

# Open Accessibility Unreal Plugin

## 0.3

Generated by Doxygen 1.9.5



|   |          |
|---|----------|
| <b>1 Hierarchical Index</b>   | <b>1</b> |
| 1.1 Class Hierarchy   | 1        |
| <b>2 Class Index</b>  | <b>3</b> |
| 2.1 Class List  | 3        |
| <b>3 File Index</b>   | <b>5</b> |
| 3.1 File List   | 5        |
| <b>4 Class Documentation</b>  | <b>9</b> |
| 4.1 <a href="#">OpenAccessibilityPy.Audio.AudioResampler Class Reference</a>                    | 9        |
| 4.1.1 Detailed Description  | 9        |
| 4.1.2 Constructor & Destructor Documentation  | 9        |
| 4.1.2.1 <code>__init__()</code>   | 9        |
| 4.1.2.2 <code>__del__()</code>  | 10       |
| 4.1.3 Member Function Documentation   | 10       |
| 4.1.3.1 <code>resample()</code>   | 10       |
| 4.2 <a href="#">OpenAccessibilityPy.CommunicationServer.CommunicationServer Class Reference</a> | 11       |
| 4.2.1 Detailed Description  | 12       |
| 4.2.2 Constructor & Destructor Documentation  | 12       |
| 4.2.2.1 <code>__init__()</code>   | 12       |
| 4.2.2.2 <code>__del__()</code>  | 13       |
| 4.2.3 Member Function Documentation   | 13       |
| 4.2.3.1 <code>EventOccured()</code>   | 13       |
| 4.2.3.2 <code>ReceiveJSON()</code>  | 14       |
| 4.2.3.3 <code>ReceiveMultipart()</code>   | 14       |
| 4.2.3.4 <code>ReceiveNDArray()</code>   | 14       |
| 4.2.3.5 <code>ReceiveNDArrayWithMeta()</code>   | 15       |
| 4.2.3.6 <code>ReceiveString()</code>  | 16       |
| 4.2.3.7 <code>RecieveRaw()</code>   | 16       |
| 4.2.3.8 <code>SendJSON()</code>   | 16       |
| 4.2.3.9 <code>SendMultipart()</code>  | 17       |
| 4.2.3.10 <code>SendMultipartWithMeta()</code>   | 17       |
| 4.2.3.11 <code>SendNDArray()</code>   | 18       |
| 4.2.3.12 <code>SendNDArrayWithMeta()</code>   | 19       |
| 4.2.3.13 <code>SendString()</code>  | 19       |
| 4.2.4 Member Data Documentation   | 20       |
| 4.2.4.1 <code>context</code>  | 20       |
| 4.2.4.2 <code>poller</code>   | 20       |
| 4.2.4.3 <code>poller_timeout_time</code>  | 20       |
| 4.2.4.4 <code>recv_socket</code>  | 20       |
| 4.2.4.5 <code>recv_socket_context</code>  | 20       |
| 4.2.4.6 <code>send_socket_context</code>  | 21       |

|   |    |
|---|----|
| 4.3 FAccessibilityNodeFactory Class Reference   | 21 |
| 4.3.1 Detailed Description                      | 21 |
| 4.3.2 Constructor & Destructor Documentation    | 21 |
| 4.3.2.1 FAccessibilityNodeFactory()             | 21 |
| 4.3.2.2 ~FAccessibilityNodeFactory()            | 22 |
| 4.3.3 Member Function Documentation             | 22 |
| 4.3.3.1 CreateNode()                            | 22 |
| 4.3.3.2 SetSharedPtr()                          | 23 |
| 4.3.3.3 WrapNodeWidget()                        | 23 |
| 4.3.3.4 WrapPinWidget()                         | 24 |
| 4.4 FAssetAccessibilityRegistry Class Reference | 25 |
| 4.4.1 Detailed Description                      | 26 |
| 4.4.2 Constructor & Destructor Documentation    | 26 |
| 4.4.2.1 FAssetAccessibilityRegistry()           | 26 |
| 4.4.2.2 ~FAssetAccessibilityRegistry()          | 26 |
| 4.4.3 Member Function Documentation             | 26 |
| 4.4.3.1 GetAllGraphIndexes() [1/2]              | 27 |
| 4.4.3.2 GetAllGraphIndexes() [2/2]              | 27 |
| 4.4.3.3 GetAllGraphKeyIndexes() [1/2]           | 27 |
| 4.4.3.4 GetAllGraphKeyIndexes() [2/2]           | 28 |
| 4.4.3.5 GetGraphIndexer()                       | 28 |
| 4.4.3.6 IsGameWorldAssetRegistered()            | 28 |
| 4.4.3.7 IsGraphAssetRegistered()                | 29 |
| 4.4.3.8 RegisterGameWorldAsset()                | 29 |
| 4.4.3.9 RegisterGraphAsset() [1/2]              | 30 |
| 4.4.3.10 RegisterGraphAsset() [2/2]             | 30 |
| 4.4.3.11 UnregisterGameWorldAsset()             | 31 |
| 4.4.3.12 UnregisterGraphAsset()                 | 32 |
| 4.4.4 Member Data Documentation                 | 32 |
| 4.4.4.1 GraphAssetIndex                         | 33 |
| 4.5 FAudioManagerSettings Struct Reference      | 33 |
| 4.5.1 Detailed Description                      | 33 |
| 4.5.2 Constructor & Destructor Documentation    | 33 |
| 4.5.2.1 FAudioManagerSettings()                 | 33 |
| 4.5.3 Member Data Documentation                 | 33 |
| 4.5.3.1 LevelThreshold                          | 34 |
| 4.5.3.2 SaveName                                | 34 |
| 4.5.3.3 SavePath                                | 34 |
| 4.6 FGraphIndexer Class Reference               | 34 |
| 4.6.1 Detailed Description                      | 35 |
| 4.6.2 Constructor & Destructor Documentation    | 35 |
| 4.6.2.1 FGraphIndexer() [1/2]                   | 35 |

|   |    |
|---|----|
| 4.6.2.2 FGraphIndexer() [2/2]                               | 36 |
| 4.6.2.3 ~FGraphIndexer()                                    | 36 |
| 4.6.3 Member Function Documentation                         | 36 |
| 4.6.3.1 AddNode() [1/2]                                     | 36 |
| 4.6.3.2 AddNode() [2/2]                                     | 37 |
| 4.6.3.3 ContainsKey()                                       | 37 |
| 4.6.3.4 ContainsNode() [1/2]                                | 38 |
| 4.6.3.5 ContainsNode() [2/2]                                | 38 |
| 4.6.3.6 GetKey() [1/2]                                      | 38 |
| 4.6.3.7 GetKey() [2/2]                                      | 40 |
| 4.6.3.8 GetNode() [1/2]                                     | 41 |
| 4.6.3.9 GetNode() [2/2]                                     | 41 |
| 4.6.3.10 GetOrAddNode() [1/2]                               | 41 |
| 4.6.3.11 GetOrAddNode() [2/2]                               | 42 |
| 4.6.3.12 GetPin() [1/2]                                     | 42 |
| 4.6.3.13 GetPin() [2/2]                                     | 43 |
| 4.6.3.14 OnGraphEvent()                                     | 43 |
| 4.6.3.15 OnGraphRebuild()                                   | 44 |
| 4.6.3.16 RemoveNode() [1/2]                                 | 44 |
| 4.6.3.17 RemoveNode() [2/2]                                 | 45 |
| 4.6.4 Member Data Documentation                             | 45 |
| 4.6.4.1 AvailableIndices                                    | 45 |
| 4.6.4.2 IndexMap  | 46 |
| 4.6.4.3 LinkedGraph   | 46 |
| 4.6.4.4 NodeSet   | 46 |
| 4.6.4.5 OnGraphChangedHandle                                | 46 |
| 4.7 FGraphLocomotionChunk Struct Reference                  | 46 |
| 4.7.1 Detailed Description                                  | 47 |
| 4.7.2 Member Function Documentation                         | 47 |
| 4.7.2.1 GetChunkBottomRight()                               | 47 |
| 4.7.2.2 GetChunkBounds()                                    | 47 |
| 4.7.2.3 GetChunkTopLeft()                                   | 47 |
| 4.7.2.4 SetChunkBounds()                                    | 48 |
| 4.7.2.5 SetVisColor()                                       | 48 |
| 4.7.3 Member Data Documentation                             | 48 |
| 4.7.3.1 BottomRight   | 48 |
| 4.7.3.2 ChunkIndexer  | 48 |
| 4.7.3.3 ChunkVisWidget                                      | 49 |
| 4.7.3.4 ChunkWidget   | 49 |
| 4.7.3.5 TopLeft   | 49 |
| 4.8 FIndexer< KeyType, ValueType > Class Template Reference | 49 |
| 4.8.1 Detailed Description                                  | 50 |

|  |    |
|--|----|
| 4.8.2 Constructor & Destructor Documentation               | 50 |
| 4.8.2.1 Findexer()   | 50 |
| 4.8.2.2 ~Findexer()  | 50 |
| 4.8.3 Member Function Documentation                        | 51 |
| 4.8.3.1 AddValue() [1/2]                                   | 51 |
| 4.8.3.2 AddValue() [2/2]                                   | 51 |
| 4.8.3.3 ContainsKey()                                      | 52 |
| 4.8.3.4 ContainsValue()                                    | 52 |
| 4.8.3.5 Empty()  | 53 |
| 4.8.3.6 GetAvailableKey() [1/2]                            | 53 |
| 4.8.3.7 GetAvailableKey() [2/2]                            | 53 |
| 4.8.3.8 GetKey() [1/2]                                     | 54 |
| 4.8.3.9 GetKey() [2/2]                                     | 54 |
| 4.8.3.10 GetKeyOrAddValue() [1/2]                          | 55 |
| 4.8.3.11 GetKeyOrAddValue() [2/2]                          | 55 |
| 4.8.3.12 GetValue() [1/2]                                  | 56 |
| 4.8.3.13 GetValue() [2/2]                                  | 56 |
| 4.8.3.14 IsEmpty()   | 57 |
| 4.8.3.15 Num() [1/2]                                       | 57 |
| 4.8.3.16 Num() [2/2]                                       | 57 |
| 4.8.3.17 RemoveValue() [1/2]                               | 58 |
| 4.8.3.18 RemoveValue() [2/2]                               | 58 |
| 4.8.3.19 Reset()   | 59 |
| 4.8.4 Member Data Documentation                            | 59 |
| 4.8.4.1 AvailableIndexes                                   | 59 |
| 4.8.4.2 IndexMap   | 59 |
| 4.9 FOpenAccessibilityAnalyticsModule Class Reference      | 60 |
| 4.9.1 Detailed Description                                 | 60 |
| 4.9.2 Member Function Documentation                        | 60 |
| 4.9.2.1 DumpTick()   | 60 |
| 4.9.2.2 Get()  | 61 |
| 4.9.2.3 LogEvent()   | 61 |
| 4.9.2.4 ShutdownModule()                                   | 62 |
| 4.9.2.5 StartupModule()                                    | 62 |
| 4.9.2.6 SupportsDynamicReloading()                         | 62 |
| 4.10 FOpenAccessibilityCommunicationModule Class Reference | 62 |
| 4.10.1 Detailed Description                                | 63 |
| 4.10.2 Member Function Documentation                       | 63 |
| 4.10.2.1 Get()   | 63 |
| 4.10.2.2 HandleKeyDownEvent()                              | 64 |
| 4.10.2.3 ShutdownModule()                                  | 64 |
| 4.10.2.4 StartupModule()                                   | 64 |

|   |    |
|---|----|
| 4.10.2.5 SupportsDynamicReloading()           | 65 |
| 4.10.2.6 Tick()                               | 65 |
| 4.10.2.7 TranscribeWaveForm()                 | 65 |
| 4.10.3 Member Data Documentation              | 66 |
| 4.10.3.1 AudioManager                         | 66 |
| 4.10.3.2 OnTranscriptionRecieved              | 66 |
| 4.10.3.3 PhraseTree                           | 67 |
| 4.10.3.4 PhraseTreeUtils                      | 67 |
| 4.10.3.5 SocketServer                         | 67 |
| 4.11 FOpenAccessibilityModule Class Reference | 67 |
| 4.11.1 Detailed Description                   | 68 |
| 4.11.2 Member Function Documentation          | 68 |
| 4.11.2.1 Get()                                | 68 |
| 4.11.2.2 ShutdownModule()                     | 68 |
| 4.11.2.3 StartupModule()                      | 69 |
| 4.11.2.4 SupportsDynamicReloading()           | 69 |
| 4.11.3 Member Data Documentation              | 69 |
| 4.11.3.1 AccessibilityNodeFactory             | 69 |
| 4.11.3.2 AssetAccessibilityRegistry           | 70 |
| 4.12 TabUtils::FOpenArea Class Reference      | 70 |
| 4.12.1 Detailed Description                   | 70 |
| 4.12.2 Member Function Documentation          | 70 |
| 4.12.2.1 GetChildNodes()                      | 70 |
| 4.13 TabUtils::FOpenSplitter Class Reference  | 71 |
| 4.13.1 Detailed Description                   | 71 |
| 4.13.2 Member Function Documentation          | 71 |
| 4.13.2.1 GetChildNodes()                      | 71 |
| 4.14 TabUtils::FOpenStack Class Reference     | 71 |
| 4.14.1 Detailed Description                   | 72 |
| 4.14.2 Member Function Documentation          | 72 |
| 4.14.2.1 GetTabs()                            | 72 |
| 4.15 FPanelViewPosition Struct Reference      | 72 |
| 4.15.1 Detailed Description                   | 72 |
| 4.15.2 Constructor & Destructor Documentation | 72 |
| 4.15.2.1 FPanelViewPosition() [1/2]           | 73 |
| 4.15.2.2 FPanelViewPosition() [2/2]           | 73 |
| 4.15.3 Member Function Documentation          | 73 |
| 4.15.3.1 operator!=(()) [1/2]                 | 73 |
| 4.15.3.2 operator!=(()) [2/2]                 | 73 |
| 4.15.4 Member Data Documentation              | 73 |
| 4.15.4.1 BotRight                             | 74 |
| 4.15.4.2 TopLeft                              | 74 |

|   |    |
|---|----|
| 4.16 FParseRecord Struct Reference                | 74 |
| 4.16.1 Detailed Description                       | 75 |
| 4.16.2 Constructor & Destructor Documentation     | 75 |
| 4.16.2.1 FParseRecord() [1/2]                     | 76 |
| 4.16.2.2 FParseRecord() [2/2]                     | 76 |
| 4.16.2.3 ~FParseRecord()                          | 76 |
| 4.16.3 Member Function Documentation              | 76 |
| 4.16.3.1 AddPhraseInput()                         | 76 |
| 4.16.3.2 AddPhraseString()                        | 77 |
| 4.16.3.3 GetContextObj() [1/4]                    | 77 |
| 4.16.3.4 GetContextObj() [2/4]                    | 77 |
| 4.16.3.5 GetContextObj() [3/4]                    | 78 |
| 4.16.3.6 GetContextObj() [4/4]                    | 78 |
| 4.16.3.7 GetContextStack() [1/2]                  | 79 |
| 4.16.3.8 GetContextStack() [2/2]                  | 79 |
| 4.16.3.9 GetPhraseInput() [1/4]                   | 79 |
| 4.16.3.10 GetPhraseInput() [2/4]                  | 80 |
| 4.16.3.11 GetPhraseInput() [3/4]                  | 80 |
| 4.16.3.12 GetPhraseInput() [4/4]                  | 81 |
| 4.16.3.13 GetPhraseInputs() [1/2]                 | 81 |
| 4.16.3.14 GetPhraseInputs() [2/2]                 | 82 |
| 4.16.3.15 GetPhraseString()                       | 82 |
| 4.16.3.16 HasContextObj() [1/2]                   | 83 |
| 4.16.3.17 HasContextObj() [2/2]                   | 83 |
| 4.16.3.18 PopContextObj() [1/2]                   | 83 |
| 4.16.3.19 PopContextObj() [2/2]                   | 84 |
| 4.16.3.20 PushContextObj()                        | 84 |
| 4.16.3.21 RemoveContextObj()                      | 84 |
| 4.16.3.22 RemovePhraseInput()                     | 85 |
| 4.16.4 Friends And Related Function Documentation | 85 |
| 4.16.4.1 FPhraseTree                              | 85 |
| 4.16.5 Member Data Documentation                  | 85 |
| 4.16.5.1 ContextObjectStack                       | 85 |
| 4.16.5.2 PhraseInputs                             | 86 |
| 4.16.5.3 PhraseRecord                             | 86 |
| 4.17 FParseResult Struct Reference                | 86 |
| 4.17.1 Detailed Description                       | 86 |
| 4.17.2 Constructor & Destructor Documentation     | 87 |
| 4.17.2.1 FParseResult() [1/3]                     | 87 |
| 4.17.2.2 FParseResult() [2/3]                     | 87 |
| 4.17.2.3 FParseResult() [3/3]                     | 87 |
| 4.17.3 Member Data Documentation                  | 87 |



|   |     |
|---|-----|
| 4.17.3.1 ReachedNode  | 87  |
| 4.17.3.2 Result   | 88  |
| 4.18 FPhrase2DDirectionalInputNode Class Reference                      | 88  |
| 4.18.1 Detailed Description   | 88  |
| 4.18.2 Constructor & Destructor Documentation                           | 88  |
| 4.18.2.1 FPhrase2DDirectionalInputNode() [1/5]                          | 89  |
| 4.18.2.2 FPhrase2DDirectionalInputNode() [2/5]                          | 89  |
| 4.18.2.3 FPhrase2DDirectionalInputNode() [3/5]                          | 89  |
| 4.18.2.4 FPhrase2DDirectionalInputNode() [4/5]                          | 89  |
| 4.18.2.5 FPhrase2DDirectionalInputNode() [5/5]                          | 90  |
| 4.19 FPhraseContextMenuNode< ContextMenuType > Class Template Reference | 90  |
| 4.19.1 Detailed Description   | 91  |
| 4.19.2 Constructor & Destructor Documentation                           | 91  |
| 4.19.2.1 FPhraseContextMenuNode() [1/7]                                 | 91  |
| 4.19.2.2 FPhraseContextMenuNode() [2/7]                                 | 92  |
| 4.19.2.3 FPhraseContextMenuNode() [3/7]                                 | 92  |
| 4.19.2.4 FPhraseContextMenuNode() [4/7]                                 | 92  |
| 4.19.2.5 FPhraseContextMenuNode() [5/7]                                 | 92  |
| 4.19.2.6 FPhraseContextMenuNode() [6/7]                                 | 93  |
| 4.19.2.7 FPhraseContextMenuNode() [7/7]                                 | 93  |
| 4.19.2.8 ~FPhraseContextMenuNode()                                      | 93  |
| 4.19.3 Member Function Documentation                                    | 93  |
| 4.19.3.1 ConstructContextChildren()                                     | 93  |
| 4.19.3.2 CreateContextObject()  | 94  |
| 4.19.3.3 HasContextObject()   | 95  |
| 4.19.3.4 ParsePhrase()  | 95  |
| 4.19.3.5 ParsePhraseAsContext()   | 96  |
| 4.19.4 Member Data Documentation  | 97  |
| 4.19.4.1 ContextMenuScalar  | 97  |
| 4.19.4.2 OnGetMenu  | 97  |
| 4.20 FPhraseContextNode< ContextType > Class Template Reference         | 97  |
| 4.20.1 Detailed Description   | 98  |
| 4.20.2 Constructor & Destructor Documentation                           | 98  |
| 4.20.2.1 FPhraseContextNode() [1/3]                                     | 98  |
| 4.20.2.2 FPhraseContextNode() [2/3]                                     | 98  |
| 4.20.2.3 FPhraseContextNode() [3/3]                                     | 99  |
| 4.20.2.4 ~FPhraseContextNode()  | 99  |
| 4.20.3 Member Function Documentation                                    | 99  |
| 4.20.3.1 ConstructContextChildren()                                     | 99  |
| 4.20.3.2 CreateContextObject()  | 100 |
| 4.20.3.3 HasContextObject()   | 100 |
| 4.20.3.4 ParsePhrase()  | 101 |

|   |     |
|---|-----|
| 4.20.3.5 ParsePhraseAsContext()                             | 101 |
| 4.21 FPhraseDirectionalInputNode Class Reference            | 102 |
| 4.21.1 Detailed Description                                 | 102 |
| 4.21.2 Constructor & Destructor Documentation               | 103 |
| 4.21.2.1 FPhraseDirectionalInputNode() [1/5]                | 103 |
| 4.21.2.2 FPhraseDirectionalInputNode() [2/5]                | 103 |
| 4.21.2.3 FPhraseDirectionalInputNode() [3/5]                | 103 |
| 4.21.2.4 FPhraseDirectionalInputNode() [4/5]                | 103 |
| 4.21.2.5 FPhraseDirectionalInputNode() [5/5]                | 104 |
| 4.22 FPhraseEnumInputNode< TEnum > Class Template Reference | 104 |
| 4.22.1 Detailed Description                                 | 105 |
| 4.22.2 Constructor & Destructor Documentation               | 105 |
| 4.22.2.1 FPhraseEnumInputNode() [1/5]                       | 105 |
| 4.22.2.2 FPhraseEnumInputNode() [2/5]                       | 105 |
| 4.22.2.3 FPhraseEnumInputNode() [3/5]                       | 105 |
| 4.22.2.4 FPhraseEnumInputNode() [4/5]                       | 106 |
| 4.22.2.5 FPhraseEnumInputNode() [5/5]                       | 106 |
| 4.22.2.6 ~FPhraseEnumInputNode()                            | 106 |
| 4.22.3 Member Function Documentation                        | 106 |
| 4.22.3.1 MeetsInputRequirements()                           | 106 |
| 4.22.3.2 RecordInput()                                      | 107 |
| 4.23 FPhraseEventNode Class Reference                       | 108 |
| 4.23.1 Detailed Description                                 | 108 |
| 4.23.2 Constructor & Destructor Documentation               | 108 |
| 4.23.2.1 FPhraseEventNode() [1/3]                           | 109 |
| 4.23.2.2 FPhraseEventNode() [2/3]                           | 109 |
| 4.23.2.3 FPhraseEventNode() [3/3]                           | 109 |
| 4.23.2.4 ~FPhraseEventNode()                                | 109 |
| 4.23.3 Member Function Documentation                        | 109 |
| 4.23.3.1 IsLeafNode()                                       | 110 |
| 4.23.3.2 ParsePhrase()                                      | 110 |
| 4.23.3.3 RequiresPhrase() [1/2]                             | 110 |
| 4.23.3.4 RequiresPhrase() [2/2]                             | 111 |
| 4.24 FPhraseInputNode< InputType > Class Template Reference | 111 |
| 4.24.1 Detailed Description                                 | 112 |
| 4.24.2 Constructor & Destructor Documentation               | 113 |
| 4.24.2.1 FPhraseInputNode() [1/5]                           | 113 |
| 4.24.2.2 FPhraseInputNode() [2/5]                           | 113 |
| 4.24.2.3 FPhraseInputNode() [3/5]                           | 113 |
| 4.24.2.4 FPhraseInputNode() [4/5]                           | 114 |
| 4.24.2.5 FPhraseInputNode() [5/5]                           | 114 |
| 4.24.2.6 ~FPhraseInputNode()                                | 114 |

|   |     |
|---|-----|
| 4.24.3 Member Function Documentation            | 114 |
| 4.24.3.1 MeetsInputRequirements()               | 114 |
| 4.24.3.2 ParsePhrase()                          | 115 |
| 4.24.3.3 RecordInput()                          | 116 |
| 4.24.3.4 RequiresPhrase() [1/2]                 | 116 |
| 4.24.3.5 RequiresPhrase() [2/2]                 | 117 |
| 4.24.4 Member Data Documentation                | 117 |
| 4.24.4.1 OnInputReceived                        | 118 |
| 4.25 FPhraseNode Class Reference                | 118 |
| 4.25.1 Detailed Description                     | 119 |
| 4.25.2 Constructor & Destructor Documentation   | 119 |
| 4.25.2.1 FPhraseNode() [1/4]                    | 119 |
| 4.25.2.2 FPhraseNode() [2/4]                    | 120 |
| 4.25.2.3 FPhraseNode() [3/4]                    | 120 |
| 4.25.2.4 FPhraseNode() [4/4]                    | 120 |
| 4.25.2.5 ~FPhraseNode()                         | 120 |
| 4.25.3 Member Function Documentation            | 121 |
| 4.25.3.1 BindChildNode()                        | 121 |
| 4.25.3.2 BindChildNodeForce()                   | 121 |
| 4.25.3.3 BindChildrenNodes()                    | 122 |
| 4.25.3.4 BindChildrenNodesForce()               | 122 |
| 4.25.3.5 CanBindChild()                         | 123 |
| 4.25.3.6 HasLeafChild() [1/2]                   | 123 |
| 4.25.3.7 HasLeafChild() [2/2]                   | 124 |
| 4.25.3.8 IsLeafNode()                           | 124 |
| 4.25.3.9 ParseChildren()                        | 124 |
| 4.25.3.10 ParsePhrase()                         | 125 |
| 4.25.3.11 ParsePhraseAsContext()                | 126 |
| 4.25.3.12 ParsePhraselfRequired()               | 126 |
| 4.25.3.13 RequiresPhrase() [1/2]                | 127 |
| 4.25.3.14 RequiresPhrase() [2/2]                | 127 |
| 4.25.4 Member Data Documentation                | 128 |
| 4.25.4.1 bHasLeafChild                          | 128 |
| 4.25.4.2 BoundPhrase                            | 128 |
| 4.25.4.3 ChildNodes                             | 128 |
| 4.25.4.4 OnPhraseParsed                         | 128 |
| 4.25.4.5 ParentNode                             | 128 |
| 4.26 FPhrasePositionalInputNode Class Reference | 129 |
| 4.26.1 Detailed Description                     | 129 |
| 4.26.2 Constructor & Destructor Documentation   | 129 |
| 4.26.2.1 FPhrasePositionalInputNode() [1/5]     | 129 |
| 4.26.2.2 FPhrasePositionalInputNode() [2/5]     | 130 |

|   |     |
|---|-----|
| 4.26.2.3 FPhrasePositionalInputNode() [ 3/5] . . . . .  | 130 |
| 4.26.2.4 FPhrasePositionalInputNode() [ 4/5] . . . . .  | 130 |
| 4.26.2.5 FPhrasePositionalInputNode() [ 5/5] . . . . .  | 130 |
| 4.27 FPhraseScrollInputNode Class Reference . . . . .   | 131 |
| 4.27.1 Detailed Description . . . . .                   | 131 |
| 4.27.2 Constructor & Destructor Documentation . . . . . | 131 |
| 4.27.2.1 FPhraseScrollInputNode() [ 1/5] . . . . .      | 131 |
| 4.27.2.2 FPhraseScrollInputNode() [ 2/5] . . . . .      | 132 |
| 4.27.2.3 FPhraseScrollInputNode() [ 3/5] . . . . .      | 132 |
| 4.27.2.4 FPhraseScrollInputNode() [ 4/5] . . . . .      | 132 |
| 4.27.2.5 FPhraseScrollInputNode() [ 5/5] . . . . .      | 132 |
| 4.28 FPhraseStringInputNode Class Reference . . . . .   | 133 |
| 4.28.1 Detailed Description . . . . .                   | 133 |
| 4.28.2 Constructor & Destructor Documentation . . . . . | 133 |
| 4.28.2.1 FPhraseStringInputNode() [ 1/4] . . . . .      | 134 |
| 4.28.2.2 FPhraseStringInputNode() [ 2/4] . . . . .      | 134 |
| 4.28.2.3 FPhraseStringInputNode() [ 3/4] . . . . .      | 134 |
| 4.28.2.4 FPhraseStringInputNode() [ 4/4] . . . . .      | 134 |
| 4.28.2.5 ~FPhraseStringInputNode() . . . . .            | 135 |
| 4.28.3 Member Function Documentation . . . . .          | 135 |
| 4.28.3.1 MeetsInputRequirements() . . . . .             | 135 |
| 4.28.3.2 RecordInput() . . . . .                        | 135 |
| 4.29 FPhraseTree Class Reference . . . . .              | 136 |
| 4.29.1 Detailed Description . . . . .                   | 137 |
| 4.29.2 Constructor & Destructor Documentation . . . . . | 137 |
| 4.29.2.1 FPhraseTree() . . . . .                        | 137 |
| 4.29.2.2 ~FPhraseTree() . . . . .                       | 137 |
| 4.29.3 Member Function Documentation . . . . .          | 137 |
| 4.29.3.1 BindBranch() . . . . .                         | 137 |
| 4.29.3.2 BindBranches() . . . . .                       | 138 |
| 4.29.3.3 GetContextManager() . . . . .                  | 138 |
| 4.29.3.4 ParsePhrase() . . . . .                        | 138 |
| 4.29.3.5 ParseTranscription() . . . . .                 | 139 |
| 4.29.3.6 Tick() . . . . .                               | 141 |
| 4.30 FPhraseTreeBranchBind Struct Reference . . . . .   | 141 |
| 4.30.1 Detailed Description . . . . .                   | 142 |
| 4.30.2 Constructor & Destructor Documentation . . . . . | 142 |
| 4.30.2.1 FPhraseTreeBranchBind() [ 1/2] . . . . .       | 142 |
| 4.30.2.2 FPhraseTreeBranchBind() [ 2/2] . . . . .       | 142 |
| 4.30.2.3 ~FPhraseTreeBranchBind() . . . . .             | 142 |
| 4.30.3 Member Data Documentation . . . . .              | 142 |
| 4.30.3.1 BranchRoot . . . . .                           | 143 |

|   |     |
|---|-----|
| 4.30.3.2 StartNode                                | 143 |
| 4.31 FPhraseTreeContextManager Struct Reference   | 143 |
| 4.31.1 Detailed Description                       | 144 |
| 4.31.2 Constructor & Destructor Documentation     | 144 |
| 4.31.2.1 FPhraseTreeContextManager()              | 144 |
| 4.31.2.2 ~FPhraseTreeContextManager()             | 144 |
| 4.31.3 Member Function Documentation              | 144 |
| 4.31.3.1 GetContextStack()                        | 144 |
| 4.31.3.2 HasContextObject()                       | 144 |
| 4.31.3.3 HasContextObjects()                      | 145 |
| 4.31.3.4 IsEmpty()                                | 145 |
| 4.31.3.5 PeekContextObject() [1/2]                | 145 |
| 4.31.3.6 PeekContextObject() [2/2]                | 145 |
| 4.31.3.7 PopContextObject() [1/3]                 | 146 |
| 4.31.3.8 PopContextObject() [2/3]                 | 146 |
| 4.31.3.9 PopContextObject() [3/3]                 | 146 |
| 4.31.3.10 PushContextObject()                     | 147 |
| 4.31.4 Friends And Related Function Documentation | 147 |
| 4.31.4.1 FPhraseTree                              | 147 |
| 4.32 FSocketCommunicationServer Class Reference   | 147 |
| 4.32.1 Detailed Description                       | 149 |
| 4.32.2 Constructor & Destructor Documentation     | 149 |
| 4.32.2.1 FSocketCommunicationServer()             | 149 |
| 4.32.2.2 ~FSocketCommunicationServer()            | 150 |
| 4.32.3 Member Function Documentation              | 150 |
| 4.32.3.1 DeserializeJSON()                        | 150 |
| 4.32.3.2 EventOccured()                           | 150 |
| 4.32.3.3 RecvArray()                              | 151 |
| 4.32.3.4 RecvJson()                               | 152 |
| 4.32.3.5 RecvMultipartWithMeta()                  | 152 |
| 4.32.3.6 RecvString()                             | 153 |
| 4.32.3.7 RecvStringMultipart()                    | 154 |
| 4.32.3.8 RecvStringMultipartWithMeta()            | 155 |
| 4.32.3.9 SendArrayBuffer() [1/3]                  | 155 |
| 4.32.3.10 SendArrayBuffer() [2/3]                 | 156 |
| 4.32.3.11 SendArrayBuffer() [3/3]                 | 156 |
| 4.32.3.12 SendArrayMessage() [1/3]                | 157 |
| 4.32.3.13 SendArrayMessage() [2/3]                | 158 |
| 4.32.3.14 SendArrayMessage() [3/3]                | 158 |
| 4.32.3.15 SendArrayMessageWithMeta() [1/3]        | 159 |
| 4.32.3.16 SendArrayMessageWithMeta() [2/3]        | 160 |
| 4.32.3.17 SendArrayMessageWithMeta() [3/3]        | 161 |

|   |     |
|---|-----|
| 4.32.3.18 SendJsonBuffer()  | 161 |
| 4.32.3.19 SendStringBuffer()  | 162 |
| 4.32.3.20 SerializeJSON()   | 163 |
| 4.32.4 Member Data Documentation  | 163 |
| 4.32.4.1 Context  | 163 |
| 4.32.4.2 Poller   | 163 |
| 4.32.4.3 PollTimeout  | 164 |
| 4.32.4.4 RecvAddress  | 164 |
| 4.32.4.5 RecvSocket   | 164 |
| 4.32.4.6 SendAddress  | 164 |
| 4.32.4.7 SendSocket   | 164 |
| 4.33 FTranscriptionVisualizer Class Reference                                     | 165 |
| 4.33.1 Detailed Description   | 165 |
| 4.33.2 Constructor & Destructor Documentation                                     | 165 |
| 4.33.2.1 FTranscriptionVisualizer()   | 166 |
| 4.33.2.2 ~FTranscriptionVisualizer()  | 166 |
| 4.33.3 Member Function Documentation  | 166 |
| 4.33.3.1 ConstructVisualizer()  | 166 |
| 4.33.3.2 GetDisplayVisualizerPosition()   | 167 |
| 4.33.3.3 GetTopScreenVisualizerPosition()   | 167 |
| 4.33.3.4 MoveVisualizer()   | 167 |
| 4.33.3.5 OnTranscriptionRecieved()  | 168 |
| 4.33.3.6 RegisterTicker()   | 168 |
| 4.33.3.7 ReparentWindow()   | 168 |
| 4.33.3.8 Tick()   | 169 |
| 4.33.3.9 UnregisterTicker()   | 169 |
| 4.33.3.10 UpdateVisualizer()  | 170 |
| 4.33.4 Member Data Documentation  | 170 |
| 4.33.4.1 TickDelegateHandle   | 170 |
| 4.33.4.2 VisContent   | 170 |
| 4.33.4.3 VisWindow  | 170 |
| 4.34 UAccessibilityGraphEditorContext::FTreeViewTickRequirements Struct Reference | 171 |
| 4.34.1 Detailed Description   | 171 |
| 4.34.2 Constructor & Destructor Documentation                                     | 171 |
| 4.34.2.1 FTreeViewTickRequirements()  | 171 |
| 4.34.3 Member Data Documentation  | 171 |
| 4.34.3.1 PrevNumGeneratedChildren   | 171 |
| 4.34.3.2 PrevNumItemsBeingObserved  | 171 |
| 4.34.3.3 PrevScrollDistance   | 172 |
| 4.34.3.4 PrevSearchText   | 172 |
| 4.35 IPhraseContextNodeBase Class Reference                                       | 172 |
| 4.35.1 Detailed Description   | 172 |

|  |     |
|--|-----|
| 4.35.2 Member Function Documentation                         | 173 |
| 4.35.2.1 ConstructContextChildren()                          | 173 |
| 4.35.2.2 CreateContextObject()                               | 173 |
| 4.35.2.3 HasContextObject()                                  | 173 |
| 4.36 IPhraseNodeBase Class Reference                         | 174 |
| 4.36.1 Detailed Description                                  | 174 |
| 4.36.2 Member Function Documentation                         | 174 |
| 4.36.2.1 HasLeafChild()                                      | 174 |
| 4.36.2.2 IsLeafNode()  | 175 |
| 4.36.2.3 ParsePhrase()                                       | 175 |
| 4.36.2.4 ParsePhraseAsContext()                              | 175 |
| 4.36.2.5 RequiresPhrase()                                    | 176 |
| 4.37 OpenAccessibilityPy.Logging.LogLevel Class Reference    | 176 |
| 4.37.1 Detailed Description                                  | 176 |
| 4.37.2 Member Data Documentation                             | 176 |
| 4.37.2.1 ERROR   | 177 |
| 4.37.2.2 INFO  | 177 |
| 4.37.2.3 WARNING   | 177 |
| 4.38 NumericParser Class Reference                           | 177 |
| 4.38.1 Detailed Description                                  | 177 |
| 4.38.2 Member Function Documentation                         | 177 |
| 4.38.2.1 IsValidNumeric()                                    | 177 |
| 4.38.2.2 StringToNumeric()                                   | 178 |
| 4.39 OAEEditorAccessibilityManager Class Reference           | 178 |
| 4.39.1 Detailed Description                                  | 178 |
| 4.39.2 Constructor & Destructor Documentation                | 179 |
| 4.39.2.1 OAEEditorAccessibilityManager()                     | 179 |
| 4.39.2.2 ~OAEEditorAccessibilityManager()                    | 179 |
| 4.40 OpenAccessibility Class Reference                       | 179 |
| 4.40.1 Detailed Description                                  | 179 |
| 4.40.2 Constructor & Destructor Documentation                | 179 |
| 4.40.2.1 OpenAccessibility()                                 | 180 |
| 4.41 OpenAccessibilityAnalytics Class Reference              | 181 |
| 4.41.1 Detailed Description                                  | 181 |
| 4.41.2 Constructor & Destructor Documentation                | 181 |
| 4.41.2.1 OpenAccessibilityAnalytics()                        | 181 |
| 4.42 OpenAccessibilityCommunication Class Reference          | 182 |
| 4.42.1 Detailed Description                                  | 182 |
| 4.42.2 Constructor & Destructor Documentation                | 182 |
| 4.42.2.1 OpenAccessibilityCommunication()                    | 183 |
| 4.43 OpenAccessibilityPy.OpenAccessibilityPy Class Reference | 184 |
| 4.43.1 Detailed Description                                  | 184 |

