T>(FindObjectsSortMode) , [Object.FindObjectsOfType<T>\(bool\)](#) ,  
Object.FindObjectsByType<T>(FindObjectsInactive, FindObjectsSortMode) ,  
Object.FindObjectOfType<T>() , [Object.FindObjectOfType<T>\(bool\)](#) ,  
Object.FindFirstObjectByType<T>() , Object.FindAnyObjectByType<T>() ,  
Object.FindFirstObjectByType<T>(FindObjectsInactive) ,  
Object.FindAnyObjectByType<T>(FindObjectsInactive) , [Object.FindObjectsOfTypeAll\(Type\)](#) ,  
[Object.FindObjectOfType\(Type\)](#) , [Object.FindFirstObjectByType\(Type\)](#) ,  
[Object.FindAnyObjectByType\(Type\)](#) , [Object.FindObjectOfType\(Type, bool\)](#) ,  
[Object.FindFirstObjectByType\(Type, FindObjectsInactive\)](#) ,  
[Object.FindAnyObjectByType\(Type, FindObjectsInactive\)](#) , Object.ToString() , Object.name ,  
Object.hideFlags , [object.Equals\(object, object\)](#) , [object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) ,  
[object.ReferenceEquals\(object, object\)](#)

## Methods

### CreateGUI()

```
public void CreateGUI()
```

## FrameToNode(Node)

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

### Parameters

node [Node](#)

## OnOpenAsset(int, int)

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

### Parameters

instanceId [int](#)

line [int](#)

### Returns

[bool](#)

## OpenWindow()

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

## SelectNode(Node)

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

### Parameters

node [Node](#)



# Class DialogueGraphView

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

Assembly: Assembly-CSharp-Editor-firstpass.dll

```
public class DialogueGraphView : GraphView, IEventHandler, IResolvedStyle, ITransform,  
ITransitionAnimations, IExperimentalFeatures, IVisualElementScheduler, ISelection
```

## Inheritance

[object](#) ← CallbackEventHandler ← Focusable ← VisualElement ← GraphView ← DialogueGraphView

## Implements

IEventHandler, IResolvedStyle, ITransform, ITransitionAnimations, IExperimentalFeatures, IVisualElementScheduler, ISelection

## Inherited Members

GraphView.ports , GraphView.UpdateViewTransform(Vector3, Vector3) ,  
GraphView.GetPortCenterOverride(Port, out Vector2) , [GraphView.AddLayer\(int\)](#) ,  
[GraphView.GetElementByGuid\(string\)](#) , [GraphView.GetNodeByGuid\(string\)](#) ,  
[GraphView.GetPortByGuid\(string\)](#) , [GraphView.GetEdgeByGuid\(string\)](#) ,  
[GraphView.SetupZoom\(float, float\)](#) , [GraphView.SetupZoom\(float, float, float, float\)](#) ,  
GraphView.ValidateTransform() , GraphView.AddToSelection(ISelectable) ,  
GraphView.RemoveFromSelection(ISelectable) , GraphView.ClearSelection() ,  
GraphView.ExecuteDefaultActionAtTarget(EventBase) , GraphView.ExecuteDefaultAction(EventBase) ,  
[GraphView.CollectElements\(IEnumerable<GraphElement>, HashSet<GraphElement>, Func<GraphElement, bool>\)](#) ,  
[GraphView.CollectCopyableGraphElements\(IEnumerable<GraphElement>, HashSet<GraphElement>\)](#) ,  
GraphView.CopySelectionCallback() , GraphView.CutSelectionCallback() , GraphView.PasteCallback() ,  
GraphView.DuplicateSelectionCallback() , GraphView.DeleteSelectionCallback(GraphView.AskUser) ,  
[GraphView.SerializeGraphElements\(IEnumerable<GraphElement>\)](#) ,  
[GraphView.CanPasteSerializedData\(string\)](#) , [GraphView.UnserializeAndPasteOperation\(string, string\)](#) ,  
[GraphView.DeleteSelectionOperation\(string, GraphView.AskUser\)](#) ,  
GraphView.AddElement(GraphElement) , GraphView.RemoveElement(GraphElement) ,  
GraphView.DeleteSelection() , [GraphView.DeleteElements\(IEnumerable<GraphElement>\)](#) ,  
GraphView.FrameAll() , GraphView.FrameSelection() , GraphView.FrameOrigin() , GraphView.FramePrev() ,  
GraphView.FrameNext() , [GraphView.FramePrev\(Func<GraphElement, bool>\)](#) ,  
[GraphView.FrameNext\(Func<GraphElement, bool>\)](#) , GraphView.CalculateRectToFitAll(VisualElement) ,  
[GraphView.CalculateFrameTransform\(Rect, Rect, int, out Vector3, out Vector3\)](#) ,  
GraphView.GetBlackboard() , GraphView.ReleaseBlackboard(Blackboard) ,

GraphView.CreatePlacematContainer() , GraphView.nodeCreationRequest ,  
GraphView.graphViewChanged , GraphView.groupTitleChanged , GraphView.elementsAddedToGroup ,  
GraphView.elementsRemovedFromGroup , GraphView.elementsInsertedToStackNode ,  
GraphView.elementsRemovedFromStackNode , GraphView.elementResized ,  
GraphView.viewTransformChanged , GraphView.supportsWindowedBlackboard ,  
GraphView.contentViewContainer , GraphView.viewport , GraphView.viewTransform ,  
GraphView.isReframable , GraphView.contentContainer , GraphView.placematContainer ,  
GraphView.graphElements , GraphView.nodes , GraphView.edges , GraphView.minScale ,  
GraphView.maxScale , GraphView.scaleStep , GraphView.referenceScale , GraphView.scale ,  
GraphView.zoomerMaxElementCountWithPixelCacheRegen , GraphView.selection ,  
GraphView.canCopySelection , GraphView.canCutSelection , GraphView.canPaste ,  
GraphView.canDuplicateSelection , GraphView.canDeleteSelection , GraphView.serializeGraphElements ,  
GraphView.canPasteSerializedData , GraphView.unserializeAndPaste , GraphView.deleteSelection ,  
VisualElement.disabledUssClassName , VisualElement.Focus() , VisualElement.SendEvent(EventBase) ,  
[VisualElement.SetEnabledFromHierarchy\(bool\)](#) , [VisualElement.SetEnabled\(bool\)](#) ,  
VisualElement.MarkDirtyRepaint() , VisualElement.ContainsPoint(Vector2) , VisualElement.Overlaps(Rect) ,  
[VisualElement.DoMeasure\(float, VisualElement.MeasureMode, float, VisualElement.MeasureMode\)](#) ,  
VisualElement.ToString() , VisualElement.GetClasses() , VisualElement.ClearClassList() ,  
[VisualElement.AddToList\(string\)](#) , [VisualElement.RemoveFromList\(string\)](#) ,  
[VisualElement.ToggleInclassList\(string\)](#) , [VisualElement.EnableInclassList\(string, bool\)](#) ,  
[VisualElement.classListContains\(string\)](#) , VisualElement.FindAncestorUserData() ,  
VisualElement.Add(VisualElement) , [VisualElement.Insert\(int, VisualElement\)](#) ,  
VisualElement.Remove(VisualElement) , [VisualElement.RemoveAt\(int\)](#) , VisualElement.Clear() ,  
[VisualElement.ElementAt\(int\)](#) , VisualElement.IndexOf(VisualElement) , VisualElement.Children() ,  
[VisualElement.Sort\(Comparison<VisualElement>\)](#) , VisualElement.BringToFront() ,  
VisualElement.SendToBack() , VisualElement.PlaceBehind(VisualElement) ,  
VisualElement.PlaceInFront(VisualElement) , VisualElement.RemoveFromHierarchy() ,  
VisualElement.GetFirstOfType<T>() , VisualElement.GetFirstAncestorOfType<T>() ,  
VisualElement.Contains(VisualElement) , VisualElement.FindCommonAncestor(VisualElement) ,  
VisualElement.resolvedStyle , VisualElement.viewDataKey , VisualElement.userData ,  
VisualElement.canGrabFocus , VisualElement.focusController , VisualElement.usageHints ,  
VisualElement.transform , VisualElement.layout , VisualElement.contentRect , VisualElement.paddingRect ,  
VisualElement.worldBound , VisualElement.localBound , VisualElement.worldTransform ,  
VisualElement.pickingMode , VisualElement.name , VisualElement.enabledInHierarchy ,  
VisualElement.enabledSelf , VisualElement.languageDirection , VisualElement.visible ,  
VisualElement.generateVisualContent , VisualElement.experimental , VisualElement.hierarchy ,  
VisualElement.cacheAsBitmap , VisualElement.parent , VisualElement.panel ,  
VisualElement.visualTreeAssetSource , [VisualElement.this\[int\]](#) , VisualElement.childCount ,  
VisualElement.schedule , VisualElement.style , VisualElement.customStyle , VisualElement.styleSheets ,  
VisualElement.tooltip , Focusable.Blur() , Focusable.focusable , Focusable.tabIndex ,

```
Focusable.delegatesFocus ,  
CallbackEventHandler.RegisterCallback<TEventType>(EventCallback<TEventType>, TrickleDown) ,  
CallbackEventHandler.RegisterCallback<TEventType, TUserArgsType>(EventCallback<TEventType,  
TUserArgsType>, TUserArgsType, TrickleDown) ,  
CallbackEventHandler.UnregisterCallback<TEventType>(EventCallback<TEventType>, TrickleDown) ,  
CallbackEventHandler.UnregisterCallback<TEventType, TUserArgsType>(EventCallback<TEventType,  
TUserArgsType>, TrickleDown) ,  
CallbackEventHandler.HandleEvent(EventBase) , CallbackEventHandler.HasTrickleDownHandlers() ,  
CallbackEventHandler.HasBubbleUpHandlers() , object.Equals\(object\) , object.Equals\(object, object\) ,  
object.GetHashCode\(\) , object.GetType\(\) , object.MemberwiseClone\(\) ,  
object.ReferenceEquals\(object, object\)
```

## Constructors

### DialogueGraphView()

```
public DialogueGraphView()
```

## Methods

### BuildContextualMenu(ContextualMenuPopulateEvent)

Add menu items to the contextual menu.