|   |     |
|---|-----|
| 4.43.2 Constructor & Destructor Documentation           | 184 |
| 4.43.2.1 <code>__init__()</code>                        | 184 |
| 4.43.2.2 <code>__del__()</code>                         | 185 |
| 4.43.3 Member Function Documentation                    | 185 |
| 4.43.3.1 <code>HandleTranscriptionRequest()</code>      | 185 |
| 4.43.3.2 <code>Shutdown()</code>                        | 186 |
| 4.43.3.3 <code>Tick()</code>                            | 187 |
| 4.43.4 Member Data Documentation                        | 187 |
| 4.43.4.1 <code>audio_resampler</code>                   | 187 |
| 4.43.4.2 <code>com_server</code>                        | 188 |
| 4.43.4.3 <code>pyshutdown_handle</code>                 | 188 |
| 4.43.4.4 <code>tick_handle</code>                       | 188 |
| 4.43.4.5 <code>whisper_interface</code>                 | 188 |
| 4.43.4.6 <code>worker_pool</code>                       | 188 |
| 4.44 SAccessibilityTranscriptionVis Class Reference     | 188 |
| 4.44.1 Detailed Description                             | 189 |
| 4.44.2 Constructor & Destructor Documentation           | 189 |
| 4.44.2.1 <code>~SAccessibilityTranscriptionVis()</code> | 189 |
| 4.44.3 Member Function Documentation                    | 189 |
| 4.44.3.1 <code>Construct()</code>                       | 190 |
| 4.44.3.2 <code>SLATE_BEGIN_ARGS()</code>                | 190 |
| 4.44.3.3 <code>Tick()</code>                            | 191 |
| 4.44.3.4 <code>UpdateTopTranscription()</code>          | 191 |
| 4.44.4 Member Data Documentation                        | 191 |
| 4.44.4.1 <code>TranscriptionContainer</code>            | 191 |
| 4.44.4.2 <code>TranscriptionSlots</code>                | 192 |
| 4.45 SContentIndexer Class Reference                    | 192 |
| 4.45.1 Detailed Description                             | 193 |
| 4.45.2 Constructor & Destructor Documentation           | 193 |
| 4.45.2.1 <code>~SContentIndexer()</code>                | 193 |
| 4.45.3 Member Function Documentation                    | 193 |
| 4.45.3.1 <code>Construct()</code>                       | 193 |
| 4.45.3.2 <code>ConstructBottomIndexer()</code>          | 194 |
| 4.45.3.3 <code>ConstructContentContainer()</code>       | 194 |
| 4.45.3.4 <code>ConstructIndexContainer()</code>         | 195 |
| 4.45.3.5 <code>ConstructIndexText()</code>              | 195 |
| 4.45.3.6 <code>ConstructLeftIndexer()</code>            | 196 |
| 4.45.3.7 <code>ConstructRightIndexer()</code>           | 196 |
| 4.45.3.8 <code>ConstructTopIndexer()</code>             | 197 |
| 4.45.3.9 <code>GetContent()</code> [1/2]                | 198 |
| 4.45.3.10 <code>GetContent()</code> [2/2]               | 198 |
| 4.45.3.11 <code>SLATE_BEGIN_ARGS()</code>               | 198 |



|   |     |
|---|-----|
| 4.45.3.12 Tick()  | 199 |
| 4.45.3.13 UpdateIndex()   | 199 |
| 4.45.4 Member Data Documentation                                  | 199 |
| 4.45.4.1 IndexedContent   | 199 |
| 4.45.4.2 IndexerWidget  | 199 |
| 4.46 SIndexer Class Reference                                     | 200 |
| 4.46.1 Detailed Description                                       | 200 |
| 4.46.2 Constructor & Destructor Documentation                     | 200 |
| 4.46.2.1 ~SIndexer()  | 200 |
| 4.46.3 Member Function Documentation                              | 201 |
| 4.46.3.1 Construct()  | 201 |
| 4.46.3.2 GetIndexText()   | 201 |
| 4.46.3.3 SLATE_BEGIN_ARGS()                                       | 201 |
| 4.46.3.4 Tick()   | 202 |
| 4.46.3.5 UpdateIndex() [1/3]                                      | 202 |
| 4.46.3.6 UpdateIndex() [2/3]                                      | 202 |
| 4.46.3.7 UpdateIndex() [3/3]                                      | 203 |
| 4.46.4 Member Data Documentation                                  | 203 |
| 4.46.4.1 IndexTextBlock   | 203 |
| 4.47 TGraphAccessibilityNodeFactory< T > Class Template Reference | 203 |
| 4.47.1 Detailed Description                                       | 204 |
| 4.47.2 Constructor & Destructor Documentation                     | 204 |
| 4.47.2.1 TGraphAccessibilityNodeFactory() [1/2]                   | 204 |
| 4.47.2.2 TGraphAccessibilityNodeFactory() [2/2]                   | 204 |
| 4.47.2.3 ~TGraphAccessibilityNodeFactory()                        | 205 |
| 4.47.3 Member Function Documentation                              | 205 |
| 4.47.3.1 CreateNodeWidget()                                       | 205 |
| 4.47.3.2 CreatePinWidget()  | 206 |
| 4.47.4 Member Data Documentation                                  | 207 |
| 4.47.4.1 AccessibilityRegistry                                    | 207 |
| 4.47.4.2 Implementation   | 208 |
| 4.48 UAccessibilityAddNodeContextMenu Class Reference             | 208 |
| 4.48.1 Detailed Description                                       | 210 |
| 4.48.2 Constructor & Destructor Documentation                     | 210 |
| 4.48.2.1 UAccessibilityAddNodeContextMenu() [1/4]                 | 210 |
| 4.48.2.2 UAccessibilityAddNodeContextMenu() [2/4]                 | 210 |
| 4.48.2.3 UAccessibilityAddNodeContextMenu() [3/4]                 | 210 |
| 4.48.2.4 UAccessibilityAddNodeContextMenu() [4/4]                 | 211 |
| 4.48.2.5 ~UAccessibilityAddNodeContextMenu()                      | 211 |
| 4.48.3 Member Function Documentation                              | 211 |
| 4.48.3.1 AppendFilterText()                                       | 211 |
| 4.48.3.2 AppendScrollDistance()                                   | 211 |

|   |     |
|---|-----|
| 4.48.3.3 ApplyAccessibilityWidget()                   | 212 |
| 4.48.3.4 Close()                                      | 212 |
| 4.48.3.5 DoesItemsRequireRefresh()                    | 213 |
| 4.48.3.6 GetFilterText()                              | 213 |
| 4.48.3.7 GetGraphActionFromIndex() [1/2]              | 213 |
| 4.48.3.8 GetGraphActionFromIndex() [2/2]              | 214 |
| 4.48.3.9 GetGraphActionFromIndexSP()                  | 214 |
| 4.48.3.10 Init() [1/3]                                | 215 |
| 4.48.3.11 Init() [2/3]                                | 216 |
| 4.48.3.12 Init() [3/3]                                | 216 |
| 4.48.3.13 PerformGraphAction()                        | 217 |
| 4.48.3.14 RefreshAccessibilityWidgets()               | 217 |
| 4.48.3.15 ResetFilterText()                           | 218 |
| 4.48.3.16 ScaleMenu()                                 | 218 |
| 4.48.3.17 SelectGraphAction()                         | 219 |
| 4.48.3.18 SetFilterText()                             | 219 |
| 4.48.3.19 SetScrollDistance()                         | 219 |
| 4.48.3.20 SetScrollDistanceBottom()                   | 220 |
| 4.48.3.21 SetScrollDistanceTop()                      | 220 |
| 4.48.3.22 Tick()                                      | 220 |
| 4.48.3.23 ToggleContextAwareness()                    | 221 |
| 4.48.3.24 UpdateAccessibilityWidget()                 | 221 |
| 4.48.4 Member Data Documentation                      | 221 |
| 4.48.4.1 ContextAwarenessCheckBox                     | 221 |
| 4.48.4.2 FilterTextBox                                | 221 |
| 4.48.4.3 GraphMenu                                    | 222 |
| 4.48.4.4 PrevFilterString                             | 222 |
| 4.48.4.5 PrevNumGeneratedChildren                     | 222 |
| 4.48.4.6 PrevNumItemsBeingObserved                    | 222 |
| 4.48.4.7 PrevScrollDistance                           | 222 |
| 4.48.4.8 TreeView                                     | 222 |
| 4.49 UAccessibilityGraphEditorContext Class Reference | 223 |
| 4.49.1 Detailed Description                           | 224 |
| 4.49.2 Constructor & Destructor Documentation         | 224 |
| 4.49.2.1 UAccessibilityGraphEditorContext()           | 224 |
| 4.49.3 Member Function Documentation                  | 224 |
| 4.49.3.1 AppendFilterText()                           | 224 |
| 4.49.3.2 AppendScrollDistance()                       | 225 |
| 4.49.3.3 Close()                                      | 225 |
| 4.49.3.4 CreateAccessibilityWrapper()                 | 226 |
| 4.49.3.5 FindGraphActionMenu()                        | 226 |
| 4.49.3.6 FindStaticComponents()                       | 227 |

|   |     |
|---|-----|
| 4.49.3.7 FindTreeView()                                   | 227 |
| 4.49.3.8 GetFilterText()                                  | 228 |
| 4.49.3.9 GetStaticIndexOffset()                           | 228 |
| 4.49.3.10 GetTreeViewAction()                             | 228 |
| 4.49.3.11 Init()  | 229 |
| 4.49.3.12 ScaleMenu()                                     | 230 |
| 4.49.3.13 SelectAction()                                  | 230 |
| 4.49.3.14 SetFilterText()                                 | 231 |
| 4.49.3.15 SetScrollDistance()                             | 231 |
| 4.49.3.16 SetScrollDistanceBottom()                       | 232 |
| 4.49.3.17 SetScrollDistanceTop()                          | 232 |
| 4.49.3.18 Tick()  | 232 |
| 4.49.3.19 TickTreeViewAccessibility()                     | 233 |
| 4.49.3.20 TreeViewCanTick()                               | 233 |
| 4.49.3.21 TreeViewRequiresTick()                          | 234 |
| 4.49.3.22 UpdateAccessibilityWidget()                     | 234 |
| 4.49.4 Member Data Documentation                          | 234 |
| 4.49.4.1 CheckBoxes                                       | 234 |
| 4.49.4.2 FilterTextBox                                    | 235 |
| 4.49.4.3 GraphMenu  | 235 |
| 4.49.4.4 TreeView   | 235 |
| 4.49.4.5 TreeViewTickRequirements                         | 235 |
| 4.50 UAccessibilityGraphLocomotionContext Class Reference | 235 |
| 4.50.1 Detailed Description                               | 236 |
| 4.50.2 Constructor & Destructor Documentation             | 236 |
| 4.50.2.1 UAccessibilityGraphLocomotionContext()           | 237 |
| 4.50.2.2 ~UAccessibilityGraphLocomotionContext()          | 237 |
| 4.50.3 Member Function Documentation                      | 237 |
| 4.50.3.1 BindFocusChangedEvent()                          | 237 |
| 4.50.3.2 CalculateVisualChunksBounds()                    | 237 |
| 4.50.3.3 CancelLocomotion()                               | 238 |
| 4.50.3.4 ChangeChunkVis()                                 | 238 |
| 4.50.3.5 Close()  | 238 |
| 4.50.3.6 ConfirmSelection()                               | 239 |
| 4.50.3.7 CreateVisualGrid()                               | 239 |
| 4.50.3.8 GenerateVisualChunks()                           | 239 |
| 4.50.3.9 HideNativeVisuals()                              | 240 |
| 4.50.3.10 Init() [1/2]                                    | 241 |
| 4.50.3.11 Init() [2/2]                                    | 241 |
| 4.50.3.12 MoveViewport() [1/2]                            | 241 |
| 4.50.3.13 MoveViewport() [2/2]                            | 242 |
| 4.50.3.14 OnFocusChanged()                                | 242 |

|  |     |
|--|-----|
| 4.50.3.15 RemoveVisualGrid()                     | 242 |
| 4.50.3.16 RevertToPreviousView()                 | 243 |
| 4.50.3.17 SelectChunk()                          | 243 |
| 4.50.3.18 UnbindFocusChangedEvent()              | 244 |
| 4.50.3.19 UnHideNativeVisuals()                  | 244 |
| 4.50.4 Member Data Documentation                 | 244 |
| 4.50.4.1 ChunkArray                              | 244 |
| 4.50.4.2 ChunkSize                               | 244 |
| 4.50.4.3 CurrentViewPosition                     | 245 |
| 4.50.4.4 GridContainer                           | 245 |
| 4.50.4.5 GridParent                              | 245 |
| 4.50.4.6 LinkedEditor                            | 245 |
| 4.50.4.7 PreviousPositions                       | 245 |
| 4.50.4.8 StartViewPosition                       | 245 |
| 4.50.4.9 StartViewZoom                           | 246 |
| 4.51 UAccessibilityWindowToolbar Class Reference | 246 |
| 4.51.1 Detailed Description                      | 246 |
| 4.51.2 Constructor & Destructor Documentation    | 246 |
| 4.51.2.1 UAccessibilityWindowToolbar()           | 247 |
| 4.51.2.2 ~UAccessibilityWindowToolbar()          | 247 |
| 4.51.3 Member Function Documentation             | 247 |
| 4.51.3.1 GetActiveToolkitWidget()                | 247 |
| 4.51.3.2 IsActiveToolbar()                       | 247 |
| 4.51.3.3 SelectToolbarItem()                     | 248 |
| 4.51.3.4 Tick()                                  | 249 |
| 4.52 UAudioManager Class Reference               | 249 |
| 4.52.1 Detailed Description                      | 250 |
| 4.52.2 Constructor & Destructor Documentation    | 250 |
| 4.52.2.1 UAudioManager()                         | 250 |
| 4.52.2.2 ~UAudioManager()                        | 251 |
| 4.52.3 Member Function Documentation             | 251 |
| 4.52.3.1 GetAudioCaptureNumChannels()            | 251 |
| 4.52.3.2 GetAudioCaptureSampleRate()             | 251 |
| 4.52.3.3 IsCapturingAudio()                      | 252 |
| 4.52.3.4 OnDefaultDeviceChanged()                | 252 |
| 4.52.3.5 PRIVATE_OnAudioGenerate()               | 252 |
| 4.52.3.6 SaveAudioBufferToWAV()                  | 253 |
| 4.52.3.7 StartCapturingAudio()                   | 253 |
| 4.52.3.8 StopCapturingAudio()                    | 253 |
| 4.52.4 Member Data Documentation                 | 254 |
| 4.52.4.1 OnAudioReadyForTranscription            | 254 |
| 4.52.4.2 Settings                                | 254 |

|   |     |
|---|-----|
| 4.53 UBAudioCapture Class Reference . . . . .           | 254 |
| 4.53.1 Detailed Description . . . . .                   | 255 |
| 4.53.2 Constructor & Destructor Documentation . . . . . | 255 |
| 4.53.2.1 UBAudioCapture() . . . . .                     | 255 |
| 4.53.2.2 ~UBAudioCapture() . . . . .                    | 255 |
| 4.53.3 Member Function Documentation . . . . .          | 255 |
| 4.53.3.1 OpenDefaultAudioStream() . . . . .             | 255 |
| 4.54 ULocalizedInputLibrary Class Reference . . . . .   | 256 |
| 4.54.1 Detailed Description . . . . .                   | 257 |
| 4.54.2 Constructor & Destructor Documentation . . . . . | 257 |
| 4.54.2.1 ULocalizedInputLibrary() . . . . .             | 257 |
| 4.54.2.2 ~ULocalizedInputLibrary() . . . . .            | 257 |
| 4.54.3 Member Function Documentation . . . . .          | 257 |
| 4.54.3.1 BindBranches() . . . . .                       | 257 |
| 4.54.3.2 KeyboardInputAdd() . . . . .                   | 258 |
| 4.54.3.3 KeyboardInputConfirm() . . . . .               | 259 |
| 4.54.3.4 KeyboardInputExit() . . . . .                  | 260 |
| 4.54.3.5 KeyboardInputRemove() . . . . .                | 260 |
| 4.54.3.6 KeyboardInputReset() . . . . .                 | 261 |
| 4.55 UNodeInteractionLibrary Class Reference . . . . .  | 262 |
| 4.55.1 Detailed Description . . . . .                   | 263 |
| 4.55.2 Constructor & Destructor Documentation . . . . . | 263 |
| 4.55.2.1 UNodeInteractionLibrary() . . . . .            | 263 |
| 4.55.2.2 ~UNodeInteractionLibrary() . . . . .           | 263 |
| 4.55.3 Member Function Documentation . . . . .          | 263 |
| 4.55.3.1 BindBranches() . . . . .                       | 263 |
| 4.55.3.2 BlueprintCompile() . . . . .                   | 267 |
| 4.55.3.3 DeleteNode() . . . . .                         | 268 |
| 4.55.3.4 LocomotionCancel() . . . . .                   | 268 |
| 4.55.3.5 LocomotionConfirm() . . . . .                  | 269 |
| 4.55.3.6 LocomotionRevert() . . . . .                   | 269 |
| 4.55.3.7 LocomotionSelect() . . . . .                   | 270 |
| 4.55.3.8 MoveNode() . . . . .                           | 270 |
| 4.55.3.9 NodeAddMenu() . . . . .                        | 271 |
| 4.55.3.10 NodeAddPinMenu() . . . . .                    | 272 |
| 4.55.3.11 NodeAddScroll() . . . . .                     | 274 |
| 4.55.3.12 NodeAddSearchAdd() . . . . .                  | 274 |
| 4.55.3.13 NodeAddSearchRemove() . . . . .               | 275 |
| 4.55.3.14 NodeAddSearchReset() . . . . .                | 275 |
| 4.55.3.15 NodeAddSelect() . . . . .                     | 275 |
| 4.55.3.16 NodeIndexFocus() . . . . .                    | 276 |
| 4.55.3.17 PinConnect() . . . . .                        | 276 |

|   |     |
|---|-----|
| 4.55.3.18 PinDisconnect()                         | 277 |
| 4.55.3.19 SelectionAlignment()                    | 278 |
| 4.55.3.20 SelectionComment()                      | 278 |
| 4.55.3.21 SelectionMove()                         | 279 |
| 4.55.3.22 SelectionNodeToggle()                   | 280 |
| 4.55.3.23 SelectionReset()                        | 280 |
| 4.55.3.24 SelectionStraighten()                   | 281 |
| 4.56 UParseEnumInput Class Reference              | 281 |
| 4.56.1 Detailed Description                       | 282 |
| 4.56.2 Constructor & Destructor Documentation     | 282 |
| 4.56.2.1 ~UParseEnumInput()                       | 282 |
| 4.56.3 Member Function Documentation              | 282 |
| 4.56.3.1 GetEnumType() [1/2]                      | 282 |
| 4.56.3.2 GetEnumType() [2/2]                      | 282 |
| 4.56.3.3 SetEnumType()                            | 283 |
| 4.56.4 Member Data Documentation                  | 283 |
| 4.56.4.1 EnumType                                 | 283 |
| 4.57 UParseInput Class Reference                  | 283 |
| 4.57.1 Detailed Description                       | 284 |
| 4.57.2 Constructor & Destructor Documentation     | 284 |
| 4.57.2.1 ~UParseInput()                           | 284 |
| 4.58 UParseIntInput Class Reference               | 284 |
| 4.58.1 Detailed Description                       | 285 |
| 4.58.2 Constructor & Destructor Documentation     | 285 |
| 4.58.2.1 ~UParseIntInput()                        | 285 |
| 4.58.3 Member Function Documentation              | 285 |
| 4.58.3.1 GetValue() [1/2]                         | 285 |
| 4.58.3.2 GetValue() [2/2]                         | 285 |
| 4.58.3.3 SetValue()                               | 286 |
| 4.58.4 Member Data Documentation                  | 286 |
| 4.58.4.1 Value                                    | 286 |
| 4.59 UParseStringInput Class Reference            | 286 |
| 4.59.1 Detailed Description                       | 287 |
| 4.59.2 Constructor & Destructor Documentation     | 287 |
| 4.59.2.1 ~UParseStringInput()                     | 287 |
| 4.59.3 Member Function Documentation              | 287 |
| 4.59.3.1 GetValue() [1/2]                         | 287 |
| 4.59.3.2 GetValue() [2/2]                         | 287 |
| 4.59.3.3 SetValue()                               | 288 |
| 4.59.4 Member Data Documentation                  | 288 |
| 4.59.4.1 Value                                    | 288 |
| 4.60 UPhraseTreeContextMenuObject Class Reference | 288 |

|   |     |
|---|-----|
| 4.60.1 Detailed Description . . . . .                     | 289 |
| 4.60.2 Constructor & Destructor Documentation . . . . .   | 289 |
| 4.60.2.1 UPhraseTreeContextMenuObject() [1/2] . . . . .   | 290 |
| 4.60.2.2 UPhraseTreeContextMenuObject() [2/2] . . . . .   | 290 |
| 4.60.2.3 ~UPhraseTreeContextMenuObject() . . . . .        | 290 |
| 4.60.3 Member Function Documentation . . . . .            | 290 |
| 4.60.3.1 BindMenuDismissed() . . . . .                    | 290 |
| 4.60.3.2 BindTickDelegate() . . . . .                     | 291 |
| 4.60.3.3 Close() . . . . .                                | 291 |
| 4.60.3.4 GetWindow() . . . . .                            | 291 |
| 4.60.3.5 Init() [1/2] . . . . .                           | 291 |
| 4.60.3.6 Init() [2/2] . . . . .                           | 292 |
| 4.60.3.7 OnMenuDismissed() . . . . .                      | 292 |
| 4.60.3.8 RemoveMenuDismissed() . . . . .                  | 293 |
| 4.60.3.9 RemoveTickDelegate() . . . . .                   | 293 |
| 4.60.3.10 ScaleMenu() . . . . .                           | 293 |
| 4.60.3.11 SetMenu() . . . . .                             | 294 |
| 4.60.3.12 Tick() . . . . .                                | 294 |
| 4.60.4 Member Data Documentation . . . . .                | 294 |
| 4.60.4.1 Menu . . . . .                                   | 294 |
| 4.60.4.2 Window . . . . .                                 | 295 |
| 4.61 UPhraseTreeContextObject Class Reference . . . . .   | 295 |
| 4.61.1 Detailed Description . . . . .                     | 295 |
| 4.61.2 Constructor & Destructor Documentation . . . . .   | 296 |
| 4.61.2.1 UPhraseTreeContextObject() . . . . .             | 296 |
| 4.61.2.2 ~UPhraseTreeContextObject() . . . . .            | 296 |
| 4.61.3 Member Function Documentation . . . . .            | 296 |
| 4.61.3.1 Close() . . . . .                                | 296 |
| 4.61.3.2 GetContextRoot() . . . . .                       | 296 |
| 4.61.3.3 GetIsActive() . . . . .                          | 297 |
| 4.61.3.4 SetContextRootNode() . . . . .                   | 297 |
| 4.61.4 Member Data Documentation . . . . .                | 297 |
| 4.61.4.1 blsActive . . . . .                              | 297 |
| 4.61.4.2 ContextRoot . . . . .                            | 298 |
| 4.62 UPhraseTreeFunctionLibrary Class Reference . . . . . | 298 |
| 4.62.1 Detailed Description . . . . .                     | 298 |
| 4.62.2 Member Function Documentation . . . . .            | 298 |
| 4.62.2.1 BindBranches() . . . . .                         | 298 |
| 4.63 UPhraseTreeUtils Class Reference . . . . .           | 299 |
| 4.63.1 Detailed Description . . . . .                     | 299 |
| 4.63.2 Constructor & Destructor Documentation . . . . .   | 299 |
| 4.63.2.1 UPhraseTreeUtils() . . . . .                     | 299 |

|  |     |
|--|-----|
| 4.63.2.2 ~UPhraseTreeUtils()   | 299 |
| 4.63.3 Member Function Documentation                                       | 300 |
| 4.63.3.1 RegisterFunctionLibrary()   | 300 |
| 4.63.3.2 SetPhraseTree()   | 301 |
| 4.63.4 Member Data Documentation   | 301 |
| 4.63.4.1 PhraseTree  | 301 |
| 4.63.4.2 RegisteredLibraries   | 302 |
| 4.64 UViewInteractionLibrary Class Reference                               | 302 |
| 4.64.1 Detailed Description  | 302 |
| 4.64.2 Constructor & Destructor Documentation                              | 302 |
| 4.64.2.1 UViewInteractionLibrary()   | 302 |
| 4.64.2.2 ~UViewInteractionLibrary()  | 303 |
| 4.64.3 Member Function Documentation                                       | 303 |
| 4.64.3.1 BindBranches()  | 303 |
| 4.64.3.2 IndexFocus()  | 304 |
| 4.64.3.3 MoveViewport()  | 304 |
| 4.64.3.4 ZoomViewport()  | 305 |
| 4.65 UWindowInteractionLibrary Class Reference                             | 306 |
| 4.65.1 Detailed Description  | 307 |
| 4.65.2 Constructor & Destructor Documentation                              | 307 |
| 4.65.2.1 UWindowInteractionLibrary()                                       | 307 |
| 4.65.2.2 ~UWindowInteractionLibrary()                                      | 307 |
| 4.65.3 Member Function Documentation                                       | 307 |
| 4.65.3.1 BindBranches()  | 307 |
| 4.65.3.2 CloseActiveWindow()   | 308 |
| 4.65.3.3 SelectToolBarItem()   | 309 |
| 4.65.3.4 SwitchNextActiveWindow()  | 309 |
| 4.65.3.5 SwitchNextTabInStack()  | 310 |
| 4.65.3.6 SwitchPrevActiveWindow()  | 311 |
| 4.65.3.7 SwitchPrevTabInStack()  | 312 |
| 4.65.4 Member Data Documentation   | 313 |
| 4.65.4.1 WindowToolBar   | 313 |
| 4.66 OpenAccessibilityPy.WhisperInterface.WhisperInterface Class Reference | 313 |
| 4.66.1 Detailed Description  | 314 |
| 4.66.2 Constructor & Destructor Documentation                              | 314 |
| 4.66.2.1 __init__()  | 314 |
| 4.66.2.2 __del__()   | 315 |
| 4.66.3 Member Function Documentation                                       | 315 |
| 4.66.3.1 process_audio_buffer()  | 315 |
| 4.66.3.2 process_file_from_dir()   | 316 |
| 4.66.4 Member Data Documentation   | 316 |
| 4.66.4.1 beam_size   | 316 |



|  |            |
|--|------------|
| 4.66.4.2 whisper_model                       | 316        |
| <b>5 File Documentation</b>                  | <b>317</b> |
| 5.1 init_unreal.py                           | 317        |
| 5.2 __init__.py                              | 318        |
| 5.3 __main__.py                              | 320        |
| 5.4 Audio.py                                 | 322        |
| 5.5 CommunicationServer.py                   | 324        |
| 5.6 LibUtils.py                              | 327        |
| 5.7 Logging.py                               | 327        |
| 5.8 WhisperInterface.py                      | 328        |
| 5.9 OpenAccessibility.Build.cs               | 329        |
| 5.10 SAccessibilityTranscriptionVis.cpp      | 330        |
| 5.11 SContentIndexer.cpp                     | 331        |
| 5.12 SIndexer.cpp                            | 333        |
| 5.13 AccessibilityAddNodeContextMenu.cpp     | 334        |
| 5.14 AccessibilityGraphEditorContext.cpp     | 338        |
| 5.15 AccessibilityGraphLocomotionContext.cpp | 343        |
| 5.16 AccessibilityWindowToolbar.cpp          | 347        |
| 5.17 AssetAccessibilityRegistry.cpp          | 351        |
| 5.18 GraphIndexer.cpp                        | 354        |
| 5.19 OAccessibilityNodeFactory.cpp           | 357        |
| 5.20 OAEEditorAccessibilityManager.cpp       | 360        |
| 5.21 OpenAccessibility.cpp                   | 360        |
| 5.22 LocalizedInputLibrary.cpp               | 365        |
| 5.23 NodeInteractionLibrary.cpp              | 368        |
| 5.24 ViewInteractionLibrary.cpp              | 379        |
| 5.25 WindowInteractionLibrary.cpp            | 381        |
| 5.26 TranscriptionVisualizer.cpp             | 386        |
| 5.27 WidgetUtils.h                           | 388        |
| 5.28 AccessibilityNodeFactory.h              | 390        |
| 5.29 SAccessibilityTranscriptionVis.h        | 392        |
| 5.30 SContentIndexer.h                       | 393        |
| 5.31 SIndexer.h                              | 394        |
| 5.32 AccessibilityAddNodeContextMenu.h       | 394        |
| 5.33 AccessibilityGraphEditorContext.h       | 396        |
| 5.34 AccessibilityGraphLocomotionContext.h   | 397        |
| 5.35 AccessibilityWindowToolbar.h            | 399        |
| 5.36 AssetAccessibilityRegistry.h            | 400        |
| 5.37 GraphIndexer.h                          | 401        |
| 5.38 Indexer.h                               | 402        |
| 5.39 OAccessibilityNodeFactory.h             | 405        |

|      |   |     |
|------|---|-----|
| 5.40 | OAEditorAccessibilityManager.h          | 405 |
| 5.41 | OpenAccessibility.h                     | 405 |
| 5.42 | OpenAccessibilityLogging.h              | 406 |
| 5.43 | LocalizedInputLibrary.h                 | 406 |
| 5.44 | NodeInteractionLibrary.h                | 407 |
| 5.45 | ViewInteractionLibrary.h                | 408 |
| 5.46 | WindowInteractionLibrary.h              | 409 |
| 5.47 | TranscriptionVisualizer.h               | 410 |
| 5.48 | OpenAccessibilityAnalytics.Build.cs     | 410 |
| 5.49 | OpenAccessibilityAnalytics.cpp          | 411 |
| 5.50 | OpenAccessibilityAnalyticsLogging.h     | 413 |
| 5.51 | OpenAccessibilityAnalytics.h            | 413 |
| 5.52 | OpenAccessibilityCommunication.Build.cs | 414 |
| 5.53 | AudioManager.cpp                        | 415 |
| 5.54 | OpenAccessibilityComLogging.cpp         | 417 |
| 5.55 | OpenAccessibilityCommunication.cpp      | 417 |
| 5.56 | PhraseTree.cpp                          | 420 |
| 5.57 | ContextMenuObject.cpp                   | 422 |
| 5.58 | PhraseEnumInputNode.cpp                 | 423 |
| 5.59 | PhraseEventNode.cpp                     | 425 |
| 5.60 | PhraseInputNode.cpp                     | 425 |
| 5.61 | PhraseNode.cpp                          | 427 |
| 5.62 | PhraseStringInputNode.cpp               | 430 |
| 5.63 | Utils.cpp                               | 430 |
| 5.64 | PhraseTreeUtils.cpp                     | 431 |
| 5.65 | SocketCommunicationServer.cpp           | 432 |
| 5.66 | UBAudioCapture.cpp                      | 437 |
| 5.67 | AudioManager.h                          | 438 |
| 5.68 | OpenAccessibilityComLogging.h           | 439 |
| 5.69 | OpenAccessibilityCommunication.h        | 439 |
| 5.70 | PhraseTree.h                            | 440 |
| 5.71 | ContextMenuObject.h                     | 443 |
| 5.72 | ContextObject.h                         | 444 |
| 5.73 | InputContainers.h                       | 444 |
| 5.74 | UParseEnumInput.h                       | 445 |
| 5.75 | UParseInput.h                           | 446 |
| 5.76 | UParseIntInput.h                        | 446 |
| 5.77 | UParseStringInput.h                     | 447 |
| 5.78 | ParseRecord.h                           | 447 |
| 5.79 | ParseResult.h                           | 450 |
| 5.80 | IPhraseContextNode.h                    | 451 |
| 5.81 | PhraseContextMenuNode.h                 | 451 |

---

|   |            |
|---|------------|
| 5.82 PhraseContextNode.h . . . . .          | 454        |
| 5.83 PhraseDirectionalInputNode.h . . . . . | 456        |
| 5.84 PhraseEnumInputNode.h . . . . .        | 457        |
| 5.85 PhraseEventNode.h . . . . .            | 458        |
| 5.86 PhraseInputNode.h . . . . .            | 458        |
| 5.87 PhraseNode.h . . . . .                 | 459        |
| 5.88 PhraseStringInputNode.h . . . . .      | 460        |
| 5.89 PhraseTreeFunctionLibrary.h . . . . .  | 460        |
| 5.90 Utils.h . . . . .                      | 460        |
| 5.91 Utils.h . . . . .                      | 463        |
| 5.92 PhraseTreeUtils.h . . . . .            | 463        |
| 5.93 SocketCommunicationServer.h . . . . .  | 464        |
| 5.94 UBAudioCapture.h . . . . .             | 465        |
| <b>Index</b>                                | <b>467</b> |



# Chapter 1

## Hierarchical Index

### 1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

|   |     |
|---|-----|
| OpenAccessibilityPy.Audio.AudioResampler . . . . .                    | 9   |
| OpenAccessibilityPy.CommunicationServer.CommunicationServer . . . . . | 11  |
| FTabManager::FArea  |     |
| TabUtils::FOpenArea . . . . .   | 70  |
| FAssetAccessibilityRegistry . . . . .                                 | 25  |
| FAudioManagerSettings . . . . .                                       | 33  |
| FGraphIndexer . . . . .   | 34  |
| FGraphLocomotionChunk . . . . .                                       | 46  |
| FGraphNodeFactory   |     |
| TGraphAccessibilityNodeFactory< T > . . . . .                         | 203 |
| FGraphPanelNodeFactory  |     |
| FAccessibilityNodeFactory . . . . .                                   | 21  |
| FIndexer< KeyType, ValueType > . . . . .                              | 49  |
| FIndexer< int32, SMultiBlockBaseWidget * > . . . . .                  | 49  |
| FPanelViewPosition . . . . .  | 72  |
| FParseRecord . . . . .  | 74  |
| FParseResult . . . . .  | 86  |
| FPhraseTreeBranchBind . . . . .                                       | 141 |
| FPhraseTreeContextManager . . . . .                                   | 143 |
| FSocketCommunicationServer . . . . .                                  | 147 |
| FTabManager::FSplitter  |     |
| TabUtils::FOpenSplitter . . . . .                                     | 71  |
| FTabManager::FStack   |     |
| TabUtils::FOpenStack . . . . .  | 71  |
| FTranscriptionVisualizer . . . . .                                    | 165 |
| UAccessibilityGraphEditorContext::FTreeViewTickRequirements . . . . . | 171 |
| IModuleInterface  |     |
| FOpenAccessibilityAnalyticsModule . . . . .                           | 60  |
| FOpenAccessibilityCommunicationModule . . . . .                       | 62  |
| FOpenAccessibilityModule . . . . .                                    | 67  |
| IPhraseContextNodeBase . . . . .                                      | 172 |
| FPhraseContextMenuNode< ContextMenuType > . . . . .                   | 90  |
| FPhraseContextNode< ContextType > . . . . .                           | 97  |
| IPhraseNodeBase . . . . .   | 174 |
| ModuleRules   |     |

|   |     |
|---|-----|
| OpenAccessibility                                   | 179 |
| OpenAccessibilityAnalytics                          | 181 |
| OpenAccessibilityCommunication                      | 182 |
| NumericParser                                       | 177 |
| OAEEditorAccessibilityManager                       | 178 |
| OpenAccessibilityPy.OpenAccessibilityPy             | 184 |
| SBox  |     |
| SAccessibilityTranscriptionVis                      | 188 |
| SContentIndexer                                     | 192 |
| SIndexer  | 200 |
| TSharedFromThis                                     |     |
| FPhraseNode   | 118 |
| FPhraselInputNode< int32 >                          | 111 |
| FPhraseEnumInputNode< EPhrase2DDirectionalInput >   | 104 |
| FPhrase2DDirectionalInputNode                       | 88  |
| FPhraseEnumInputNode< EPhraseDirectionalInput >     | 104 |
| FPhraseDirectionalInputNode                         | 102 |
| FPhraseEnumInputNode< EPhrasePositionalInput >      | 104 |
| FPhrasePositionalInputNode                          | 129 |
| FPhraseEnumInputNode< EPhraseScrollInput >          | 104 |
| FPhraseScrollInputNode                              | 131 |
| FPhraseEnumInputNode< TEnum >                       | 104 |
| FPhraselInputNode< FString >                        | 111 |
| FPhraseStringInputNode                              | 133 |
| FPhraseContextMenuNode< ContextMenuType >           | 90  |
| FPhraseContextNode< ContextType >                   | 97  |
| FPhraseEventNode                                    | 108 |
| FPhraselInputNode< InputType >                      | 111 |
| FPhraseTree   | 136 |
| UAudioCapture                                       |     |
| UBAudioCapture                                      | 254 |
| UObject   |     |
| UAccessibilityWindowToolbar                         | 246 |
| UAudioManager                                       | 249 |
| UParseInput   | 283 |
| UParseIntInput                                      | 284 |
| UParseEnumInput                                     | 281 |
| UParseStringInput                                   | 286 |
| UPhraseTreeContextObject                            | 295 |
| UAccessibilityGraphLocomotionContext                | 235 |
| UPhraseTreeContextMenuObject                        | 288 |
| UAccessibilityAddNodeContextMenu                    | 208 |
| UAccessibilityGraphEditorContext                    | 223 |
| UPhraseTreeFunctionLibrary                          | 298 |
| ULocalizedInputLibrary                              | 256 |
| UNodeInteractionLibrary                             | 262 |
| UViewInteractionLibrary                             | 302 |
| UWindowInteractionLibrary                           | 306 |
| UPhraseTreeUtils                                    | 299 |
| OpenAccessibilityPy.WisperInterface.WisperInterface | 313 |
| Enum  |     |
| OpenAccessibilityPy.Logging.LogLevel                | 176 |

## Chapter 2

# Class Index

### 2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

|  |     |
|--|-----|
| <a href="#">OpenAccessibilityPy.Audio.AudioResampler</a>                       | 9   |
| <a href="#">OpenAccessibilityPy.CommunicationServer.CommunicationServer</a>    | 11  |
| <a href="#">FAccessibilityNodeFactory</a>                                      | 21  |
| <a href="#">FAssetAccessibilityRegistry</a>                                    | 25  |
| <a href="#">FAudioManagerSettings</a>  | 33  |
| <a href="#">FGraphIndexer</a>  | 34  |
| <a href="#">FGraphLocomotionChunk</a>  | 46  |
| <a href="#">FIndexer&lt; KeyType, ValueType &gt;</a>                           | 49  |
| <a href="#">FOpenAccessibilityAnalyticsModule</a>                              | 60  |
| <a href="#">FOpenAccessibilityCommunicationModule</a>                          | 62  |
| <a href="#">FOpenAccessibilityModule</a>                                       | 67  |
| <a href="#">TabUtils::FOpenArea</a>  | 70  |
| <a href="#">TabUtils::FOpenSplitter</a>  | 71  |
| <a href="#">TabUtils::FOpenStack</a>   | 71  |
| <a href="#">FPanelViewPosition</a>   | 72  |
| <a href="#">FParseRecord</a>   |     |
| The Collected Information from the Propagation of the Phrase through the tree. | 74  |
| <a href="#">FParseResult</a>   |     |
| Contains the Result of Propagation through the Phrase Tree.                    | 86  |
| <a href="#">FPhrase2DDirectionalInputNode</a>                                  | 88  |
| <a href="#">FPhraseContextMenuNode&lt; ContextMenuType &gt;</a>                | 90  |
| <a href="#">FPhraseContextNode&lt; ContextType &gt;</a>                        | 97  |
| <a href="#">FPhraseDirectionalInputNode</a>                                    | 102 |
| <a href="#">FPhraseEnumInputNode&lt; TEnum &gt;</a>                            | 104 |
| <a href="#">FPhraseEventNode</a>   | 108 |
| <a href="#">FPhraseInputNode&lt; InputType &gt;</a>                            | 111 |
| <a href="#">FPhraseNode</a>  | 118 |
| <a href="#">FPhrasePositionalInputNode</a>                                     | 129 |
| <a href="#">FPhraseScrollInputNode</a>   | 131 |
| <a href="#">FPhraseStringInputNode</a>   | 133 |
| <a href="#">FPhraseTree</a>  | 136 |
| <a href="#">FPhraseTreeBranchBind</a>  | 141 |
| <a href="#">FPhraseTreeContextManager</a>                                      | 143 |
| <a href="#">FSocketCommunicationServer</a>                                     | 147 |
| <a href="#">FTranscriptionVisualizer</a>                                       | 165 |

|   |     |
|---|-----|
| UAccessibilityGraphEditorContext::FTreeViewTickRequirements                             | 171 |
| IPhraseContextNodeBase  |     |
| Base Abstract Class For Phrase Context Nodes, that are required to have a Context Node. | 172 |
| IPhraseNodeBase   | 174 |
| OpenAccessibilityPy.Logging.LogLevel  | 176 |
| NumericParser   | 177 |
| OAEditorAccessibilityManager  | 178 |
| OpenAccessibility   | 179 |
| OpenAccessibilityAnalytics  | 181 |
| OpenAccessibilityCommunication  | 182 |
| OpenAccessibilityPy.OpenAccessibilityPy   | 184 |
| SAccessibilityTranscriptionVis  | 188 |
| SContentIndexer   | 192 |
| SIndexer  | 200 |
| TGraphAccessibilityNodeFactory< T >   | 203 |
| UAccessibilityAddNodeContextMenu  | 208 |
| UAccessibilityGraphEditorContext  | 223 |
| UAccessibilityGraphLocomotionContext  | 235 |
| UAccessibilityWindowToolbar   | 246 |
| UAudioManager   | 249 |
| UBAudioCapture  | 254 |
| ULocalizedInputLibrary  | 256 |
| UNodeInteractionLibrary   | 262 |
| UParseEnumInput   | 281 |
| UParseInput   | 283 |
| UParseIntInput  | 284 |
| UParseStringInput   | 286 |
| UPhraseTreeContextMenuObject  | 288 |
| UPhraseTreeContextObject  | 295 |
| UPhraseTreeFunctionLibrary  | 298 |
| UPhraseTreeUtils  | 299 |
| UViewInteractionLibrary   | 302 |
| UWindowInteractionLibrary   | 306 |
| OpenAccessibilityPy.WhisperInterface.WhisperInterface                                   | 313 |



## Chapter 3

# File Index

### 3.1 File List

Here is a list of all documented files with brief descriptions:

|  |     |
|--|-----|
| Content/Python/init_unreal.py  | 317 |
| Content/Python/OpenAccessibilityPy/__init__.py   | 318 |
| Content/Python/OpenAccessibilityPy/__main__.py   | 320 |
| Content/Python/OpenAccessibilityPy/Audio.py  | 322 |
| Content/Python/OpenAccessibilityPy/CommunicationServer.py                                      | 324 |
| Content/Python/OpenAccessibilityPy/LibUtils.py   | 327 |
| Content/Python/OpenAccessibilityPy/Logging.py  | 327 |
| Content/Python/OpenAccessibilityPy/WhisperInterface.py   | 328 |
| Source/OpenAccessibility/OpenAccessibility.Build.cs  | 329 |
| Source/OpenAccessibility/Private/AssetAccessibilityRegistry.cpp                                | 351 |
| Source/OpenAccessibility/Private/GraphIndexer.cpp  | 354 |
| Source/OpenAccessibility/Private/OAccessibilityNodeFactory.cpp                                 | 357 |
| Source/OpenAccessibility/Private/OAEditorAccessibilityManager.cpp                              | 360 |
| Source/OpenAccessibility/Private/OpenAccessibility.cpp   | 360 |
| Source/OpenAccessibility/Private/TranscriptionVisualizer.cpp                                   | 386 |
| Source/OpenAccessibility/Private/AccessibilityWidgets/SAccessibilityTranscriptionVis.cpp       | 330 |
| Source/OpenAccessibility/Private/AccessibilityWidgets/SContentIndexer.cpp                      | 331 |
| Source/OpenAccessibility/Private/AccessibilityWidgets/SIndexer.cpp                             | 333 |
| Source/OpenAccessibility/Private/AccessibilityWrappers/AccessibilityAddNodeContextMenu.cpp     | 334 |
| Source/OpenAccessibility/Private/AccessibilityWrappers/AccessibilityGraphEditorContext.cpp     | 338 |
| Source/OpenAccessibility/Private/AccessibilityWrappers/AccessibilityGraphLocomotionContext.cpp | 343 |
| Source/OpenAccessibility/Private/AccessibilityWrappers/AccessibilityWindowToolbar.cpp          | 347 |
| Source/OpenAccessibility/Private/PhraseEvents/LocalizedInputLibrary.cpp                        | 365 |
| Source/OpenAccessibility/Private/PhraseEvents/NodeInteractionLibrary.cpp                       | 368 |
| Source/OpenAccessibility/Private/PhraseEvents/ViewInteractionLibrary.cpp                       | 379 |
| Source/OpenAccessibility/Private/PhraseEvents/WindowInteractionLibrary.cpp                     | 381 |
| Source/OpenAccessibility/Private/Utils/WidgetUtils.h   | 388 |
| Source/OpenAccessibility/Public/AccessibilityNodeFactory.h                                     | 390 |
| Source/OpenAccessibility/Public/AssetAccessibilityRegistry.h                                   | 400 |
| Source/OpenAccessibility/Public/GraphIndexer.h   | 401 |
| Source/OpenAccessibility/Public/OAccessibilityNodeFactory.h                                    | 405 |
| Source/OpenAccessibility/Public/OAEditorAccessibilityManager.h                                 | 405 |
| Source/OpenAccessibility/Public/OpenAccessibility.h  | 405 |
| Source/OpenAccessibility/Public/OpenAccessibilityLogging.h                                     | 406 |
| Source/OpenAccessibility/Public/TranscriptionVisualizer.h                                      | 410 |

|   |     |
|---|-----|
| Source/OpenAccessibility/Public/AccessibilityWidgets/SAccessibilityTranscriptionVis.h       | 392 |
| Source/OpenAccessibility/Public/AccessibilityWidgets/SContentIndexer.h                      | 393 |
| Source/OpenAccessibility/Public/AccessibilityWidgets/SIndexer.h                             | 394 |
| Source/OpenAccessibility/Public/AccessibilityWrappers/AccessibilityAddNodeContextMenu.h     | 394 |
| Source/OpenAccessibility/Public/AccessibilityWrappers/AccessibilityGraphEditorContext.h     | 396 |
| Source/OpenAccessibility/Public/AccessibilityWrappers/AccessibilityGraphLocomotionContext.h | 397 |
| Source/OpenAccessibility/Public/AccessibilityWrappers/AccessibilityWindowToolBar.h          | 399 |
| Source/OpenAccessibility/Public/Indexers/Indexer.h  | 402 |
| Source/OpenAccessibility/Public/PhraseEvents/LocalizedInputLibrary.h                        | 406 |
| Source/OpenAccessibility/Public/PhraseEvents/NodeInteractionLibrary.h                       | 407 |
| Source/OpenAccessibility/Public/PhraseEvents/Utils.h  | 460 |
| Source/OpenAccessibility/Public/PhraseEvents/ViewInteractionLibrary.h                       | 408 |
| Source/OpenAccessibility/Public/PhraseEvents/WindowInteractionLibrary.h                     | 409 |
| Source/OpenAccessibilityAnalytics/OpenAccessibilityAnalytics.Build.cs                       | 410 |
| Source/OpenAccessibilityAnalytics/Private/OpenAccessibilityAnalytics.cpp                    | 411 |
| Source/OpenAccessibilityAnalytics/Private/OpenAccessibilityAnalyticsLogging.h               | 413 |
| Source/OpenAccessibilityAnalytics/Public/OpenAccessibilityAnalytics.h                       | 413 |
| Source/OpenAccessibilityCommunication/OpenAccessibilityCommunication.Build.cs               | 414 |
| Source/OpenAccessibilityCommunication/Private/AudioManager.cpp                              | 415 |
| Source/OpenAccessibilityCommunication/Private/OpenAccessibilityComLogging.cpp               | 417 |
| Source/OpenAccessibilityCommunication/Private/OpenAccessibilityCommunication.cpp            | 417 |
| Source/OpenAccessibilityCommunication/Private/PhraseTree.cpp                                | 420 |
| Source/OpenAccessibilityCommunication/Private/PhraseTreeUtils.cpp                           | 431 |
| Source/OpenAccessibilityCommunication/Private/SocketCommunicationServer.cpp                 | 432 |
| Source/OpenAccessibilityCommunication/Private/UBAudioCapture.cpp                            | 437 |
| Source/OpenAccessibilityCommunication/Private/PhraseTree/PhraseEnumInputNode.cpp            | 423 |
| Source/OpenAccessibilityCommunication/Private/PhraseTree/PhraseEventNode.cpp                | 425 |
| Source/OpenAccessibilityCommunication/Private/PhraseTree/PhraseInputNode.cpp                | 425 |
| Source/OpenAccessibilityCommunication/Private/PhraseTree/PhraseNode.cpp                     | 427 |
| Source/OpenAccessibilityCommunication/Private/PhraseTree/PhraseStringInputNode.cpp          | 430 |
| Source/OpenAccessibilityCommunication/Private/PhraseTree/Utils.cpp                          | 430 |
| Source/OpenAccessibilityCommunication/Private/PhraseTree/Containers/ContextMenuObject.cpp   | 422 |
| Source/OpenAccessibilityCommunication/Public/AudioManager.h                                 | 438 |
| Source/OpenAccessibilityCommunication/Public/OpenAccessibilityComLogging.h                  | 439 |
| Source/OpenAccessibilityCommunication/Public/OpenAccessibilityCommunication.h               | 439 |
| Source/OpenAccessibilityCommunication/Public/PhraseTree.h                                   | 440 |
| Source/OpenAccessibilityCommunication/Public/PhraseTreeUtils.h                              | 463 |
| Source/OpenAccessibilityCommunication/Public/SocketCommunicationServer.h                    | 464 |
| Source/OpenAccessibilityCommunication/Public/UBAudioCapture.h                               | 465 |
| Source/OpenAccessibilityCommunication/Public/PhraseTree/IPhaseContextNode.h                 | 451 |
| Source/OpenAccessibilityCommunication/Public/PhraseTree/PhraseContextMenuNode.h             | 451 |
| Source/OpenAccessibilityCommunication/Public/PhraseTree/PhraseContextNode.h                 | 454 |
| Source/OpenAccessibilityCommunication/Public/PhraseTree/PhraseDirectionalInputNode.h        | 456 |
| Source/OpenAccessibilityCommunication/Public/PhraseTree/PhraseEnumInputNode.h               | 457 |
| Source/OpenAccessibilityCommunication/Public/PhraseTree/PhraseEventNode.h                   | 458 |
| Source/OpenAccessibilityCommunication/Public/PhraseTree/PhraseInputNode.h                   | 458 |
| Source/OpenAccessibilityCommunication/Public/PhraseTree/PhraseNode.h                        | 459 |
| Source/OpenAccessibilityCommunication/Public/PhraseTree/PhraseStringInputNode.h             | 460 |
| Source/OpenAccessibilityCommunication/Public/PhraseTree/PhraseTreeFunctionLibrary.h         | 460 |
| Source/OpenAccessibilityCommunication/Public/PhraseTree/Utils.h                             | 463 |
| Source/OpenAccessibilityCommunication/Public/PhraseTree/Containers/ContextMenuObject.h      | 443 |
| Source/OpenAccessibilityCommunication/Public/PhraseTree/Containers/ContextObject.h          | 444 |
| Source/OpenAccessibilityCommunication/Public/PhraseTree/Containers/ParseRecord.h            | 447 |
| Source/OpenAccessibilityCommunication/Public/PhraseTree/Containers/ParseResult.h            | 450 |
| Source/OpenAccessibilityCommunication/Public/PhraseTree/Containers/Input/InputContainers.h  | 444 |
| Source/OpenAccessibilityCommunication/Public/PhraseTree/Containers/Input/UParseEnumInput.h  | 445 |
| Source/OpenAccessibilityCommunication/Public/PhraseTree/Containers/Input/UParseInput.h      | 446 |
| Source/OpenAccessibilityCommunication/Public/PhraseTree/Containers/Input/UParseIntInput.h   | 446 |

---

Source/OpenAccessibilityCommunication/Public/PhraseTree/Containers/Input/[UParseStringInput.h](#) . . . 447



# Chapter 4

## Class Documentation

### 4.1 OpenAccessibilityPy.Audio.AudioResampler Class Reference

#### Public Member Functions

- `def __init__` (self, int target\_sample\_rate=16000)
- `def __del__` (self)
- `np.ndarray resample` (self, np.ndarray audio\_data, int buffer\_sample\_rate=48000, int buffer\_num\_↵ channels=2)

#### 4.1.1 Detailed Description

Audio Resampler for Resampling Incoming Audio to the Target Sample Rate. Using FFmpeg for Resampling.

Definition at line 15 of file [Audio.py](#).

#### 4.1.2 Constructor & Destructor Documentation

##### 4.1.2.1 \_\_init\_\_()

```
def OpenAccessibilityPy.Audio.AudioResampler.__init__ (
    self,
    int target_sample_rate = 16000 )
```

Constructor of Audio Resampler Class

Args:

target\_sample\_rate (int, optional): The Target for all incoming resampling requests. Defaults to 16000 (Re

Definition at line 18 of file [Audio.py](#).

```
00018     def __init__(self, target_sample_rate: int = 16000):
00019         """Constructor of Audio Resampler Class
00020
00021         Args:
00022             target_sample_rate (int, optional): The Target for all incoming resampling requests.
00023             Defaults to 16000 (Required by Whisper).
00024
00025             self._audio_resampler = av.AudioResampler(
00026                 format="s16", layout="mono", rate=target_sample_rate
00027             )
00028             self._resample_mutex = Lock()
00029
```

#### 4.1.2.2 `__del__()`

```
def OpenAccessibilityPy.Audio.AudioResampler.__del__ (
    self )
```

Destructor of Audio Resampler Class.

Ensures PyAV Resampler Object is Properly Deleted, calling Garbage Collection in the process.

Definition at line 30 of file [Audio.py](#).

```
00030     def __del__(self):
00031         """Destructor of Audio Resampler Class.
00032
00033         Ensures PyAV Resampler Object is Properly Deleted, calling Garbage Collection in the process.
00034         """
00035
00036         # Try Deleting the resampler object to cleanly free up memory
00037         try:
00038             del self._audio_resampler
00039         except:
00040             pass
00041
00042         try: # Delete the mutex
00043             del self._resample_mutex
00044         except:
00045             pass
00046
00047         # Force Garbage Collection, due to resampler not being properly deleted otherwise.
00048         gc.collect()
00049
```

### 4.1.3 Member Function Documentation

#### 4.1.3.1 `resample()`

```
np.ndarray OpenAccessibilityPy.Audio.AudioResampler.resample (
    self,
    np.ndarray audio_data,
    int buffer_sample_rate = 48000,
    int buffer_num_channels = 2 )
```

Resamples the Incoming Audio Data to the Classes Assigned Target Sample Rate.

Args:

```
audio_data (np.ndarray): Audio Data to Resample.
buffer_sample_rate (int, optional): Sample Rate of the Incoming Audio Data. Defaults to 48000.
buffer_num_channels (int, optional): Number of Channels in the Incoming Audio Data. Defaults to 2 (Stereo)
```

Returns:

```
np.ndarray: Resampled Version of the Incoming Audio Data.
```

Definition at line 50 of file [Audio.py](#).

```
00055     ) -> np.ndarray:
00056         """Resamples the Incoming Audio Data to the Classes Assigned Target Sample Rate.
00057
00058         Args:
00059             audio_data (np.ndarray): Audio Data to Resample.
00060             buffer_sample_rate (int, optional): Sample Rate of the Incoming Audio Data. Defaults to
00061             48000.
00062             buffer_num_channels (int, optional): Number of Channels in the Incoming Audio Data.
00063             Defaults to 2 (Stereo).
```

```

00063         Returns:
00064             np.ndarray: Resampled Version of the Incoming Audio Data.
00065         """
00066
00067         audio_data = self._convert_to_s16(audio_data).reshape(-1, 1)
00068
00069         frame: av.AudioFrame = av.AudioFrame.from_ndarray(
00070             audio_data.T,
00071             format="s16",
00072             layout="stereo" if buffer_num_channels == 2 else "mono",
00073         )
00074
00075         frame.sample_rate = buffer_sample_rate
00076
00077         resampled_frames: list[av.AudioFrame] = []
00078         with self._resample_mutex:
00079             resampled_frames = self._audio_resampler.resample(frame)
00080
00081         return self._convert_to_float32(resampled_frames[0].to_ndarray()).reshape(
00082             -1,
00083         )
00084

```

The documentation for this class was generated from the following file:

- Content/Python/OpenAccessibilityPy/Audio.py

## 4.2 OpenAccessibilityPy.CommunicationServer.CommunicationServer Class Reference

### Public Member Functions

- def `__init__` (self, int send\_socket\_type, int recv\_socket\_type, str send\_socket\_addr="tcp://127.0.0.1:5556", str recv\_socket\_addr="tcp://127.0.0.1:5555", int poll\_timeout=10)
- def `__del__` (self)
- bool `EventOccured` (self)
- bool `SendString` (self, str message)
- bool `SendJSON` (self, dict message)
- bool `SendNDArray` (self, np.ndarray message)
- bool `SendNDArrayWithMeta` (self, np.ndarray message, dict meta)
- bool `SendMultipart` (self, list message)
- bool `SendMultipartWithMeta` (self, list message, dict meta)
- def `RecieveRaw` (self)
- str `ReceiveString` (self)
- def `ReceiveJSON` (self)
- np.ndarray `ReceiveNDArray` (self, dtype=np.float32)
- tuple[np.ndarray, dict] `ReceiveNDArrayWithMeta` (self, dtype=np.float32)
- list[bytes] `ReceiveMultipart` (self)

### Public Attributes

- `context`
- `send_socket_context`
- `recv_socket`
- `recv_socket_context`
- `poller`
- `poller_timeout_time`

## 4.2.1 Detailed Description

Communication Server Class for Handling Communication Between Python and C++.

Using ZeroMQ for Socket Communication. (Push / PULL Architecture)

Definition at line 11 of file [CommunicationServer.py](#).

## 4.2.2 Constructor & Destructor Documentation

### 4.2.2.1 \_\_init\_\_()

```
def OpenAccessibilityPy.CommunicationServer.CommunicationServer.__init__ (
    self,
    int send_socket_type,
    int recv_socket_type,
    str send_socket_addr = "tcp://127.0.0.1:5556",
    str recv_socket_addr = "tcp://127.0.0.1:5555",
    int poll_timeout = 10 )
```

Constructor of Communication Server Class

Args:

send\_socket\_type (int): ZeroMQ Socket Type for Sending Messages.  
recv\_socket\_type (int): ZeroMQ Socket Type for Receiving Messages.  
send\_socket\_addr (str, optional): Local Address / Port for Sending Communication Data. Defaults to "tcp://127.0.0.1:5556".  
recv\_socket\_addr (str, optional): Local Address / Port for Receiving Communication Data. Defaults to "tcp://127.0.0.1:5555".  
poll\_timeout (int, optional): Amount of time (ms) for event polling on the Receive Socket. Defaults to 10.

Definition at line 17 of file [CommunicationServer.py](#).

```
00024     ):
00025         """Constructor of Communication Server Class
00026
00027         Args:
00028             send_socket_type (int): ZeroMQ Socket Type for Sending Messages.
00029             recv_socket_type (int): ZeroMQ Socket Type for Receiving Messages.
00030             send_socket_addr (str, optional): Local Address / Port for Sending Communication Data.
00031             Defaults to "tcp://127.0.0.1:5556".
00032             recv_socket_addr (str, optional): Local Address / Port for Receiving Communication Data.
00033             Defaults to "tcp://127.0.0.1:5555".
00034             poll_timeout (int, optional): Amount of time (ms) for event polling on the Receive Socket.
00035             Defaults to 10.
00036         """
00037
00038         # Create the Context
00039         self.context = zmq.Context()
00040
00041         # Create a Socket
00042         self.send_socket: zmq.Socket = self.context.socket(send_socket_type)
00043         self.send_socket_context = self.send_socket.connect(send_socket_addr)
00044
00045         self.recv_socket = self.context.socket(recv_socket_type)
00046         self.recv_socket_context = self.recv_socket.bind(recv_socket_addr)
00047
00048         self.poller = zmq.Poller()
00049         self.poller.register(self.recv_socket, zmq.POLLIN)
00050         self.poller_timeout_time = poll_timeout
```



#### 4.2.2.2 `__del__()`

```
def OpenAccessibilityPy.CommunicationServer.CommunicationServer.__del__ (
    self )
```

Destructor of Communication Server Class.

Closes the Sockets and Terminates the ZeroMQ Context.

Definition at line 49 of file [CommunicationServer.py](#).

```
00049     def __del__(self):
00050         """Destructor of Communication Server Class.
00051
00052         Closes the Sockets and Terminates the ZeroMQ Context.
00053         """
00054
00055         self.send_socket.close()
00056         self.recv_socket.close()
00057
00058         self.context.term()
00059
```

### 4.2.3 Member Function Documentation

#### 4.2.3.1 `EventOccured()`

```
bool OpenAccessibilityPy.CommunicationServer.CommunicationServer.EventOccured (
    self )
```

Checks if a Receive Event has Occured on the Receive Socket.

Returns:

bool: True if an Event has Occured, False Otherwise.

Definition at line 60 of file [CommunicationServer.py](#).

```
00060     def EventOccured(self) -> bool:
00061         """Checks if a Receive Event has Occured on the Receive Socket.
00062
00063         Returns:
00064             bool: True if an Event has Occured, False Otherwise.
00065         """
00066
00067         polled_events = dict(self.poller.poll(self.poller_timeout_time))
00068         if len(polled_events) > 0 and polled_events.get(self.recv_socket) == zmq.POLLIN:
00069             return True
00070         else:
00071             return False
00072
```

#### 4.2.3.2 ReceiveJSON()

```
def OpenAccessibilityPy.CommunicationServer.CommunicationServer.ReceiveJSON (
    self )
```

Receive a JSON Message from the Receive Socket.

Returns:

dict: Dictionary of the Received JSON Message.

Definition at line 211 of file [CommunicationServer.py](#).

```
00211     def ReceiveJSON(self):
00212         """Receive a JSON Message from the Receive Socket.
00213
00214         Returns:
00215             dict: Dictionary of the Received JSON Message.
00216         """
00217
00218         return json.loads(self.recv_socket.recv_json(zmq.DONTWAIT))
00219
```

#### 4.2.3.3 ReceiveMultipart()

```
list[bytes] OpenAccessibilityPy.CommunicationServer.CommunicationServer.ReceiveMultipart (
    self )
```

Receieved a Raw Multipart Message from the Receive Socket.

Returns:

list[bytes]: Raw List of Bytes from the Received Multipart Message.

Definition at line 262 of file [CommunicationServer.py](#).

```
00262     def ReceiveMultipart(self) -> list[bytes]:
00263         """Receieved a Raw Multipart Message from the Receive Socket.
00264
00265         Returns:
00266             list[bytes]: Raw List of Bytes from the Received Multipart Message.
00267         """
00268
00269         return self.recv_socket.recv_multipart(zmq.DONTWAIT)
```

#### 4.2.3.4 ReceiveNDArray()

```
np.ndarray OpenAccessibilityPy.CommunicationServer.CommunicationServer.ReceiveNDArray (
    self,
    dtype = np.float32 )
```

Receives a Numpy NDArray from the Receive Socket.

Args:

dtype (optional): Type of NDArray of Received Data. Defaults to np.float32.

Returns:

np.ndarray: Receieved NDArray Message.

Definition at line 220 of file [CommunicationServer.py](#).

```
00220     def ReceiveNDArray(self, dtype=np.float32) -> np.ndarray:
00221         """Receives a Numpy NDArray from the Receive Socket.
00222
00223         Args:
00224             dtype (optional): Type of NDArray of Received Data. Defaults to np.float32.
00225
00226         Returns:
00227             np.ndarray: Received NDArray Message.
00228         """
00229
00230         return np.frombuffer(
00231             self.recv_socket.recv(zmq.DONTWAIT),
00232             dtype=dtype,
00233         )
00234
```

#### 4.2.3.5 ReceiveNDArrayWithMeta()

```
tuple[np.ndarray, dict] OpenAccessibilityPy.CommunicationServer.CommunicationServer.Receive↵
NDArrayWithMeta (
    self,
    dtype = np.float32 )
```

Receives a Numpy NDArray with Metadata from the Receive Socket.

Args:

dtype (optional): Type of NDArray of Received Data. Defaults to np.float32.

Returns:

tuple[np.ndarray, dict]: Tuple of Received NDArray and Dict Metadata Object.

Definition at line 235 of file [CommunicationServer.py](#).

```
00235     def ReceiveNDArrayWithMeta(self, dtype=np.float32) -> tuple[np.ndarray, dict]:
00236         """Receives a Numpy NDArray with Metadata from the Receive Socket.
00237
00238         Args:
00239             dtype (optional): Type of NDArray of Received Data. Defaults to np.float32.
00240
00241         Returns:
00242             tuple[np.ndarray, dict]: Tuple of Received NDArray and Dict Metadata Object.
00243         """
00244
00245         recv_message = self.recv_socket.recv_multipart(zmq.DONTWAIT)
00246
00247         if len(recv_message) > 1:
00248             return (
00249                 np.frombuffer(recv_message[1], dtype=dtype),
00250                 json.loads(recv_message[0]),
00251             )
00252
00253         elif len(recv_message) == 1:
00254             Log(
00255                 "CommunicationServer | Error Receiving NDArray With Meta. Only Contains One Message,
Assumed Data.",
00256                 LogLevel.WARNING,
00257             )
00258             return (np.frombuffer(recv_message[0], dtype=dtype), {})
00259
00260             Log("CommunicationServer | Error Receiving NDArray With Meta", LogLevel.WARNING)
00261
```

#### 4.2.3.6 ReceiveString()

```
str OpenAccessibilityPy.CommunicationServer.CommunicationServer.ReceiveString (
    self )
```

Receives a String Message from the Receive Socket.

Returns:  
str: Received String Message.

Definition at line 202 of file [CommunicationServer.py](#).

```
00202     def ReceiveString(self) -> str:
00203         """Receives a String Message from the Receive Socket.
00204
00205         Returns:
00206             str: Received String Message.
00207         """
00208
00209         return self.recv_socket.recv_string(zmq.DONTWAIT)
00210
```

#### 4.2.3.7 RecieveRaw()

```
def OpenAccessibilityPy.CommunicationServer.CommunicationServer.RecieveRaw (
    self )
```

Receives a Raw Message of Bytes from the Receive Socket.

Returns:  
bytes: Raw Received Bytes from the Receive Socket.

Definition at line 193 of file [CommunicationServer.py](#).

```
00193     def RecieveRaw(self):
00194         """Receives a Raw Message of Bytes from the Receive Socket.
00195
00196         Returns:
00197             bytes: Raw Received Bytes from the Receive Socket.
00198         """
00199
00200         return self.recv_socket.recv(zmq.DONTWAIT)
00201
```

#### 4.2.3.8 SendJSON()

```
bool OpenAccessibilityPy.CommunicationServer.CommunicationServer.SendJSON (
    self,
    dict message )
```

Sends a JSON Message on the Send Socket.

Args:  
message (dict): Stringified JSON Message to Send.

Returns:  
bool: True if the Message was Sent Successfully, False Otherwise.

Definition at line 90 of file [CommunicationServer.py](#).

```
00090     def SendJSON(self, message: dict) -> bool:
00091         """Sends a JSON Message on the Send Socket.
00092
00093         Args:
00094             message (dict): Stringified JSON Message to Send.
00095
00096         Returns:
00097             bool: True if the Message was Sent Successfully, False Otherwise.
00098         """
00099
00100         try:
00101             self.send_socket.send_json(message)
00102             return True
00103         except:
00104             Log(
00105                 "CommunicationServer | Error Sending JSON Message",
00106                 LogLevel.WARNING,
00107             )
00108             return False
00109
```

#### 4.2.3.9 SendMultipart()

```
bool OpenAccessibilityPy.CommunicationServer.CommunicationServer.SendMultipart (
    self,
    list message )
```

Sends a Multipart Message on the Send Socket.

Args:  
message (list): List of Messages to Send.

Returns:  
bool: True if the MultiPart Message was Sent Successfully, False Otherwise.

Definition at line 152 of file [CommunicationServer.py](#).

```
00152     def SendMultipart(self, message: list) -> bool:
00153         """Sends a Multipart Message on the Send Socket.
00154
00155         Args:
00156             message (list): List of Messages to Send.
00157
00158         Returns:
00159             bool: True if the MultiPart Message was Sent Successfully, False Otherwise.
00160         """
00161
00162         try:
00163             self.send_socket.send_multipart(message)
00164             return True
00165         except:
00166             Log(
00167                 "CommunicationServer | Error Sending Multipart Message",
00168                 LogLevel.WARNING,
00169             )
00170             return False
00171
```

#### 4.2.3.10 SendMultipartWithMeta()

```
bool OpenAccessibilityPy.CommunicationServer.CommunicationServer.SendMultipartWithMeta (
    self,
    list message,
    dict meta )
```

Sends a Multipart Message with Metadata on the Send Socket.

Args:

message (list): List of Messages to Send.  
meta (dict): Metadata to Send.

Returns:

bool: True if the MultiPart Message with Metadata was Sent Successfully, False Otherwise.

Definition at line 172 of file [CommunicationServer.py](#).

```
00172     def SendMultipartWithMeta(self, message: list, meta: dict) -> bool:
00173         """Sends a Multipart Message with Metadata on the Send Socket.
00174
00175         Args:
00176             message (list): List of Messages to Send.
00177             meta (dict): Metadata to Send.
00178
00179         Returns:
00180             bool: True if the MultiPart Message with Metadata was Sent Successfully, False Otherwise.
00181         """
00182
00183         try:
00184             self.send_socket.send_multipart([json.dumps(meta).encode(), *message])
00185             return True
00186         except:
00187             Log(
00188                 "CommunicationServer | Error Sending Multipart With Meta Message",
00189                 LogLevel.WARNING,
00190             )
00191             return False
00192
```

#### 4.2.3.11 SendNDArray()

```
bool OpenAccessibilityPy.CommunicationServer.CommunicationServer.SendNDArray (
    self,
    np.ndarray message )
```

Sends a Numpy NDArray Message on the Send Socket.

Args:

message (np.ndarray): NDArray of Data to Send.

Returns:

bool: True if the Data was Sent Successfully, False Otherwise.

Definition at line 110 of file [CommunicationServer.py](#).

```
00110     def SendNDArray(self, message: np.ndarray) -> bool:
00111         """Sends a Numpy NDArray Message on the Send Socket.
00112
00113         Args:
00114             message (np.ndarray): NDArray of Data to Send.
00115
00116         Returns:
00117             bool: True if the Data was Sent Successfully, False Otherwise.
00118         """
00119
00120         try:
00121             self.send_socket.send(message)
00122             return True
00123         except:
00124             Log(
00125                 "CommunicationServer | Error Sending NDArray Message",
00126                 LogLevel.WARNING,
00127             )
00128             return False
00129
```

#### 4.2.3.12 SendNDArrayWithMeta()

```
bool OpenAccessibilityPy.CommunicationServer.CommunicationServer.SendNDArrayWithMeta (
    self,
    np.ndarray message,
    dict meta )
```

Sends a Numpy NDArray Message with Metadata on the Send Socket.

Args:

message (np.ndarray): NDArray of Data to Send.  
meta (dict): A Dictionary of Metadata to Send.

Returns:

bool: True if the Data was Sent Successfully, False Otherwise.

Definition at line 130 of file [CommunicationServer.py](#).

```
00130     def SendNDArrayWithMeta(self, message: np.ndarray, meta: dict) -> bool:
00131         """Sends a Numpy NDArray Message with Metadata on the Send Socket.
00132
00133         Args:
00134             message (np.ndarray): NDArray of Data to Send.
00135             meta (dict): A Dictionary of Metadata to Send.
00136
00137         Returns:
00138             bool: True if the Data was Sent Successfully, False Otherwise.
00139         """
00140
00141         try:
00142             self.send_socket.send_multipart([json.dumps(meta).encode(), message.data])
00143
00144             return True
00145         except:
00146             Log(
00147                 "CommunicationServer | Error Sending NDArray With Meta Message",
00148                 LogLevel.WARNING,
00149             )
00150             return False
00151
```

#### 4.2.3.13 SendString()

```
bool OpenAccessibilityPy.CommunicationServer.CommunicationServer.SendString (
    self,
    str message )
```

Sends a String Message on the Send Socket.

Args:

message (str): String Message to Send.

Returns:

bool: True if the Message was Sent Successfully, False Otherwise.

Definition at line 73 of file [CommunicationServer.py](#).

```
00073     def SendString(self, message: str) -> bool:
00074         """Sends a String Message on the Send Socket.
00075
00076         Args:
00077             message (str): String Message to Send.
00078
00079         Returns:
00080             bool: True if the Message was Sent Successfully, False Otherwise.
00081         """
00082
00083         try:
00084             self.send_socket.send_string(message)
00085             return True
00086         except:
00087             Log("CommunicationServer | Error Sending String Message", LogLevel.WARNING)
00088             return False
00089
```

## 4.2.4 Member Data Documentation

### 4.2.4.1 context

`OpenAccessibilityPy.CommunicationServer.CommunicationServer.context`

Definition at line 36 of file [CommunicationServer.py](#).

### 4.2.4.2 poller

`OpenAccessibilityPy.CommunicationServer.CommunicationServer.poller`

Definition at line 45 of file [CommunicationServer.py](#).

### 4.2.4.3 poller\_timeout\_time

`OpenAccessibilityPy.CommunicationServer.CommunicationServer.poller_timeout_time`

Definition at line 47 of file [CommunicationServer.py](#).

### 4.2.4.4 recv\_socket

`OpenAccessibilityPy.CommunicationServer.CommunicationServer.recv_socket`

Definition at line 42 of file [CommunicationServer.py](#).

### 4.2.4.5 recv\_socket\_context

`OpenAccessibilityPy.CommunicationServer.CommunicationServer.recv_socket_context`

Definition at line 43 of file [CommunicationServer.py](#).



#### 4.2.4.6 send\_socket\_context

OpenAccessibilityPy.CommunicationServer.CommunicationServer.send\_socket\_context

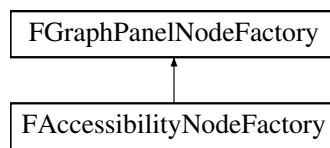
Definition at line 40 of file [CommunicationServer.py](#).

The documentation for this class was generated from the following file:

- Content/Python/OpenAccessibilityPy/CommunicationServer.py

## 4.3 FAccessibilityNodeFactory Class Reference

Inheritance diagram for FAccessibilityNodeFactory:



### Public Member Functions

- virtual TSharedPtr< class SGraphNode > [CreateNode](#) (UEdGraphNode \*Node) const override
- void [WrapNodeWidget](#) (UEdGraphNode \*Node, TSharedPtr< SGraphNode > NodeWidget, int NodeIndex) const  
*Wraps the Node Widget with Accessibility Indexing.*
- void [WrapPinWidget](#) (UEdGraphPin \*Pin, TSharedPtr< SGraphPin > PinWidget, int PinIndex, SGraphNode \*OwnerNode) const  
*Wraps the Pin Widget with Accessibility Indexing.*
- void [SetSharedPtr](#) (TSharedPtr< [FAccessibilityNodeFactory](#) > InSharedPtr)

#### 4.3.1 Detailed Description

Definition at line 11 of file [OAccessibilityNodeFactory.h](#).