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

#### Parameters

**evt** ContextualMenuPopulateEvent

The event holding the menu to populate.

### FindNodeView(Node)

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

Parameters

**node** [Node](#)

Returns

[NodeView](#)

## GetCompatiblePorts(Port, NodeAdapter)

Get all ports compatible with given port.

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

Parameters

**startPort** Port

Start port to validate against.

**nodeAdapter** NodeAdapter

Node adapter.

Returns

[List](#) <Port>

List of compatible ports.

## Refresh()

```
public void Refresh()
```

## Events

### OnNodeSelected

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

Event Type

[Action](#) <[NodeView](#)>

## OnPopulateView

```
public event Action OnPopulateView
```

Event Type

[Action](#)

# Class DialogueGraphView.UxmlFactory

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

Assembly: Assembly-CSharp-Editor-firstpass.dll

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

## Inheritance

[object](#) ← BaseUxmlFactory<[DialogueGraphView](#), VisualElement.UxmlTraits> ←  
UxmlFactory<[DialogueGraphView](#), VisualElement.UxmlTraits> ← DialogueGraphView.UxmlFactory

## Implements

IUxmlFactory, IBaseUxmlFactory

## Inherited Members

UxmlFactory<DialogueGraphView, VisualElement.UxmlTraits>.Create(IUxmlAttributes, CreationContext) ,  
BaseUxmlFactory<DialogueGraphView, VisualElement.UxmlTraits>.AcceptsAttributeBag(IUxmlAttributes, CreationContext) ,  
BaseUxmlFactory<DialogueGraphView, VisualElement.UxmlTraits>.uxmlName ,  
BaseUxmlFactory<DialogueGraphView, VisualElement.UxmlTraits>.uxmlNamespace ,  
BaseUxmlFactory<DialogueGraphView, VisualElement.UxmlTraits>.uxmlQualifiedName ,  
BaseUxmlFactory<DialogueGraphView, VisualElement.UxmlTraits>.uxmlType ,  
BaseUxmlFactory<DialogueGraphView, VisualElement.UxmlTraits>.canHaveAnyAttribute ,  
BaseUxmlFactory<DialogueGraphView, VisualElement.UxmlTraits>.uxmlAttributesDescription ,  
BaseUxmlFactory<DialogueGraphView, VisualElement.UxmlTraits>.uxmlChildElementsDescription ,  
BaseUxmlFactory<DialogueGraphView, VisualElement.UxmlTraits>.substituteForTypeName ,  
BaseUxmlFactory<DialogueGraphView, VisualElement.UxmlTraits>.substituteForTypeNamespace ,  
BaseUxmlFactory<DialogueGraphView, VisualElement.UxmlTraits>.substituteForTypeQualifiedName ,  
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

# Class InspectorView

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

Assembly: Assembly-CSharp-Editor-firstpass.dll

```
public class InspectorView : VisualElement, IEventHandler, IResolvedStyle, ITransform,  
ITransitionAnimations, IExperimentalFeatures, IVisualElementScheduler
```

## Inheritance

[object](#) ← CallbackEventHandler ← Focusable ← VisualElement ← InspectorView

## Implements

IEventHandler, IResolvedStyle, ITransform, ITransitionAnimations, IExperimentalFeatures, IVisualElementScheduler

## Inherited Members

VisualElement.disabledUssClassName , VisualElement.ExecuteDefaultAction(EventBase) ,  
VisualElement.Focus() , VisualElement.SendEvent(EventBase) ,  
[VisualElement.SetEnabledFromHierarchy\(bool\)](#) , [VisualElement.SetEnabled\(bool\)](#) ,  
VisualElement.MarkDirtyRepaint() , VisualElement.ContainsPoint(Vector2) , VisualElement.Overlaps(Rect) ,  
[VisualElement.DoMeasure\(float, VisualElement.MeasureMode, float, VisualElement.MeasureMode\)](#) ,  
VisualElement.ToString() , VisualElement.GetClasses() , VisualElement.ClearClassList() ,  
[VisualElement.AddToClassList\(string\)](#) , [VisualElement.RemoveFromClassList\(string\)](#) ,  
[VisualElement.ToggleInClassList\(string\)](#) , [VisualElement.EnableInClassList\(string, bool\)](#) ,  
[VisualElement.ClassListContains\(string\)](#) , VisualElement.FindAncestorUserData() ,  
VisualElement.Add(VisualElement) , [VisualElement.Insert\(int, VisualElement\)](#) ,  
VisualElement.Remove(VisualElement) , [VisualElement.RemoveAt\(int\)](#) , VisualElement.Clear() ,  
[VisualElement.ElementAt\(int\)](#) , VisualElement.IndexOf(VisualElement) , VisualElement.Children() ,  
[VisualElement.Sort\(Comparison<VisualElement>\)](#) , VisualElement.BringToFront() ,  
VisualElement.SendToBack() , VisualElement.PlaceBehind(VisualElement) ,  
VisualElement.PlaceInFront(VisualElement) , VisualElement.RemoveFromHierarchy() ,  
VisualElement.GetFirstOfType<T>() , VisualElement.GetFirstAncestorOfType<T>() ,  
VisualElement.Contains(VisualElement) , VisualElement.FindCommonAncestor(VisualElement) ,  
VisualElement.resolvedStyle , VisualElement.viewDataKey , VisualElement.userData ,  
VisualElement.canGrabFocus , VisualElement.focusController , VisualElement.usageHints ,  
VisualElement.transform , VisualElement.layout , VisualElement.contentRect , VisualElement.paddingRect ,  
VisualElement.worldBound , VisualElement.localBound , VisualElement.worldTransform ,  
VisualElement.pickingMode , VisualElement.name , VisualElement.enabledInHierarchy ,  
VisualElement.enabledSelf , VisualElement.languageDirection , VisualElement.visible ,

VisualElement.generateVisualContent , VisualElement.experimental , VisualElement.hierarchy ,  
VisualElement.cacheAsBitmap , VisualElement.parent , VisualElement.panel ,  
VisualElement.contentContainer , VisualElement.visualTreeAssetSource , [VisualElement.this\[int\]](#) ,  
VisualElement.childCount , VisualElement.schedule , VisualElement.style , VisualElement.customStyle ,  
VisualElement.styleSheets , VisualElement.tooltip , Focusable.Blur() , Focusable.focusable ,  
Focusable.tabIndex , Focusable.delegatesFocus ,  
CallbackEventHandler.RegisterCallback<TEventType>(EventCallback<TEventType>, TrickleDown) ,  
CallbackEventHandler.RegisterCallback<TEventType, TUserArgsType>(EventCallback<TEventType,  
TUserArgsType>, TUserArgsType, TrickleDown) ,  
CallbackEventHandler.UnregisterCallback<TEventType>(EventCallback<TEventType>, TrickleDown) ,  
CallbackEventHandler.UnregisterCallback<TEventType, TUserArgsType>(EventCallback<TEventType,  
TUserArgsType>, TrickleDown) ,  
CallbackEventHandler.HandleEvent(EventBase) , CallbackEventHandler.HasTrickleDownHandlers() ,  
CallbackEventHandler.HasBubbleUpHandlers() ,  
CallbackEventHandler.ExecuteDefaultActionAtTarget(EventBase) , [object.Equals\(object\)](#) ,  
[object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

## Constructors

### InspectorView()

```
public InspectorView()
```

# Class InspectorView.UxmlFactory

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

Assembly: Assembly-CSharp-Editor-firstpass.dll

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

## Inheritance

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

## Implements

IUxmlFactory, IBaseUxmlFactory

## Inherited Members

UxmlFactory<InspectorView, VisualElement.UxmlTraits>.Create(IUxmlAttributes, CreationContext) ,  
BaseUxmlFactory<InspectorView, VisualElement.UxmlTraits>.AcceptsAttributeBag(IUxmlAttributes, CreationContext) ,  
BaseUxmlFactory<InspectorView, VisualElement.UxmlTraits>.uxmlName ,  
BaseUxmlFactory<InspectorView, VisualElement.UxmlTraits>.uxmlNamespace ,  
BaseUxmlFactory<InspectorView, VisualElement.UxmlTraits>.uxmlQualifiedName ,  
BaseUxmlFactory<InspectorView, VisualElement.UxmlTraits>.uxmlType ,  
BaseUxmlFactory<InspectorView, VisualElement.UxmlTraits>.canHaveAnyAttribute ,  
BaseUxmlFactory<InspectorView, VisualElement.UxmlTraits>.uxmlAttributesDescription ,  
BaseUxmlFactory<InspectorView, VisualElement.UxmlTraits>.uxmlChildElementsDescription ,  
BaseUxmlFactory<InspectorView, VisualElement.UxmlTraits>.substituteForTypeName ,  
BaseUxmlFactory<InspectorView, VisualElement.UxmlTraits>.substituteForTypeNamespace ,  
BaseUxmlFactory<InspectorView, VisualElement.UxmlTraits>.substituteForTypeQualifiedName ,  
[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

# Class NodeView

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

Assembly: Assembly-CSharp-Editor-firstpass.dll

```
public class NodeView : Node, IEventHandler, IResolvedStyle, ITransform,  
ITransitionAnimations, IExperimentalFeatures, IVisualElementScheduler,  
ISelectable, ICollectibleElement
```

## Inheritance

[object](#) ↗ ← CallbackEventHandler ← Focusable ← VisualElement ← GraphElement ← Node ← NodeView

## Implements

IEventHandler, IResolvedStyle, ITransform, ITransitionAnimations, IExperimentalFeatures,  
IVisualElementScheduler, ISelectable, ICollectibleElement

## Inherited Members

Node.m\_CollapseButton , Node.m\_ButtonContainer , Node.RefreshExpandedState() , Node.GetPosition() ,  
Node.OnPortRemoved(Port) , [Node.InstantiatePort\(Orientation, Direction, Port.Capacity, Type\)](#) ↗ ,  
Node.RefreshPorts() , Node.ToggleCollapse() , Node.UseDefaultStyling() ,  
Node.BuildContextualMenu(ContextualMenuPopulateEvent) ,  
[Node.CollectElements\(HashSet<GraphElement>, Func<GraphElement, bool>\)](#) ↗ , Node.mainContainer ,  
Node.titleContainer , Node.inputContainer , Node.outputContainer , Node.titleButtonContainer ,  
Node.topContainer , Node.extensionContainer , Node.expanded , Node.title ,  
GraphElement.ResetLayer() , GraphElement.OnCustomStyleResolved(ICustomStyle) ,  
GraphElement.IsSelectable() , GraphElement.IsMovable() , GraphElement.IsResizable() ,  
GraphElement.IsDroppable() , GraphElement.IsAscendable() , GraphElement.IsRenamable() ,  
GraphElement.IsCopyable() , GraphElement.IsSnappable() , GraphElement.IsGroupable() ,  
GraphElement.IsStackable() , GraphElement.GetGlobalCenter() ,  
GraphElement.UpdatePresenterPosition() , GraphElement.HitTest(Vector2) ,  
[GraphElement.Select\(VisualElement, bool\)](#) ↗ , GraphElement.Unselect(VisualElement) ,  
GraphElement.isSelected(VisualElement) , GraphElement.elementTypeColor , GraphElement.layer ,  
GraphElement.showInMiniMap , GraphElement.capabilities , GraphElement.selected ,  
VisualElement.disabledUssClassName , VisualElement.ExecuteDefaultAction(EventBase) ,  
VisualElement.Focus() , VisualElement.SendEvent(EventBase) ,  
[VisualElement.SetEnabledFromHierarchy\(bool\)](#) ↗ , [VisualElement.SetEnabled\(bool\)](#) ↗ ,  
VisualElement.MarkDirtyRepaint() , VisualElement.ContainsPoint(Vector2) , VisualElement.Overlaps(Rect) ,  
[VisualElement.DoMeasure\(float, VisualElement.MeasureMode, float, VisualElement.MeasureMode\)](#) ↗ ,  
VisualElement.ToString() , VisualElement.GetClasses() , VisualElement.ClearClassList() ,

[VisualElement.AddToClassList\(string\)](#) , [VisualElement.RemoveFromClassList\(string\)](#) ,  
[VisualElement.ToggleInClassList\(string\)](#) , [VisualElement.EnableInClassList\(string, bool\)](#) ,  
[VisualElement.ClassListContains\(string\)](#) , VisualElement.FindAncestorUserData() ,  
VisualElement.Add(VisualElement) , [VisualElement.Insert\(int, VisualElement\)](#) ,  
VisualElement.Remove(VisualElement) , [VisualElement.RemoveAt\(int\)](#) , VisualElement.Clear() ,  
[VisualElement.ElementAt\(int\)](#) , VisualElement.IndexOf(VisualElement) , VisualElement.Children() ,  
[VisualElement.Sort\(Comparison<VisualElement>\)](#) , VisualElement.BringToFront() ,  
VisualElement.SendToBack() , VisualElement.PlaceBehind(VisualElement) ,  
VisualElement.PlaceInFront(VisualElement) , VisualElement.RemoveFromHierarchy() ,  
VisualElement.GetFirstOfType<T>() , VisualElement.GetFirstAncestorOfType<T>() ,  
VisualElement.Contains(VisualElement) , VisualElement.FindCommonAncestor(VisualElement) ,  
VisualElement.resolvedStyle , VisualElement.viewDataKey , VisualElement.userData ,  
VisualElement.canGrabFocus , VisualElement.focusController , VisualElement.usageHints ,  
VisualElement.transform , VisualElement.layout , VisualElement.contentRect , VisualElement.paddingRect ,  
VisualElement.worldBound , VisualElement.localBound , VisualElement.worldTransform ,  
VisualElement.pickingMode , VisualElement.name , VisualElement.enabledInHierarchy ,  
VisualElement.enabledSelf , VisualElement.languageDirection , VisualElement.visible ,  
VisualElement.generateVisualContent , VisualElement.experimental , VisualElement.hierarchy ,  
VisualElement.cacheAsBitmap , VisualElement.parent , VisualElement.panel ,  
VisualElement.contentContainer , VisualElement.visualTreeAssetSource , [VisualElement.this\[int\]](#) ,  
VisualElement.childCount , VisualElement.schedule , VisualElement.style , VisualElement.customStyle ,  
VisualElement.styleSheets , VisualElement.tooltip , Focusable.Blur() , Focusable.focusable ,  
Focusable.TabIndex , Focusable.delegatesFocus ,  
CallbackEventHandler.RegisterCallback<TEventType>(EventCallback<TEventType>, TrickleDown) ,  
CallbackEventHandler.RegisterCallback<TEventType, TUserArgsType>(EventCallback<TEventType, TUserArgsType>, TUserArgsType, TrickleDown) ,  
CallbackEventHandler.UnregisterCallback<TEventType>(EventCallback<TEventType>, TrickleDown) ,  
CallbackEventHandler.UnregisterCallback<TEventType, TUserArgsType>(EventCallback<TEventType, TUserArgsType>, TrickleDown) ,  
CallbackEventHandler.HandleEvent(EventBase) , CallbackEventHandler.HasTrickleDownHandlers() ,  
CallbackEventHandler.HasBubbleUpHandlers() ,  
CallbackEventHandler.ExecuteDefaultActionAtTarget(EventBase) , [object.Equals\(object\)](#) ,  
[object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#)

## Constructors

### NodeView(Node)

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

Parameters

node [Node](#)

## Fields

Input

```
public Port Input
```

Field Value

Port

K\_PERSONDEPENDENT\_CLASSNAME

```
public static string K_PERSONDEPENDENT_CLASSNAME
```

Field Value

[string](#) ↗

Node

```
public Node Node
```

Field Value

[Node](#)

OnNodeSelected

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

## Field Value

[Action](#) <[NodeView](#)>

## Outputs

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

## Field Value

[List](#) <[Port](#)>

## m\_serializedNode

```
protected SerializedObject m_serializedNode
```

## Field Value

SerializedObject

## Properties

### Master

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

## Property Value

[DialogueGraphView](#)

## Methods

## OnSelected()

Called when the GraphElement is selected.

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

## OnUnselected()

Called when the GraphElement is unselected.

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

## SetPosition(Rect)

Set node position.

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

### Parameters

**newPos** Rect

New position.

# Class VariableBankCreationHandler

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

Assembly: Assembly-CSharp-Editor-firstpass.dll

```
public class VariableBankCreationHandler
```

## Inheritance

[object](#) ← VariableBankCreationHandler

## Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

# Namespace com.absence.dialoguesystem.internals

## Classes

### [ActionNode](#)

Node which invokes some actions on the flow.

### [AdditionalSpeechData](#)

### [Blackboard](#)

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

### [ConditionNode](#)

Node which re-routes the flow under some conditions.

### [DecisionSpeechNode](#)

Node which displays a speech with options.

### [DialoguePartNode](#)

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

### [FastSpeechNode](#)

Node which displays a speech without options.

### [GotoNode](#)

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

### [Node](#)

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

### [Option](#)

The type to hold references to dialogue options.

### [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 [AdditionalSpeech Data](#).

## [IContainVariableManipulators](#)

Any node subtype with this interface implemented will refresh its com.absence.variablesystem.Variable Comparers and com.absence.variablesystem.VariableSetters to have the correct reference to the [Bank](#) of the current [Dialogue](#) everytime the editor window refreshes.

# Enums

## [ConditionNode.ProcessType](#)

## [Node.NodeState](#)

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

# Class ActionNode

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

Assembly: Assembly-CSharp-firstpass.dll

Node which invokes some actions on the flow.