#### 4.3.2 Constructor & Destructor Documentation

##### 4.3.2.1 FAccessibilityNodeFactory()

FAccessibilityNodeFactory::FAccessibilityNodeFactory ( )

Definition at line 23 of file [OAccessibilityNodeFactory.cpp](#).

```

00023                                     : FGraphPanelNodeFactory()
00024 {
00025     UE_LOGFMT(LogOpenAccessibility, Display, "Accessibility Node Factory Constructed");
00026 }
```

#### 4.3.2.2 ~FAccessibilityNodeFactory()

FAccessibilityNodeFactory::~FAccessibilityNodeFactory ( ) [virtual]

Definition at line 28 of file [OAccessibilityNodeFactory.cpp](#).

```
00029 {
00030
00031 }
```

### 4.3.3 Member Function Documentation

#### 4.3.3.1 CreateNode()

TSharedPtr< class SGraphNode > FAccessibilityNodeFactory::CreateNode ( UEdGraphNode \* Node ) const [override], [virtual]

Definition at line 33 of file [OAccessibilityNodeFactory.cpp](#).

```
00034 {
00035     UE_LOG(LogOpenAccessibility, Display, TEXT("Accessibility Node Factory Used to construct %s"),
        *InNode->GetName());
00036
00037     check(InNode);
00038
00039     // Hack to get around the possible infinite loop of using
00040     // this factory to create the node from the factory itself.
00041
00042     FEdGraphUtilities::UnregisterVisualNodeFactory(FOpenAccessibilityModule::Get().AccessibilityNodeFactory);
00043     TSharedPtr<SGraphNode> OutNode = FNodeFactory::CreateNodeWidget(InNode);
00044     FEdGraphUtilities::RegisterVisualNodeFactory(FOpenAccessibilityModule::Get().AccessibilityNodeFactory);
00045
00046     // Get Node Accessibility Index, from registry
00047     TSharedRef<FGraphIndexer> GraphIndexer = FOpenAccessibilityModule::Get()
        .AssetAccessibilityRegistry->GetGraphIndexer(InNode->GetGraph());
00048
00049     int NodeIndex = -1;
00050     GraphIndexer->GetOrAddNode(InNode, NodeIndex);
00051
00052     {
00053         // Create Accessibility Widgets For Pins
00054         TArray<UEdGraphPin*> Pins = InNode->GetAllPins();
00055         TSharedPtr<SGraphPin> PinWidget;
00056
00057         for (int i = 0; i < Pins.Num(); i++)
00058         {
00059             UEdGraphPin* Pin = Pins[i];
00060
00061             PinWidget = OutNode->FindWidgetForPin(Pin);
00062             if (!PinWidget.IsValid())
00063             {
00064                 continue;
00065             }
00066
00067             WrapPinWidget(Pin, PinWidget.ToSharedRef(), i, OutNode.Get());
00068         }
00069
00070         PinWidget.Reset();
00071     }
00072
00073     // Wrap The Node Widget
00074     WrapNodeWidget(InNode, OutNode.ToSharedRef(), NodeIndex);
00075
00076     return OutNode;
00077 }
```

## 4.3.3.2 SetSharedPtr()

```
void FAccessibilityNodeFactory::SetSharedPtr (
    TSharedPtr< FAccessibilityNodeFactory > InSharedPtr ) [inline]
```

Definition at line 40 of file [OAccessibilityNodeFactory.h](#).

```
00041 {
00042     ThisPtr = InSharedPtr;
00043 }
```

## 4.3.3.3 WrapNodeWidget()

```
void FAccessibilityNodeFactory::WrapNodeWidget (
    UEdGraphNode * Node,
    TSharedPtr< SGraphNode > NodeWidget,
    int NodeIndex ) const [inline]
```

Wraps the Node Widget with Accessibility Indexing.

## Parameters

|                   |  |
|-------------------|--|
| <i>Node</i>       | The Node Object That is Being Wrapped. |
| <i>NodeWidget</i> | The Node Widget That is Being Wrapped. |
| <i>NodeIndex</i>  | The Index of the Node.                 |

Definition at line 79 of file [OAccessibilityNodeFactory.cpp](#).

```
00080 {
00081     TSharedPtr<SWidget> WidgetToWrap = NodeWidget->GetSlot(ENodeZone::Center)->GetWidget();
00082     check(WidgetToWrap != SNullWidget::NullWidget);
00083
00084     NodeWidget->GetOrAddSlot(ENodeZone::Center)
00085         .HAlign(HAlign_Fill)
00086     [
00087         SNew(SVerticalBox)
00088
00089         + SVerticalBox::Slot()
00090         .HAlign(HAlign_Fill)
00091         .AutoHeight()
00092         .Padding(FMargin(1.5f, 0.25f))
00093     [
00094         SNew(SOverlay)
00095
00096         + SOverlay::Slot()
00097     [
00098         SNew(SImage)
00099         .Image(FAppStyle::Get().GetBrush("Graph.Node.Body"))
00100     ]
00101
00102         + SOverlay::Slot()
00103         .Padding(FMargin(4.0f, 0.0f))
00104     [
00105         SNew(SHorizontalBox)
00106         + SHorizontalBox::Slot()
00107         .HAlign(HAlign_Right)
00108         .VAlign(VAlign_Center)
00109         .Padding(1.f)
00110     [
00111         SNew(SOverlay)
00112         + SOverlay::Slot()
00113     [
00114         SNew(SIndexer)
00115         .IndexValue(NodeIndex)
00116         .TextColor(FLinearColor::White)
00117         .BorderColor(FLinearColor::Gray)
00118     ]
00119     ]
00120     ]
00121 }
```

```

00121         ]
00122
00123         + SVerticalBox::Slot()
00124         .HAlign(HAlign_Fill)
00125         .AutoHeight()
00126         [
00127             WidgetToWrap
00128         ]
00129     };
00130 }

```

#### 4.3.3.4 WrapPinWidget()

```

void FAccessibilityNodeFactory::WrapPinWidget (
    UEdGraphPin * Pin,
    TSharedRef< SGraphPin > PinWidget,
    int PinIndex,
    SGraphNode * OwnerNode ) const [inline]

```

Wraps the Pin Widget with Accessibility Indexing.

##### Parameters

|                  |  |
|------------------|--|
| <i>Pin</i>       | The Pin Object That is Being Wrapped.  |
| <i>PinWidget</i> | The Node Widget That is Being Wrapped. |
| <i>PinIndex</i>  | The Index of the Pin.                  |
| <i>OwnerNode</i> | The Owning Node Widget of the Pin.     |

Definition at line 132 of file [OAccessibilityNodeFactory.cpp](#).

```

00133 {
00134     TSharedRef<SWidget> PinWidgetContent = PinWidget->GetContent();
00135     check(PinWidgetContent != SNullWidget::NullWidget);
00136
00137     TSharedRef<SWidget> AccessibilityWidget = SNew(SOverlay)
00138         .Visibility_Lambda([OwnerNode]() -> EVisibility {
00139
00140             if (OwnerNode->HasAnyUserFocusOrFocusedDescendants() || OwnerNode->IsHovered() ||
00141                 OwnerNode->GetOwnerPanel()->SelectionManager.IsNodeSelected(OwnerNode->GetNodeObj()))
00142                 return EVisibility::Visible;
00143
00144             return EVisibility::Hidden;
00145         })
00146         + SOverlay::Slot()
00147         [
00148             SNew(SIndexer)
00149             .IndexValue(PinIndex)
00150             .TextColor(FLinearColor::White)
00151             .BorderColor(FLinearColor::Gray)
00152         ];
00153     switch (Pin->Direction)
00154     {
00155     case EEdGraphPinDirection::EGPD_Input:
00156     {
00157         PinWidget->SetContent(
00158             SNew(SHorizontalBox)
00159             + SHorizontalBox::Slot()
00160             .AutoWidth()
00161             [
00162                 PinWidgetContent
00163             ]
00164             + SHorizontalBox::Slot()
00165             .AutoWidth()
00166             [
00167                 AccessibilityWidget
00168             ]
00169         );
00170         break;
00171     }
00172     }
00173 }

```

```

00172     }
00173
00174     case EEdGraphPinDirection::EGPD_Output:
00175     {
00176         PinWidget->SetContent (
00177             SNew (SHorizontalBox)
00178                 + SHorizontalBox::Slot ()
00179                 .AutoWidth ()
00180                 [
00181                     AccessibilityWidget
00182                 ]
00183                 + SHorizontalBox::Slot ()
00184                 .AutoWidth ()
00185                 [
00186                     PinWidgetContent
00187                 ]
00188         );
00189         break;
00190     }
00191
00192     default:
00193     {
00194         UE_LOG (LogOpenAccessibility, Error, TEXT ("Pin Direction Not Recognized"));
00195         break;
00196     }
00197 }
00198 }

```

The documentation for this class was generated from the following files:

- Source/OpenAccessibility/Public/OAccessibilityNodeFactory.h
- Source/OpenAccessibility/Private/OAccessibilityNodeFactory.cpp

## 4.4 FAssetAccessibilityRegistry Class Reference

### Public Member Functions

- bool [IsGraphAssetRegistered](#) (const UEdGraph \*InGraph) const  
*Checks if the provided graph asset has been registered with the registry.*
- bool [RegisterGraphAsset](#) (const UEdGraph \*InGraph)  
*Registers the provided graph asset with the registry.*
- bool [RegisterGraphAsset](#) (const UEdGraph \*InGraph, const TSharedRef< [FGraphIndexer](#) > InGraphIndexer)  
*Registers the provided graph asset with the registry.*
- bool [UnregisterGraphAsset](#) (const UEdGraph \*InGraph)  
*Unregisters the provided graph asset from the registry.*
- TSharedRef< [FGraphIndexer](#) > [GetGraphIndexer](#) (const UEdGraph \*InGraph) const  
*Gets the Graph Indexer linked to the provided graph asset.*
- void [GetAllGraphKeyIndexes](#) (TArray< FGuid > &OutGraphKeys) const  
*Gets the Guids of all the Graphs that have been registered with the registry.*
- TArray< FGuid > [GetAllGraphKeyIndexes](#) () const  
*Gets the Guids of all the Graphs that have been registered with the registry.*
- void [GetAllGraphIndexes](#) (TArray< TSharedPtr< [FGraphIndexer](#) > > &OutGraphIndexes) const  
*Gets all the Graph Indexers that have been registered with the registry.*
- TArray< TSharedPtr< [FGraphIndexer](#) > > [GetAllGraphIndexes](#) ()  
*Gets all the Graph Indexers that have been registered with the registry.*
- bool [IsGameWorldAssetRegistered](#) (const UWorld \*InWorld) const  
*Checks if the provided world asset has been registered with the registry.*
- bool [RegisterGameWorldAsset](#) (const UWorld \*InWorld)  
*Registers the UWorld Asset with the Registry.*
- bool [UnregisterGameWorldAsset](#) (const UWorld \*InWorld)  
*Unregisters the provided UWorld Asset from the Registry.*

## Public Attributes

- TMap< FGuid, TSharedPtr< [FGraphIndexer](#) > > [GraphAssetIndex](#)

*A Map containing all the Graph Indexers that have been created for the registered Graph Assets.*

### 4.4.1 Detailed Description

Definition at line 14 of file [AssetAccessibilityRegistry.h](#).

### 4.4.2 Constructor & Destructor Documentation

#### 4.4.2.1 FAssetAccessibilityRegistry()

FAssetAccessibilityRegistry::FAssetAccessibilityRegistry ( )

Definition at line 15 of file [AssetAccessibilityRegistry.cpp](#).

```
00016 {
00017     GraphAssetIndex = TMap<FGuid, TSharedPtr<FGraphIndexer>>();
00018     //GameWorldAssetIndex = TMap<FGuid, FGameWorldIndexer*>();
00019
00020     AssetOpenedInEditorHandle =
00021         GEditor->GetEditorSubsystem<UAssetEditorSubsystem>()->OnAssetOpenedInEditor()
00022         .AddRaw(this, &FAssetAccessibilityRegistry::OnAssetOpenedInEditor);
00023
00024     AssetEditorRequestCloseHandle =
00025         GEditor->GetEditorSubsystem<UAssetEditorSubsystem>()->OnAssetEditorRequestClose()
00026         .AddRaw(this, &FAssetAccessibilityRegistry::OnAssetEditorRequestClose);
00027 }
```

#### 4.4.2.2 ~FAssetAccessibilityRegistry()

FAssetAccessibilityRegistry::~~FAssetAccessibilityRegistry ( )

Definition at line 27 of file [AssetAccessibilityRegistry.cpp](#).

```
00028 {
00029     GEditor->GetEditorSubsystem<UAssetEditorSubsystem>()->OnAssetOpenedInEditor()
00030     .Remove(AssetOpenedInEditorHandle);
00031
00032     GEditor->GetEditorSubsystem<UAssetEditorSubsystem>()->OnAssetEditorRequestClose()
00033     .Remove(AssetEditorRequestCloseHandle);
00034
00035     EmptyGraphAssetIndex();
00036 }
```

### 4.4.3 Member Function Documentation

**4.4.3.1 GetAllGraphIndexes() [1/2]**

```
TArray< TSharedPtr< FGraphIndexer > > FAssetAccessibilityRegistry::GetAllGraphIndexes ( )
```

Gets all the Graph Indexers that have been registered with the registry.

**Returns**

An Array of all the Found Graph Indexers registered with the registry.

Definition at line 150 of file [AssetAccessibilityRegistry.cpp](#).

```
00151 {
00152     TArray<TSharedPtr<FGraphIndexer> GraphIndexArray;
00153
00154     GraphAssetIndex.GenerateValueArray(GraphIndexArray);
00155
00156     return GraphIndexArray;
00157 }
```

**4.4.3.2 GetAllGraphIndexes() [2/2]**

```
void FAssetAccessibilityRegistry::GetAllGraphIndexes (
    TArray< TSharedPtr< FGraphIndexer > > & OutGraphIndexes ) const
```

Gets all the Graph Indexers that have been registered with the registry.

**Parameters**

|                        |   |
|------------------------|---|
| <i>OutGraphIndexes</i> | The Array to apply all the registered graph indexers. |
|------------------------|---|

Definition at line 145 of file [AssetAccessibilityRegistry.cpp](#).

```
00146 {
00147     return GraphAssetIndex.GenerateValueArray(OutGraphIndexes);
00148 }
```

**4.4.3.3 GetAllGraphKeyIndexes() [1/2]**

```
TArray< FGuid > FAssetAccessibilityRegistry::GetAllGraphKeyIndexes ( ) const
```

Gets the Guids of all the Graphs that have been registered with the registry.

**Returns**

An Array of all Found Guids registered with the registry.

Definition at line 137 of file [AssetAccessibilityRegistry.cpp](#).

```
00138 {
00139     TArray<FGuid> GraphKeys;
00140     GraphAssetIndex.GetKeys(GraphKeys);
00141
00142     return GraphKeys;
00143 }
```

#### 4.4.3.4 GetAllGraphKeyIndexes() [2/2]

```
void FAssetAccessibilityRegistry::GetAllGraphKeyIndexes (
    TArray< FGuid > & OutGraphKeys ) const
```

Gets the Guids of all the Graphs that have been registered with the registry.

##### Parameters

|                     |   |
|---------------------|---|
| <i>OutGraphKeys</i> | The Array of Guids to Apply the found Guids to. |
|---------------------|---|

Definition at line 132 of file [AssetAccessibilityRegistry.cpp](#).

```
00133 {
00134     GraphAssetIndex.GetKeys (OutGraphKeys);
00135 }
```

#### 4.4.3.5 GetGraphIndexer()

```
TSharedRef< FGraphIndexer > FAssetAccessibilityRegistry::GetGraphIndexer (
    const UEdGraph * InGraph ) const [inline]
```

Gets the Graph Indexer linked to the provided graph asset.

##### Parameters

|                |   |
|----------------|---|
| <i>InGraph</i> | The Graph to Find the Linked Indexer For. |
|----------------|---|

##### Returns

Returns the Found SharedReference of the GraphIndexer when found successfully. Returns nullptr on fail.

Definition at line 50 of file [AssetAccessibilityRegistry.h](#).

```
00050 {
00051     if (GraphAssetIndex.Contains(InGraph->GraphGuid))
00052         return GraphAssetIndex[InGraph->GraphGuid].ToSharedRef();
00053
00054     return TSharedRef<FGraphIndexer>();
00055 }
```

#### 4.4.3.6 IsGameWorldAssetRegistered()

```
bool FAssetAccessibilityRegistry::IsGameWorldAssetRegistered (
    const UWorld * InWorld ) const
```

Checks if the provided world asset has been registered with the registry.

##### Parameters

|                |   |
|----------------|---|
| <i>InWorld</i> | The UWorld Asset to Check if Registered |
|----------------|---|



### Returns

True, if the UWorld Asset is Registered. False, if the UWorld Asset is not.

Definition at line 159 of file [AssetAccessibilityRegistry.cpp](#).

```
00160 {  
00161     throw std::exception("The method or operation is not implemented.");  
00162 }
```

#### 4.4.3.7 IsGraphAssetRegistered()

```
bool FAssetAccessibilityRegistry::IsGraphAssetRegistered (  
    const UEdGraph * InGraph ) const
```

Checks if the provied graph asset has been registered with the registry.

### Parameters

|                |                           |
|----------------|---------------------------|
| <i>InGraph</i> | The Graph Asset to Check. |
|----------------|---------------------------|

### Returns

True, if the graph has been registered. False, if the graph has not.

Definition at line 71 of file [AssetAccessibilityRegistry.cpp](#).

```
00072 {  
00073     return GraphAssetIndex.Contains(InUEdGraph->GraphGuid);  
00074 }
```

#### 4.4.3.8 RegisterGameWorldAsset()

```
bool FAssetAccessibilityRegistry::RegisterGameWorldAsset (  
    const UWorld * InWorld )
```

Registered the UWorld Asset with the Registry.

### Parameters

|                |                               |
|----------------|-------------------------------|
| <i>InWorld</i> | The UWorld Asset to Register. |
|----------------|-------------------------------|

### Returns

True, if the Asset was Successfully Registered. False, if the asset could not be registered.

Definition at line 164 of file [AssetAccessibilityRegistry.cpp](#).

```
00165 {  
00166     throw std::exception("The method or operation is not implemented.");  
00167 }
```

#### 4.4.3.9 RegisterGraphAsset() [1/2]

```
bool FAssetAccessibilityRegistry::RegisterGraphAsset (
    const UEdGraph * InGraph )
```

Registers the provided graph asset with the registry.

##### Parameters

|                |                        |
|----------------|------------------------|
| <i>InGraph</i> | The Graph to Register. |
|----------------|------------------------|

##### Returns

True, if the Graph was Successfully Registered. False, if the Graph Could Not Be Registered.

Definition at line 76 of file [AssetAccessibilityRegistry.cpp](#).

```
00077 {
00078     if (!InGraph->IsValidLowLevel())
00079         return false;
00080     GraphAssetIndex.Add(InGraph->GraphGuid, MakeShared<FGraphIndexer>(InGraph));
00081     for (auto& ChildGraph : InGraph->SubGraphs)
00082     {
00083         if (!RegisterGraphAsset(ChildGraph))
00084         {
00085             UE_LOG(LogOpenAccessibility, Error, TEXT("|| AssetRegistry || Error When Logging Child
00086 Graph: { %s } From Parent: { %s }||"), *ChildGraph->GetName(), *InGraph->GetName())
00087             return false;
00088         }
00089     }
00090     return true;
00091 }
00092 }
00093 }
00094 }
```

#### 4.4.3.10 RegisterGraphAsset() [2/2]

```
bool FAssetAccessibilityRegistry::RegisterGraphAsset (
    const UEdGraph * InGraph,
    const TSharedRef< FGraphIndexer > InGraphIndexer )
```

Definition at line 96 of file [AssetAccessibilityRegistry.cpp](#).

```
00097 {
00098     if (!InGraph->IsValidLowLevel())
00099         return false;
00100     GraphAssetIndex.Add(InGraph->GraphGuid, InGraphIndexer.ToSharedPtr());
00101     for (auto& ChildGraph : InGraph->SubGraphs)
00102     {
00103         if (!RegisterGraphAsset(ChildGraph))
00104         {
00105             UE_LOG(LogOpenAccessibility, Error, TEXT("|| AssetRegistry || Error When Logging Child
00106 Graph: { %s } From Parent: { %s }||"), *ChildGraph->GetName(), *InGraph->GetName());
00107             return false;
00108         }
00109     }
00110     return true;
00111 }
00112 }
00113 }
```

#### 4.4.3.11 UnregisterGameWorldAsset()

```
bool FAssetAccessibilityRegistry::UnregisterGameWorldAsset (
    const UWorld * InWorld )
```

Unregisters the provided UWorld Asset from the Registry.

**Parameters**

|                |                                 |
|----------------|---------------------------------|
| <i>InWorld</i> | The UWorld Asset to Unregister. |
|----------------|---------------------------------|

**Returns**

True, if the Asset was Successfully Registered. False, if the asset could not be registered.

Definition at line 169 of file [AssetAccessibilityRegistry.cpp](#).

```
00170 {
00171     throw std::exception("The method or operation is not implemented.");
00172 }
```

**4.4.3.12 UnregisterGraphAsset()**

```
bool FAssetAccessibilityRegistry::UnregisterGraphAsset (
    const UEdGraph * InGraph )
```

Unregisters the provided graph asset from the registry.

**Parameters**

|                |                          |
|----------------|--------------------------|
| <i>InGraph</i> | The Graph To Unregister. |
|----------------|--------------------------|

**Returns**

True, if the provided graph was unregistered successfully. False, if the Graph Could Not Be Unregistered.

Definition at line 115 of file [AssetAccessibilityRegistry.cpp](#).

```
00116 {
00117     GraphAssetIndex.Remove(UEdGraph->GraphGuid);
00118
00119     for (auto& ChildGraph : UEdGraph->SubGraphs)
00120     {
00121         if (!UnregisterGraphAsset(ChildGraph))
00122         {
00123             UE_LOG(LogOpenAccessibility, Error, TEXT("|| AssetRegistry || Error When Unregistering
Child Graph: { %s } From Parent: { %s }||"), *ChildGraph->GetName(), *UEdGraph->GetName())
00124
00125             return false;
00126         }
00127     }
00128
00129     return true;
00130 }
```

**4.4.4 Member Data Documentation**

#### 4.4.4.1 GraphAssetIndex

```
TMap<FGuid, TSharedPtr<FGraphIndexer> > FAssetAccessibilityRegistry::GraphAssetIndex
```

A Map containing all the Graph Indexers that have been created for the registered Graph Assets.

Definition at line 162 of file [AssetAccessibilityRegistry.h](#).

The documentation for this class was generated from the following files:

- Source/OpenAccessibility/Public/AssetAccessibilityRegistry.h
- Source/OpenAccessibility/Private/AssetAccessibilityRegistry.cpp

## 4.5 FAudioManagerSettings Struct Reference

### Public Attributes

- float [LevelThreshold](#)
- FString [SaveName](#)  
*The Name of the Audio File to be saved to.*
- FString [SavePath](#)  
*The Path to save recorded audio files to.*

### 4.5.1 Detailed Description

Definition at line 15 of file [AudioManager.h](#).

### 4.5.2 Constructor & Destructor Documentation

#### 4.5.2.1 FAudioManagerSettings()

```
FAudioManagerSettings::FAudioManagerSettings ( ) [inline]
```

Definition at line 20 of file [AudioManager.h](#).

```
00021 {  
00022     // Default Settings  
00023     LevelThreshold = -2.5f;  
00024     SaveName = FString("Captured_User_Audio");  
00025     SavePath = FString("./OpenAccessibility/Audioclips/");  
00026 }
```

### 4.5.3 Member Data Documentation

#### 4.5.3.1 LevelThreshold

```
float FAudioManagerSettings::LevelThreshold
```

Definition at line 30 of file [AudioManager.h](#).

#### 4.5.3.2 SaveName

```
FString FAudioManagerSettings::SaveName
```

The Name of the Audio File to be saved to.

Definition at line 36 of file [AudioManager.h](#).

#### 4.5.3.3 SavePath

```
FString FAudioManagerSettings::SavePath
```

The Path to save recorded audio files to.

Definition at line 42 of file [AudioManager.h](#).

The documentation for this struct was generated from the following file:

- Source/OpenAccessibilityCommunication/Public/AudioManager.h

## 4.6 FGraphIndexer Class Reference

### Public Member Functions

- [FGraphIndexer](#) (const UEdGraph \*GraphToIndex)
- bool [ContainsKey](#) (const int &InKey)  
*Checks if the Provided Key is Contained within the Indexer.*
- int [ContainsNode](#) (UEdGraphNode \*InNode)  
*Checks that the provided Node is Indexed, and retrieves its Key.*
- void [ContainsNode](#) (UEdGraphNode \*InNode, int &OutIndex)  
*Checks that the provided Node is Indexed, and retrieves its Key.*
- int [GetKey](#) (const UEdGraphNode \*InNode)  
*Gets the Key Linked to the Provided Node in the Indexer.*
- bool [GetKey](#) (const UEdGraphNode \*InNode, int &OutKey)  
*Gets the Key Linked to the Provided Node in the Indexer.*
- void [GetNode](#) (const int &InIndex, UEdGraphNode \*OutNode)  
*Gets the Node Linked to the Provided Index.*
- UEdGraphNode \* [GetNode](#) (const int &InIndex)  
*Gets the Node Linked to the Provided Index.*

- void [GetPin](#) (const int &InNodeIndex, const int &InPinIndex, UEdGraphPin \*OutPin)  
*Gets the Pin Linked to the Provided Index, of the Provided Node Index.*
- UEdGraphPin \* [GetPin](#) (const int &InNodeIndex, const int &InPinIndex)  
*Gets the Pin Linked to the Provided Index, of the Provided Node Index.*
- int [AddNode](#) (const UEdGraphNode \*Node)  
*Adds the Given Node to the Indexer.*
- void [AddNode](#) (int &OutIndex, const UEdGraphNode &InNode)  
*Adds the Given Node to the Indexer.*
- int [GetOrAddNode](#) (const UEdGraphNode \*InNode)  
*Gets or Adds the Provided Node to the Indexer.*
- void [GetOrAddNode](#) (const UEdGraphNode \*InNode, int &OutIndex)  
*Gets or Adds the Provided Node to the Indexer.*
- void [RemoveNode](#) (const int &InIndex)  
*Removes the Node from the Indexer, linked to the Provided Index.*
- void [RemoveNode](#) (const UEdGraphNode \*InNode)  
*Removes the Node from the Indexer, finds the Index in the Process.*
- void [OnGraphEvent](#) (const FEdGraphEditAction &InAction)  
*Callback for when the Linked Graph for the Indexer has been Modified.*
- void [OnGraphRebuild](#) ()  
*Calls a Full Rebuild of the Indexer, to ensure all Nodes are Indexed.*

## Protected Attributes

- TMap< int, UEdGraphNode \* > [IndexMap](#)  
*Map of the Index to the Node.*
- TSet< int32 > [NodeSet](#)  
*Look-Up Set of the Nodes Contained in the Indexer.*
- TQueue< int32 > [AvailableIndices](#)  
*A Queue of the Available Indices for the Indexer, that was previously in use but made vacant.*
- UEdGraph \* [LinkedGraph](#)  
*The Graph Being Indexed By This Indexer.*
- FDelegateHandle [OnGraphChangedHandle](#)

### 4.6.1 Detailed Description

Definition at line 14 of file [GraphIndexer.h](#).

### 4.6.2 Constructor & Destructor Documentation

#### 4.6.2.1 FGraphIndexer() [1/2]

```
FGraphIndexer::FGraphIndexer ( )
```

Definition at line 12 of file [GraphIndexer.cpp](#).

```
00013 {
00014
00015 }
```

#### 4.6.2.2 FGraphIndexer() [2/2]

```
FGraphIndexer::FGraphIndexer (
    const UEdGraph * GraphToIndex )
```

Definition at line 17 of file [GraphIndexer.cpp](#).

```
00018 : LinkedGraph(const\_cast<UEdGraph*>(GraphToIndex))
00019 {
00020     BuildGraphIndex();
00021
00022     OnGraphChangedHandle = LinkedGraph->AddOnGraphChangedHandler(
00023         FOnGraphChanged::FDelegate::CreateRaw(this, &FGraphIndexer::OnGraphEvent)
00024     );
00025 }
```

#### 4.6.2.3 ~FGraphIndexer()

```
FGraphIndexer::~~FGraphIndexer ( )
```

Definition at line 27 of file [GraphIndexer.cpp](#).

```
00028 {
00029     IndexMap.Empty();
00030     NodeSet.Empty();
00031     AvailableIndices.Empty();
00032
00033     LinkedGraph->RemoveOnGraphChangedHandler(OnGraphChangedHandle);
00034
00035     LinkedGraph = nullptr;
00036 }
```

### 4.6.3 Member Function Documentation

#### 4.6.3.1 AddNode() [1/2]

```
int FGraphIndexer::AddNode (
    const UEdGraphNode * Node )
```

Adds the Given Node to the Indexer.

##### Parameters

|             |                                 |
|-------------|---------------------------------|
| <i>Node</i> | The Node To Add To The Indexer. |
|-------------|---------------------------------|

##### Returns

The Index of the Node in the Indexer.

Definition at line 134 of file [GraphIndexer.cpp](#).

```
00135 {
00136     check (InNode != nullptr);
00137
00138     if (!InNode->IsValidLowLevelFast())
00139     {
```



```

00140         UE_LOG(LogOpenAccessibility, Warning, TEXT("Provided Node is not valid.))
00141     }
00142
00143     int Index = ContainsNode(const_cast<UEdGraphNode*>(InNode));
00144     if (Index != -1)
00145     {
00146         return Index;
00147     }
00148
00149     GetAvailableIndex(Index);
00150
00151     NodeSet.Add(InNode->GetUniqueID());
00152     IndexMap.Add(Index, const_cast<UEdGraphNode*>(InNode));
00153
00154     return Index;
00155 }

```

#### 4.6.3.2 AddNode() [2/2]

```

void FGraphIndexer::AddNode (
    int & OutIndex,
    const UEdGraphNode & InNode )

```

Adds the Given Node to the Indexer.

##### Parameters

|                 |                                       |
|-----------------|---------------------------------------|
| <i>OutIndex</i> | The Index of the Node in the Indexer. |
| <i>InNode</i>   | The Node To Add to the Indexer.       |

Definition at line 157 of file [GraphIndexer.cpp](#).

```

00158 {
00159     OutIndex = AddNode (&InNode);
00160 }

```

#### 4.6.3.3 ContainsKey()

```

bool FGraphIndexer::ContainsKey (
    const int & InKey )

```

Checks if the Provided Key is Contained within the Indexer.

##### Parameters

|              |  |
|--------------|--|
| <i>InKey</i> | The Key To Check if used in the Indexer. |
|--------------|--|

##### Returns

True, if the Key is Used for Indexing. False, if the Key is Not Used for Indexing.

Definition at line 38 of file [GraphIndexer.cpp](#).

```

00039 {
00040     return IndexMap.Contains(InKey);
00041 }

```

**4.6.3.4 ContainsNode()** [1/2]

```
int FGraphIndexer::ContainsNode (
    UEdGraphNode * InNode )
```

Checks that the provided Node is Indexed, and retrieves its Key.

**Parameters**

|               |                   |
|---------------|-------------------|
| <i>InNode</i> | The Node to Find. |
|---------------|-------------------|

**Returns**

The Key Used to Index The Provided Node. -1 if Unsuccessful in finding the Node.

Definition at line 43 of file [GraphIndexer.cpp](#).

```
00044 {
00045     check(InNode != nullptr);
00046
00047     if (!InNode->IsValidLowLevelFast() || !NodeSet.Contains(InNode->GetUniqueID()))
00048         return -1;
00049
00050     const int* ReturnedIndex = IndexMap.FindKey(InNode);
00051
00052     if (ReturnedIndex != nullptr)
00053     {
00054         return *ReturnedIndex;
00055     }
00056     else return -1;
00057 }
```

**4.6.3.5 ContainsNode()** [2/2]

```
void FGraphIndexer::ContainsNode (
    UEdGraphNode * InNode,
    int & OutIndex )
```

Checks that the provided Node is Indexed, and retrieves its Key.

**Parameters**

|                 |  |
|-----------------|--|
| <i>InNode</i>   | The Node to Find.                      |
| <i>OutIndex</i> | The Index Linked to the Provided Node. |

Definition at line 59 of file [GraphIndexer.cpp](#).

```
00060 {
00061     OutIndex = ContainsNode(InNode);
00062 }
```

**4.6.3.6 GetKey()** [1/2]

```
int FGraphIndexer::GetKey (
    const UEdGraphNode * InNode )
```

Gets the Key Linked to the Provided Node in the Indexer.

**Parameters**

|               |                                |
|---------------|--------------------------------|
| <i>InNode</i> | The Node to find the Index of. |
|---------------|--------------------------------|

**Returns**

The Index of the Provided Node. -1 on Failure.

Definition at line 64 of file [GraphIndexer.cpp](#).

```

00065 {
00066     check(InNode != nullptr);
00067
00068     if (!InNode->IsValidLowLevelFast())
00069         return -1;
00070
00071     const int* FoundKey = IndexMap.FindKey(const_cast<UEdGraphNode*>(InNode));
00072
00073     if (FoundKey != nullptr) return *FoundKey;
00074     else return -1;
00075 }
```

**4.6.3.7 GetKey() [2/2]**

```

bool FGraphIndexer::GetKey (
    const UEdGraphNode * InNode,
    int & OutKey )
```

Gets the Key Linked to the Provided Node in the Indexer.

**Parameters**

|               |                                 |
|---------------|---------------------------------|
| <i>InNode</i> | The Node to find the Index of.  |
| <i>OutKey</i> | The Index of the Provided Node. |

**Returns**

True, if the Key Was Found. False, if the Key Could Not Be Found.

Definition at line 77 of file [GraphIndexer.cpp](#).

```

00078 {
00079     check(InNode != nullptr);
00080
00081     if (!InNode->IsValidLowLevelFast())
00082         return false;
00083
00084     const int* FoundKey = IndexMap.FindKey(const_cast<UEdGraphNode*>(InNode));
00085     if (FoundKey != nullptr)
00086     {
00087         OutKey = *FoundKey;
00088         return true;
00089     }
00090     else return false;
00091 }
```

**4.6.3.8 GetNode() [1/2]**

```
UEdGraphNode * FGraphIndexer::GetNode (
    const int & InIndex )
```

Gets the Node Linked to the Provided Index.

**Parameters**

|                |                                |
|----------------|--------------------------------|
| <i>InIndex</i> | The Index to Find the Node of. |
|----------------|--------------------------------|

**Returns**

The Found Graph Node, nullptr on Failure.

Definition at line 93 of file [GraphIndexer.cpp](#).

```
00094 {
00095     if (!IndexMap.Contains(InIndex))
00096     {
00097         UE_LOG(LogOpenAccessibility, Warning, TEXT("Provided Index is not recognised"))
00098     }
00099     return nullptr;
00100 }
00101
00102 return IndexMap[InIndex];
00103 }
```

**4.6.3.9 GetNode() [2/2]**

```
void FGraphIndexer::GetNode (
    const int & InIndex,
    UEdGraphNode * OutNode )
```

Gets the Node Linked to the Provided Index.

**Parameters**

|                |                                       |
|----------------|---------------------------------------|
| <i>InIndex</i> | The Index to Find the Node of.        |
| <i>OutNode</i> | Applies the Found Node, else nullptr. |

Definition at line 129 of file [GraphIndexer.cpp](#).

```
00130 {
00131     OutNode = GetNode(InIndex);
00132 }
```

**4.6.3.10 GetOrAddNode() [1/2]**

```
int FGraphIndexer::GetOrAddNode (
    const UEdGraphNode * InNode )
```

Gets or Adds the Provided Node to the Indexer.

**Parameters**

|               |  |
|---------------|--|
| <i>InNode</i> | The Node to Get or Look-Up in the Indexer. |
|---------------|--|

**Returns**

The Index of the Node in the Graph.

Definition at line 162 of file [GraphIndexer.cpp](#).

```

00163 {
00164     int Key = GetKey(InNode);
00165     if (Key != -1)
00166     {
00167         return Key;
00168     }
00169     return AddNode(InNode);
00170 }
00171 }
```

**4.6.3.11 GetOrAddNode() [2/2]**

```

void FGraphIndexer::GetOrAddNode (
    const UEdGraphNode * InNode,
    int & OutIndex )
```

Gets or Adds the Provided Node to the Indexer.

**Parameters**

|                 |  |
|-----------------|--|
| <i>InNode</i>   | The Node to Get or Look-Up in the Indexer. |
| <i>OutIndex</i> | The Index of the Node in the Graph.        |

Definition at line 173 of file [GraphIndexer.cpp](#).

```

00174 {
00175     OutIndex = GetKey(InNode);
00176     if (OutIndex != -1)
00177     {
00178         return;
00179     }
00180     OutIndex = AddNode(InNode);
00181 }
00182 }
```

**4.6.3.12 GetPin() [1/2]**

```

UEdGraphPin * FGraphIndexer::GetPin (
    const int & InNodeIndex,
    const int & InPinIndex )
```

Gets the Pin Linked to the Provided Index, of the Provided Node Index.

## Parameters

|                    |   |
|--------------------|---|
| <i>InNodeIndex</i> | The Index of the Node to find the Pin From. |
| <i>InPinIndex</i>  | The Index of the Pin on the Provided Node.  |

## Returns

The Found Pin on the Provided Node, nullptr on Failure.

Definition at line 117 of file [GraphIndexer.cpp](#).

```

00118 {
00119     UEdGraphNode* Node = GetNode(InNodeIndex);
00120     if (Node == nullptr)
00121     {
00122         UE_LOG(LogOpenAccessibility, Warning, TEXT("Requested Node at index %d is not valid."),
00123             InNodeIndex);
00124         return nullptr;
00125     }
00126     return Node->GetPinAt(InPinIndex); // Returns nullptr if invalid
00127 }
```

## 4.6.3.13 GetPin() [2/2]

```

void FGraphIndexer::GetPin (
    const int & InNodeIndex,
    const int & InPinIndex,
    UEdGraphPin * OutPin )
```

Gets the Pin Linked to the Provided Index, of the Provided Node Index.

## Parameters

|                    |   |
|--------------------|---|
| <i>InNodeIndex</i> | The Index of the Node to find the Pin From. |
| <i>InPinIndex</i>  | The Index of the Pin on the Provided Node.  |
| <i>OutPin</i>      | The Found Pin on the Provided Node.         |

Definition at line 105 of file [GraphIndexer.cpp](#).

```

00106 {
00107     UEdGraphNode* Node = GetNode(InNodeIndex);
00108     if (Node == nullptr)
00109     {
00110         UE_LOG(LogOpenAccessibility, Warning, TEXT("Requested Node at index %d is not valid."),
00111             InNodeIndex);
00112         return;
00113     }
00114     OutPin = Node->GetPinAt(InPinIndex); // Returns nullptr if invalid
00115 }
```

## 4.6.3.14 OnGraphEvent()

```

void FGraphIndexer::OnGraphEvent (
    const FEdGraphEditAction & InAction )
```

Callback for when the Linked Graph for the Indexer has been Modified.

## Parameters

|                 |  |
|-----------------|--|
| <i>InAction</i> |  |
|-----------------|--|

Definition at line 225 of file [GraphIndexer.cpp](#).

```

00226 {
00227     if (InAction.Graph != LinkedGraph)
00228     {
00229         return;
00230     }
00231
00232     switch (InAction.Action)
00233     {
00234         case EEdGraphActionType::GRAPHACTION_AddNode:
00235         {
00236             for (const UEdGraphNode* Node : InAction.Nodes)
00237             {
00238                 AddNode (Node);
00239             }
00240
00241             break;
00242         }
00243
00244         case EEdGraphActionType::GRAPHACTION_RemoveNode:
00245         {
00246             for (const UEdGraphNode* Node : InAction.Nodes)
00247             {
00248                 RemoveNode (Node);
00249             }
00250
00251             break;
00252         }
00253     }
00254 }
```

#### 4.6.3.15 OnGraphRebuild()

```
void FGraphIndexer::OnGraphRebuild ( )
```

Calls a Full Rebuild of the Indexer, to ensure all Nodes are Indexed.