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

## 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>() , Object.GetInstanceID() , Object.GetHashCode() ,  
[Object.Equals\(object\)](#) , Object.Instantiate(Object, Vector3, Quaternion) ,  
Object.Instantiate(Object, Vector3, Quaternion, Transform) , Object.Instantiate(Object) ,  
Object.Instantiate(Object, Transform) , [Object.Instantiate\(Object, Transform, bool\)](#) ,  
Object.Instantiate<T>(T) , Object.Instantiate<T>(T, Vector3, Quaternion) ,  
Object.Instantiate<T>(T, Vector3, Quaternion, Transform) , Object.Instantiate<T>(T, Transform) ,  
[Object.Instantiate<T>\(T, Transform, bool\)](#) , [Object.Destroy\(Object, float\)](#) , Object.Destroy(Object) ,  
[Object.DestroyImmediate\(Object, bool\)](#) , Object.DestroyImmediate(Object) ,  
[Object.FindObjectsOfType\(Type\)](#) , [Object.FindObjectsOfType\(Type, bool\)](#) ,  
[Object.FindObjectsByType\(Type, FindObjectsSortMode\)](#) ,  
[Object.FindObjectsByType\(Type, FindObjectsInactive, FindObjectsSortMode\)](#) ,  
Object.DontDestroyOnLoad(Object) , [Object.DestroyObject\(Object, float\)](#) ,  
Object.DestroyObject(Object) , [Object.FindSceneObjectsOfType\(Type\)](#) ,  
[Object.FindObjectsOfTypeIncludingAssets\(Type\)](#) , Object.FindObjectsOfType<T>() ,  
Object.FindObjectsByType<T>(FindObjectsSortMode) , [Object.FindObjectsOfType<T>\(bool\)](#) ,

```
Object.FindObjectsOfType<T>(FindObjectsInactive, FindObjectsSortMode),  
Object.FindObjectOfType<T>() , Object.FindObjectOfType<T>\(bool\) ,  
Object.FindFirstObjectByType<T>() , Object.FindAnyObjectByType<T>() ,  
Object.FindFirstObjectByType<T>(FindObjectsInactive) ,  
Object.FindAnyObjectByType<T>(FindObjectsInactive) , Object.FindObjectsOfTypeAll\(Type\) ,  
Object.FindObjectType\(Type\) , Object.FindFirstObjectType\(Type\) ,  
Object.FindAnyObjectType\(Type\) , Object.FindObjectType\(Type, bool\) ,  
Object.FindFirstObjectType\(Type, FindObjectsInactive\) ,  
Object.FindAnyObjectType\(Type, FindObjectsInactive\) , Object.ToString() , Object.name ,  
Object.hideFlags , object.Equals\(object, object\) , object.GetType\(\) , object.MemberwiseClone\(\) ,  
object.ReferenceEquals\(object, object\)
```

## Fields

### Next

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

### Field Value

[Node](#)

## UnityEvents

```
public UnityEvent UnityEvents
```

### Field Value

UnityEvent

## VBActions

```
public List<VariableSetter> VBActions
```

## Field Value

[List ↗](#) <VariableSetter>

## Methods

### AddNextNode\_Inline(Node, int)

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

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

#### Parameters

nextWillBeAdded [Node](#)

atPort [int ↗](#)

### CustomAction()

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

### DelayedClone(Dialogue)

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

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

#### Parameters

originalDialogue [Dialogue](#)

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.

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

Returns

[string](#)

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

## GetComparers()

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

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

Returns

[List](#)<VariableComparer>

## 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.

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

Parameters

[result](#) [List](#)<(int portIndex, Node node)>

## GetSetters()

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

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

Returns

[List](#) <VariableSetter>

## GetTitle()

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

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

Returns

[string](#)

The title as a string.

## Pass\_Inline(params object[])

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

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

Parameters

passData [object](#)[]

## Reach\_Inline()

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

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

## RemoveNextNode\_Inline(int)

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

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

Parameters

atPort [int](#)

## Traverse(Action<Node>)

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

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

Parameters

action [Action](#)<[Node](#)>

# Class AdditionalSpeechData

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

Assembly: Assembly-CSharp-firstpass.dll

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

## Inheritance

[object](#) ← AdditionalSpeechData

## Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

## Properties

### AnimatorMemberName

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

#### Property Value

[string](#)

### AudioClip

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

#### Property Value

AudioClip

### CustomInfo

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

Property Value

[string](#)[]

## Sprite

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

Property Value

Sprite

# Class Blackboard

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

Assembly: Assembly-CSharp-firstpass.dll

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

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

## Inheritance

[object](#) ← Blackboard

## Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

## Fields

### Bank

Bank of this blackboard.

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

## Field Value

VariableBank

## Methods

### Clone()

Use to clone this blackboard.

```
public Blackboard Clone()
```

Returns

[Blackboard](#)

# Class ConditionNode

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

Assembly: Assembly-CSharp-firstpass.dll

Node which re-routes the flow under some conditions.

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

## 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>\(\)](#) , [Object.GetInstanceID\(\)](#) ,  
[Object.GetHashCode\(\)](#) , [Object.Equals\(object\)](#) , [Object.Instantiate\(Object, Vector3, Quaternion\)](#) ,  
[Object.Instantiate\(Object, Vector3, Quaternion, Transform\)](#) , [Object.Instantiate\(Object\)](#) ,  
[Object.Instantiate\(Object, Transform\)](#) , [Object.Instantiate\(Object, Transform, bool\)](#) ,  
[Object.Instantiate<T>\(T\)](#) , [Object.Instantiate<T>\(T, Vector3, Quaternion\)](#) ,  
[Object.Instantiate<T>\(T, Vector3, Quaternion, Transform\)](#) , [Object.Instantiate<T>\(T, Transform\)](#) ,  
[Object.Instantiate<T>\(T, Transform, bool\)](#) , [Object.Destroy\(Object, float\)](#) , [Object.Destroy\(Object\)](#) ,  
[Object.DestroyImmediate\(Object, bool\)](#) , [Object.DestroyImmediate\(Object\)](#) ,  
[Object.FindObjectsOfType\(Type\)](#) , [Object.FindObjectsOfType\(Type, bool\)](#) ,  
[Object.FindObjectsByType\(Type, FindObjectsSortMode\)](#) ,  
[Object.FindObjectsByType\(Type, FindObjectsInactive, FindObjectsSortMode\)](#) ,  
[Object.DontDestroyOnLoad\(Object\)](#) , [Object.DestroyObject\(Object, float\)](#) ,  
[Object.DestroyObject\(Object\)](#) , [Object.FindSceneObjectsOfType\(Type\)](#) ,  
[Object.FindObjectsOfTypeIncludingAssets\(Type\)](#) , [Object.FindObjectsOfType<T>\(\)](#) ,  
[Object.FindObjectsByType<T>\(FindObjectsSortMode\)](#) , [Object.FindObjectsOfType<T>\(bool\)](#) ,  
[Object.FindObjectsByType<T>\(FindObjectsInactive, FindObjectsSortMode\)](#) ,

Object.FindObjectOfType<T>() , [Object.FindObjectOfType<T>\(bool\)](#) ,  
Object.FindFirstObjectByType<T>() , Object.FindAnyObjectByType<T>() ,  
Object.FindFirstObjectByType<T>(FindObjectsInactive) ,  
Object.FindAnyObjectByType<T>(FindObjectsInactive) , [Object.FindObjectsOfTypeAll\(Type\)](#) ,  
[Object.FindObjectOfType\(Type\)](#) , [Object.FindFirstObjectByType\(Type\)](#) ,  
[Object.FindAnyObjectByType\(Type\)](#) , [Object.FindObjectOfType\(Type, bool\)](#) ,  
[Object.FindFirstObjectByType\(Type, FindObjectsInactive\)](#) ,  
[Object.FindAnyObjectByType\(Type, FindObjectsInactive\)](#) , Object.ToString() , Object.name ,  
Object.hideFlags , [object.Equals\(object, object\)](#) , [object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) ,  
[object.ReferenceEquals\(object, object\)](#)

## Fields

### Comparers

```
public List<VariableComparer> Comparers
```

#### Field Value

[List](#)<VariableComparer>

### FalseNext

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

#### Field Value

[Node](#)

### Processor

```
public ConditionNode.ProcessType Processor
```

#### Field Value

## TrueNext

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

### Field Value

[Node](#)

## Methods

### AddNextNode\_Inline(Node, int)

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

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

### Parameters

nextWillBeAdded [Node](#)

atPort [int](#)

### DelayedClone(Dialogue)

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

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

### Parameters

originalDialogue [Dialogue](#)

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.

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

Returns

[string](#)

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

## GetComparers()

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

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

Returns

[List](#)<VariableComparer>

## 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.

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

Parameters

[result](#) [List](#)<[int](#) [portIndex](#), [Node](#) [node](#)>

## GetOutputPortNamesForCreation()

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

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

Returns

[List](#) <[string](#)>

Returns the port names as a list of strings. Return an empty list if you want no output ports.

## GetSetters()

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

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

Returns

[List](#) <[VariableSetter](#)>

## GetTitle()

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

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

Returns

[string](#)

The title as a string.

## Pass\_Inline(params object[])

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

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

Parameters

`passData object[]`

## Process()

`protected virtual bool Process()`

Returns

`bool`

## Reach\_Inline()

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

`protected override void Reach_Inline()`

## RemoveNextNode\_Inline(int)

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

`protected override void RemoveNextNode_Inline(int atPort)`

Parameters

`atPort int`

## Traverse(Action<Node>)

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

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

## Parameters

action [Action](#) <Node>

# Enum ConditionNode.ProcessType

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

Assembly: Assembly-CSharp-firstpass.dll

```
public enum ConditionNode.ProcessType
```

## Fields

All = 0

Any = 1

# Class DecisionSpeechNode

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

Assembly: Assembly-CSharp-firstpass.dll

Node which displays a speech with options.

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

## 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>\(\)](#), [Object.GetInstanceID\(\)](#), [Object.GetHashCode\(\)](#),  
[Object.Equals\(object\)](#) ↗, [Object.Instantiate\(Object, Vector3, Quaternion\)](#),  
[Object.Instantiate\(Object, Vector3, Quaternion, Transform\)](#), [Object.Instantiate\(Object\)](#),  
[Object.Instantiate\(Object, Transform\)](#), [Object.Instantiate\(Object, Transform, bool\)](#) ↗,  
[Object.Instantiate<T>\(T\)](#), [Object.Instantiate<T>\(T, Vector3, Quaternion\)](#),  
[Object.Instantiate<T>\(T, Vector3, Quaternion, Transform\)](#), [Object.Instantiate<T>\(T, Transform\)](#),  
[Object.Instantiate<T>\(T, Transform, bool\)](#) ↗, [Object.Destroy\(Object, float\)](#) ↗, [Object.Destroy\(Object\)](#),  
[Object.DestroyImmediate\(Object, bool\)](#) ↗, [Object.DestroyImmediate\(Object\)](#),  
[Object.FindObjectsOfType\(Type\)](#) ↗, [Object.FindObjectsOfType\(Type, bool\)](#) ↗,  
[Object.FindObjectsByType\(Type, FindObjectsSortMode\)](#) ↗,  
[Object.FindObjectsByType\(Type, FindObjectsInactive, FindObjectsSortMode\)](#) ↗,  
[Object.DontDestroyOnLoad\(Object\)](#), [Object.DestroyObject\(Object, float\)](#) ↗,  
[Object.DestroyObject\(Object\)](#), [Object.FindSceneObjectsOfType\(Type\)](#) ↗,  
[Object.FindObjectsOfTypeIncludingAssets\(Type\)](#) ↗, [Object.FindObjectsOfType<T>\(\)](#),  
[Object.FindObjectsByType<T>\(FindObjectsSortMode\)](#), [Object.FindObjectsOfType<T>\(bool\)](#) ↗,  
[Object.FindObjectsByType<T>\(FindObjectsInactive, FindObjectsSortMode\)](#),

Object.FindObjectOfType<T>() , [Object.FindObjectOfType<T>\(bool\)](#) ,  
Object.FindFirstObjectByType<T>() , Object.FindAnyObjectByType<T>() ,  
Object.FindFirstObjectByType<T>(FindObjectsInactive) ,  
Object.FindAnyObjectByType<T>(FindObjectsInactive) , [Object.FindObjectsOfTypeAll\(Type\)](#) ,  
[Object.FindObjectOfType\(Type\)](#) , [Object.FindFirstObjectByType\(Type\)](#) ,  
[Object.FindAnyObjectByType\(Type\)](#) , [Object.FindObjectOfType\(Type, bool\)](#) ,  
[Object.FindFirstObjectByType\(Type, FindObjectsInactive\)](#) ,  
[Object.FindAnyObjectByType\(Type, FindObjectsInactive\)](#) , Object.ToString() , Object.name ,  
Object.hideFlags , [object.Equals\(object, object\)](#) , [object.GetType\(\)](#) ,  
[object.ReferenceEquals\(object, object\)](#)

## Fields

### Options

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

### Field Value

[List](#)<[Option](#)>

### Speech

[HideInInspector]  
`public string Speech`

### Field Value

[string](#)

## Properties

### PersonDependent

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

Property Value

[bool](#)

## Methods

### AddNextNode\_Inline(Node, int)

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

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

Parameters

nextWillBeAdded [Node](#)

atPort [int](#)

### DelayedClone(Dialogue)

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

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

Parameters

originalDialogue [Dialogue](#)

This is the dialogue the cloned dialogue had cloned from.

### GetAdditionalSpeechData()

```
public AdditionalSpeechData GetAdditionalSpeechData()
```

Returns

[AdditionalSpeechData](#)

## GetClassName()

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

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

Returns

[string](#)

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

## GetComparers()

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

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

Returns

[List](#)<VariableComparer>

## 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.

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

Parameters

result [List](#)<(int portIndex, Node node)>

## GetOptions()

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

Returns

[List](#) <[Option](#)>

## GetOutputPortNamesForCreation()

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

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

Returns

[List](#) <[string](#)>

Returns the port names as a list of strings. Return an empty list if you want no output ports.

## GetSetters()

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

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

Returns

[List](#) <[VariableSetter](#)>

## GetSpeech()

```
public string GetSpeech()
```

Returns

[string](#)

## GetTitle()

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

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

Returns

[string](#)

The title as a string.

## Pass\_Inline(params object[])

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

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

Parameters

passData [object](#)[]

## Reach\_Inline()

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

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

## RemoveNextNode\_Inline(int)

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

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

## Parameters

atPort [int](#)

## Traverse(Action<Node>)

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

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

## Parameters

action [Action](#)<[Node](#)>

# Class DialoguePartNode

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

Assembly: Assembly-CSharp-firstpass.dll

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

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

## 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>() , Object.GetInstanceID() ,  
Object.GetHashCode() , [Object.Equals\(object\)](#) ↗ , Object.Instantiate(Object, Vector3, Quaternion) ,  
Object.Instantiate(Object, Vector3, Quaternion, Transform) , Object.Instantiate(Object) ,  
Object.Instantiate(Object, Transform) , [Object.Instantiate\(Object, Transform, bool\)](#) ↗ ,  
Object.Instantiate<T>(T) , Object.Instantiate<T>(T, Vector3, Quaternion) ,  
Object.Instantiate<T>(T, Vector3, Quaternion, Transform) , Object.Instantiate<T>(T, Transform) ,  
[Object.Instantiate<T>\(T, Transform, bool\)](#) ↗ , [Object.Destroy\(Object, float\)](#) ↗ , Object.Destroy(Object) ,  
[Object.DestroyImmediate\(Object, bool\)](#) ↗ , Object.DestroyImmediate(Object) ,  
[Object.FindObjectsOfType\(Type\)](#) ↗ , [Object.FindObjectsOfType\(Type, bool\)](#) ↗ ,  
[Object.FindObjectsByType\(Type, FindObjectsSortMode\)](#) ↗ ,  
[Object.FindObjectsByType\(Type, FindObjectsInactive, FindObjectsSortMode\)](#) ↗ ,  
Object.DontDestroyOnLoad(Object) , [Object.DestroyObject\(Object, float\)](#) ↗ ,  
Object.DestroyObject(Object) , [Object.FindSceneObjectsOfType\(Type\)](#) ↗ ,  
[Object.FindObjectsOfTypeIncludingAssets\(Type\)](#) ↗ , Object.FindObjectsOfType<T>() ,  
Object.FindObjectsByType<T>(FindObjectsSortMode) , [Object.FindObjectsOfType<T>\(bool\)](#) ↗ ,  
Object.FindObjectsByType<T>(FindObjectInactive, FindObjectsSortMode) ,  
Object.FindObjectOfType<T>() , [Object.FindObjectOfType<T>\(bool\)](#) ↗ ,  
Object.FindFirstObjectOfType<T>() , Object.FindAnyObjectOfType<T>() ,  
Object.FindFirstObjectOfType<T>(FindObjectInactive) ,

Object.FindAnyObjectByType<T>(FindObjectsInactive) , [Object.FindObjectsOfTypeAll\(Type\)](#) ,  
[Object.FindObjectOfType\(Type\)](#) , [Object.FindFirstObjectByType\(Type\)](#) ,  
[Object.FindAnyObjectByType\(Type\)](#) , [Object.FindObjectOfType\(Type, bool\)](#) ,  
[Object.FindFirstObjectByType\(Type, FindObjectsInactive\)](#) ,  
[Object.FindAnyObjectByType\(Type, FindObjectsInactive\)](#) , Object.ToString() , Object.name ,  
Object.hideFlags , [object.Equals\(object, object\)](#) , [object.GetType\(\)](#) ,  
[object.ReferenceEquals\(object, object\)](#)

## Fields

### DialoguePartName

```
public string DialoguePartName
```

#### Field Value

[string](#)

### Next

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

#### Field Value

[Node](#)

## Properties

### DisplayState

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

#### Property Value

[bool](#)

## Methods

### AddNextNode\_Inline(Node, int)

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

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

#### Parameters

nextWillBeAdded [Node](#)

atPort [int](#)

### DelayedClone(Dialogue)

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

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

#### Parameters

originalDialogue [Dialogue](#)

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.

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

#### Returns

[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.

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

Returns

[string](#)

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

## 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.

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

Parameters

[result](#) [List](#)<[int](#), [Node](#)>

## GetTitle()

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

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

Returns

[string](#)

The title as a string.

## Pass\_Inline(params object[])

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

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

Parameters

passData [object](#)[]

## Reach\_Inline()

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

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

## RemoveNextNode\_Inline(int)

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

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

Parameters

atPort [int](#)

## Traverse(Action<Node>)

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

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

Parameters

action [Action](#)<[Node](#)>



# Class FastSpeechNode

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

Assembly: Assembly-CSharp-firstpass.dll

Node which displays a speech without options.