Definition at line 256 of file [GraphIndexer.cpp](#).

```

00257 {
00258     IndexMap.Reset ();
00259     NodeSet.Reset ();
00260     AvailableIndices.Empty ();
00261
00262     BuildGraphIndex ();
00263 }
```

#### 4.6.3.16 RemoveNode() [1/2]

```
void FGraphIndexer::RemoveNode (
    const int & InIndex )
```

Removes the Node from the Indexer, linked to the Provided Index.

## Parameters

|                |  |
|----------------|--|
| <i>InIndex</i> | The Index to Remove from the Indexer, and its Linked Node. |
|----------------|--|



Definition at line 184 of file [GraphIndexer.cpp](#).

```
00185 {
00186     if (!IndexMap.Contains(InIndex))
00187     {
00188         UE_LOG(LogOpenAccessibility, Warning, TEXT("Provided Index is not recognised"))
00189     }
00190
00191     const UEdGraphNode* Node = IndexMap[InIndex];
00192
00193     if (Node->IsValidLowLevelFast())
00194     {
00195         NodeSet.Remove(Node->GetUniqueID());
00196         IndexMap.Remove(InIndex);
00197         AvailableIndices.Enqueue(InIndex);
00198     }
00199     else
00200     {
00201         UE_LOG(LogOpenAccessibility, Warning, TEXT("Stored Node in IndexMap is not valid.))
00202     }
00203 }
```

#### 4.6.3.17 RemoveNode() [2/2]

```
void FGraphIndexer::RemoveNode (
    const UEdGraphNode * InNode )
```

Removes the Node from the Indexer, finds the Index in the Process.

##### Parameters

|               |  |
|---------------|--|
| <i>InNode</i> | The Node To Remove from the Indexer, and its Linked Index. |
|---------------|--|

Definition at line 205 of file [GraphIndexer.cpp](#).

```
00206 {
00207     check(InNode != nullptr);
00208
00209     int Key = GetKey(InNode);
00210     if (Key == -1)
00211     {
00212         UE_LOG(LogOpenAccessibility, Warning, TEXT("Node does not exist in IndexMap.))
00213         return;
00214     }
00215
00216     RemoveNode(Key);
00217 }
```

### 4.6.4 Member Data Documentation

#### 4.6.4.1 AvailableIndices

```
TQueue<int32> FGraphIndexer::AvailableIndices [protected]
```

A Queue of the Available Indices for the Indexer, that was previously in use but made vacant.

Definition at line 173 of file [GraphIndexer.h](#).

#### 4.6.4.2 IndexMap

```
TMap<int, UEdGraphNode*> FGraphIndexer::IndexMap [protected]
```

Map of the Index to the Node.

Definition at line 163 of file [GraphIndexer.h](#).

#### 4.6.4.3 LinkedGraph

```
UEdGraph* FGraphIndexer::LinkedGraph [protected]
```

The Graph Being Indexed By This Indexer.

Definition at line 178 of file [GraphIndexer.h](#).

#### 4.6.4.4 NodeSet

```
TSet<int32> FGraphIndexer::NodeSet [protected]
```

Look-Up Set of the Nodes Contained in the Indexer.

Definition at line 168 of file [GraphIndexer.h](#).

#### 4.6.4.5 OnGraphChangedHandle

```
FDelegateHandle FGraphIndexer::OnGraphChangedHandle [protected]
```

Definition at line 180 of file [GraphIndexer.h](#).

The documentation for this class was generated from the following files:

- Source/OpenAccessibility/Public/GraphIndexer.h
- Source/OpenAccessibility/Private/GraphIndexer.cpp

## 4.7 FGraphLocomotionChunk Struct Reference

### Public Member Functions

- void [SetChunkBounds](#) (FVector2D InTopLeft, FVector2D InBottomRight)
- void [GetChunkBounds](#) (FVector2D &OutTopLeft, FVector2D &OutBottomRight) const
- FVector2D [GetChunkTopLeft](#) () const
- FVector2D [GetChunkBottomRight](#) () const
- void [SetVisColor](#) (const FLinearColor &NewColor) const

## Public Attributes

- FVector2D [TopLeft](#)  
*Visual Chunks Top Left Corner.*
- FVector2D [BottomRight](#)  
*Visual Chunks Bottom Right Corner.*
- TWeakPtr< SBox > [ChunkWidget](#)  
*Weak Pointer to the Chunks Visual Widget.*
- TWeakPtr< SBorder > [ChunkVisWidget](#)  
*Weak Pointer to the Chunks Visual Widget.*
- TWeakPtr< class [SIndexer](#) > [ChunkIndexer](#)  
*Weak Pointer to the Chunks Indexer Widget.*

### 4.7.1 Detailed Description

Definition at line 13 of file [AccessibilityGraphLocomotionContext.h](#).

### 4.7.2 Member Function Documentation

#### 4.7.2.1 GetChunkBottomRight()

```
FVector2D FGraphLocomotionChunk::GetChunkBottomRight ( ) const [inline]
```

Definition at line 33 of file [AccessibilityGraphLocomotionContext.h](#).

```
00033 { return BottomRight; }
```

#### 4.7.2.2 GetChunkBounds()

```
void FGraphLocomotionChunk::GetChunkBounds (
    FVector2D & OutTopLeft,
    FVector2D & OutBottomRight ) const [inline]
```

Definition at line 25 of file [AccessibilityGraphLocomotionContext.h](#).

```
00026 {
00027     OutTopLeft = TopLeft;
00028     OutBottomRight = BottomRight;
00029 }
```

#### 4.7.2.3 GetChunkTopLeft()

```
FVector2D FGraphLocomotionChunk::GetChunkTopLeft ( ) const [inline]
```

Definition at line 31 of file [AccessibilityGraphLocomotionContext.h](#).

```
00031 { return TopLeft; }
```

#### 4.7.2.4 SetChunkBounds()

```
void FGraphLocomotionChunk::SetChunkBounds (
    FVector2D InTopLeft,
    FVector2D InBottomRight ) [inline]
```

Definition at line 19 of file [AccessibilityGraphLocomotionContext.h](#).

```
00020 {
00021     TopLeft = InTopLeft;
00022     BottomRight = InBottomRight;
00023 }
```

#### 4.7.2.5 SetVisColor()

```
void FGraphLocomotionChunk::SetVisColor (
    const FLinearColor & NewColor ) const [inline]
```

Definition at line 35 of file [AccessibilityGraphLocomotionContext.h](#).

```
00036 {
00037     if (ChunkVisWidget.IsValid())
00038         ChunkVisWidget.Pin()->SetBorderBackgroundColor(NewColor);
00039 }
```

### 4.7.3 Member Data Documentation

#### 4.7.3.1 BottomRight

```
FVector2D FGraphLocomotionChunk::BottomRight
```

Visual Chunks Bottom Right Corner.

Definition at line 51 of file [AccessibilityGraphLocomotionContext.h](#).

#### 4.7.3.2 ChunkIndexer

```
TWeakPtr<class SIndexer> FGraphLocomotionChunk::ChunkIndexer
```

Weak Pointer to the Chunks Indexer Widget.

Definition at line 66 of file [AccessibilityGraphLocomotionContext.h](#).

#### 4.7.3.3 ChunkVisWidget

```
TWeakPtr<SBorder> FGraphLocomotionChunk::ChunkVisWidget
```

Weak Pointer to the Chunks Visual Widget.

Definition at line 61 of file [AccessibilityGraphLocomotionContext.h](#).

#### 4.7.3.4 ChunkWidget

```
TWeakPtr<SBox> FGraphLocomotionChunk::ChunkWidget
```

Weak Pointer to the Chunks Visual Widget.

Definition at line 56 of file [AccessibilityGraphLocomotionContext.h](#).

#### 4.7.3.5 TopLeft

```
FVector2D FGraphLocomotionChunk::TopLeft
```

Visual Chunks Top Left Corner.

Definition at line 46 of file [AccessibilityGraphLocomotionContext.h](#).

The documentation for this struct was generated from the following file:

- Source/OpenAccessibility/Public/AccessibilityWrappers/AccessibilityGraphLocomotionContext.h

## 4.8 FIndexer< KeyType, ValueType > Class Template Reference

```
#include <Indexer.h>
```

### Public Member Functions

- bool [IsEmpty](#) () const
- void [Reset](#) ()
- void [Empty](#) ()
- int32 [Num](#) () const
- void [Num](#) (int32 &OutNum) const
- bool [ContainsKey](#) (const KeyType &InKey)
- bool [ContainsValue](#) (const ValueType &InValue)
- const KeyType [GetKey](#) (const ValueType &InValue)
- bool [GetKey](#) (const ValueType &InValue, KeyType &OutKey)
- ValueType [GetValue](#) (const KeyType &InKey)
- bool [GetValue](#) (const KeyType &InKey, ValueType &OutValue)
- KeyType [AddValue](#) (const ValueType &InValue)
- void [AddValue](#) (const ValueType &InValue, KeyType &OutKey)
- KeyType [GetKeyOrAddValue](#) (const ValueType &InValue)
- void [GetKeyOrAddValue](#) (const ValueType &InValue, KeyType &OutKey)
- void [RemoveValue](#) (const KeyType &InKey)
- void [RemoveValue](#) (const ValueType &InValue)

## Protected Member Functions

- void [GetAvailableKey](#) (KeyType &OutKey)
- KeyType [GetAvailableKey](#) ()

## Protected Attributes

- TMap< KeyType, ValueType > [IndexMap](#)
- TQueue< KeyType > [AvailableIndexes](#)

### 4.8.1 Detailed Description

```
template<typename KeyType, typename ValueType>
class FIndexer< KeyType, ValueType >
```

A Templated Indexer for Indexing Assets in a TMap.

#### Template Parameters

|                  |   |
|------------------|---|
| <i>KeyType</i>   | Type of the Key Element of the Index.   |
| <i>ValueType</i> | Type of the Value Element of the Index. |

Definition at line 15 of file [Indexer.h](#).

### 4.8.2 Constructor & Destructor Documentation

#### 4.8.2.1 FIndexer()

```
template<typename KeyType , typename ValueType >
FIndexer< KeyType, ValueType >::FIndexer ( ) [inline]
```

Definition at line 19 of file [Indexer.h](#).

```
00020     {
00021
00022     }
```

#### 4.8.2.2 ~FIndexer()

```
template<typename KeyType , typename ValueType >
virtual FIndexer< KeyType, ValueType >::~~FIndexer ( ) [inline], [virtual]
```

Definition at line 24 of file [Indexer.h](#).

```
00025     {
00026
00027     }
```

### 4.8.3 Member Function Documentation

#### 4.8.3.1 AddValue() [1/2]

```
template<typename KeyType , typename ValueType >
KeyType FIndexer< KeyType, ValueType >::AddValue (
    const ValueType & InValue ) [inline]
```

Inserts the specified value into the indexer, and provides its key.

##### Parameters

|                |                      |
|----------------|----------------------|
| <i>InValue</i> | The value to insert. |
|----------------|----------------------|

##### Returns

The Key of the associated to the inserted value in the indexer.

Definition at line 166 of file [Indexer.h](#).

```
00167     {
00168         check(InValue != nullptr);
00169
00170         if (ContainsValue(InValue))
00171         {
00172             return GetKey(InValue);
00173         }
00174
00175         KeyType NewKey;
00176         GetAvailableKey(NewKey);
00177
00178         IndexMap.Add(NewKey, InValue);
00179
00180         return NewKey;
00181     }
```

#### 4.8.3.2 AddValue() [2/2]

```
template<typename KeyType , typename ValueType >
void FIndexer< KeyType, ValueType >::AddValue (
    const ValueType & InValue,
    KeyType & OutKey ) [inline]
```

Inserts the specified value into the indexer, and provides its key.

##### Parameters

|                |  |
|----------------|--|
| <i>InValue</i> | The value to insert.                                   |
| <i>OutKey</i>  | The Key of the associated to the newly inserted value. |

Definition at line 188 of file [Indexer.h](#).

```
00189     {
00190         check(InValue != nullptr);
```

```

00191
00192         if (ContainsValue(InValue))
00193         {
00194             OutKey = GetKey(InValue);
00195             return;
00196         }
00197
00198         OutKey = GetAvailableKey();
00199
00200         IndexMap.Add(OutKey, InValue);
00201     }

```

#### 4.8.3.3 ContainsKey()

```

template<typename KeyType , typename ValueType >
bool FIndexer< KeyType, ValueType >::ContainsKey (
    const KeyType & InKey ) [inline]

```

Checks if the indexer contains the specified key.

##### Parameters

|              |                        |
|--------------|------------------------|
| <i>InKey</i> | The Key to Search For. |
|--------------|------------------------|

##### Returns

True if the Key is in use in the Indexer, otherwise False.

Definition at line 80 of file [Indexer.h](#).

```

00081     {
00082         return IndexMap.Contains(InKey);
00083     }

```

#### 4.8.3.4 ContainsValue()

```

template<typename KeyType , typename ValueType >
bool FIndexer< KeyType, ValueType >::ContainsValue (
    const ValueType & InValue ) [inline]

```

Checks if the Indexer contains the specified value.

##### Parameters

|                |                          |
|----------------|--------------------------|
| <i>InValue</i> | The Value to Search For. |
|----------------|--------------------------|

##### Returns

True if the specified value is associated with the Indexer.

Definition at line 90 of file [Indexer.h](#).



```

00091     {
00092         check(InValue != nullptr);
00093
00094         const KeyType* FoundKey = IndexMap.FindKey(InValue);
00095
00096         return FoundKey != nullptr;
00097     }

```

#### 4.8.3.5 Empty()

```

template<typename KeyType , typename ValueType >
void FIndexer< KeyType, ValueType >::Empty ( ) [inline]

```

Empties the Indexer, preserving no space allocated.

Definition at line 51 of file [Indexer.h](#).

```

00052     {
00053         IndexMap.Empty();
00054         AvailableIndexes.Empty();
00055     }

```

#### 4.8.3.6 GetAvailableKey() [1/2]

```

template<typename KeyType , typename ValueType >
KeyType FIndexer< KeyType, ValueType >::GetAvailableKey ( ) [inline], [protected]

```

Gets the Next Available Key in the Indexer.

##### Returns

The next available key in the indexer.

Definition at line 285 of file [Indexer.h](#).

```

00286     {
00287         if (!AvailableIndexes.IsEmpty())
00288         {
00289             KeyType OutKey;
00290             if (AvailableIndexes.Dequeue(OutKey))
00291                 return OutKey;
00292         }
00293
00294         return IndexMap.Num();
00295     }

```

#### 4.8.3.7 GetAvailableKey() [2/2]

```

template<typename KeyType , typename ValueType >
void FIndexer< KeyType, ValueType >::GetAvailableKey (
    KeyType & OutKey ) [inline], [protected]

```

Gets the Next Available Key in the Indexer.

**Parameters**

|               |                              |
|---------------|------------------------------|
| <i>OutKey</i> | Sets the Next Available Key. |
|---------------|------------------------------|

Definition at line 273 of file [Indexer.h](#).

```
00274     {
00275         if (!AvailableIndexes.IsEmpty() && AvailableIndexes.Dequeue(OutKey))
00276             return;
00277
00278         OutKey = IndexMap.Num();
00279     }
```

**4.8.3.8 GetKey() [1/2]**

```
template<typename KeyType , typename ValueType >
const KeyType FIndexer< KeyType, ValueType >::GetKey (
    const ValueType & InValue ) [inline]
```

Gets the associated Key with the specified value.

**Parameters**

|                |                            |
|----------------|----------------------------|
| <i>InValue</i> | The value to search using. |
|----------------|----------------------------|

**Returns**

The associated key for the specified value.

Definition at line 104 of file [Indexer.h](#).

```
00105     {
00106         check(InValue != nullptr);
00107
00108         return *IndexMap.FindKey(InValue);
00109     }
```

**4.8.3.9 GetKey() [2/2]**

```
template<typename KeyType , typename ValueType >
bool FIndexer< KeyType, ValueType >::GetKey (
    const ValueType & InValue,
    KeyType & OutKey ) [inline]
```

Gets the associated Key with the specified value.

**Parameters**

|                |   |
|----------------|---|
| <i>InValue</i> | The value to search using.                      |
| <i>OutKey</i>  | Sets the associated key for the specified value |

**Returns**

True if the associated key was found, otherwise False.

Definition at line 117 of file [Indexer.h](#).

```
00118     {
00119         check(InValue != nullptr);
00120
00121         const KeyType* FoundKey = IndexMap.FindKey(InValue);
00122
00123         if (FoundKey != nullptr)
00124         {
00125             OutKey = *FoundKey;
00126
00127             return true;
00128         }
00129         else return false;
00130     }
```

**4.8.3.10 GetKeyOrAddValue() [1/2]**

```
template<typename KeyType , typename ValueType >
KeyType FIndexer< KeyType, ValueType >::GetKeyOrAddValue (
    const ValueType & InValue ) [inline]
```

Finds or Inserts the specified value into the Indexer.

**Parameters**

|                |   |
|----------------|---|
| <i>InValue</i> | The value to find or insert into the indexer. |
|----------------|---|

**Returns**

The Key of the associated value.

Definition at line 208 of file [Indexer.h](#).

```
00209     {
00210         check(InValue != nullptr);
00211
00212         KeyType FoundKey;
00213         if (GetKey(InValue, FoundKey))
00214             return FoundKey;
00215
00216         return AddValue(InValue);
00217     }
```

**4.8.3.11 GetKeyOrAddValue() [2/2]**

```
template<typename KeyType , typename ValueType >
void FIndexer< KeyType, ValueType >::GetKeyOrAddValue (
    const ValueType & InValue,
    KeyType & OutKey ) [inline]
```

Finds or Inserts the specified value into the Indexer.

## Parameters

|                |   |
|----------------|---|
| <i>InValue</i> | The value to find or insert into the indexer. |
| <i>OutKey</i>  | Sets the Key of the associated value.         |

Definition at line 224 of file [Indexer.h](#).

```

00225     {
00226         check(InValue != nullptr);
00227
00228         if (GetKey(InValue, OutKey))
00229             return;
00230
00231         OutKey = AddValue(InValue);
00232     }

```

#### 4.8.3.12 GetValue() [1/2]

```

template<typename KeyType , typename ValueType >
ValueType FIndexer< KeyType, ValueType >::GetValue (
    const KeyType & InKey ) [inline]

```

Gets the value linked to the specified key.

## Parameters

|              |                          |
|--------------|--------------------------|
| <i>InKey</i> | The Key to Search using. |
|--------------|--------------------------|

## Returns

The associated value of the specified key.

Definition at line 137 of file [Indexer.h](#).

```

00138     {
00139         return *IndexMap.Find(InKey);
00140     }

```

#### 4.8.3.13 GetValue() [2/2]

```

template<typename KeyType , typename ValueType >
bool FIndexer< KeyType, ValueType >::GetValue (
    const KeyType & InKey,
    ValueType & OutValue ) [inline]

```

Gets the value linked to the specified key.

## Parameters

|                 |   |
|-----------------|---|
| <i>InKey</i>    | The Key to Search using.                        |
| <i>OutValue</i> | Sets the associated value of the specified key. |

**Returns**

True if an associated value was found, otherwise False.

Definition at line 148 of file [Indexer.h](#).

```
00149     {
00150         if (!IndexMap.Contains(InKey))
00151         {
00152             UE_LOG(LogOpenAccessibility, Warning, TEXT("Provided Key is not recognised.));
00153             return false;
00154         }
00155
00156         OutValue = *IndexMap.Find(InKey);
00157
00158         return true;
00159     }
```

**4.8.3.14 IsEmpty()**

```
template<typename KeyType , typename ValueType >
bool FIndexer< KeyType, ValueType >::IsEmpty ( ) const [inline]
```

Checks if the Indexer is Empty.

**Returns**

True if the Indexer is Empty, otherwise False.

Definition at line 34 of file [Indexer.h](#).

```
00035     {
00036         return IndexMap.IsEmpty();
00037     }
```

**4.8.3.15 Num() [1/2]**

```
template<typename KeyType , typename ValueType >
int32 FIndexer< KeyType, ValueType >::Num ( ) const [inline]
```

Gets the Number of Items Currently in the Indexer.

**Returns**

Number of Items being Indexed.

Definition at line 61 of file [Indexer.h](#).

```
00062     {
00063         return IndexMap.Num();
00064     }
```

**4.8.3.16 Num() [2/2]**

```
template<typename KeyType , typename ValueType >
void FIndexer< KeyType, ValueType >::Num (
    int32 & OutNum ) const [inline]
```

Gets the Number of Items Currently in the Indexer.

**Parameters**

|               |  |
|---------------|--|
| <i>OutNum</i> | Sets to the Number of Items Being Indexed. |
|---------------|--|

Definition at line 70 of file [Indexer.h](#).

```
00071     {
00072         OutNum = IndexMap.Num();
00073     }
```

**4.8.3.17 RemoveValue() [1/2]**

```
template<typename KeyType , typename ValueType >
void FIndexer< KeyType, ValueType >::RemoveValue (
    const KeyType & InKey ) [inline]
```

Removes the specified key from the Indexer.

**Parameters**

|              |                                     |
|--------------|-------------------------------------|
| <i>InKey</i> | The key to remove from the indexer. |
|--------------|-------------------------------------|

Definition at line 238 of file [Indexer.h](#).

```
00239     {
00240         if (!IndexMap.Contains(InKey))
00241         {
00242             UE_LOG(LogOpenAccessibility, Warning, TEXT("Provided Key Has No Pair in Index."));
00243             return;
00244         }
00245         IndexMap.Remove(InKey);
00246         AvailableIndexes.Enqueue(InKey);
00247     }
```

**4.8.3.18 RemoveValue() [2/2]**

```
template<typename KeyType , typename ValueType >
void FIndexer< KeyType, ValueType >::RemoveValue (
    const ValueType & InValue ) [inline]
```

Removes the specified value and its associated key from the Indexer.

**Parameters**

|                |                                       |
|----------------|---------------------------------------|
| <i>InValue</i> | The value to remove from the Indexer. |
|----------------|---------------------------------------|

Definition at line 254 of file [Indexer.h](#).

```
00255     {
00256         check(InValue != nullptr);
00257         KeyType FoundKey;
00258         if (GetKey(InValue, FoundKey))
00259         {
00260             IndexMap.Remove(FoundKey);
00261         }
```

```

00262         AvailableIndexes.Enqueue(FoundKey);
00263     }
00264     else UE_LOG(LogOpenAccessibility, Log, TEXT("Provided Value Had No Associated Key.));
00265 }

```

#### 4.8.3.19 Reset()

```

template<typename KeyType , typename ValueType >
void FIndexer< KeyType, ValueType >::Reset ( ) [inline]

```

Empties the Indexer, but preserves all Allocations.

Definition at line 42 of file [Indexer.h](#).

```

00043     {
00044         IndexMap.Reset();
00045         AvailableIndexes.Empty();
00046     }

```

### 4.8.4 Member Data Documentation

#### 4.8.4.1 AvailableIndexes

```

template<typename KeyType , typename ValueType >
TQueue<KeyType> FIndexer< KeyType, ValueType >::AvailableIndexes [protected]

```

The Queue of Available Indexes from Previous Associations.

Definition at line 310 of file [Indexer.h](#).

#### 4.8.4.2 IndexMap

```

template<typename KeyType , typename ValueType >
TMap<KeyType, ValueType> FIndexer< KeyType, ValueType >::IndexMap [protected]

```

The Map of Keys to Associated Values.

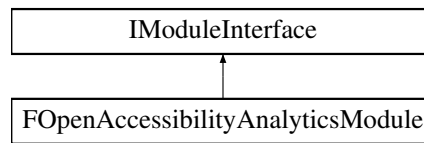
Definition at line 305 of file [Indexer.h](#).

The documentation for this class was generated from the following file:

- [Source/OpenAccessibility/Public/Indexers/Indexer.h](#)

## 4.9 FOpenAccessibilityAnalyticsModule Class Reference

Inheritance diagram for FOpenAccessibilityAnalyticsModule:



### Public Member Functions

- virtual void [StartupModule](#) () override
- virtual void [ShutdownModule](#) () override
- virtual bool [SupportsDynamicReloading](#) () override
- bool [DumpTick](#) (float DeltaTime)
- void [LogEvent](#) (const TCHAR \*EventTitle, const TCHAR \*LogString,...)

### Static Public Member Functions

- static [FOpenAccessibilityAnalyticsModule & Get](#) ()

### 4.9.1 Detailed Description

Definition at line 15 of file [OpenAccessibilityAnalytics.h](#).

### 4.9.2 Member Function Documentation

#### 4.9.2.1 DumpTick()

```
bool FOpenAccessibilityAnalyticsModule::DumpTick (
    float DeltaTime )
```

The Tick Event for Handling the Dump Event.

#### Parameters

|                  |                       |
|------------------|-----------------------|
| <i>DeltaTime</i> | Time since last Tick. |
|------------------|-----------------------|

#### Returns

Definition at line 23 of file [OpenAccessibilityAnalytics.cpp](#).



```

00024 {
00025     if (EventBuffer.IsEmpty())
00026         return true;
00027
00028     if (SessionBufferFile.IsEmpty())
00029         SessionBufferFile = GenerateFileForSessionLog();
00030
00031     UE_LOG(LogOpenAccessibilityAnalytics, Log, TEXT("Dumping Event Log To File.));
00032
00033     if (!WriteBufferToFile())
00034     {
00035         UE_LOG(LogOpenAccessibilityAnalytics, Warning, TEXT("EventLog Dumping Failed.));
00036     }
00037
00038     return true;
00039 }

```

#### 4.9.2.2 Get()

```

static FOpenAccessibilityAnalyticsModule & FOpenAccessibilityAnalyticsModule::Get ( ) [inline],
[static]

```

End IModuleInterface Implementation

Definition at line 28 of file [OpenAccessibilityAnalytics.h](#).

```

00029 {
00030     return
    FModuleManager::GetModuleChecked<FOpenAccessibilityAnalyticsModule>("OpenAccessibilityAnalytics");
00031 }

```

#### 4.9.2.3 LogEvent()

```

FORCEINLINE void FOpenAccessibilityAnalyticsModule::LogEvent (
    const TCHAR * EventTitle,
    const TCHAR * LogString,
    ... )

```

Logs the Event to the Analytics Module.

Parameters

|                   |                         |
|-------------------|-------------------------|
| <i>EventTitle</i> | Title of the Log Event. |
| <i>LogString</i>  | Body of the Log Event   |
| ...               |                         |

Definition at line 135 of file [OpenAccessibilityAnalytics.h](#).

```

00136 {
00137     va_list Args;
00138
00139     va_start(Args, LogString);
00140     TStringBuilder<1024> Message;
00141     Message.AppendV(LogString, Args);
00142     va_end(Args);
00143
00144     EventBuffer.Add(
00145         LoggedEvent(EventTitle, *Message)
00146     );
00147 }

```

#### 4.9.2.4 ShutdownModule()

```
void FOpenAccessibilityAnalyticsModule::ShutdownModule ( ) [override], [virtual]
```

Definition at line 17 of file [OpenAccessibilityAnalytics.cpp](#).

```
00018 {
00019     DisableDumpTick();
00020     RemoveConsoleCommands();
00021 }
```

#### 4.9.2.5 StartupModule()

```
void FOpenAccessibilityAnalyticsModule::StartupModule ( ) [override], [virtual]
```

IModuleInterface Implementation

Definition at line 9 of file [OpenAccessibilityAnalytics.cpp](#).

```
00010 {
00011     SessionBufferFile = GenerateFileForSessionLog();
00012
00013     EnableDumpTick();
00014     AddConsoleCommands();
00015 }
```

#### 4.9.2.6 SupportsDynamicReloading()

```
virtual bool FOpenAccessibilityAnalyticsModule::SupportsDynamicReloading ( ) [inline], [override],
[virtual]
```

Definition at line 24 of file [OpenAccessibilityAnalytics.h](#).

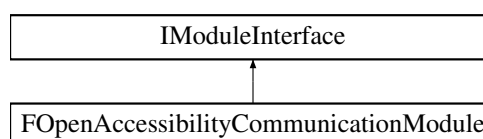
```
00024 { return false; }
```

The documentation for this class was generated from the following files:

- Source/OpenAccessibilityAnalytics/Public/OpenAccessibilityAnalytics.h
- Source/OpenAccessibilityAnalytics/Private/OpenAccessibilityAnalytics.cpp

## 4.10 FOpenAccessibilityCommunicationModule Class Reference

Inheritance diagram for FOpenAccessibilityCommunicationModule:



## Public Member Functions

- virtual void [StartupModule](#) () override
- virtual void [ShutdownModule](#) () override
- virtual bool [SupportsDynamicReloading](#) () override
- bool [Tick](#) (const float DeltaTime)
- void [HandleKeyDownEvent](#) (const FKeyEvent &InKeyEvent)
- void [TranscribeWaveForm](#) (TArray< float > AudioBufferToTranscribe)

*Sends the Audio Buffer to the Transcription Service.*

## Static Public Member Functions

- static [FOpenAccessibilityCommunicationModule](#) & [Get](#) ()

## Public Attributes

- TMulticastDelegate< void(TArray< FString >)> [OnTranscriptionRecieved](#)  
*A Delegate for when Transcriptions are recieved back from the Transcription Service.*
- class [UAudioManager](#) \* [AudioManager](#)  
*The AudioManager, Managing any Audio Capture Component.*
- TSharedPtr< class [FSocketCommunicationServer](#) > [SocketServer](#)  
*The Socket Communication Server, Managing Socket Communication for the Transcription Service.*
- TSharedPtr< [FPhraseTree](#) > [PhraseTree](#)  
*The PhraseTree, Containing any Bound Phrase Nodes and Commands to Execute from Transcriptions.*
- class [UPhraseTreeUtils](#) \* [PhraseTreeUtils](#)  
*Phrase Tree Utility Class, For Dealing With Phrase Tree Function Libraries.*

### 4.10.1 Detailed Description

Definition at line 16 of file [OpenAccessibilityCommunication.h](#).

### 4.10.2 Member Function Documentation

#### 4.10.2.1 Get()

```
static FOpenAccessibilityCommunicationModule & FOpenAccessibilityCommunicationModule::Get ( )
[inline], [static]
```

End IModuleInterface Implementation

Definition at line 31 of file [OpenAccessibilityCommunication.h](#).

```
00032     {
00033         return
00034         FModuleManager::GetModuleChecked<FOpenAccessibilityCommunicationModule>("OpenAccessibilityCommunication");
00034     }
```

#### 4.10.2.2 HandleKeyDownEvent()

```
void FOpenAccessibilityCommunicationModule::HandleKeyDownEvent (
    const FKeyEvent & InKeyEvent )
```

Definition at line 92 of file [OpenAccessibilityCommunication.cpp](#).

```
00093 {
00094     // If the Space Key is pressed, we will send a request to the Accessibility Server
00095     if (InKeyEvent.GetKey() == EKeys::SpaceBar)
00096     {
00097         if (InKeyEvent.IsShiftDown())
00098         {
00099             OA_LOG(LogOpenAccessibilityCom, Log, TEXT("AudioCapture Change"), TEXT("Stopping Audio
Capture"));
00100             AudioManager->StopCapturingAudio();
00101         }
00102         else
00103         {
00104             OA_LOG(LogOpenAccessibilityCom, Log, TEXT("AudioCapture Change"), TEXT("Starting Audio
Capture"));
00105             AudioManager->StartCapturingAudio();
00106         }
00107     }
00108 }
```

#### 4.10.2.3 ShutdownModule()

```
void FOpenAccessibilityCommunicationModule::ShutdownModule ( ) [override], [virtual]
```

Definition at line 55 of file [OpenAccessibilityCommunication.cpp](#).

```
00056 {
00057     // This function may be called during shutdown to clean up your module. For modules that support
dynamic reloading,
00058     // we call this function before unloading the module.
00059     UE_LOG(LogOpenAccessibilityCom, Display, TEXT("OpenAccessibilityComModule::ShutdownModule()"));
00060
00061     AudioManager->RemoveFromRoot();
00062     PhraseTreeUtils->RemoveFromRoot();
00063
00064     F SlateApplication::Get().OnApplicationPreInputKeyDownListener().Remove(KeyDownEventHandle);
00065
00066     UnloadZMQDLL();
00067
00068     UnregisterConsoleCommands();
00069 }
```

#### 4.10.2.4 StartupModule()

```
void FOpenAccessibilityCommunicationModule::StartupModule ( ) [override], [virtual]
```

#### IModuleInterface Implementation

Definition at line 24 of file [OpenAccessibilityCommunication.cpp](#).

```
00025 {
00026     LoadZMQDLL();
00027
00028     // This code will execute after your module is loaded into memory; the exact timing is specified
in the .uplugin file per-module
00029     UE_LOG(LogOpenAccessibilityCom, Display, TEXT("OpenAccessibilityComModule::StartupModule()"));
00030
00031     // Initialize AudioManager
00032     AudioManager = NewObject<UAudioManager>();
00033     AudioManager->AddToRoot();
00034
00035     AudioManager->OnAudioReadyForTranscription
00036         .BindRaw(this, &FOpenAccessibilityCommunicationModule::TranscribeWaveForm);
```

```

00037
00038     // Initialize Socket Server
00039     SocketServer = MakeShared<FSocketCommunicationServer>();
00040
00041     // Build The Phrase Tree
00042     BuildPhraseTree();
00043
00044     // Bind Tick Event
00045     TickDelegate = FTickerDelegate::CreateRaw(this, &FOpenAccessibilityCommunicationModule::Tick);
00046     TickDelegateHandle = FTSTicker::GetCoreTicker().AddTicker(TickDelegate);
00047
00048     // Bind Input Events
00049     KeyDownEventHandle = FSlateApplication::Get().OnApplicationPreInputKeyDownListener().AddRaw(this,
&FOpenAccessibilityCommunicationModule::HandleKeyDownEvent);
00050
00051     // Register Console Commands
00052     RegisterConsoleCommands();
00053 }

```

#### 4.10.2.5 SupportsDynamicReloading()

```

virtual bool FOpenAccessibilityCommunicationModule::SupportsDynamicReloading ( ) [inline],
[override], [virtual]

```

Definition at line 25 of file [OpenAccessibilityCommunication.h](#).

```

00026     {
00027         return false;
00028     }

```

#### 4.10.2.6 Tick()

```

bool FOpenAccessibilityCommunicationModule::Tick (
    const float DeltaTime )

```

Definition at line 71 of file [OpenAccessibilityCommunication.cpp](#).

```

00072 {
00073     // Detect if any events are ready to be received.
00074     if (SocketServer->EventOccured())
00075     {
00076         TArray<FString> RecvStrings;
00077         TSharedPtr<FJsonObject> RecvMetadata;
00078
00079         // Receive the Detected Event, with separate transcriptions and metadata.
00080         if (SocketServer->RecvStringMultipartWithMeta(RecvStrings, RecvMetadata))
00081         {
00082             OA_LOG(LogOpenAccessibilityCom, Log, TEXT("TRANSCRIPTION RECIEVED"), TEXT("Recieved
Multipart - Message Count: %d"), RecvStrings.Num());
00083
00084             // Send Received Transcriptions to any bound events.
00085             OnTranscriptionRecieved.Broadcast(RecvStrings);
00086         }
00087     }
00088
00089     return true;
00090 }

```

#### 4.10.2.7 TranscribeWaveForm()

```

void FOpenAccessibilityCommunicationModule::TranscribeWaveForm (
    TArray< float > AudioBufferToTranscribe )

```

Sends the Audio Buffer to the Transcription Service.

## Parameters

|                                |  |
|--------------------------------|--|
| <i>AudioBufferToTranscribe</i> | - The Audiobuffer To Send For Transcription. |
|--------------------------------|--|

Definition at line 110 of file [OpenAccessibilityCommunication.cpp](#).

```

00111 {
00112     if (AudioBufferToTranscribe.Num() == 0)
00113     {
00114         UE_LOG(LogOpenAccessibilityCom, Warning, TEXT("|| Transcription Ready || Audio Buffer is Empty
||"));
00115         return;
00116     }
00117     PrevAudioBuffer = TArray(AudioBufferToTranscribe);
00118     UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| WaveForm Transcription || Array Size: %d || Byte
Size: %s ||"), AudioBufferToTranscribe.Num(), *FString::FromInt(AudioBufferToTranscribe.Num() *
sizeof(float)));
00121     // Create Metadata of Audio Source.
00122     TSharedPtr<FJsonObject> AudioBufferMetadata = MakeShared<FJsonObject>();
00123     AudioBufferMetadata->SetNumberField(TEXT("sample_rate"),
AudioManager->GetAudioCaptureSampleRate());
00125     AudioBufferMetadata->SetNumberField(TEXT("num_channels"),
AudioManager->GetAudioCaptureNumChannels());
00126     bool bArrayMessageSent = SocketServer->SendArrayMessageWithMeta(AudioBufferToTranscribe,
AudioBufferMetadata.ToSharedRef(), ComSendFlags::none);
00128     OA_LOG(LogOpenAccessibilityCom, Log, TEXT("TRANSCRIPTION SENT"), TEXT("{%s} Send Audiobuffer
(float x %d / %d Hz / %d channels)"),
00130     bArrayMessageSent ? TEXT("Success") : TEXT("Failed"),
00131     AudioBufferToTranscribe.Num(), AudioManager->GetAudioCaptureSampleRate(),
AudioManager->GetAudioCaptureNumChannels());
00132 }

```

## 4.10.3 Member Data Documentation

### 4.10.3.1 AudioManager

```
class UAudioManager* FOpenAccessibilityCommunicationModule::AudioManager
```

The AudioManager, Managing any Audio Capture Component.

Definition at line 82 of file [OpenAccessibilityCommunication.h](#).

### 4.10.3.2 OnTranscriptionRecieved

```
TMulticastDelegate<void(TArray<FString>>> FOpenAccessibilityCommunicationModule::OnTranscription↵
Recieved
```

A Delegate for when Transcriptions are recived back from the Transcription Service.

Definition at line 77 of file [OpenAccessibilityCommunication.h](#).

#### 4.10.3.3 PhraseTree

```
TSharedPtr<FPhraseTree> FOpenAccessibilityCommunicationModule::PhraseTree
```

The PhraseTree, Containing any Bound Phrase Nodes and Commands to Execute from Transcriptions.

Definition at line 92 of file [OpenAccessibilityCommunication.h](#).