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

## 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>\(\)](#) , [Object.GetInstanceID\(\)](#) ,  
[Object.GetHashCode\(\)](#) , [Object.Equals\(object\)](#) ↗ , [Object.Instantiate\(Object, Vector3, Quaternion\)](#) ,  
[Object.Instantiate\(Object, Vector3, Quaternion, Transform\)](#) , [Object.Instantiate\(Object\)](#) ,  
[Object.Instantiate\(Object, Transform\)](#) , [Object.Instantiate\(Object, Transform, bool\)](#) ↗ ,  
[Object.Instantiate<T>\(T\)](#) , [Object.Instantiate<T>\(T, Vector3, Quaternion\)](#) ,  
[Object.Instantiate<T>\(T, Vector3, Quaternion, Transform\)](#) , [Object.Instantiate<T>\(T, Transform\)](#) ,  
[Object.Instantiate<T>\(T, Transform, bool\)](#) ↗ , [Object.Destroy\(Object, float\)](#) ↗ , [Object.Destroy\(Object\)](#) ,  
[Object.DestroyImmediate\(Object, bool\)](#) ↗ , [Object.DestroyImmediate\(Object\)](#) ,  
[Object.FindObjectsOfType\(Type\)](#) ↗ , [Object.FindObjectsOfType\(Type, bool\)](#) ↗ ,  
[Object.FindObjectsByType\(Type, FindObjectsSortMode\)](#) ↗ ,  
[Object.FindObjectsByType\(Type, FindObjectsInactive, FindObjectsSortMode\)](#) ↗ ,  
[Object.DontDestroyOnLoad\(Object\)](#) , [Object.DestroyObject\(Object, float\)](#) ↗ ,  
[Object.DestroyObject\(Object\)](#) , [Object.FindSceneObjectsOfType\(Type\)](#) ↗ ,  
[Object.FindObjectsOfTypeIncludingAssets\(Type\)](#) ↗ , [Object.FindObjectsOfType<T>\(\)](#) ,  
[Object.FindObjectsByType<T>\(FindObjectsSortMode\)](#) , [Object.FindObjectsOfType<T>\(bool\)](#) ↗ ,  
[Object.FindObjectsByType<T>\(FindObjectsInactive, FindObjectsSortMode\)](#) ,

Object.FindObjectOfType<T>() , [Object.FindObjectOfType<T>\(bool\)](#) ,  
Object.FindFirstObjectByType<T>() , Object.FindAnyObjectByType<T>() ,  
Object.FindFirstObjectByType<T>(FindObjectsInactive) ,  
Object.FindAnyObjectByType<T>(FindObjectsInactive) , [Object.FindObjectsOfTypeAll\(Type\)](#) ,  
[Object.FindObjectOfType\(Type\)](#) , [Object.FindFirstObjectByType\(Type\)](#) ,  
[Object.FindAnyObjectByType\(Type\)](#) , [Object.FindObjectOfType\(Type, bool\)](#) ,  
[Object.FindFirstObjectByType\(Type, FindObjectsInactive\)](#) ,  
[Object.FindAnyObjectByType\(Type, FindObjectsInactive\)](#) , Object.ToString() , Object.name ,  
Object.hideFlags , [object.Equals\(object, object\)](#) , [object.GetType\(\)](#) ,  
[object.ReferenceEquals\(object, object\)](#)

## Fields

### Next

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

### Field Value

[Node](#)

### Speech

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

### Field Value

[string](#)

## Properties

### PersonDependent

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

Property Value

[bool](#)

## Methods

### AddNextNode\_Inline(Node, int)

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

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

Parameters

nextWillBeAdded [Node](#)

atPort [int](#)

### DelayedClone(Dialogue)

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

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

Parameters

originalDialogue [Dialogue](#)

This is the dialogue the cloned dialogue had cloned from.

### GetAdditionalSpeechData()

```
public AdditionalSpeechData GetAdditionalSpeechData()
```

Returns

[AdditionalSpeechData](#)

## GetClassName()

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

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

Returns

[string](#)

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

## 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.

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

Parameters

[result](#) [List](#)<[int](#), [Node](#)>

## GetOptions()

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

Returns

[List](#)<[Option](#)>

## GetSpeech()

```
public string GetSpeech()
```

Returns

[string](#)

## GetTitle()

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

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

Returns

[string](#)

The title as a string.

## Pass\_Inline(params object[])

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

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

Parameters

[passData](#) [object](#)[]

## Reach\_Inline()

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

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

## RemoveNextNode\_Inline(int)

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

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

Parameters

atPort [int](#)

## Traverse(Action<Node>)

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

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

Parameters

action [Action](#)<[Node](#)>

# Class GotoNode

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

Assembly: Assembly-CSharp-firstpass.dll

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

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

## 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>\(\)](#) , [Object.GetInstanceID\(\)](#) ,  
[Object.GetHashCode\(\)](#) , [Object.Equals\(object\)](#) ↗ , [Object.Instantiate\(Object, Vector3, Quaternion\)](#) ,  
[Object.Instantiate\(Object, Vector3, Quaternion, Transform\)](#) , [Object.Instantiate\(Object\)](#) ,  
[Object.Instantiate\(Object, Transform\)](#) , [Object.Instantiate\(Object, Transform, bool\)](#) ↗ ,  
[Object.Instantiate<T>\(T\)](#) , [Object.Instantiate<T>\(T, Vector3, Quaternion\)](#) ,  
[Object.Instantiate<T>\(T, Vector3, Quaternion, Transform\)](#) , [Object.Instantiate<T>\(T, Transform\)](#) ,  
[Object.Instantiate<T>\(T, Transform, bool\)](#) ↗ , [Object.Destroy\(Object, float\)](#) ↗ , [Object.Destroy\(Object\)](#) ,  
[Object.DestroyImmediate\(Object, bool\)](#) ↗ , [Object.DestroyImmediate\(Object\)](#) ,  
[Object.FindObjectsOfType\(Type\)](#) ↗ , [Object.FindObjectsOfType\(Type, bool\)](#) ↗ ,  
[Object.FindObjectsByType\(Type, FindObjectsSortMode\)](#) ↗ ,  
[Object.FindObjectsByType\(Type, FindObjectsInactive, FindObjectsSortMode\)](#) ↗ ,  
[Object.DontDestroyOnLoad\(Object\)](#) , [Object.DestroyObject\(Object, float\)](#) ↗ ,  
[Object.DestroyObject\(Object\)](#) , [Object.FindSceneObjectsOfType\(Type\)](#) ↗ ,  
[Object.FindObjectsOfTypeIncludingAssets\(Type\)](#) ↗ , [Object.FindObjectsOfType<T>\(\)](#) ,  
[Object.FindObjectsByType<T>\(FindObjectsSortMode\)](#) , [Object.FindObjectsOfType<T>\(bool\)](#) ↗ ,  
[Object.FindObjectsByType<T>\(FindObjectInactive, FindObjectsSortMode\)](#) ,  
[Object.FindObjectOfType<T>\(\)](#) , [Object.FindObjectOfType<T>\(bool\)](#) ↗ ,  
[Object.FindFirstObjectOfType<T>\(\)](#) , [Object.FindAnyObjectOfType<T>\(\)](#) ,  
[Object.FindFirstObjectOfType<T>\(FindObjectInactive\)](#) ,

Object.FindAnyObjectByType<T>(FindObjectsInactive) , [Object.FindObjectsOfTypeAll\(Type\)](#) ,  
[Object.FindObjectOfType\(Type\)](#) , [Object.FindFirstObjectByType\(Type\)](#) ,  
[Object.FindAnyObjectByType\(Type\)](#) , [Object.FindObjectOfType\(Type, bool\)](#) ,  
[Object.FindFirstObjectByType\(Type, FindObjectsInactive\)](#) ,  
[Object.FindAnyObjectByType\(Type, FindObjectsInactive\)](#) , Object.ToString() , Object.name ,  
Object.hideFlags , [object.Equals\(object, object\)](#) , [object.GetType\(\)](#) ,  
[object.ReferenceEquals\(object, object\)](#)

## Fields

### TargetDialogPartName

`public string TargetDialogPartName`

#### Field Value

`string`

## Methods

### AddNextNode\_Inline(Node, int)

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

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

#### Parameters

`nextWillBeAdded Node`

`atPort int`

### GetClassName()

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

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

Returns

string

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

## 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.

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

Parameters

result List<(int portIndex, Node node)>

## GetOutputPortNamesForCreation()

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

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

Returns

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.

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

Returns

[string](#)

The title as a string.

## Pass\_Inline(params object[])

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

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

Parameters

passData [object](#)[]

## Reach\_Inline()

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

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

## RemoveNextNode\_Inline(int)

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

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

Parameters

atPort [int](#)

# Interface IContainSpeech

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

Assembly: Assembly-CSharp-firstpass.dll

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

```
public interface IContainSpeech
```

## Methods

### GetAdditionalSpeechData()

```
AdditionalSpeechData GetAdditionalSpeechData()
```

Returns

[AdditionalSpeechData](#)

### GetOptions()

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

Returns

[List](#) <[Option](#)>

### GetSpeech()

```
string GetSpeech()
```

Returns

[string](#) ↗

# Interface IContainVariableManipulators

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

Assembly: Assembly-CSharp-firstpass.dll

Any node subtype with this interface implemented will refresh its com.absence.variablesystem.VariableComparers and com.absence.variablesystem.VariableSetters to have the correct reference to the [Bank](#) of the current [Dialogue](#) everytime the editor window refreshes.

```
public interface IContainVariableManipulators
```

## Methods

### GetComparers()

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

```
List<VariableComparer> GetComparers()
```

Returns

[List](#) <VariableComparer>

### GetSetters()

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

```
List<VariableSetter> GetSetters()
```

Returns

[List](#) <VariableSetter>

# Class Node

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

Assembly: Assembly-CSharp-firstpass.dll

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

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

## Inheritance

[object](#) ← Object ← ScriptableObject ← Node

## Derived

[ActionNode](#), [ConditionNode](#), [DecisionSpeechNode](#), [DialoguePartNode](#), [FastSpeechNode](#), [GotoNode](#),  
[RootNode](#), [StickyNoteNode](#), [TitleNode](#)

## Inherited Members

ScriptableObject.SetDirty(), [ScriptableObject.CreateInstance\(string\)](#) ,  
[ScriptableObject.CreateInstance\(Type\)](#) , ScriptableObject.CreateInstance<T>() , Object.GetInstanceId() ,  
Object.GetHashCode() , [Object.Equals\(object\)](#) , Object.Instantiate(Object, Vector3, Quaternion) ,  
Object.Instantiate(Object, Vector3, Quaternion, Transform) , Object.Instantiate(Object) ,  
Object.Instantiate(Object, Transform) , [Object.Instantiate\(Object, Transform, bool\)](#) ,  
Object.Instantiate<T>(T) , Object.Instantiate<T>(T, Vector3, Quaternion) ,  
Object.Instantiate<T>(T, Vector3, Quaternion, Transform) , Object.Instantiate<T>(T, Transform) ,  
[Object.Instantiate<T>\(T, Transform, bool\)](#) , [Object.Destroy\(Object, float\)](#) , Object.Destroy(Object) ,  
[Object.DestroyImmediate\(Object, bool\)](#) , Object.DestroyImmediate(Object) ,  
[Object.FindObjectsOfType\(Type\)](#) , [Object.FindObjectsOfType\(Type, bool\)](#) ,  
[Object.FindObjectsByType\(Type, FindObjectsSortMode\)](#) ,  
[Object.FindObjectsByType\(Type, FindObjectsInactive, FindObjectsSortMode\)](#) ,  
Object.DontDestroyOnLoad(Object) , [Object.DestroyObject\(Object, float\)](#) ,  
Object.DestroyObject(Object) , [Object.FindSceneObjectsOfType\(Type\)](#) ,  
[Object.FindObjectsOfTypeIncludingAssets\(Type\)](#) , Object.FindObjectsOfType<T>() ,  
Object.FindObjectsByType<T>(FindObjectsSortMode) , [Object.FindObjectsOfType<T>\(bool\)](#) ,  
Object.FindObjectsByType<T>(FindObjectInactive, FindObjectsSortMode) ,  
Object.FindObjectOfType<T>() , [Object.FindObjectOfType<T>\(bool\)](#) ,  
Object.FindFirstObjectOfType<T>() , Object.FindAnyObjectOfType<T>() ,  
Object.FindFirstObjectOfType<T>(FindObjectInactive) ,  
Object.FindAnyObjectOfType<T>(FindObjectInactive) , [Object.FindObjectsOfTypeAll\(Type\)](#) ,  
[Object.FindObjectOfType\(Type\)](#) , [Object.FindFirstObjectOfType\(Type\)](#) ,

[Object.FindAnyObjectByType\(Type\)](#) , [ObjectFindObjectOfType\(Type, bool\)](#) ,  
[Object.FindFirstObjectByType\(Type, FindObjectsInactive\)](#) ,  
[Object.FindAnyObjectByType\(Type, FindObjectsInactive\)](#) , Object.ToString() , Object.name ,  
Object.hideFlags , [object.Equals\(object, object\)](#) , [object.GetType\(\)](#) , [object.MemberwiseClone\(\)](#) ,  
[object.ReferenceEquals\(object, object\)](#)

## Fields

### Blackboard

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

#### Field Value

[Blackboard](#)

### ExitDialogAfterwards

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

#### Field Value

[bool](#)

### Guid

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

#### Field Value

[string](#)

# MasterDialogue

```
[HideInInspector]  
public Dialogue MasterDialogue
```

Field Value

[Dialogue](#)

# PersonIndex

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

Field Value

[int](#)

# Position

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

Field Value

[Vector2](#)

# State

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

Field Value

[Node.NodeState](#)

# Properties

## DisplayState

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

### Property Value

[bool](#) ↗

## Person

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

### Property Value

Person

## PersonDependent

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

### Property Value

[bool](#) ↗

## ShowInMinimap

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

### Property Value

## Methods

### AddNextNode(Node, int)

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

#### Parameters

nextWillBeAdded [Node](#)

atPort [int](#) ↗

### AddNextNode\_Inline(Node, int)

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

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

#### Parameters

nextWillBeAdded [Node](#)

atPort [int](#) ↗

### Clone()

Use to clone this node.

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

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

Returns

[Node](#)

## GetClassName()

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

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

Returns

[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.

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

Returns

[string](#)

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

## GetNextNodes()

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

Returns

[List](#)<[int](#), [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.

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

Parameters

result [List](#)<(int [portIndex](#), [Node](#) [node](#))>

## GetOutputPortNamesForCreation()

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

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

Returns

[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.

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

Returns

[string](#)

The title as a string.

## OnRemoval()

```
public void OnRemoval()
```

## Pass(params object[])

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

### Parameters

passData object[]

## Pass\_Inline(params object[])

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

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

### Parameters

passData object[]

## Reach()

```
public void Reach()
```

## Reach\_Inline()

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

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

## RemoveNextNode(int)

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

Parameters

atPort [int](#)

## RemoveNextNode\_Inline(int)

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

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

Parameters

atPort [int](#)

## SetState(NodeState)

Use to set the flow state of this node.

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

Parameters

newState [Node.NodeState](#)

## Traverse(Action<Node>)

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

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

Parameters

action [Action](#)<Node>

## Events

### OnPass

`public event Action OnPass`

Event Type

[Action](#)

### OnReach

`public event Action OnReach`

Event Type

[Action](#)

### OnRemove

`public event Action OnRemove`

Event Type

[Action](#)

### OnSetState

`public event Action<Node.NodeState> OnSetState`

Event Type

[Action](#) <Node.NodeState>

## OnValidation

```
public event Action OnValidation
```

Event Type

[Action](#)

# Enum Node.NodeState

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

Assembly: Assembly-CSharp-firstpass.dll

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

```
public enum Node.NodeState
```

## Fields

Current = 1

Past = 2

Unreached = 0

# Class Option

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

Assembly: Assembly-CSharp-firstpass.dll

The type to hold references to dialogue options.

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

Inheritance

[object](#) ← Option

Inherited Members

[object.Equals\(object\)](#) , [object.Equals\(object, object\)](#) , [object.GetHashCode\(\)](#) , [object.GetType\(\)](#) ,  
[object.MemberwiseClone\(\)](#) , [object.ReferenceEquals\(object, object\)](#) , [object.ToString\(\)](#)

## Fields

### AdditionalData

Additional speech data this option contains.

```
public AdditionalSpeechData AdditionalData
```

Field Value

[AdditionalSpeechData](#)

### LeadsTo

The node this option leads to.

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

Field Value

[Node](#)

## ShowIf

The condition checker which decides the visibility of the option.

```
[HideInInspector]  
public VariableComparer ShowIf
```

Field Value

VariableComparer

## Speech

Speech of this option.

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

Field Value

[string](#)

## UseShowIf

Boolean which decides if [ShowIf](#) will be used.

```
[HideInInspector]  
public bool UseShowIf
```

Field Value

[bool](#)

# Methods

## Clone(VariantBank)

Use to get a clone of this option.

```
public Option Clone(VariantBank overrideBank)
```

### Parameters

**overrideBank** VariantBank

### Returns

[Option](#)

# Class RootNode

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

Assembly: Assembly-CSharp-firstpass.dll

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

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

## 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>() , Object.GetInstanceID() ,  
Object.GetHashCode() , [Object.Equals\(object\)](#) , Object.Instantiate(Object, Vector3, Quaternion) ,  
Object.Instantiate(Object, Vector3, Quaternion, Transform) , Object.Instantiate(Object) ,  
Object.Instantiate(Object, Transform) , [Object.Instantiate\(Object, Transform, bool\)](#) ,  
Object.Instantiate<T>(T) , Object.Instantiate<T>(T, Vector3, Quaternion) ,  
Object.Instantiate<T>(T, Vector3, Quaternion, Transform) , Object.Instantiate<T>(T, Transform) ,  
[Object.Instantiate<T>\(T, Transform, bool\)](#) , [Object.Destroy\(Object, float\)](#) , Object.Destroy(Object) ,  
[Object.DestroyImmediate\(Object, bool\)](#) , Object.DestroyImmediate(Object) ,  
[Object.FindObjectsOfType\(Type\)](#) , [Object.FindObjectsOfType\(Type, bool\)](#) ,  
[Object.FindObjectsByType\(Type, FindObjectsSortMode\)](#) ,  
[Object.FindObjectsByType\(Type, FindObjectsInactive, FindObjectsSortMode\)](#) ,  
Object.DontDestroyOnLoad(Object) , [Object.DestroyObject\(Object, float\)](#) ,  
Object.DestroyObject(Object) , [Object.FindSceneObjectsOfType\(Type\)](#) ,  
[Object.FindObjectsOfTypeIncludingAssets\(Type\)](#) , Object.FindObjectsOfType<T>() ,  
Object.FindObjectsByType<T>(FindObjectsSortMode) , [Object.FindObjectsOfType<T>\(bool\)](#) ,  
Object.FindObjectsByType<T>(FindObjectInactive, FindObjectsSortMode) ,  
Object.FindObjectOfType<T>() , [Object.FindObjectOfType<T>\(bool\)](#) ,  
Object.FindFirstObjectOfType<T>() , Object.FindAnyObjectOfType<T>() ,  
Object.FindFirstObjectOfType<T>(FindObjectInactive) ,  
Object.FindAnyObjectOfType<T>(FindObjectInactive) , [Object.FindObjectsOfTypeAll\(Type\)](#) ,

[Object.FindObjectOfType\(Type\)](#) , [Object.FindFirstObjectOfType\(Type\)](#) ,  
[Object.FindAnyObjectOfType\(Type\)](#) , [Object.FindObjectOfType\(Type, bool\)](#) ,  
[Object.FindFirstObjectOfType\(Type, FindObjectsInactive\)](#) ,  
[Object.FindAnyObjectOfType\(Type, FindObjectsInactive\)](#) , Object.ToString() , Object.name ,  
Object.hideFlags , [object.Equals\(object, object\)](#) , [object.GetType\(\)](#) ,  
[object.ReferenceEquals\(object, object\)](#)

## Fields

### Next

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

### Field Value

[Node](#)

## Properties

### DisplayState

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

### Property Value

[bool](#)

## Methods

### AddNextNode\_Inline(Node, int)

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

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

## Parameters

`nextWillBeAdded` [Node](#)

`atPort` [int](#)

## DelayedClone(Dialogue)

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

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

## Parameters

`originalDialogue` [Dialogue](#)

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.