#### 4.10.3.4 PhraseTreeUtils

```
class UPhraseTreeUtils* FOpenAccessibilityCommunicationModule::PhraseTreeUtils
```

Phrase Tree Utility Class, For Dealing With Phrase Tree Function Libraries.

Definition at line 97 of file [OpenAccessibilityCommunication.h](#).

#### 4.10.3.5 SocketServer

```
TSharedPtr<class FSocketCommunicationServer> FOpenAccessibilityCommunicationModule::SocketServer
```

The Socket Communication Server, Managing Socket Communication for the Transcription Service.

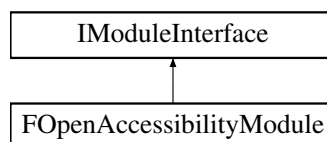
Definition at line 87 of file [OpenAccessibilityCommunication.h](#).

The documentation for this class was generated from the following files:

- Source/OpenAccessibilityCommunication/Public/OpenAccessibilityCommunication.h
- Source/OpenAccessibilityCommunication/Private/OpenAccessibilityCommunication.cpp

## 4.11 FOpenAccessibilityModule Class Reference

Inheritance diagram for FOpenAccessibilityModule:



### Public Member Functions

- virtual void [StartupModule](#) () override
- virtual void [ShutdownModule](#) () override
- virtual bool [SupportsDynamicReloading](#) () override

## Static Public Member Functions

- static [FOpenAccessibilityModule](#) & [Get](#) ()

## Public Attributes

- TSharedPtr< class [FAccessibilityNodeFactory](#) > [AccessibilityNodeFactory](#)  
*The Node Factory for Generating Accessibility Graph Nodes.*
- TSharedPtr< class [FAssetAccessibilityRegistry](#) > [AssetAccessibilityRegistry](#)  
*The Registry for Any Asset Accessibility Information.*

### 4.11.1 Detailed Description

Definition at line 11 of file [OpenAccessibility.h](#).

### 4.11.2 Member Function Documentation

#### 4.11.2.1 Get()

```
static FOpenAccessibilityModule & FOpenAccessibilityModule::Get ( ) [inline], [static]
```

End IModuleInterface Implementation

Definition at line 21 of file [OpenAccessibility.h](#).

```
00022 {
00023     return FModuleManager::GetModuleChecked<FOpenAccessibilityModule>("OpenAccessibility");
00024 }
```

#### 4.11.2.2 ShutdownModule()

```
void FOpenAccessibilityModule::ShutdownModule ( ) [override], [virtual]
```

Definition at line 75 of file [OpenAccessibility.cpp](#).

```
00076 {
00077     UE_LOG(LogOpenAccessibility, Display, TEXT("OpenAccessibilityModule::ShutdownModule()"));
00078
00079     UnregisterConsoleCommands();
00080 }
```



### 4.11.2.3 StartupModule()

```
void FOpenAccessibilityModule::StartupModule ( ) [override], [virtual]
```

#### IModuleInterface Implementation

Definition at line 37 of file [OpenAccessibility.cpp](#).

```
00038 {
00039     UE_LOG(LogOpenAccessibility, Display, TEXT("OpenAccessibilityModule::StartupModule()"));
00040
00041     // Create the Asset Registry
00042     AssetAccessibilityRegistry = MakeShared<FAssetAccessibilityRegistry, ESPMode::ThreadSafe>();
00043
00044     // Register the Accessibility Node Factory
00045     AccessibilityNodeFactory = MakeShared<FAccessibilityNodeFactory, ESPMode::ThreadSafe>();
00046     FEdGraphUtilities::RegisterVisualNodeFactory(AccessibilityNodeFactory);
00047
00048     // Construct Base Phrase Tree Libraries
00049     FOpenAccessibilityCommunicationModule::Get()
00050     .PhraseTreeUtils->RegisterFunctionLibrary(
00051         NewObject<ULocalizedInputLibrary>()
00052     );
00053
00054     FOpenAccessibilityCommunicationModule::Get()
00055     .PhraseTreeUtils->RegisterFunctionLibrary(
00056         NewObject<UWindowInteractionLibrary>()
00057     );
00058
00059     FOpenAccessibilityCommunicationModule::Get()
00060     .PhraseTreeUtils->RegisterFunctionLibrary(
00061         NewObject<UViewInteractionLibrary>()
00062     );
00063
00064     FOpenAccessibilityCommunicationModule::Get()
00065     .PhraseTreeUtils->RegisterFunctionLibrary(
00066         NewObject<UNodeInteractionLibrary>()
00067     );
00068
00069     CreateTranscriptionVisualization();
00070
00071     // Register Console Commands
00072     RegisterConsoleCommands();
00073 }
```

### 4.11.2.4 SupportsDynamicReloading()

```
virtual bool FOpenAccessibilityModule::SupportsDynamicReloading ( ) [inline], [override],
[virtual]
```

Definition at line 26 of file [OpenAccessibility.h](#).

```
00027 {
00028     return false;
00029 }
```

## 4.11.3 Member Data Documentation

### 4.11.3.1 AccessibilityNodeFactory

TSharedPtr<class [FAccessibilityNodeFactory](#)> FOpenAccessibilityModule::AccessibilityNodeFactory

The Node Factory for Generating Accessibility Graph Nodes.

Definition at line 81 of file [OpenAccessibility.h](#).

### 4.11.3.2 AssetAccessibilityRegistry

TSharedPtr<class [FAssetAccessibilityRegistry](#)> FOpenAccessibilityModule::AssetAccessibilityRegistry

The Registry for Any Asset Accessibility Information.

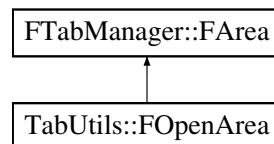
Definition at line 86 of file [OpenAccessibility.h](#).

The documentation for this class was generated from the following files:

- Source/OpenAccessibility/Public/OpenAccessibility.h
- Source/OpenAccessibility/Private/OpenAccessibility.cpp

## 4.12 TabUtils::FOpenArea Class Reference

Inheritance diagram for TabUtils::FOpenArea:



### Public Member Functions

- const TArray< TSharedRef< FLayoutNode > > & [GetChildNodes](#) ()

### 4.12.1 Detailed Description

Definition at line 170 of file [WindowInteractionLibrary.cpp](#).

### 4.12.2 Member Function Documentation

#### 4.12.2.1 GetChildNodes()

```
const TArray< TSharedRef< FLayoutNode > > & TabUtils::FOpenArea::GetChildNodes ( ) [inline]
```

Definition at line 173 of file [WindowInteractionLibrary.cpp](#).

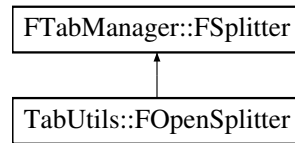
```
00174     {
00175         return ChildNodes;
00176     }
```

The documentation for this class was generated from the following file:

- Source/OpenAccessibility/Private/PhraseEvents/WindowInteractionLibrary.cpp

## 4.13 TabUtils::FOpenSplitter Class Reference

Inheritance diagram for TabUtils::FOpenSplitter:



### Public Member Functions

- `const TArray< TSharedRef< FLayoutNode > > & GetChildNodes ()`

#### 4.13.1 Detailed Description

Definition at line 188 of file [WindowInteractionLibrary.cpp](#).

#### 4.13.2 Member Function Documentation

##### 4.13.2.1 GetChildNodes()

```
const TArray< TSharedRef< FLayoutNode > > & TabUtils::FOpenSplitter::GetChildNodes ( ) [inline]
```

Definition at line 191 of file [WindowInteractionLibrary.cpp](#).

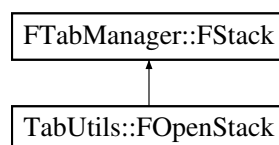
```
00192     {
00193         return ChildNodes;
00194     }
```

The documentation for this class was generated from the following file:

- `Source/OpenAccessibility/Private/PhraseEvents/WindowInteractionLibrary.cpp`

## 4.14 TabUtils::FOpenStack Class Reference

Inheritance diagram for TabUtils::FOpenStack:



## Public Member Functions

- `const TArray< FTabManager::FTab > & GetTabs ()`

### 4.14.1 Detailed Description

Definition at line 179 of file [WindowInteractionLibrary.cpp](#).

### 4.14.2 Member Function Documentation

#### 4.14.2.1 GetTabs()

```
const TArray< FTabManager::FTab > & TabUtils::FOpenStack::GetTabs ( ) [inline]
```

Definition at line 182 of file [WindowInteractionLibrary.cpp](#).

```
00183     {  
00184         return Tabs;  
00185     }
```

The documentation for this class was generated from the following file:

- [Source/OpenAccessibility/Private/PhraseEvents/WindowInteractionLibrary.cpp](#)

## 4.15 FPanelViewPosition Struct Reference

### Public Member Functions

- [FPanelViewPosition](#) (FVector2D InTopLeft, FVector2D InBotRight)
- `bool operator!= (const FVector2D &Other)`
- `bool operator!= (const FPanelViewPosition &Other)`

### Public Attributes

- FVector2D [TopLeft](#)
- FVector2D [BotRight](#)

### 4.15.1 Detailed Description

Definition at line 70 of file [AccessibilityGraphLocomotionContext.h](#).

### 4.15.2 Constructor & Destructor Documentation

#### 4.15.2.1 FPanelViewPosition() [1/2]

```
FPanelViewPosition::FPanelViewPosition ( ) [inline]
```

Definition at line 74 of file [AccessibilityGraphLocomotionContext.h](#).

```
00075         : TopLeft(FVector2D::ZeroVector)
00076         , BotRight(FVector2D::ZeroVector)
00077     { }
```

#### 4.15.2.2 FPanelViewPosition() [2/2]

```
FPanelViewPosition::FPanelViewPosition (
    FVector2D InTopLeft,
    FVector2D InBotRight ) [inline]
```

Definition at line 79 of file [AccessibilityGraphLocomotionContext.h](#).

```
00080         : TopLeft(InTopLeft)
00081         , BotRight(InBotRight)
00082     { }
```

### 4.15.3 Member Function Documentation

#### 4.15.3.1 operator!=(()) [1/2]

```
bool FPanelViewPosition::operator!= (
    const FPanelViewPosition & Other ) [inline]
```

Definition at line 89 of file [AccessibilityGraphLocomotionContext.h](#).

```
00090     {
00091         return TopLeft != Other.TopLeft || BotRight != Other.BotRight;
00092     }
```

#### 4.15.3.2 operator!=(()) [2/2]

```
bool FPanelViewPosition::operator!= (
    const FVector2D & Other ) [inline]
```

Definition at line 84 of file [AccessibilityGraphLocomotionContext.h](#).

```
00085     {
00086         return TopLeft != Other || BotRight != Other;
00087     }
```

### 4.15.4 Member Data Documentation

#### 4.15.4.1 BotRight

FVector2D FPanelViewPosition::BotRight

Definition at line 95 of file [AccessibilityGraphLocomotionContext.h](#).

#### 4.15.4.2 TopLeft

FVector2D FPanelViewPosition::TopLeft

Definition at line 94 of file [AccessibilityGraphLocomotionContext.h](#).

The documentation for this struct was generated from the following file:

- Source/OpenAccessibility/Public/AccessibilityWrappers/AccessibilityGraphLocomotionContext.h

## 4.16 FParseRecord Struct Reference

The Collected Information from the Propagation of the Phrase through the tree.

```
#include <ParseRecord.h>
```

### Public Member Functions

- [FParseRecord](#) (TArray< [UPhraseTreeContextObject](#) \* > InContextObjects)
- FString [GetPhraseString](#) () const  
*Gets the Recorded Phrase String for this record of propagation.*
- void [AddPhraseString](#) (FString StringToRecord)
- [UParseInput](#) \* [GetPhraseInput](#) (const FString &InString)  
*Gets the Input for the Provided Phrase, if it exists.*
- template<class CastToType >  
CastToType \* [GetPhaselInput](#) (const FString &InString)  
*Gets the Input for the Provided Phrase, if it exists.*
- void [GetPhaselInput](#) (const FString &InString, [UParseInput](#) \*OutInput)  
*Gets the Input for the Provided Phrase, if it exists.*
- template<class CastToType >  
void [GetPhaselInput](#) (const FString &InString, CastToType \*OutInput)  
*Gets the Input for the Provided Phrase, if it exists.*
- void [GetPhaselInputs](#) (const FString &InString, TArray< [UParseInput](#) \* > &OutInputs, const bool MaintainOrder=true)  
*Gets an Array of Phrase Inputs for the Provided Phrase.*
- TArray< [UParseInput](#) \* > [GetPhaselInputs](#) (const FString &InString, const bool MaintainOrder=true)  
*Gets an Array of Phrase Inputs for the Provided Phrase.*
- void [AddPhaselInput](#) (const FString &InString, [UParseInput](#) \*InInput)  
*Adds a Phrase Input to the Record.*
- void [RemovePhaselInput](#) (const FString &InString)  
*Removes a Phrase Input From The Record.*

- void [PushContextObj](#) ([UPhraseTreeContextObject](#) \*InObject)  
*Pushes a Context Object onto the Stack.*
- void [PopContextObj](#) ()  
*Pops the Top Context Object From The Stack.*
- void [PopContextObj](#) ([UPhraseTreeContextObject](#) \*OutObject)  
*Pops the Top Context Object From The Stack.*
- void [RemoveContextObj](#) ([UPhraseTreeContextObject](#) \*InObject)  
*Removes a Select Context Object From The Stack.*
- bool [HasContextObj](#) ()  
*Checks if there is a Context Object on the Stack.*
- bool [HasContextObj](#) ([UPhraseTreeContextObject](#) \*InObject)  
*Checks if a specific Context Object is on the Stack.*
- [UPhraseTreeContextObject](#) \* [GetContextObj](#) ()  
*Gets the Top Context Object On The Stack.*
- void [GetContextObj](#) ([UPhraseTreeContextObject](#) \*OutObject)  
*Gets the Top Context Object On The Stack.*
- template<class CastToType >  
CastToType \* [GetContextObj](#) ()  
*Gets the Top Context Object On The Stack.*
- template<class CastToType >  
void [GetContextObj](#) (CastToType \*OutObject)  
*Gets the Top Context Object On The Stack.*
- void [GetContextStack](#) (TArray< [UPhraseTreeContextObject](#) \* > OutContextStack)  
*Gets the Entire Context Stack.*
- TArray< [UPhraseTreeContextObject](#) \* > [GetContextStack](#) ()  
*Gets the Entire Context Stack.*

## Protected Attributes

- TArray< [UPhraseTreeContextObject](#) \* > [ContextObjectStack](#) = TArray<[UPhraseTreeContextObject](#)\*>()  
*The Context Stack of Context Objects.*
- TArray< FString > [PhraseRecord](#)  
*A Record of the Phrase String used through-out propagation.*
- TMultiMap< FString, [UParseInput](#) \* > [PhraseInputs](#)  
*Map of all the Provided Phrase Inputs, to their Respective Phrases.*

## Friends

- class [FPhraseTree](#)

### 4.16.1 Detailed Description

The Collected Information from the Propagation of the Phrase through the tree.

Definition at line 16 of file [ParseRecord.h](#).

### 4.16.2 Constructor & Destructor Documentation

#### 4.16.2.1 FParseRecord() [1/2]

FParseRecord::FParseRecord ( ) [inline]

Definition at line 23 of file [ParseRecord.h](#).

```
00024     {
00025         PhraseInputs = TMultiMap<FString, UParseInput*>();
00026         ContextObjectStack = TArray<UPhraseTreeContextObject*>();
00027     }
```

#### 4.16.2.2 FParseRecord() [2/2]

FParseRecord::FParseRecord (   
                   TArray< UPhraseTreeContextObject \* > InContextObjects ) [inline]

Definition at line 29 of file [ParseRecord.h](#).

```
00030     {
00031         PhraseInputs = TMultiMap<FString, UParseInput*>();
00032         ContextObjectStack = InContextObjects;
00033     }
```

#### 4.16.2.3 ~FParseRecord()

FParseRecord::~FParseRecord ( ) [inline]

Definition at line 35 of file [ParseRecord.h](#).

```
00036     {
00037         PhraseInputs.Empty();
00038     }
```

### 4.16.3 Member Function Documentation

#### 4.16.3.1 AddPhraseInput()

```
void FParseRecord::AddPhraseInput (   

                      const FString & InString,   

                      UParseInput * InInput ) [inline]
```

Adds a Phrase Input to the Record.

##### Parameters

|                 |  |
|-----------------|--|
| <i>InString</i> | - The Phrase to Bind the Input To.               |
| <i>InInput</i>  | - The Phrase Input Object Containing Input Data. |

Definition at line 162 of file [ParseRecord.h](#).



```

00163     {
00164         PhraseInputs.Add(InString.ToUpper(), InInput);
00165     }

```

#### 4.16.3.2 AddPhraseString()

```

void FParseRecord::AddPhraseString (
    FString StringToRecord ) [inline]

```

Definition at line 51 of file [ParseRecord.h](#).

```

00052     {
00053         PhraseRecord.Add(StringToRecord);
00054     }

```

#### 4.16.3.3 GetContextObj() [1/4]

```
UPhraseTreeContextObject * FParseRecord::GetContextObj ( ) [inline]
```

Gets the Top Context Object On The Stack.

##### Returns

The Top Context Object On The Stack.

Definition at line 249 of file [ParseRecord.h](#).

```

00250     {
00251         if (ContextObjectStack.IsEmpty())
00252             return nullptr;
00253
00254         return this->ContextObjectStack.Last();
00255     }

```

#### 4.16.3.4 GetContextObj() [2/4]

```

template<class CastToType >
CastToType * FParseRecord::GetContextObj ( ) [inline]

```

Gets the Top Context Object On The Stack.

##### Template Parameters

|                   |   |
|-------------------|---|
| <i>CastToType</i> | DownCast Type For the Context Object (Must Derrive From <a href="#">UPhraseTreeContextObject</a> ). |
|-------------------|---|

##### Returns

The DownCasted Context Object, otherwise nullptr.

Definition at line 278 of file [ParseRecord.h](#).

```

00279     {
00280         if (ContextObjectStack.IsEmpty())
00281             return nullptr;
00282
00283         return Cast<CastToType>(this->ContextObjectStack.Last());
00284     }

```

#### 4.16.3.5 GetContextObj() [3/4]

```

template<class CastToType >
void FParseRecord::GetContextObj (
    CastToType * OutObject ) [inline]

```

Gets the Top Context Object On The Stack.

##### Template Parameters

|                   |   |
|-------------------|---|
| <i>CastToType</i> | DownCast Type For the Context Object (Must Derrive From <a href="#">UPhraseTreeContextObject</a> ). |
|-------------------|---|

##### Parameters

|                  |   |
|------------------|---|
| <i>OutObject</i> | - Returns the Downcasted Context Object, otherwise nullptr. |
|------------------|---|

Definition at line 292 of file [ParseRecord.h](#).

```

00293     {
00294         if (ContextObjectStack.IsEmpty())
00295         {
00296             OutObject = nullptr;
00297             return;
00298         }
00299
00300         OutObject = Cast<CastToType>(this->ContextObjectStack.Last());
00301     }

```

#### 4.16.3.6 GetContextObj() [4/4]

```

void FParseRecord::GetContextObj (
    UPhraseTreeContextObject * OutObject ) [inline]

```

Gets the Top Context Object On The Stack.

##### Parameters

|                  |  |
|------------------|--|
| <i>OutObject</i> | - Returns the Top Context Object On The Stack. |
|------------------|--|

Definition at line 261 of file [ParseRecord.h](#).

```

00262     {
00263         if (ContextObjectStack.IsEmpty())
00264         {
00265             OutObject = nullptr;
00266             return;
00267         }
00268

```

```

00269         OutObject = this->ContextObjectStack.Last();
00270     }

```

#### 4.16.3.7 GetContextStack() [1/2]

```
TArray< UPhraseTreeContextObject * > FParseRecord::GetContextStack ( ) [inline]
```

Gets the Entire Context Stack.

##### Returns

The Current Context Stack.

Definition at line 316 of file [ParseRecord.h](#).

```

00317     {
00318         return ContextObjectStack;
00319     }

```

#### 4.16.3.8 GetContextStack() [2/2]

```

void FParseRecord::GetContextStack (
    TArray< UPhraseTreeContextObject * > OutContextStack ) [inline]

```

Gets the Entire Context Stack.

##### Parameters

|                        |                                      |
|------------------------|--------------------------------------|
| <i>OutContextStack</i> | - Returns the Current Context Stack. |
|------------------------|--------------------------------------|

Definition at line 307 of file [ParseRecord.h](#).

```

00308     {
00309         OutContextStack = ContextObjectStack;
00310     }

```

#### 4.16.3.9 GetPhraseInput() [1/4]

```

UParseInput * FParseRecord::GetPhraseInput (
    const FString & InString ) [inline]

```

Gets the Input for the Provided Phrase, if it exists.

##### Parameters

|                 |                                     |
|-----------------|-------------------------------------|
| <i>InString</i> | - The Phrase To Check For An Input. |
|-----------------|-------------------------------------|

**Returns**

The Found PhraseInput For the Phrase, otherwise nullptr.

Definition at line 64 of file [ParseRecord.h](#).

```
00065     {
00066         // Check If The Phrase Exits
00067         // This Error Will Be Thrown If: InString Is In Correct (Requires UpperCase) or The Phrase
        Does Not Exist.
00068         check(PhraseInputs.Contains(InString))
00069
00070         return *PhraseInputs.Find(InString);
00071     }
```

**4.16.3.10 GetPhraseInput() [2/4]**

```
template<class CastToType >
CastToType * FParseRecord::GetPhraseInput (
    const FString & InString ) [inline]
```

Gets the Input for the Provided Phrase, if it exists.

**Template Parameters**

|                   |   |
|-------------------|---|
| <i>CastToType</i> | DownCast Type For the Phrase Input (Must Derive From UPhraseInput). |
|-------------------|---|

**Parameters**

|                 |                                     |
|-----------------|-------------------------------------|
| <i>InString</i> | - The Phrase To Check For An Input. |
|-----------------|-------------------------------------|

**Returns**

The Found DownCasted PhraseInput, otherwise nullptr.

Definition at line 80 of file [ParseRecord.h](#).

```
00081     {
00082         // Check If The Phrase Exits
00083         // This Error Will Be Thrown If: InString Is In Correct (Requires UpperCase) or The Phrase
        Does Not Exist.
00084         check(PhraseInputs.Contains(InString))
00085
00086         return Cast<CastToType>(*PhraseInputs.Find(InString));
00087     }
```

**4.16.3.11 GetPhraseInput() [3/4]**

```
template<class CastToType >
void FParseRecord::GetPhraseInput (
    const FString & InString,
    CastToType * OutInput ) [inline]
```

Gets the Input for the Provided Phrase, if it exists.

## Template Parameters

|                   |   |
|-------------------|---|
| <i>CastToType</i> | DownCast Type For the Phrase Input (Must Derive From UPhraseInput). |
|-------------------|---|

## Parameters

|                 |  |
|-----------------|--|
| <i>InString</i> | - The Phrase To Check For An Input.              |
| <i>OutInput</i> | - Returns the Found DownCasted Input or nullptr. |

Definition at line 110 of file [ParseRecord.h](#).

```

00111     {
00112         // Check If The Phrase Exits
00113         // This Error Will Be Thrown If: InString Is In Correct (Requires UpperCase) or The Phrase
        Does Not Exist.
00114         check(PhraseInputs.Contains(InString))
00115
00116         OutInput = Cast<CastToType>(*PhraseInputs.Find(InString));
00117     }
```

## 4.16.3.12 GetPhraseInput() [4/4]

```

void FParseRecord::GetPhraseInput (
    const FString & InString,
    UParseInput * OutInput ) [inline]
```

Gets the Input for the Provided Phrase, if it exists.

## Parameters

|                 |                                       |
|-----------------|---------------------------------------|
| <i>InString</i> | - The Phrase To Check For An Input.   |
| <i>OutInput</i> | - Returns the Found Input or nullptr. |

Definition at line 94 of file [ParseRecord.h](#).

```

00095     {
00096         // Check If The Phrase Exits
00097         // This Error Will Be Thrown If: InString Is In Correct (Requires UpperCase) or The Phrase
        Does Not Exist.
00098         check(PhraseInputs.Contains(InString))
00099
00100         OutInput = *PhraseInputs.Find(InString);
00101     }
```

## 4.16.3.13 GetPhraseInputs() [1/2]

```

TArray< UParseInput * > FParseRecord::GetPhraseInputs (
    const FString & InString,
    const bool MaintainOrder = true ) [inline]
```

Gets an Array of Phrase Inputs for the Provided Phrase.

**Parameters**

|                      |   |
|----------------------|---|
| <i>InString</i>      | - The Phrase To Check For A Multi-Input.                                  |
| <i>MaintainOrder</i> | - Should the Returned Array Maintain the Order the Inputs where Inserted. |

**Returns**

The Array of Found Inputs.

Definition at line 142 of file [ParseRecord.h](#).

```

00143     {
00144         // Check If The Phrase Exits
00145         // This Error Will Be Thrown If: InString Is In Correct (Requires UpperCase) or The Phrase
        Does Not Exist.
00146         check(PhraseInputs.Contains(InString))
00147
00148         TArray<UParseInput*> OutInputs;
00149
00150         PhraseInputs.MultiFind(InString, OutInputs, MaintainOrder);
00151
00152         return OutInputs;
00153     }

```

**4.16.3.14 GetPhraseInputs() [2/2]**

```

void FParseRecord::GetPhraseInputs (
    const FString & InString,
    TArray< UParseInput * > & OutInputs,
    const bool MaintainOrder = true ) [inline]

```

Gets an Array of Phrase Inputs for the Provided Phrase.

**Parameters**

|                      |   |
|----------------------|---|
| <i>InString</i>      | - The Phrase To Check For A Multi-Input.                                  |
| <i>OutInputs</i>     | - Returns An Array of Inputs.   |
| <i>MaintainOrder</i> | - Should the Returned Array Maintain the Order the Inputs where Inserted. |

Definition at line 127 of file [ParseRecord.h](#).

```

00128     {
00129         // Check If The Phrase Exits
00130         // This Error Will Be Thrown If: InString Is In Correct (Requires UpperCase) or The Phrase
        Does Not Exist.
00131         check(PhraseInputs.Contains(InString))
00132
00133         PhraseInputs.MultiFind(InString, OutInputs, MaintainOrder);
00134     }

```

**4.16.3.15 GetPhraseString()**

```

FString FParseRecord::GetPhraseString ( ) const [inline]

```

Gets the Recorded Phrase String for this record of propagation.

## Returns

Definition at line 46 of file [ParseRecord.h](#).

```

00047     {
00048         return FString::Join(PhraseRecord, TEXT(" "));
00049     }

```

### 4.16.3.16 HasContextObj() [1/2]

```
bool FParseRecord::HasContextObj ( ) [inline]
```

Checks if there is a Context Object on the Stack.

## Returns

Definition at line 228 of file [ParseRecord.h](#).

```

00229     {
00230         return this->ContextObjectStack.Num() > 0;
00231     }

```

### 4.16.3.17 HasContextObj() [2/2]

```
bool FParseRecord::HasContextObj (
    UPhraseTreeContextObject * InObject ) [inline]
```

Checks if a specific Context Object is on the Stack.

## Parameters

|                 |  |
|-----------------|--|
| <i>InObject</i> | - The Context Object To Check if On The Stack. |
|-----------------|--|

## Returns

True, if the Object is on the Stack. False, if the Object is not on the stack.

Definition at line 238 of file [ParseRecord.h](#).

```

00239     {
00240         return HasContextObj() && this->ContextObjectStack.Contains(InObject);
00241     }

```

### 4.16.3.18 PopContextObj() [1/2]

```
void FParseRecord::PopContextObj ( ) [inline]
```

Pops the Top Context Object From The Stack.

Definition at line 190 of file [ParseRecord.h](#).

```
00191     {
00192         if (ContextObjectStack.IsEmpty())
00193             return;
00194
00195         this->ContextObjectStack.Pop();
00196     }
```

#### 4.16.3.19 PopContextObj() [2/2]

```
void FParseRecord::PopContextObj (
    UPhraseTreeContextObject * OutObject ) [inline]
```

Pops the Top Context Object From The Stack.

##### Parameters

|                  |                              |
|------------------|------------------------------|
| <i>OutObject</i> | - The Popped Context Object. |
|------------------|------------------------------|

Definition at line 202 of file [ParseRecord.h](#).

```
00203     {
00204         if (ContextObjectStack.IsEmpty())
00205         {
00206             OutObject = nullptr;
00207             return;
00208         }
00209
00210         OutObject = this->ContextObjectStack.Pop();
00211     }
```

#### 4.16.3.20 PushContextObj()

```
void FParseRecord::PushContextObj (
    UPhraseTreeContextObject * InObject ) [inline]
```

Pushes a Context Object onto the Stack.

##### Parameters

|                 |  |
|-----------------|--|
| <i>InObject</i> | - The Context Object To Push onto The Stack. |
|-----------------|--|

Definition at line 182 of file [ParseRecord.h](#).

```
00183     {
00184         this->ContextObjectStack.Push(InObject);
00185     }
```

#### 4.16.3.21 RemoveContextObj()

```
void FParseRecord::RemoveContextObj (
    UPhraseTreeContextObject * InObject ) [inline]
```



Removes a Select Context Object From The Stack.

#### Parameters

|                 |  |
|-----------------|--|
| <i>InObject</i> |  |
|-----------------|--|

Definition at line 217 of file [ParseRecord.h](#).

```
00218     {
00219         this->ContextObjectStack.Remove(InObject);
00220     }
```

#### 4.16.3.22 RemovePhraseInput()

```
void FParseRecord::RemovePhraseInput (
    const FString & InString ) [inline]
```

Removes a Phrase Input From The Record.

#### Parameters

|                 |   |
|-----------------|---|
| <i>InString</i> | - The Phrase To Remove All Bound Inputs from. |
|-----------------|---|

Definition at line 171 of file [ParseRecord.h](#).

```
00172     {
00173         PhraseInputs.Remove(InString);
00174     }
```

### 4.16.4 Friends And Related Function Documentation

#### 4.16.4.1 FPhraseTree

```
friend class FPhraseTree [friend]
```

Definition at line 21 of file [ParseRecord.h](#).

### 4.16.5 Member Data Documentation

#### 4.16.5.1 ContextObjectStack

```
TArray<UPhraseTreeContextObject*> FParseRecord::ContextObjectStack = TArray<UPhraseTreeContextObject*>()
[protected]
```

The Context Stack of Context Objects.

Definition at line 326 of file [ParseRecord.h](#).

#### 4.16.5.2 PhraseInputs

```
TMultiMap<FString, UParseInput*> FParseRecord::PhraseInputs [protected]
```

Map of all the Provided Phrase Inputs, to their Respective Phrases.

Definition at line 336 of file [ParseRecord.h](#).

#### 4.16.5.3 PhraseRecord

```
TArray<FString> FParseRecord::PhraseRecord [protected]
```

A Record of the Phrase String used through-out propagation.

Definition at line 331 of file [ParseRecord.h](#).

The documentation for this struct was generated from the following file:

- [Source/OpenAccessibilityCommunication/Public/PhraseTree/Containers/ParseRecord.h](#)

### 4.17 FParseResult Struct Reference

Contains the Result of Propagation through the Phrase Tree.

```
#include <ParseResult.h>
```

#### Public Member Functions

- [FParseResult](#) (PhrasePropogationType InResult)
- [FParseResult](#) (PhrasePropogationType InResult, TSharedPtr< [FPhraseNode](#) > InReachedNode)

#### Public Attributes

- uint8\_t [Result](#)  
*The Result of the Propogation.*
- TSharedPtr< [FPhraseNode](#) > [ReachedNode](#)  
*The Node that was reached in the tree.*

#### 4.17.1 Detailed Description

Contains the Result of Propagation through the Phrase Tree.

Definition at line 51 of file [ParseResult.h](#).

## 4.17.2 Constructor & Destructor Documentation

### 4.17.2.1 FParseResult() [1/3]

```
FParseResult::FParseResult ( ) [inline]
```

Definition at line 53 of file [ParseResult.h](#).

```
00054     {  
00055         Result = PHRASE_NOT_PARSED;  
00056     }
```

### 4.17.2.2 FParseResult() [2/3]

```
FParseResult::FParseResult (  
    PhrasePropogationType InResult ) [inline]
```

Definition at line 58 of file [ParseResult.h](#).

```
00059     {  
00060         Result = InResult;  
00061     }
```

### 4.17.2.3 FParseResult() [3/3]

```
FParseResult::FParseResult (  
    PhrasePropogationType InResult,  
    TSharedPtr< FPhraseNode > InReachedNode ) [inline]
```

Definition at line 63 of file [ParseResult.h](#).

```
00064     {  
00065         Result = InResult;  
00066         ReachedNode = InReachedNode;  
00067     }
```

## 4.17.3 Member Data Documentation

### 4.17.3.1 ReachedNode

```
TSharedPtr<FPhraseNode> FParseResult::ReachedNode
```

The Node that was reached in the tree.

Definition at line 79 of file [ParseResult.h](#).

#### 4.17.3.2 Result

```
uint8_t FParseResult::Result
```

The Result of the Propagation.

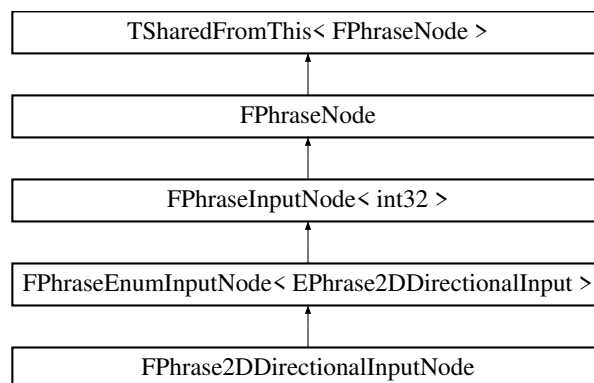
Definition at line 74 of file [ParseResult.h](#).

The documentation for this struct was generated from the following file:

- Source/OpenAccessibilityCommunication/Public/PhraseTree/Containers/ParseResult.h

### 4.18 FPhrase2DDirectionalInputNode Class Reference

Inheritance diagram for FPhrase2DDirectionalInputNode:



#### Public Member Functions

- [FPhrase2DDirectionalInputNode](#) (const TCHAR \*NodeName)
- [FPhrase2DDirectionalInputNode](#) (const TCHAR \*NodeName, TPhraseNodeArray InChildNodes)
- [FPhrase2DDirectionalInputNode](#) (const TCHAR \*NodeName, TDelegate< void([FParseRecord](#) &Record)> InOnPhraseParsed, TPhraseNodeArray InChildNodes)
- [FPhrase2DDirectionalInputNode](#) (const TCHAR \*NodeName, TPhraseNodeArray InChildNodes, TDelegate< void(int32 Input)> InOnInputRecieved)
- [FPhrase2DDirectionalInputNode](#) (const TCHAR \*NodeName, TDelegate< void([FParseRecord](#) &Record)> InOnPhraseParsed, TPhraseNodeArray InChildNodes, TDelegate< void(int32 Input)> InOnInputRecieved)

#### Additional Inherited Members

#### 4.18.1 Detailed Description

Definition at line 32 of file [PhraseDirectionalInputNode.h](#).

#### 4.18.2 Constructor & Destructor Documentation

**4.18.2.1 FPhrase2DDirectionalInputNode() [1/5]**

```
FPhrase2DDirectionalInputNode::FPhrase2DDirectionalInputNode (
    const TCHAR * NodeName ) [inline]
```

Definition at line 35 of file [PhraseDirectionalInputNode.h](#).

```
00036 : FPhraseEnumInputNode<EPhrase2DDirectionalInput>(NodeName)
00037 {}
```

**4.18.2.2 FPhrase2DDirectionalInputNode() [2/5]**

```
FPhrase2DDirectionalInputNode::FPhrase2DDirectionalInputNode (
    const TCHAR * NodeName,
    TPhraseNodeArray InChildNodes ) [inline]
```

Definition at line 39 of file [PhraseDirectionalInputNode.h](#).

```
00040 : FPhraseEnumInputNode<EPhrase2DDirectionalInput>(NodeName, InChildNodes)
00041 {}
```

**4.18.2.3 FPhrase2DDirectionalInputNode() [3/5]**

```
FPhrase2DDirectionalInputNode::FPhrase2DDirectionalInputNode (
    const TCHAR * NodeName,
    TDelegate< void(FParseRecord &Record)> InOnPhraseParsed,
    TPhraseNodeArray InChildNodes ) [inline]
```

Definition at line 43 of file [PhraseDirectionalInputNode.h](#).

```
00044 : FPhraseEnumInputNode<EPhrase2DDirectionalInput>(NodeName, InOnPhraseParsed, InChildNodes)
00045 {}
```

**4.18.2.4 FPhrase2DDirectionalInputNode() [4/5]**

```
FPhrase2DDirectionalInputNode::FPhrase2DDirectionalInputNode (
    const TCHAR * NodeName,
    TPhraseNodeArray InChildNodes,
    TDelegate< void(int32 Input)> InOnInputRecieved ) [inline]
```

Definition at line 47 of file [PhraseDirectionalInputNode.h](#).

```
00048 : FPhraseEnumInputNode<EPhrase2DDirectionalInput>(NodeName, InChildNodes, InOnInputRecieved)
00049 {}
```

#### 4.18.2.5 FPhrase2DDirectionalInputNode() [5/5]

```
FPhrase2DDirectionalInputNode::FPhrase2DDirectionalInputNode (
    const TCHAR * NodeName,
    TDelegate< void(FParseRecord &Record)> InOnPhraseParsed,
    TPhraseNodeArray InChildNodes,
    TDelegate< void(int32 Input)> InOnInputRecieved ) [inline]
```

Definition at line 51 of file [PhraseDirectionalInputNode.h](#).

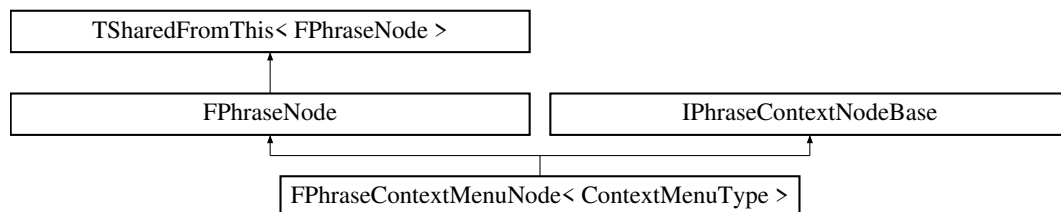
```
00052 : FPhraseEnumInputNode<EPhrase2DDirectionalInput>(NodeName, InOnPhraseParsed, InChildNodes,
00053     InOnInputRecieved)
    {}
```

The documentation for this class was generated from the following file:

- Source/OpenAccessibilityCommunication/Public/PhraseTree/PhraseDirectionalInputNode.h

### 4.19 FPhraseContextMenuNode< ContextMenuType > Class Template Reference

Inheritance diagram for FPhraseContextMenuNode< ContextMenuType >:



#### Public Member Functions

- [FPhraseContextMenuNode](#) (const TCHAR \*InInputString)
- [FPhraseContextMenuNode](#) (const TCHAR \*InInputString, TPhraseNodeArray InChildNodes)
- [FPhraseContextMenuNode](#) (const TCHAR \*InInputString, TDelegate< TSharedPtr< IMenu >(FParseRecord &Record)> InOnGetMenu, TPhraseNodeArray InChildNodes)
- [FPhraseContextMenuNode](#) (const TCHAR \*InInputString, const float InMenuScalar, TPhraseNodeArray InChildNodes)
- [FPhraseContextMenuNode](#) (const TCHAR \*InInputString, const float InMenuScalar, TDelegate< TSharedPtr< IMenu >(FParseRecord &Record)> InOnGetMenu, TPhraseNodeArray InChildNodes)
- [FPhraseContextMenuNode](#) (const TCHAR \*InInputString, const float InMenuScalar, TDelegate< void(FParseRecord &Record)> InOnPhraseParsed, TPhraseNodeArray InChildNodes)
- [FPhraseContextMenuNode](#) (const TCHAR \*InInputString, const float InMenuScalar, TDelegate< TSharedPtr< IMenu >(FParseRecord &Record)> InOnGetMenu, TDelegate< void(FParseRecord &Record)> InOnPhraseParsed, TPhraseNodeArray InChildNodes)
- virtual [FParseResult ParsePhrase](#) (TArray< FString > &InPhraseWordArray, [FParseRecord](#) &InParseRecord) override

*Parses the phrase down the given Node, propagating down child nodes if required.*

- virtual [FParseResult ParsePhraseAsContext](#) (TArray< FString > &InPhraseWordArray, [FParseRecord](#) &InParseRecord) override

*Parses the phrase down the given node, propagating down child nodes if required. Missed Pop of the Phrase Array from this Node.*

## Protected Member Functions

- virtual bool [HasContextObject](#) (TArray< [UPhraseTreeContextObject](#) \* > InContextObjects) const override  
*Checks if the Given Context Array Contains Context Objects.*
- virtual [UPhraseTreeContextObject](#) \* [CreateContextObject](#) (FParseRecord &Record) override  
*Creates a Context Object, using Record Inputs.*
- virtual void [ConstructContextChildren](#) (TPhraseNodeArray &InChildNodes) override  
*Constructs the Context Nodes Children, from Given Child Nodes. Allowing for Inclusion of Utility Nodes in relation to the Context.*

## Protected Attributes

- const float [ContextMenuScalar](#)  
*Scalar for the Initialized Menu Elements.*
- TDelegate< TSharedPtr< IMenu > (FParseRecord &Record)> [OnGetMenu](#)  
*Delegate for Initializing of the Menu.*

## Additional Inherited Members

### 4.19.1 Detailed Description

```
template<typename ContextMenuType = UPhraseTreeContextMenuObject>
class FPhraseContextMenuNode< ContextMenuType >
```

Definition at line 14 of file [PhraseContextMenuNode.h](#).

### 4.19.2 Constructor & Destructor Documentation

#### 4.19.2.1 FPhraseContextMenuNode() [1/7]

```
template<typename ContextMenuType = UPhraseTreeContextMenuObject>
FPhraseContextMenuNode< ContextMenuType >::FPhraseContextMenuNode (
    const TCHAR * InInputString ) [inline]
```

Definition at line 20 of file [PhraseContextMenuNode.h](#).

```
00021     : FPhraseNode(InInputString)
00022     , ContextMenuScalar(1.0f)
00023     {
00024     this->ChildNodes = TPhraseNodeArray();
00025     };
```

**4.19.2.2 FPhraseContextMenuNode() [2/7]**

```
template<typename ContextMenuType = UPhraseTreeContextMenuObject>
FPhraseContextMenuNode< ContextMenuType >::FPhraseContextMenuNode (
    const TCHAR * InInputString,
    TPhraseNodeArray InChildNodes ) [inline]
```

Definition at line 27 of file [PhraseContextMenuNode.h](#).

```
00028 : FPhraseNode(InInputString)
00029 , ContextMenuScalar(1.0f)
00030 {
00031     ConstructContextChildren(InChildNodes);
00032 };
```

**4.19.2.3 FPhraseContextMenuNode() [3/7]**

```
template<typename ContextMenuType = UPhraseTreeContextMenuObject>
FPhraseContextMenuNode< ContextMenuType >::FPhraseContextMenuNode (
    const TCHAR * InInputString,
    TDelegate< TSharedPtr< IMenu > (FParseRecord &Record)> InOnGetMenu,
    TPhraseNodeArray InChildNodes ) [inline]
```

Definition at line 34 of file [PhraseContextMenuNode.h](#).

```
00035 : FPhraseNode(InInputString)
00036 , ContextMenuScalar(1.0f)
00037 , OnGetMenu(InOnGetMenu)
00038 {
00039     ConstructContextChildren(InChildNodes);
00040 };
```

**4.19.2.4 FPhraseContextMenuNode() [4/7]**

```
template<typename ContextMenuType = UPhraseTreeContextMenuObject>
FPhraseContextMenuNode< ContextMenuType >::FPhraseContextMenuNode (
    const TCHAR * InInputString,
    const float InMenuScalar,
    TPhraseNodeArray InChildNodes ) [inline]
```

Definition at line 42 of file [PhraseContextMenuNode.h](#).

```
00043 : FPhraseNode(InInputString)
00044 , ContextMenuScalar(InMenuScalar)
00045 {
00046     ConstructContextChildren(InChildNodes);
00047 };
```

**4.19.2.5 FPhraseContextMenuNode() [5/7]**

```
template<typename ContextMenuType = UPhraseTreeContextMenuObject>
FPhraseContextMenuNode< ContextMenuType >::FPhraseContextMenuNode (
    const TCHAR * InInputString,
    const float InMenuScalar,
    TDelegate< TSharedPtr< IMenu > (FParseRecord &Record)> InOnGetMenu,
    TPhraseNodeArray InChildNodes ) [inline]
```

Definition at line 49 of file [PhraseContextMenuNode.h](#).

```
00050 : FPhraseNode(InInputString)
00051 , ContextMenuScalar(InMenuScalar)
00052 , OnGetMenu(InOnGetMenu)
00053 {
00054     ConstructContextChildren(InChildNodes);
00055 }
```



## 4.19.2.6 FPhraseContextMenuNode() [6/7]

```
template<typename ContextMenuType = UPhraseTreeContextMenuObject>
FPhraseContextMenuNode< ContextMenuType >::FPhraseContextMenuNode (
    const TCHAR * InInputString,
    const float InMenuScalar,
    TDelegate< void(FParseRecord &Record)> InOnPhraseParsed,
    TPhraseNodeArray InChildNodes ) [inline]
```

Definition at line 57 of file [PhraseContextMenuNode.h](#).

```
00058 : FPhraseNode(InInputString, InOnPhraseParsed)
00059 , ContextMenuScalar(InMenuScalar)
00060 {
00061     ConstructContextChildren(InChildNodes);
00062 }
```

## 4.19.2.7 FPhraseContextMenuNode() [7/7]

```
template<typename ContextMenuType = UPhraseTreeContextMenuObject>
FPhraseContextMenuNode< ContextMenuType >::FPhraseContextMenuNode (
    const TCHAR * InInputString,
    const float InMenuScalar,
    TDelegate< TSharedPtr< IMenu > (FParseRecord &Record)> InOnGetMenu,
    TDelegate< void(FParseRecord &Record)> InOnPhraseParsed,
    TPhraseNodeArray InChildNodes ) [inline]
```

Definition at line 64 of file [PhraseContextMenuNode.h](#).

```
00065 : FPhraseNode(InInputString, InOnPhraseParsed)
00066 , ContextMenuScalar(InMenuScalar)
00067 , OnGetMenu(InOnGetMenu)
00068 {
00069     ConstructContextChildren(InChildNodes);
00070 }
```

## 4.19.2.8 ~FPhraseContextMenuNode()

```
template<typename ContextMenuType = UPhraseTreeContextMenuObject>
virtual FPhraseContextMenuNode< ContextMenuType >::~~FPhraseContextMenuNode ( ) [inline],
[virtual]
```

Definition at line 72 of file [PhraseContextMenuNode.h](#).

```
00073 {
00074
00075 }
```

## 4.19.3 Member Function Documentation

## 4.19.3.1 ConstructContextChildren()

```
template<typename ContextMenuType >
void FPhraseContextMenuNode< ContextMenuType >::ConstructContextChildren (
    TPhraseNodeArray & InChildNodes ) [override], [protected], [virtual]
```

Constructs the Context Nodes Children, from Given Child Nodes. Allowing for Inclusion of Utility Nodes in relation to the Context.

## Parameters

|                     |                                   |
|---------------------|-----------------------------------|
| <i>InChildNodes</i> | - An Array of the Nodes Children. |
|---------------------|-----------------------------------|

Definition at line 225 of file [PhraseContextMenuNode.h](#).

```

00226 {
00227     // Construct Context Specific Children Nodes,
00228     // With Linked Functionality to the Context Menu Object and Root Node.
00229     TSharedPtr<FPhraseEventNode> CloseContextNode = MakeShared<FPhraseEventNode>();
00230     CloseContextNode->OnPhraseParsed.BindLambda(
00231         [this](FParseRecord& Record) {
00232
00233             UPhraseTreeContextMenuObject* ContextMenu =
00234                 Record.GetContextObj<UPhraseTreeContextMenuObject>();
00235             if (ContextMenu->GetContextRoot() == this->AsShared())
00236             {
00237                 ContextMenu->Close();
00238                 ContextMenu->RemoveFromRoot();
00239                 Record.PopContextObj();
00240             }
00241         }
00242     );
00243
00244     this->ChildNodes = TPhraseNodeArray{
00245         MakeShared<FPhraseNode>(TEXT("CLOSE"),
00246             TPhraseNodeArray {
00247                 CloseContextNode
00248             })
00249     };
00250
00251     this->ChildNodes.Append(InChildNodes);
00252 }
```

#### 4.19.3.2 CreateContextObject()

```

template<typename ContextMenuType >
UPhraseTreeContextObject * FPhraseContextMenuNode< ContextMenuType >::CreateContextObject (
    FParseRecord & Record ) [override], [protected], [virtual]
```

Creates a Context Object, using Record Inputs.

## Returns

The Created Context Object, otherwise nullptr

Implements [IPhraseContextNodeBase](#).

Definition at line 200 of file [PhraseContextMenuNode.h](#).

```

00201 {
00202     if (!OnGetMenu.IsBound())
00203     {
00204         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("OnGetMenu Delegate Not Bound. Cannot Create Context
Object, linked to a Menu.));
00205         return nullptr;
00206     }
00207
00208     TSharedPtr<IMenu> NewMenu = OnGetMenu.Execute(Record);
00209
00210     if (!NewMenu.IsValid())
00211     {
00212         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("OnGetMenu Delegate Returned Invalid Menu. Cannot
Create Context Object.));
00213         return nullptr;
00214     }
00215
00216     ContextMenuType* NewContextObject = NewObject<ContextMenuType>();
00217     NewContextObject->Init(NewMenu.ToSharedRef(), this->AsShared());
00218
00219     NewContextObject->ScaleMenu(ContextMenuScalar);
00220
00221     return NewContextObject;
00222 }
```

### 4.19.3.3 HasContextObject()

```
template<typename ContextMenuType >
bool FPhraseContextMenuNode< ContextMenuType >::HasContextObject (
    TArray< UPhraseTreeContextObject * > InContextObjects ) const [override], [protected],
[virtual]
```

Checks if the Given Context Array Contains Context Objects.

#### Parameters

|                         |   |
|-------------------------|---|
| <i>InContextObjects</i> | - The Array To Check For Context Objects. |
|-------------------------|---|

#### Returns

True, if their is Context Objects in the Given Array.

Implements [IPhraseContextNodeBase](#).

Definition at line 186 of file [PhraseContextMenuNode.h](#).

```
00187 {
00188     for (auto& ContextObject : InContextObjects)
00189     {
00190         if (ContextObject->IsA(ContextMenuType::StaticClass()) && ContextObject->GetContextRoot() ==
AsShared())
00191         {
00192             return true;
00193         }
00194     }
00195     return false;
00196 }
00197 }
```

### 4.19.3.4 ParsePhrase()

```
template<typename ContextMenuType >
FParseResult FPhraseContextMenuNode< ContextMenuType >::ParsePhrase (
    TArray< FString > & InPhraseWordArray,
    FParseRecord & InParseRecord ) [override], [virtual]
```

Parses the phrase down the given Node, propagating down child nodes if required.

#### Parameters

|                          |  |
|--------------------------|--|
| <i>InPhraseWordArray</i> | The Array of Phrase Strings to Propagate against.            |
| <i>InParseRecord</i>     | The Record of Propagation of collected context's and inputs. |

#### Returns

Returns the Result of the propogation, including any key nodes met.

Reimplemented from [FPhraseNode](#).

Definition at line 138 of file [PhraseContextMenuNode.h](#).

```

00139 {
00140     if (!HasContextObject(InParseRecord.GetContextStack()))
00141     {
00142         UPhraseTreeContextObject* NewObject = CreateContextObject(InParseRecord);
00143         InParseRecord.PushContextObj(NewObject);
00144     }
00145     if (InPhraseWordArray.IsEmpty())
00146     {
00147         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Emptied Phrase Array ||"))
00148         return FParseResult(PHRASE_REQUIRES_MORE, AsShared());
00149     }
00150     InPhraseWordArray.Pop();
00151     OnPhraseParsed.ExecuteIfBound(InParseRecord);
00152     return ParseChildren(InPhraseWordArray, InParseRecord);
00153 }
00154 return FPhraseNode::ParsePhrase(InPhraseWordArray, InParseRecord);
00155 }

```

#### 4.19.3.5 ParsePhraseAsContext()

```

template<typename ContextMenuType >
FParseResult FPhraseContextMenuNode< ContextMenuType >::ParsePhraseAsContext (
    TArray< FString > & InPhraseWordArray,
    FParseRecord & InParseRecord ) [inline], [override], [virtual]

```

Parses the phrase down the given node, propagating down child nodes if required. Missed Pop of the Phrase Array from this Node.

##### Parameters

|                          |  |
|--------------------------|--|
| <i>InPhraseWordArray</i> |  |
| <i>InParseRecord</i>     |  |

##### Returns

Returns the Result of the propogation, including any key nodes met.

Reimplemented from [FPhraseNode](#).

Definition at line 164 of file [PhraseContextMenuNode.h](#).

```

00165 {
00166     if (!HasContextObject(InParseRecord.GetContextStack()))
00167     {
00168         UPhraseTreeContextObject* NewObject = CreateContextObject(InParseRecord);
00169         InParseRecord.PushContextObj(NewObject);
00170     }
00171     if (InPhraseWordArray.IsEmpty())
00172     {
00173         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Emptied Phrase Array ||"))
00174         return FParseResult(PHRASE_REQUIRES_MORE, AsShared());
00175     }
00176     InPhraseWordArray.Pop();
00177     OnPhraseParsed.ExecuteIfBound(InParseRecord);
00178     return ParseChildren(InPhraseWordArray, InParseRecord);
00179 }

```

## 4.19.4 Member Data Documentation

### 4.19.4.1 ContextMenuScalar

```
template<typename ContextMenuType = UPhraseTreeContextMenuObject>
const float FPhraseContextMenuNode< ContextMenuType >::ContextMenuScalar [protected]
```

Scalar for the Initialized Menu Elements.

Definition at line 129 of file [PhraseContextMenuNode.h](#).

### 4.19.4.2 OnGetMenu

```
template<typename ContextMenuType = UPhraseTreeContextMenuObject>
TDelegate<TSharedPtr<IMenu>(FParseRecord& Record)> FPhraseContextMenuNode< ContextMenuType
>::OnGetMenu [protected]
```

Delegate for Initializing of the Menu.

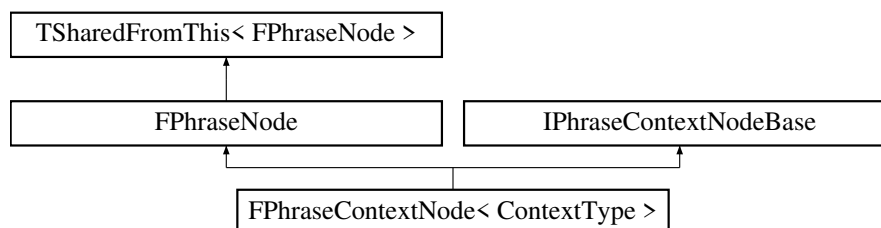
Definition at line 134 of file [PhraseContextMenuNode.h](#).

The documentation for this class was generated from the following file:

- [Source/OpenAccessibilityCommunication/Public/PhraseTree/PhraseContextMenuNode.h](#)

## 4.20 FPhraseContextNode< ContextType > Class Template Reference

Inheritance diagram for FPhraseContextNode< ContextType >:



## Public Member Functions

- [FPhraseContextNode](#) (const TCHAR \*InInputString)
- [FPhraseContextNode](#) (const TCHAR \*InInputString, TPhraseNodeArray InChildNodes)
- [FPhraseContextNode](#) (const TCHAR \*InInputString, TDelegate< void([FParseRecord](#) &Record)> InOnPhraseParsed, TPhraseNodeArray InChildNodes)
- virtual [FPhraseResult](#) [ParsePhrase](#) (TArray< FString > &InPhraseWordArray, [FParseRecord](#) &InParseRecord) override  
*Parses The Phrase Down This Node, Propagating Down Any Child Nodes If Required.*
- virtual [FPhraseResult](#) [ParsePhraseAsContext](#) (TArray< FString > &InPhraseWordArray, [FParseRecord](#) &InParseRecord) override  
*Parses the Phrase Down This Node, Propagating Down Any Child Nodes If Required. Does not Pop the Phrase Array.*

## Protected Member Functions

- virtual bool [HasContextObject](#) (TArray< [UPhraseTreeContextObject](#) \* > InContextObjects) const override  
*Checks if the Given Context Array Contains Context Objects.*
- virtual [UPhraseTreeContextObject](#) \* [CreateContextObject](#) (FParseRecord &Record) override  
*Creates a Context Object, using Record Inputs.*
- virtual void [ConstructContextChildren](#) (TPhraseNodeArray &InChildNodes) override

## Additional Inherited Members

### 4.20.1 Detailed Description

```
template<class ContextType = UPhraseTreeContextObject>
class FPhraseContextNode< ContextType >
```

Definition at line 14 of file [PhraseContextNode.h](#).

### 4.20.2 Constructor & Destructor Documentation

#### 4.20.2.1 FPhraseContextNode() [1/3]

```
template<class ContextType = UPhraseTreeContextObject>
FPhraseContextNode< ContextType >::FPhraseContextNode (
    const TCHAR * InInputString ) [inline]
```

Definition at line 18 of file [PhraseContextNode.h](#).

```
00019     : FPhraseNode(InInputString)
00020     {
00021         static_assert(std::is_base_of<UPhraseTreeContextObject, ContextType>::value, "ContextType must
be a subclass of UPhraseTreeContextObject");
00022
00023         TPhraseNodeArray EmptyArray = TPhraseNodeArray();
00024         ConstructContextChildren(EmptyArray);
00025     }
```

#### 4.20.2.2 FPhraseContextNode() [2/3]

```
template<class ContextType = UPhraseTreeContextObject>
FPhraseContextNode< ContextType >::FPhraseContextNode (
    const TCHAR * InInputString,
    TPhraseNodeArray InChildNodes ) [inline]
```

Definition at line 27 of file [PhraseContextNode.h](#).

```
00028     : FPhraseNode(InInputString, InChildNodes)
00029     {
00030         static_assert(std::is_base_of<UPhraseTreeContextObject, ContextType>::value, "ContextType must
be a subclass of UPhraseTreeContextObject");
00031
00032         ConstructContextChildren(InChildNodes);
00033     }
```

## 4.20.2.3 FPhraseContextNode() [3/3]

```
template<class ContextType = UPhraseTreeContextObject>
FPhraseContextNode< ContextType >::FPhraseContextNode (
    const TCHAR * InInputString,
    TDelegate< void(FParseRecord &Record)> InOnPhraseParsed,
    TPhraseNodeArray InChildNodes ) [inline]
```

Definition at line 35 of file [PhraseContextNode.h](#).

```
00036         : FPhraseNode(InInputString, InOnPhraseParsed)
00037     {
00038         static_assert(std::is_base_of<UPhraseTreeContextObject, ContextType>::value, "ContextType must
be a subclass of UPhraseTreeContextObject");
00039
00040         ConstructContextChildren(InChildNodes);
00041     }
```

## 4.20.2.4 ~FPhraseContextNode()

```
template<class ContextType = UPhraseTreeContextObject>
virtual FPhraseContextNode< ContextType >::~~FPhraseContextNode ( ) [inline], [virtual]
```

Definition at line 43 of file [PhraseContextNode.h](#).

```
00044     {
00045
00046     }
```

## 4.20.3 Member Function Documentation

## 4.20.3.1 ConstructContextChildren()

```
template<class ContextType >
void FPhraseContextNode< ContextType >::ConstructContextChildren (
    TPhraseNodeArray & InChildNodes ) [override], [protected], [virtual]
```

Definition at line 132 of file [PhraseContextNode.h](#).

```
00133 {
00134     TSharedPtr<FPhraseEventNode> CloseContextNode = MakeShared<FPhraseEventNode>();
00135     CloseContextNode->OnPhraseParsed.BindLambda(
00136         [this](FParseRecord& Record) {
00137
00138             UPhraseTreeContextObject* ContextObject = Record.GetContextObj();
00139             if (ContextObject->GetContextRoot() == this->AsShared())
00140             {
00141                 ContextObject->Close();
00142                 ContextObject->RemoveFromRoot();
00143
00144                 Record.PopContextObj();
00145             }
00146         }
00147     );
00148
00149     this->ChildNodes = TPhraseNodeArray{
00150         MakeShared<FPhraseNode>(TEXT("CLOSE"),
00151             TPhraseNodeArray {
00152                 CloseContextNode
00153             })
00154     };
00155
00156     this->ChildNodes.Append(InChildNodes);
00157 }
```

#### 4.20.3.2 CreateContextObject()

```
template<class ContextType >
UPhraseTreeContextObject * FPhraseContextNode< ContextType >::CreateContextObject (
    FParseRecord & Record ) [override], [protected], [virtual]
```

Creates a Context Object, using Record Inputs.

##### Returns

The Created Context Object, otherwise nullptr

Implements [IPhraseContextNodeBase](#).

Definition at line 122 of file [PhraseContextNode.h](#).

```
00123 {
00124     ContextType* NewContextObject = NewObject<ContextType>();
00125     NewContextObject->Init();
00126     NewContextObject->SetContextRootNode(AsShared());
00127
00128     return NewContextObject;
00129 }
```

#### 4.20.3.3 HasContextObject()

```
template<class ContextType >
bool FPhraseContextNode< ContextType >::HasContextObject (
    TArray< UPhraseTreeContextObject * > InContextObjects ) const [override], [protected],
[virtual]
```

Checks if the Given Context Array Contains Context Objects.

##### Parameters

|                         |   |
|-------------------------|---|
| <i>InContextObjects</i> | - The Array To Check For Context Objects. |
|-------------------------|---|

##### Returns

True, if their is Context Objects in the Given Array.

Implements [IPhraseContextNodeBase](#).

Definition at line 107 of file [PhraseContextNode.h](#).

```
00108 {
00109     for (auto& ContextObject : InContextObjects)
00110     {
00111         if (ContextObject->IsA(ContextType::StaticClass()) && ContextObject->GetContextRoot() ==
AsShared())
00112         {
00113             return true;
00114         }
00115     }
00116     return false;
00117 }
00118 }
```



#### 4.20.3.4 ParsePhrase()

```
template<class ContextType >
FParseResult FPhraseContextNode< ContextType >::ParsePhrase (
    TArray< FString > & InPhraseWordArray,
    FParseRecord & InParseRecord ) [override], [virtual]
```

Parses The Phrase Down This Node, Propagating Down Any Child Nodes If Required.

##### Parameters

|                          |  |
|--------------------------|--|
| <i>InPhraseWordArray</i> | - The Current Array of Transcription Phrases.  |
| <i>InParseRecord</i>     | - The Parse Record of the Current Propagation. |

##### Returns

The Result of the Parsing of the Phrase, and any Propagation.

Reimplemented from [FPhraseNode](#).

Definition at line 71 of file [PhraseContextNode.h](#).

```
00072 {
00073     if (!HasContextObject (InParseRecord.GetContextStack ()))
00074     {
00075         UPhraseTreeContextObject* NewObject = CreateContextObject (InParseRecord);
00076         InParseRecord.PushContextObj (NewObject);
00077     }
00078 }
00079
00080 return FPhraseNode::ParsePhrase (InPhraseWordArray, InParseRecord);
00081 }
```

#### 4.20.3.5 ParsePhraseAsContext()

```
template<class ContextType >
FParseResult FPhraseContextNode< ContextType >::ParsePhraseAsContext (
    TArray< FString > & InPhraseWordArray,
    FParseRecord & InParseRecord ) [override], [virtual]
```

Parses the Phrase Down This Node, Propagating Down Any Child Nodes If Required. Does not Pop the Phrase Array.

##### Parameters

|                          |  |
|--------------------------|--|
| <i>InPhraseWordArray</i> | - The Current Array of Transcription Phrases.  |
| <i>InParseRecord</i>     | - The Parse Record of the Current Propagation. |

##### Returns

The Result of the Parsing of the Phrase, and any Propagation.

Reimplemented from [FPhraseNode](#).

Definition at line 84 of file [PhraseContextNode.h](#).

```

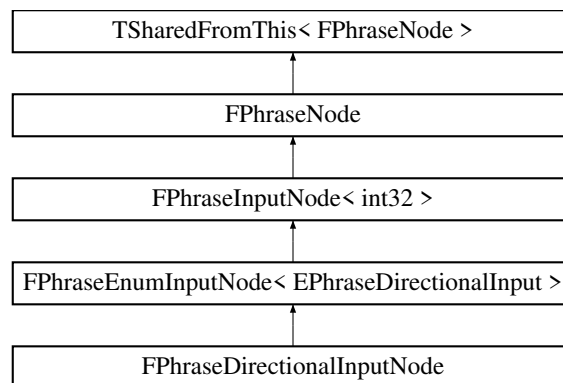
00085 {
00086     if (!HasContextObject(InParseRecord.GetContextStack()))
00087     {
00088         UPhraseTreeContextObject* NewObject = CreateContextObject(InParseRecord);
00089         InParseRecord.PushContextObj(NewObject);
00090     }
00091 }
00092
00093 if (InPhraseWordArray.IsEmpty())
00094 {
00095     UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Emptied Phrase Array ||"))
00096     return FParseResult(PHRASE_REQUIRES_MORE, AsShared());
00097 }
00098
00099 OnPhraseParsed.ExecuteIfBound(InParseRecord);
00100
00101 // Pass
00102 return ParseChildren(InPhraseWordArray, InParseRecord);
00103 }
00104 }
```

The documentation for this class was generated from the following file:

- Source/OpenAccessibilityCommunication/Public/PhraseTree/PhraseContextNode.h

## 4.21 FPhraseDirectionalInputNode Class Reference

Inheritance diagram for FPhraseDirectionalInputNode:



### Public Member Functions

- [FPhraseDirectionalInputNode](#) (const TCHAR \*NodeName)
- [FPhraseDirectionalInputNode](#) (const TCHAR \*NodeName, TPhraseNodeArray InChildNodes)
- [FPhraseDirectionalInputNode](#) (const TCHAR \*NodeName, TDelegate< void([FParseRecord](#) &Record)> InOnPhraseParsed, TPhraseNodeArray InChildNodes)
- [FPhraseDirectionalInputNode](#) (const TCHAR \*NodeName, TPhraseNodeArray InChildNodes, TDelegate< void(int32 Input)> InOnInputRecieved)
- [FPhraseDirectionalInputNode](#) (const TCHAR \*NodeName, TDelegate< void([FParseRecord](#) &Record)> InOnPhraseParsed, TPhraseNodeArray InChildNodes, TDelegate< void(int32 Input)> InOnInputRecieved)

### Additional Inherited Members

#### 4.21.1 Detailed Description

Definition at line 8 of file [PhraseDirectionalInputNode.h](#).

## 4.21.2 Constructor & Destructor Documentation

### 4.21.2.1 FPhraseDirectionalInputNode() [1/5]

```
FPhraseDirectionalInputNode::FPhraseDirectionalInputNode (
    const TCHAR * NodeName ) [inline]
```

Definition at line 11 of file [PhraseDirectionalInputNode.h](#).

```
00012     : FPhraseEnumInputNode<EPhraseDirectionalInput>(NodeName)
00013     {}
```

### 4.21.2.2 FPhraseDirectionalInputNode() [2/5]

```
FPhraseDirectionalInputNode::FPhraseDirectionalInputNode (
    const TCHAR * NodeName,
    TPhraseNodeArray InChildNodes ) [inline]
```

Definition at line 15 of file [PhraseDirectionalInputNode.h](#).

```
00016     : FPhraseEnumInputNode<EPhraseDirectionalInput>(NodeName, InChildNodes)
00017     {}
```

### 4.21.2.3 FPhraseDirectionalInputNode() [3/5]

```
FPhraseDirectionalInputNode::FPhraseDirectionalInputNode (
    const TCHAR * NodeName,
    TDelegate< void(FParseRecord &Record)> InOnPhraseParsed,
    TPhraseNodeArray InChildNodes ) [inline]
```

Definition at line 19 of file [PhraseDirectionalInputNode.h](#).

```
00020     : FPhraseEnumInputNode<EPhraseDirectionalInput>(NodeName, InOnPhraseParsed, InChildNodes)
00021     {}
```

### 4.21.2.4 FPhraseDirectionalInputNode() [4/5]

```
FPhraseDirectionalInputNode::FPhraseDirectionalInputNode (
    const TCHAR * NodeName,
    TPhraseNodeArray InChildNodes,
    TDelegate< void(int32 Input)> InOnInputRecieved ) [inline]
```

Definition at line 23 of file [PhraseDirectionalInputNode.h](#).

```
00024     : FPhraseEnumInputNode<EPhraseDirectionalInput>(NodeName, InChildNodes, InOnInputRecieved)
00025     {}
```

#### 4.21.2.5 FPhraseDirectionalInputNode() [5/5]

```
FPhraseDirectionalInputNode::FPhraseDirectionalInputNode (
    const TCHAR * NodeName,
    TDelegate< void(FParseRecord &Record)> InOnPhraseParsed,
    TPhraseNodeArray InChildNodes,
    TDelegate< void(int32 Input)> InOnInputRecieved ) [inline]
```

Definition at line 27 of file [PhraseDirectionalInputNode.h](#).

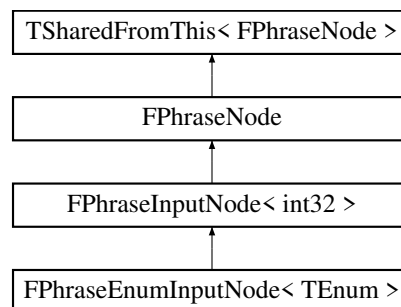
```
00028 : FPhraseEnumInputNode<EPhraseDirectionalInput>(NodeName, InOnPhraseParsed, InChildNodes,
00029     InOnInputRecieved)
    {}
```

The documentation for this class was generated from the following file:

- Source/OpenAccessibilityCommunication/Public/PhraseTree/PhraseDirectionalInputNode.h

## 4.22 FPhraseEnumInputNode< TEnum > Class Template Reference

Inheritance diagram for FPhraseEnumInputNode< TEnum >:



### Public Member Functions

- [FPhraseEnumInputNode](#) (const TCHAR \*InInputString)
- [FPhraseEnumInputNode](#) (const TCHAR \*InInputString, TPhraseNodeArray InChildNodes)
- [FPhraseEnumInputNode](#) (const TCHAR \*InInputString, TDelegate< void([FParseRecord](#) &Record)> InOnPhraseParsed, TPhraseNodeArray InChildNodes)
- [FPhraseEnumInputNode](#) (const TCHAR \*InInputString, TPhraseNodeArray InChildNodes, TDelegate< void(int32 Input)> InOnInputRecieved)
- [FPhraseEnumInputNode](#) (const TCHAR \*InInputString, TDelegate< void([FParseRecord](#) &Record)> InOnPhraseParsed, TPhraseNodeArray InChildNodes, TDelegate< void(int32 Input)> InOnInputRecieved)

### Protected Member Functions

- virtual bool [MeetsInputRequirements](#) (const FString &InPhrase) override  
*Checks if the Given Phrase Meets Requirements for usage as Input. In Correlation to this Nodes Input Specifications.*
- virtual bool [RecordInput](#) (const FString &InInput, [FParseRecord](#) &OutParseRecord) override  
*Records the Input onto the Parse Record.*

## Additional Inherited Members

### 4.22.1 Detailed Description

```
template<typename TEnum>
class FPhraseEnumInputNode< TEnum >
```

Definition at line 13 of file [PhraseEnumInputNode.h](#).

### 4.22.2 Constructor & Destructor Documentation

#### 4.22.2.1 FPhraseEnumInputNode() [1/5]

```
template<typename TEnum >
FPhraseEnumInputNode< TEnum >::FPhraseEnumInputNode (
    const TCHAR * InInputString )
```

Definition at line 9 of file [PhraseEnumInputNode.cpp](#).

```
00010     : FPhraseInputNode(NodeName)
00011 {
00012     static_assert(TIsEnum<TEnum>::Value, "Passed EnumType Must be an Enum.");
00013 };
```

#### 4.22.2.2 FPhraseEnumInputNode() [2/5]

```
template<typename TEnum >
FPhraseEnumInputNode< TEnum >::FPhraseEnumInputNode (
    const TCHAR * InInputString,
    TPhraseNodeArray InChildNodes )
```

Definition at line 16 of file [PhraseEnumInputNode.cpp](#).

```
00017     : FPhraseInputNode(NodeName, InChildNodes)
00018 {
00019     static_assert(TIsEnum<TEnum>::Value, "Passed EnumType Must be an Enum.");
00020 }
```

#### 4.22.2.3 FPhraseEnumInputNode() [3/5]

```
template<typename TEnum >
FPhraseEnumInputNode< TEnum >::FPhraseEnumInputNode (
    const TCHAR * InInputString,
    TDelegate< void(FParseRecord &Record)> InOnPhraseParsed,
    TPhraseNodeArray InChildNodes )
```

Definition at line 23 of file [PhraseEnumInputNode.cpp](#).

```
00024     : FPhraseInputNode(InInputString, InOnPhraseParsed, InChildNodes)
00025 {
00026     static_assert(TIsEnum<TEnum>::Value, "Passed EnumType Must be an Enum.");
00027 }
```

#### 4.22.2.4 FPhraseEnumInputNode() [4/5]

```
template<typename TEnum >
FPhraseEnumInputNode< TEnum >::FPhraseEnumInputNode (
    const TCHAR * InInputString,
    TPhraseNodeArray InChildNodes,
    TDelegate< void(int32 Input)> InOnInputRecieved )
```

Definition at line 30 of file [PhraseEnumInputNode.cpp](#).

```
00031 : FPhraseInputNode(InInputString, InChildNodes, InOnInputRecieved)
00032 {
00033     static_assert(TIsEnum<TEnum>::Value, "Passed EnumType Must be an Enum");
00034 }
```

#### 4.22.2.5 FPhraseEnumInputNode() [5/5]

```
template<typename TEnum >
FPhraseEnumInputNode< TEnum >::FPhraseEnumInputNode (
    const TCHAR * InInputString,
    TDelegate< void(FParseRecord &Record)> InOnPhraseParsed,
    TPhraseNodeArray InChildNodes,
    TDelegate< void(int32 Input)> InOnInputRecieved )
```

Definition at line 37 of file [PhraseEnumInputNode.cpp](#).

```
00038 : FPhraseInputNode(InInputString, InOnPhraseParsed, InChildNodes, InOnInputRecieved)
00039 {
00040     static_assert(TIsEnum<TEnum>::Value, "Passed EnumType Must be an Enum");
00041 }
```

#### 4.22.2.6 ~FPhraseEnumInputNode()

```
template<typename TEnum >
FPhraseEnumInputNode< TEnum >::~~FPhraseEnumInputNode [virtual]
```

Definition at line 44 of file [PhraseEnumInputNode.cpp](#).

```
00045 {
00046
00047 }
```

### 4.22.3 Member Function Documentation

#### 4.22.3.1 MeetsInputRequirements()

```
template<typename TEnum >
bool FPhraseEnumInputNode< TEnum >::MeetsInputRequirements (
    const FString & InPhrase ) [override], [protected], [virtual]
```

Checks if the Given Phrase Meets Requirements for usage as Input. In Correlation to this Nodes Input Specifications.

## Parameters

|                 |   |
|-----------------|---|
| <i>InPhrase</i> | - The Phrase To Check If It Meets Requirements. |
|-----------------|---|

## Returns

True, if the Phrase Meets Requirements. Otherwise False.

Reimplemented from [FPhraseInputNode< int32 >](#).

Definition at line 50 of file [PhraseEnumInputNode.cpp](#).

```
00051 {
00052     UEnum* EnumPtr = StaticEnum<TEnum>();
00053     if (!EnumPtr)
00054     {
00055         UE_LOG(LogTemp, Error, TEXT("FPhraseEnumInputNode::MeetsInputRequirements: EnumPtr is NULL"));
00056         return false;
00057     }
00058     return EnumPtr->IsValidEnumName(*EnumPtr->GenerateFullEnumName(*InPhrase.ToUpper()));
00059 }
00060 }
```

## 4.22.3.2 RecordInput()

```
template<typename TEnum >
bool FPhraseEnumInputNode< TEnum >::RecordInput (
    const FString & InInput,
    FParseRecord & OutParseRecord ) [override], [protected], [virtual]
```

Records the Input onto the Parse Record.