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

## Returns

[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.

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

## Returns

## string

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

## 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.

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

### Parameters

result List<(int portIndex, Node node)>

## GetOutputPortNamesForCreation()

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

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

### Returns

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.

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

### Returns

string

The title as a string.

## Pass\_Inline(params object[])

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

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

Parameters

passData [object](#)[]

## Reach\_Inline()

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

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

## RemoveNextNode\_Inline(int)

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

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

Parameters

atPort [int](#)

## Traverse(Action<Node>)

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

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

Parameters

action [Action](#)<[Node](#)>



# Class StickyNoteNode

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

Assembly: Assembly-CSharp-firstpass.dll

Node which contains a user defined string.

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

## 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>(), Object.GetInstanceID(), Object.GetHashCode(),  
[Object.Equals\(object\)](#) ↗, Object.Instantiate(Object, Vector3, Quaternion),  
Object.Instantiate(Object, Vector3, Quaternion, Transform), Object.Instantiate(Object),  
Object.Instantiate(Object, Transform), [Object.Instantiate\(Object, Transform, bool\)](#) ↗,  
Object.Instantiate<T>(T), Object.Instantiate<T>(T, Vector3, Quaternion),  
Object.Instantiate<T>(T, Vector3, Quaternion, Transform), Object.Instantiate<T>(T, Transform),  
[Object.Instantiate<T>\(T, Transform, bool\)](#) ↗, [Object.Destroy\(Object, float\)](#) ↗, Object.Destroy(Object),  
[Object.DestroyImmediate\(Object, bool\)](#) ↗, Object.DestroyImmediate(Object),  
[Object.FindObjectsOfType\(Type\)](#) ↗, [Object.FindObjectsOfType\(Type, bool\)](#) ↗,  
[Object.FindObjectsByType\(Type, FindObjectsSortMode\)](#) ↗,  
[Object.FindObjectsByType\(Type, FindObjectsInactive, FindObjectsSortMode\)](#) ↗,  
Object.DontDestroyOnLoad(Object), [Object.DestroyObject\(Object, float\)](#) ↗,  
Object.DestroyObject(Object), [Object.FindSceneObjectsOfType\(Type\)](#) ↗,  
[Object.FindObjectsOfTypeIncludingAssets\(Type\)](#) ↗, Object.FindObjectsOfType<T>(),  
Object.FindObjectsByType<T>(FindObjectsSortMode), [Object.FindObjectsOfType<T>\(bool\)](#) ↗,  
Object.FindObjectsByType<T>(FindObjectInactive, FindObjectsSortMode),  
Object.FindObjectOfType<T>(), [Object.FindObjectOfType<T>\(bool\)](#) ↗,  
Object.FindFirstObjectOfType<T>(), Object.FindAnyObjectOfType<T>(),  
Object.FindFirstObjectOfType<T>(FindObjectInactive),

Object.FindAnyObjectByType<T>(FindObjectsInactive) , [Object.FindObjectsOfTypeAll\(Type\)](#) ,  
[Object.FindObjectOfType\(Type\)](#) , [Object.FindFirstObjectByType\(Type\)](#) ,  
[Object.FindAnyObjectByType\(Type\)](#) , [Object.FindObjectOfType\(Type, bool\)](#) ,  
[Object.FindFirstObjectByType\(Type, FindObjectsInactive\)](#) ,  
[Object.FindAnyObjectByType\(Type, FindObjectsInactive\)](#) , Object.ToString() , Object.name ,  
Object.hideFlags , [object.Equals\(object, object\)](#) , [object.GetType\(\)](#) ,  
[object.ReferenceEquals\(object, object\)](#)

## Fields

### Speech

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

#### Field Value

[string](#)

## Properties

### DisplayState

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

#### Property Value

[bool](#)

### ShowInMinimap

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

#### Property Value

[bool](#)

## Methods

### AddNextNode\_Inline(Node, int)

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

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

#### Parameters

nextWillBeAdded [Node](#)

atPort [int](#)

### GetClassName()

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

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

#### Returns

[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.

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

#### Returns

[string](#)

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

## 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.

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

Parameters

result [List](#)<(int [portIndex](#), [Node](#) [node](#))>

## GetOutputPortNamesForCreation()

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

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

Returns

[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.

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

Returns

[string](#)

The title as a string.

## Pass\_Inline(params object[])

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

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

Parameters

passData object[]

## Reach\_Inline()

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

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

## RemoveNextNode\_Inline(int)

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

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

Parameters

atPort int

# Class TitleNode

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

Assembly: Assembly-CSharp-firstpass.dll

Node which is simply [StickyNoteNode](#) but bigger.

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

## 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>(), Object.GetInstanceID(), Object.GetHashCode(),  
[Object.Equals\(object\)](#) ↗, Object.Instantiate(Object, Vector3, Quaternion),  
Object.Instantiate(Object, Vector3, Quaternion, Transform), Object.Instantiate(Object),  
Object.Instantiate(Object, Transform), [Object.Instantiate\(Object, Transform, bool\)](#) ↗,  
Object.Instantiate<T>(T), Object.Instantiate<T>(T, Vector3, Quaternion),  
Object.Instantiate<T>(T, Vector3, Quaternion, Transform), Object.Instantiate<T>(T, Transform),  
[Object.Instantiate<T>\(T, Transform, bool\)](#) ↗, [Object.Destroy\(Object, float\)](#) ↗, Object.Destroy(Object),  
[Object.DestroyImmediate\(Object, bool\)](#) ↗, Object.DestroyImmediate(Object),  
[Object.FindObjectsOfType\(Type\)](#) ↗, [Object.FindObjectsOfType\(Type, bool\)](#) ↗,  
[Object.FindObjectsByType\(Type, FindObjectsSortMode\)](#) ↗,  
[Object.FindObjectsByType\(Type, FindObjectsInactive, FindObjectsSortMode\)](#) ↗,  
Object.DontDestroyOnLoad(Object), [Object.DestroyObject\(Object, float\)](#) ↗,  
Object.DestroyObject(Object), [Object.FindSceneObjectsOfType\(Type\)](#) ↗,  
[Object.FindObjectsOfTypeIncludingAssets\(Type\)](#) ↗, Object.FindObjectsOfType<T>(),  
Object.FindObjectsByType<T>(FindObjectsSortMode), [Object.FindObjectsOfType<T>\(bool\)](#) ↗,  
Object.FindObjectsByType<T>(FindObjectInactive, FindObjectsSortMode),  
Object.FindObjectOfType<T>(), [Object.FindObjectOfType<T>\(bool\)](#) ↗,  
Object.FindFirstObjectOfType<T>(), Object.FindAnyObjectOfType<T>(),  
Object.FindFirstObjectOfType<T>(FindObjectInactive),

Object.FindAnyObjectByType<T>(FindObjectsInactive) , [Object.FindObjectsOfTypeAll\(Type\)](#) ,  
[Object.FindObjectOfType\(Type\)](#) , [Object.FindFirstObjectByType\(Type\)](#) ,  
[Object.FindAnyObjectByType\(Type\)](#) , [Object.FindObjectOfType\(Type, bool\)](#) ,  
[Object.FindFirstObjectByType\(Type, FindObjectsInactive\)](#) ,  
[Object.FindAnyObjectByType\(Type, FindObjectsInactive\)](#) , Object.ToString() , Object.name ,  
Object.hideFlags , [object.Equals\(object, object\)](#) , [object.GetType\(\)](#) ,  
[object.ReferenceEquals\(object, object\)](#)

## Fields

### Speech

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

#### Field Value

[string](#)

## Properties

### DisplayState

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

#### Property Value

[bool](#)

### ShowInMinimap

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

#### Property Value

[bool](#)

## Methods

### AddNextNode\_Inline(Node, int)

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

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

#### Parameters

nextWillBeAdded [Node](#)

atPort [int](#)

### GetClassName()

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

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

#### Returns

[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.

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

#### Returns

[string](#)

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

## 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.

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

Parameters

result [List](#)<(int [portIndex](#), [Node](#) [node](#))>

## GetOutputPortNamesForCreation()

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

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

Returns

[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.

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

Returns

[string](#)

The title as a string.

## Pass\_Inline(params object[])

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

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

Parameters

passData object[]

## Reach\_Inline()

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

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

## RemoveNextNode\_Inline(int)

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

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

Parameters

atPort int