## Parameters

|                       |   |
|-----------------------|---|
| <i>InInput</i>        | - The Phrase To Record onto the Parse Record. |
| <i>OutParseRecord</i> | - Returns the Updated ParseRecord.            |

## Returns

True, if the Input Was Successful in Recording. Otherwise False.

Reimplemented from [FPhraseInputNode< int32 >](#).

Definition at line 63 of file [PhraseEnumInputNode.cpp](#).

```
00064 {
00065     UEnum* EnumPtr = StaticEnum<TEnum>();
00066     if (!EnumPtr)
00067     {
00068         UE_LOG(LogTemp, Error, TEXT("FPhraseEnumInputNode::RecordInput: EnumPtr is NULL"));
00069         return false;
00070     }
00071     int32 Val = EnumPtr->GetValueByNameString(EnumPtr->GenerateFullEnumName(*InInput.ToUpper()));
00072     if (Val == INDEX_NONE)
00073     {
00074         return false;
00075     }
00076     UParseEnumInput* ParseInput = MakeParseInput<UParseEnumInput>();
00077 }
00078 }
```

```

00079     ParseInput->SetValue(Val);
00080     ParseInput->SetEnumType(EnumPtr);
00081
00082     OutParseRecord.AddPhraseInput(BoundPhrase, ParseInput);
00083
00084     return true;
00085 }

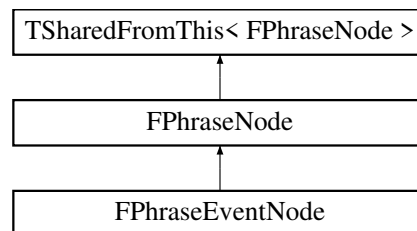
```

The documentation for this class was generated from the following files:

- Source/OpenAccessibilityCommunication/Public/PhraseTree/PhraseEnumInputNode.h
- Source/OpenAccessibilityCommunication/Private/PhraseTree/PhraseEnumInputNode.cpp

## 4.23 FPhraseEventNode Class Reference

Inheritance diagram for FPhraseEventNode:



### Public Member Functions

- [FPhraseEventNode](#) (TDelegate< void([FParseRecord](#) &)> InEvent)
- [FPhraseEventNode](#) (TFunction< void([FParseRecord](#) &)> InEventFunction)
- virtual bool [IsLeafNode](#) () const override  
*Checks if the Node is a Leaf Node.*
- virtual bool [RequiresPhrase](#) (const FString InPhrase) override  
*Checks if the Node Requires the Given Phrase.*
- virtual bool [RequiresPhrase](#) (const FString InPhrase, int32 &OutDistance) override  
*Checks if the Node Requires the Given Phrase, and Returns the Distance of the Phrase.*
- virtual [FParseResult](#) [ParsePhrase](#) (TArray< FString > &InPhraseArray, [FParseRecord](#) &InParseRecord) override  
*Parses The Phrase Down This Node, Propagating Down Any Child Nodes If Required.*

### Additional Inherited Members

#### 4.23.1 Detailed Description

Definition at line 11 of file [PhraseEventNode.h](#).

#### 4.23.2 Constructor & Destructor Documentation



#### 4.23.2.1 FPhraseEventNode() [1/3]

FPhraseEventNode::FPhraseEventNode ( )

Definition at line 7 of file [PhraseEventNode.cpp](#).

```
00008      : FPhraseNode(TEXT("EVENT_NODE"))
00009 {
00010     OnPhraseParsed = TDelegate<void(FParseRecord&)>();
00011 }
```

#### 4.23.2.2 FPhraseEventNode() [2/3]

FPhraseEventNode::FPhraseEventNode (   
TDelegate< void(FParseRecord &)> InEvent )

Definition at line 13 of file [PhraseEventNode.cpp](#).

```
00014      : FPhraseNode(TEXT("EVENT_NODE"), InEvent)
00015 {
00016
00017 }
```

#### 4.23.2.3 FPhraseEventNode() [3/3]

FPhraseEventNode::FPhraseEventNode (   
TFunction< void(FParseRecord &)> InEventFunction )

Definition at line 19 of file [PhraseEventNode.cpp](#).

```
00020      : FPhraseNode(TEXT("EVENT_NODE"), TDelegate<void(FParseRecord&)>::CreateLambda(InEventFunction))
00021 {
00022
00023 }
```

#### 4.23.2.4 ~FPhraseEventNode()

FPhraseEventNode::~FPhraseEventNode ( ) [virtual]

Definition at line 25 of file [PhraseEventNode.cpp](#).

```
00026 {
00027
00028 }
```

### 4.23.3 Member Function Documentation

#### 4.23.3.1 IsLeafNode()

```
virtual bool FPhraseEventNode::IsLeafNode ( ) const [inline], [override], [virtual]
```

Checks if the Node is a Leaf Node.

##### Returns

True, if the Node is a Leaf Node. Otherwise False.

Reimplemented from [FPhraseNode](#).

Definition at line 21 of file [PhraseEventNode.h](#).

```
00021 { return true; }
```

#### 4.23.3.2 ParsePhrase()

```
FParseResult FPhraseEventNode::ParsePhrase (
    TArray< FString > & InPhraseWordArray,
    FParseRecord & InParseRecord ) [override], [virtual]
```

Parses The Phrase Down This Node, Propagating Down Any Child Nodes If Required.

##### Parameters

|                          |  |
|--------------------------|--|
| <i>InPhraseWordArray</i> | - The Current Array of Transcription Phrases.  |
| <i>InParseRecord</i>     | - The Parse Record of the Current Propagation. |

##### Returns

The Result of the Parsing of the Phrase, and any Propagation.

Reimplemented from [FPhraseNode](#).

Definition at line 41 of file [PhraseEventNode.cpp](#).

```
00042 {
00043     if (OnPhraseParsed.ExecuteIfBound(InParseRecord))
00044     {
00045         return FParseResult (PHRASE_PARSED_AND_EXECUTED);
00046     }
00047     UE_LOG(LogOpenAccessibilityCom, Warning, TEXT("|| Unable to Execute Event ||"))
00048     return FParseResult (PHRASE_UNABLE_TO_PARSE, AsShared());
00050 }
00051 }
```

#### 4.23.3.3 RequiresPhrase() [1/2]

```
bool FPhraseEventNode::RequiresPhrase (
    const FString InPhrase ) [override], [virtual]
```

Checks if the Node Requires the Given Phrase.

**Parameters**

|                 |  |
|-----------------|--|
| <i>InPhrase</i> | - The Phrase To Check if Required By The Node. |
|-----------------|--|

**Returns**

True, if the Phrase is Required. Otherwise False.

Reimplemented from [FPhraseNode](#).

Definition at line 30 of file [PhraseEventNode.cpp](#).

```
00031 {
00032     return true;
00033 }
```

**4.23.3.4 RequiresPhrase() [2/2]**

```
bool FPhraseEventNode::RequiresPhrase (
    const FString InPhrase,
    int32 & OutDistance ) [override], [virtual]
```

Checks if the Node Requires the Given Phrase, and Returns the Distance of the Phrase.

**Parameters**

|                    |  |
|--------------------|--|
| <i>InPhrase</i>    | - The Phrase To Check if Required By The Node.                       |
| <i>OutDistance</i> | - The Returned Distancing from the Target Phrase To The BoundPhrase. |

**Returns**

True, if the Phrase is Required. Otherwise False.

Reimplemented from [FPhraseNode](#).

Definition at line 35 of file [PhraseEventNode.cpp](#).

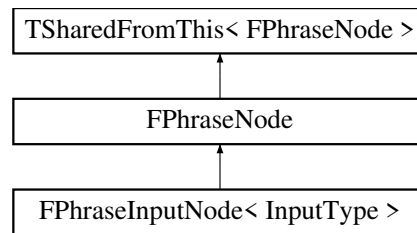
```
00036 {
00037     OutDistance = 0;
00038     return true;
00039 }
```

The documentation for this class was generated from the following files:

- Source/OpenAccessibilityCommunication/Public/PhraseTree/PhraseEventNode.h
- Source/OpenAccessibilityCommunication/Private/PhraseTree/PhraseEventNode.cpp

**4.24 FPhraseInputNode< InputType > Class Template Reference**

Inheritance diagram for FPhraseInputNode< InputType >:



## Public Member Functions

- [FPhraseInputNode](#) (const TCHAR \*InInputString)
- [FPhraseInputNode](#) (const TCHAR \*InInputString, TPhraseNodeArray InChildNodes)
- [FPhraseInputNode](#) (const TCHAR \*InInputString, TDelegate< void([FParseRecord](#) &Record)> InOnPhraseParsed, TPhraseNodeArray InChildNodes)
- [FPhraseInputNode](#) (const TCHAR \*InInputString, TPhraseNodeArray InChildNodes, TDelegate< void(InputType Input)> InOnInputRecieved)
- [FPhraseInputNode](#) (const TCHAR \*InInputString, TDelegate< void([FParseRecord](#) &Record)> InOnPhraseParsed, TPhraseNodeArray InChildNodes, TDelegate< void(InputType Input)> InOnInputRecieved)
- virtual bool [RequiresPhrase](#) (const FString InPhrase) override  
*Checks if the Node Requires the Given Phrase.*
- virtual bool [RequiresPhrase](#) (const FString InPhrase, int32 &OutDistance) override  
*Checks if the Node Requires the Given Phrase, and Returns the Distance of the Phrase.*
- virtual [FParseResult](#) [ParsePhrase](#) (TArray< FString > &InPhraseArray, [FParseRecord](#) &InParseRecord) override  
*Parses The Phrase Down This Node, Propagating Down Any Child Nodes If Required.*

## Public Attributes

- TDelegate< void(InputType ReceivedInput)> [OnInputReceived](#)

## Protected Member Functions

- virtual bool [MeetsInputRequirements](#) (const FString &InPhrase)  
*Checks if the Given Phrase Meets Requirements for usage as Input. In Correlation to this Nodes Input Specifications.*
- virtual bool [RecordInput](#) (const FString &InInput, [FParseRecord](#) &OutParseRecord)  
*Records the Input onto the Parse Record.*

## Additional Inherited Members

### 4.24.1 Detailed Description

```

template<typename InputType = int32>
class FPhraseInputNode< InputType >

```

Definition at line 12 of file [PhraseInputNode.h](#).

## 4.24.2 Constructor & Destructor Documentation

### 4.24.2.1 FPhraseInputNode() [1/5]

```
template<typename InputType >
FPhraseInputNode< InputType >::FPhraseInputNode (
    const TCHAR * InInputString )
```

Definition at line 10 of file [PhraseInputNode.cpp](#).

```
00011      : FPhraseNode(InInputString)
00012 {
00013
00014 }
```

### 4.24.2.2 FPhraseInputNode() [2/5]

```
template<typename InputType >
FPhraseInputNode< InputType >::FPhraseInputNode (
    const TCHAR * InInputString,
    TPhraseNodeArray InChildNodes )
```

Definition at line 17 of file [PhraseInputNode.cpp](#).

```
00018      : FPhraseNode(InInputString, InChildNodes)
00019 {
00020
00021 }
```

### 4.24.2.3 FPhraseInputNode() [3/5]

```
template<typename InputType >
FPhraseInputNode< InputType >::FPhraseInputNode (
    const TCHAR * InInputString,
    TDelegate< void(FParseRecord &Record)> InOnPhraseParsed,
    TPhraseNodeArray InChildNodes )
```

Definition at line 24 of file [PhraseInputNode.cpp](#).

```
00025      : FPhraseNode(InInputString, InOnPhraseParsed, InChildNodes)
00026 {
00027
00028 }
```

#### 4.24.2.4 FPhraseInputNode() [4/5]

```
template<typename InputType >
FPhraseInputNode< InputType >::FPhraseInputNode (
    const TCHAR * InInputString,
    TPhraseNodeArray InChildNodes,
    TDelegate< void(InputType Input)> InOnInputRecieved )
```

Definition at line 31 of file [PhraseInputNode.cpp](#).

```
00032 : FPhraseNode(InInputString, InChildNodes)
00033 {
00034     OnInputReceived = InOnInputRecieved;
00035 }
```

#### 4.24.2.5 FPhraseInputNode() [5/5]

```
template<typename InputType >
FPhraseInputNode< InputType >::FPhraseInputNode (
    const TCHAR * InInputString,
    TDelegate< void(FParseRecord &Record)> InOnPhraseParsed,
    TPhraseNodeArray InChildNodes,
    TDelegate< void(InputType Input)> InOnInputRecieved )
```

Definition at line 38 of file [PhraseInputNode.cpp](#).

```
00039 : FPhraseNode(InInputString, InOnPhraseParsed, InChildNodes)
00040 {
00041     OnInputReceived = InOnInputRecieved;
00042 }
```

#### 4.24.2.6 ~FPhraseInputNode()

```
template<typename InputType >
FPhraseInputNode< InputType >::~~FPhraseInputNode [virtual]
```

Definition at line 45 of file [PhraseInputNode.cpp](#).

```
00046 {
00047
00048 }
```

### 4.24.3 Member Function Documentation

#### 4.24.3.1 MeetsInputRequirements()

```
template<typename InputType >
bool FPhraseInputNode< InputType >::MeetsInputRequirements (
    const FString & InPhrase ) [protected], [virtual]
```

Checks if the Given Phrase Meets Requirements for usage as Input. In Correlation to this Nodes Input Specifications.

## Parameters

|                 |   |
|-----------------|---|
| <i>InPhrase</i> | - The Phrase To Check If It Meets Requirements. |
|-----------------|---|

## Returns

True, if the Phrase Meets Requirements. Otherwise False.

Reimplemented in [FPhraseEnumInputNode< TEnum >](#), [FPhraseEnumInputNode< EPhrase2DDirectionalInput >](#), [FPhraseEnumInputNode< EPhraseDirectionalInput >](#), [FPhraseEnumInputNode< EPhrasePositionalInput >](#), [FPhraseEnumInputNode< EPhraseScrollInput >](#), and [FPhraseStringInputNode](#).

Definition at line 104 of file [PhraseInputNode.cpp](#).

```
00105 {
00106     return InPhrase.IsNumeric() || NumericParser::IsValidNumeric(InPhrase, false);
00107 }
```

## 4.24.3.2 ParsePhrase()

```
template<typename InputType >
FParseResult FPhraseInputNode< InputType >::ParsePhrase (
    TArray< FString > & InPhraseWordArray,
    FParseRecord & InParseRecord ) [override], [virtual]
```

Parses The Phrase Down This Node, Propagating Down Any Child Nodes If Required.

## Parameters

|                          |  |
|--------------------------|--|
| <i>InPhraseWordArray</i> | - The Current Array of Transcription Phrases.  |
| <i>InParseRecord</i>     | - The Parse Record of the Current Propagation. |

## Returns

The Result of the Parsing of the Phrase, and any Propagation.

Reimplemented from [FPhraseNode](#).

Definition at line 66 of file [PhraseInputNode.cpp](#).

```
00067 {
00068     if (InPhraseArray.Num() == 0)
00069     {
00070         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Emptied Phrase Array ||"))
00071     }
00072     return FParseResult(PHASE_REQUIRES_MORE, AsShared());
00073 }
00074
00075 if (MeetsInputRequirements(InPhraseArray.Last()))
00076 {
00077     // Get the Input String.
00078     FString InputToRecord = InPhraseArray.Pop();
00079
00080     // Append the Input String to the Record.
00081     InParseRecord.AddPhraseString(InputToRecord);
00082
00083     if (!InputToRecord.IsNumeric() && NumericParser::IsValidNumeric(InputToRecord, false))
00084     {
00085         NumericParser::StringToNumeric(InputToRecord, false);
00086     }
00087 }
```

```

00086     }
00087
00088     if (!RecordInput(InputToRecord, InParseRecord))
00089     {
00090         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Unable to Record Input ||"))
00091
00092         return FParseResult(PHRASE_UNABLE_TO_PARSE, AsShared());
00093     }
00094
00095     OnPhraseParsed.ExecuteIfBound(InParseRecord);
00096
00097     return ParseChildren(InPhraseArray, InParseRecord);
00098 }
00099
00100 return FParseResult(PHRASE_UNABLE_TO_PARSE, AsShared());
00101 }

```

#### 4.24.3.3 RecordInput()

```

template<typename InputType >
bool FPhraseInputNode< InputType >::RecordInput (
    const FString & InInput,
    FParseRecord & OutParseRecord ) [protected], [virtual]

```

Records the Input onto the Parse Record.

##### Parameters

|                       |   |
|-----------------------|---|
| <i>InInput</i>        | - The Phrase To Record onto the Parse Record. |
| <i>OutParseRecord</i> | - Returns the Updated ParseRecord.            |

##### Returns

True, if the Input Was Successful in Recording. Otherwise False.

Reimplemented in [FPhraseEnumInputNode< TEnum >](#), [FPhraseEnumInputNode< EPhrase2DDirectionalInput >](#), [FPhraseEnumInputNode< EPhraseDirectionalInput >](#), [FPhraseEnumInputNode< EPhrasePositionalInput >](#), [FPhraseEnumInputNode< EPhraseScrollInput >](#), and [FPhraseStringInputNode](#).

Definition at line 110 of file [PhrasInputNode.cpp](#).

```

00111 {
00112     return false;
00113 }

```

#### 4.24.3.4 RequiresPhrase() [1/2]

```

template<typename InputType >
bool FPhraseInputNode< InputType >::RequiresPhrase (
    const FString InPhrase ) [override], [virtual]

```

Checks if the Node Requires the Given Phrase.



**Parameters**

|                 |  |
|-----------------|--|
| <i>InPhrase</i> | - The Phrase To Check if Required By The Node. |
|-----------------|--|

**Returns**

True, if the Phrase is Required. Otherwise False.

Reimplemented from [FPhraseNode](#).

Definition at line 51 of file [PhraseInputNode.cpp](#).

```
00052 {  
00053     return MeetsInputRequirements (InPhrase);  
00054 }
```

**4.24.3.5 RequiresPhrase() [2/2]**

```
template<typename InputType >  
bool FPhraseInputNode< InputType >::RequiresPhrase (  
    const FString InPhrase,  
    int32 & OutDistance ) [override], [virtual]
```

Checks if the Node Requires the Given Phrase, and Returns the Distance of the Phrase.

**Parameters**

|                    |  |
|--------------------|--|
| <i>InPhrase</i>    | - The Phrase To Check if Required By The Node.                       |
| <i>OutDistance</i> | - The Returned Distancing from the Target Phrase To The BoundPhrase. |

**Returns**

True, if the Phrase is Required. Otherwise False.

Reimplemented from [FPhraseNode](#).

Definition at line 57 of file [PhraseInputNode.cpp](#).

```
00058 {  
00059     bool bMeetsRequirements = MeetsInputRequirements (InPhrase);  
00060     OutDistance = bMeetsRequirements ? 0 : INT32_MAX;  
00061  
00062     return bMeetsRequirements;  
00063 }
```

**4.24.4 Member Data Documentation**



## Public Attributes

- TWeakPtr< [FPhraseNode](#) > [ParentNode](#)  
*This Nodes Parent Node.*
- TPhraseNodeArray [ChildNodes](#)  
*The Child Nodes of the Node.*
- FString [BoundPhrase](#)  
*The Phrase Bound to this*
- TDelegate< void([FParseRecord](#) &Record)> [OnPhraseParsed](#)

## Protected Member Functions

- bool [HasLeafChild](#) ()  
*Filters through the children, to check if it contains a Leaf Child.*

## Protected Attributes

- bool [bHasLeafChild](#)  
*Records if the Node has a Leaf Child.*

### 4.25.1 Detailed Description

Definition at line 54 of file [PhraseNode.h](#).

### 4.25.2 Constructor & Destructor Documentation

#### 4.25.2.1 FPhraseNode() [1/4]

```
FPhraseNode::FPhraseNode (
    const TCHAR * InBoundPhrase )
```

Definition at line 9 of file [PhraseNode.cpp](#).

```
00010 {
00011     BoundPhrase = InBoundPhrase;
00012     BoundPhrase.ToUpperInline();
00013
00014     ChildNodes = TArray<TSharedPtr<FPhraseNode>>();
00015 }
```

**4.25.2.2 FPhraseNode() [2/4]**

```
FPhraseNode::FPhraseNode (
    const TCHAR * InBoundPhrase,
    TDelegate< void(FParseRecord &Record)> InOnPhraseParsed )
```

Definition at line 17 of file [PhraseNode.cpp](#).

```
00018 {
00019     BoundPhrase = InBoundPhrase;
00020     BoundPhrase.ToUpperInline();
00021
00022     OnPhraseParsed = InOnPhraseParsed;
00023     ChildNodes = TArray<TSharedPtr<FPhraseNode>>();
00024 }
```

**4.25.2.3 FPhraseNode() [3/4]**

```
FPhraseNode::FPhraseNode (
    const TCHAR * InBoundPhrase,
    TPhraseNodeArray InChildNodes )
```

Definition at line 26 of file [PhraseNode.cpp](#).

```
00027 {
00028     BoundPhrase = InBoundPhrase;
00029     BoundPhrase.ToUpperInline();
00030
00031     ChildNodes = InChildNodes;
00032 }
```

**4.25.2.4 FPhraseNode() [4/4]**

```
FPhraseNode::FPhraseNode (
    const TCHAR * InBoundPhrase,
    TDelegate< void(FParseRecord &Record)> InOnPhraseParsed,
    TPhraseNodeArray InChildNodes )
```

Definition at line 34 of file [PhraseNode.cpp](#).

```
00035 {
00036     BoundPhrase = InBoundPhrase;
00037     BoundPhrase.ToUpperInline();
00038
00039     OnPhraseParsed = InOnPhraseParsed;
00040     ChildNodes = InChildNodes;
00041 }
```

**4.25.2.5 ~FPhraseNode()**

```
FPhraseNode::~FPhraseNode ( ) [virtual]
```

Definition at line 43 of file [PhraseNode.cpp](#).

```
00044 {
00045
00046 }
```

### 4.25.3 Member Function Documentation

#### 4.25.3.1 BindChildNode()

```
bool FPhraseNode::BindChildNode (
    TPhraseNode InNode )
```

Binds the Given Node as a Child Node.

##### Parameters

|               |   |
|---------------|---|
| <i>InNode</i> | - The Node To Bind as a Child of This Node. |
|---------------|---|

##### Returns

True, if the Node was Successfully Bound. Otherwise False.

Definition at line 124 of file [PhraseNode.cpp](#).

```
00125 {
00126     if (!InNode.IsValid())
00127         return false;
00128
00129     for (auto& ChildNode : ChildNodes)
00130     {
00131         if (ChildNode->RequiresPhrase (InNode->BoundPhrase))
00132         {
00133             return ChildNode->BindChildrenNodes (InNode->ChildNodes);
00134         }
00135         else
00136         {
00137             ChildNodes.AddUnique (ChildNode);
00138             return true;
00139         }
00140     }
00141
00142     return false;
00143 }
```

#### 4.25.3.2 BindChildNodeForce()

```
bool FPhraseNode::BindChildNodeForce (
    TPhraseNode InNode )
```

Forcefully Binds the Given Node as a Child, performing no checks.

##### Parameters

|               |                                      |
|---------------|--------------------------------------|
| <i>InNode</i> | - The Node To Force Bind as a Child. |
|---------------|--------------------------------------|

##### Returns

True, if the Node was Successfully Bound. Otherwise False.

Definition at line 145 of file [PhraseNode.cpp](#).

```
00146 {
00147     ChildNodes.AddUnique(InNode);
00148
00149     return true;
00150 }
```

#### 4.25.3.3 BindChildrenNodes()

```
bool FPhraseNode::BindChildrenNodes (
    TPhraseNodeArray InNodes )
```

Binds an Array of Nodes as Children of this Node.

##### Parameters

|                |   |
|----------------|---|
| <i>InNodes</i> | - The Array of Nodes To Bind as Children. |
|----------------|---|

##### Returns

True, if the Nodes were Successfully Bound. Otherwise False.

Definition at line 152 of file [PhraseNode.cpp](#).

```
00153 {
00154     for (auto& InNode : InNodes)
00155     {
00156         for (auto& ChildNode : ChildNodes)
00157         {
00158             if (ChildNode->RequiresPhrase(InNode->BoundPhrase))
00159             {
00160                 return ChildNode->BindChildrenNodes(InNode->ChildNodes);
00161             }
00162             else
00163             {
00164                 ChildNodes.AddUnique(ChildNode);
00165                 return true;
00166             }
00167         }
00168     }
00169
00170     return false;
00171 }
```

#### 4.25.3.4 BindChildrenNodesForce()

```
bool FPhraseNode::BindChildrenNodesForce (
    TPhraseNodeArray InNodes )
```

Forcefully Binds an Array of Nodes as Children of this Node, performing no checks.

##### Parameters

|                |   |
|----------------|---|
| <i>InNodes</i> | - The Array of Nodes To Bind sa Children. |
|----------------|---|

**Returns**

True, if the Nodes were successfully bound. Otherwise False.

Definition at line 173 of file [PhraseNode.cpp](#).

```
00174 {
00175     for (auto& InNode : InNodes)
00176     {
00177         ChildNodes.AddUnique(InNode);
00178     }
00179
00180     return true;
00181 }
```

**4.25.3.5 CanBindChild()**

```
bool FPhraseNode::CanBindChild (
    TPhraseNode & InNode )
```

Checks if the Given Node Can Be Bound as a Child Node.

**Parameters**

|               |   |
|---------------|---|
| <i>InNode</i> | - The Node To Check If It Can Be Bound. |
|---------------|---|

**Returns**

True, if the Node Can Be Bound as a Child. Otherwise False.

Definition at line 111 of file [PhraseNode.cpp](#).

```
00112 {
00113     for (auto& ChildNode : ChildNodes)
00114     {
00115         if (ChildNode->RequiresPhrase(InNode->BoundPhrase) || ChildNode->IsLeafNode())
00116         {
00117             return false;
00118         }
00119     }
00120
00121     return true;
00122 }
```

**4.25.3.6 HasLeafChild() [1/2]**

```
bool FPhraseNode::HasLeafChild ( ) [protected]
```

Filters through the children, to check if it contains a Leaf Child.

Definition at line 183 of file [PhraseNode.cpp](#).

```
00184 {
00185     return ChildNodes.Num() == 1 && ChildNodes[0]->IsLeafNode();
00186 }
```

#### 4.25.3.7 HasLeafChild() [2/2]

```
bool FPhraseNode::HasLeafChild ( ) const [virtual]
```

Definition at line 48 of file [PhraseNode.cpp](#).

```
00049 {
00050     return bHasLeafChild;
00051 }
```

#### 4.25.3.8 IsLeafNode()

```
virtual bool FPhraseNode::IsLeafNode ( ) const [inline], [virtual]
```

Checks if the Node is a Leaf Node.

##### Returns

True, if the Node is a Leaf Node. Otherwise False.

Reimplemented in [FPhraseEventNode](#).

Definition at line 69 of file [PhraseNode.h](#).

```
00069 { return false; }
```

#### 4.25.3.9 ParseChildren()

```
FParseResult FPhraseNode::ParseChildren (
    TArray< FString > & InPhraseArray,
    FParseRecord & InParseRecord ) [virtual]
```

Parses The Children Node of this Node.

##### Parameters

|                      |  |
|----------------------|--|
| <i>InPhraseArray</i> | - The Current Array of Transcription Phrases.  |
| <i>InParseRecord</i> | - The Parse Record of the Current Propagation. |

##### Returns

The Result of the Parsing of the Phrase, and any Propagation.

Definition at line 188 of file [PhraseNode.cpp](#).

```
00189 {
00190     if (HasLeafChild())
00191         return ChildNodes[0]->ParsePhrase(InPhraseArray, InParseRecord);
00192     if (InPhraseArray.IsEmpty())
00193         return FParseResult(PHRASE_REQUIRES_MORE, AsShared());
00194
00195     // Below Can Be Optimized.
00196     // Maybe bypass the loop if Distance == 0 and Sort ChildNodes with Derived PhraseNodes Last?
00197 }
```



```

00198     int FoundChildIndex = -1;
00199     {
00200         int32 FoundChildDistance = INT32_MAX, CurrentDistance = INT32_MAX;
00201
00202         for (int i = 0; i < ChildNodes.Num(); i++)
00203         {
00204             // Child Nodes Require Unique Phrases to Siblings.
00205             if (ChildNodes[i]->RequiresPhrase(InPhraseArray.Last(), CurrentDistance))
00206             {
00207                 if (FoundChildDistance > CurrentDistance)
00208                 {
00209                     FoundChildIndex = i;
00210                     FoundChildDistance = CurrentDistance;
00211                 }
00212             }
00213         }
00214     }
00215
00216     if (FoundChildIndex != -1)
00217     {
00218         return ChildNodes[FoundChildIndex]->ParsePhrase(InPhraseArray, InParseRecord);
00219     }
00220
00221     /*else if (!InPhraseArray.IsEmpty())
00222     {
00223         return FParseResult(PHRASE_REQUIRES_MORE_CORRECT_PHRASES, AsShared());
00224     }*/
00225
00226     return FParseResult(PHRASE_UNABLE_TO_PARSE, AsShared());
00227 }

```

#### 4.25.3.10 ParsePhrase()

```

FParseResult FPhraseNode::ParsePhrase (
    TArray< FString > & InPhraseWordArray,
    FParseRecord & InParseRecord ) [virtual]

```

Parses The Phrase Down This Node, Propagating Down Any Child Nodes If Required.

##### Parameters

|                          |  |
|--------------------------|--|
| <i>InPhraseWordArray</i> | - The Current Array of Transcription Phrases.  |
| <i>InParseRecord</i>     | - The Parse Record of the Current Propagation. |

##### Returns

The Result of the Parsing of the Phrase, and any Propagation.

Reimplemented in [FPhraseEventNode](#), [FPhrasInputNode< InputType >](#), [FPhrasInputNode< int32 >](#), [FPhrasInputNode< FString >](#), [FPhraseTree](#), [FPhraseContextMenuNode< ContextMenuType >](#), and [FPhraseContextNode< ContextMenuType >](#).

Definition at line 65 of file [PhraseNode.cpp](#).

```

00066     {
00067         if (InPhraseArray.IsEmpty())
00068         {
00069             UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Emptied Phrase Array ||"))
00070
00071             return FParseResult(PHRASE_REQUIRES_MORE, AsShared());
00072         }
00073
00074         // Pop the Phrase Linked to this Node.
00075         // Apply to the Record.
00076         FString LinkedPhrase = InPhraseArray.Pop();
00077
00078         // Append Removed Phrase To Record.
00079         InParseRecord.AddPhraseString(LinkedPhrase);
00080     }

```

```

00081     OnPhraseParsed.ExecuteIfBound(InParseRecord);
00082
00083     // Pass
00084     return ParseChildren(InPhraseArray, InParseRecord);
00085 }

```

#### 4.25.3.11 ParsePhraseAsContext()

```

FParseResult FPhraseNode::ParsePhraseAsContext (
    TArray< FString > & InPhraseWordArray,
    FParseRecord & InParseRecord ) [virtual]

```

Parses the Phrase Down This Node, Propagating Down Any Child Nodes If Required. Does not Pop the Phrase Array.

##### Parameters

|                          |  |
|--------------------------|--|
| <i>InPhraseWordArray</i> | - The Current Array of Transcription Phrases.  |
| <i>InParseRecord</i>     | - The Parse Record of the Current Propagation. |

##### Returns

The Result of the Parsing of the Phrase, and any Propagation.

Reimplemented in [FPhraseContextMenuNode< ContextMenuType >](#), and [FPhraseContextNode< ContextType >](#).

Definition at line 87 of file [PhraseNode.cpp](#).

```

00088 {
00089     if (InPhraseWordArray.IsEmpty())
00090     {
00091         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Emptied Phrase Array ||"))
00092
00093         return FParseResult(PHRASE_REQUIRES_MORE, AsShared());
00094     }
00095
00096     OnPhraseParsed.ExecuteIfBound(InParseRecord);
00097
00098     return ParseChildren(InPhraseWordArray, InParseRecord);
00099 }

```

#### 4.25.3.12 ParsePhraselfRequired()

```

FParseResult FPhraseNode::ParsePhraseIfRequired (
    TArray< FString > & InPhraseWordArray,
    FParseRecord & InParseRecord ) [virtual]

```

If the Phrase If Required, Parses the Phrase Down This Node, Propagating Down Any Child Nodes If Required.

Definition at line 101 of file [PhraseNode.cpp](#).

```

00102 {
00103     if (RequiresPhrase(InPhraseWordArray.Last()))
00104     {
00105         return ParsePhrase(InPhraseWordArray, InParseRecord);
00106     }
00107
00108     return FParseResult(PHRASE_UNABLE_TO_PARSE);
00109 }

```

**4.25.3.13 RequiresPhrase()** [1/2]

```
bool FPhraseNode::RequiresPhrase (
    const FString InPhrase ) [virtual]
```

Checks if the Node Requires the Given Phrase.

**Parameters**

|                 |  |
|-----------------|--|
| <i>InPhrase</i> | - The Phrase To Check if Required By The Node. |
|-----------------|--|

**Returns**

True, if the Phrase is Required. Otherwise False.

Reimplemented in [FPhraseEventNode](#), [FPhraseInputNode< InputType >](#), [FPhraseInputNode< int32 >](#), and [FPhraseInputNode< FString >](#).

Definition at line 53 of file [PhraseNode.cpp](#).

```
00054 {
00055     return InPhrase.Equals(BoundPhrase, ESearchCase::IgnoreCase) ||
        Algo::LevenshteinDistance(BoundPhrase, InPhrase) < 3;
00056 }
```

**4.25.3.14 RequiresPhrase()** [2/2]

```
bool FPhraseNode::RequiresPhrase (
    const FString InPhrase,
    int32 & OutDistance ) [virtual]
```

Checks if the Node Requires the Given Phrase, and Returns the Distance of the Phrase.

**Parameters**

|                    |  |
|--------------------|--|
| <i>InPhrase</i>    | - The Phrase To Check if Required By The Node.                       |
| <i>OutDistance</i> | - The Returned Distancing from the Target Phrase To The BoundPhrase. |

**Returns**

True, if the Phrase is Required. Otherwise False.

Reimplemented in [FPhraseEventNode](#), [FPhraseInputNode< InputType >](#), [FPhraseInputNode< int32 >](#), and [FPhraseInputNode< FString >](#).

Definition at line 58 of file [PhraseNode.cpp](#).

```
00059 {
00060     OutDistance = Algo::LevenshteinDistance(BoundPhrase, InPhrase);
00061
00062     return InPhrase.Equals(BoundPhrase, ESearchCase::IgnoreCase) || OutDistance < 3;
00063 }
```

## 4.25.4 Member Data Documentation

### 4.25.4.1 bHasLeafChild

```
bool FPhraseNode::bHasLeafChild [protected]
```

Records if the Node has a Leaf Child.

Definition at line 185 of file [PhraseNode.h](#).

### 4.25.4.2 BoundPhrase

```
FString FPhraseNode::BoundPhrase
```

The Phrase Bound to this

Definition at line 175 of file [PhraseNode.h](#).

### 4.25.4.3 ChildNodes

```
TPhraseNodeArray FPhraseNode::ChildNodes
```

The Child Nodes of the Node.

Definition at line 170 of file [PhraseNode.h](#).

### 4.25.4.4 OnPhraseParsed

```
TDelegate<void (FParseRecord& Record)> FPhraseNode::OnPhraseParsed
```

Definition at line 178 of file [PhraseNode.h](#).

### 4.25.4.5 ParentNode

```
TWeakPtr<FPhraseNode> FPhraseNode::ParentNode
```

This Nodes Parent Node.

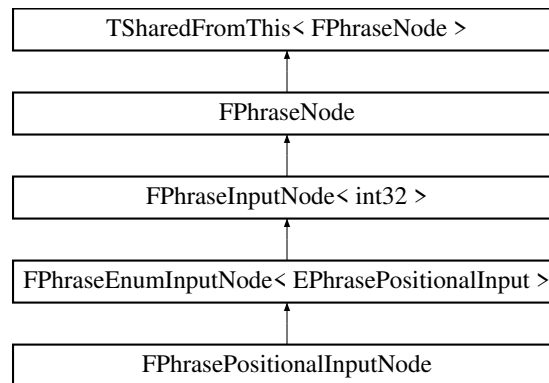
Definition at line 165 of file [PhraseNode.h](#).

The documentation for this class was generated from the following files:

- Source/OpenAccessibilityCommunication/Public/PhraseTree/PhraseNode.h
- Source/OpenAccessibilityCommunication/Private/PhraseTree/PhraseNode.cpp

## 4.26 FPhrasePositionalInputNode Class Reference

Inheritance diagram for FPhrasePositionalInputNode:



### Public Member Functions

- [FPhrasePositionalInputNode](#) (const TCHAR \*NodeName)
- [FPhrasePositionalInputNode](#) (const TCHAR \*NodeName, TPhraseNodeArray InChildNodes)
- [FPhrasePositionalInputNode](#) (const TCHAR \*NodeName, TDelegate< void([FParseRecord](#) &Record)> InOnPhraseParsed, TPhraseNodeArray InChildNodes)
- [FPhrasePositionalInputNode](#) (const TCHAR \*NodeName, TPhraseNodeArray InChildNodes, TDelegate< void(int32 Input)> InOnInputRecieved)
- [FPhrasePositionalInputNode](#) (const TCHAR \*NodeName, TDelegate< void([FParseRecord](#) &Record)> InOnPhraseParsed, TPhraseNodeArray InChildNodes, TDelegate< void(int32 Input)> InOnInputRecieved)

### Additional Inherited Members

#### 4.26.1 Detailed Description

Definition at line 80 of file [PhraseDirectionalInputNode.h](#).

#### 4.26.2 Constructor & Destructor Documentation

##### 4.26.2.1 FPhrasePositionalInputNode() [1/5]

```
FPhrasePositionalInputNode::FPhrasePositionalInputNode (
    const TCHAR * NodeName ) [inline]
```

Definition at line 83 of file [PhraseDirectionalInputNode.h](#).

```
00084 : FPhraseEnumInputNode<EPhrasePositionalInput> (NodeName)
00085 { }
```

**4.26.2.2 FPhrasePositionalInputNode() [2/5]**

```
FPhrasePositionalInputNode::FPhrasePositionalInputNode (
    const TCHAR * NodeName,
    TPhraseNodeArray InChildNodes ) [inline]
```

Definition at line 87 of file [PhraseDirectionalInputNode.h](#).

```
00088      : FPhraseEnumInputNode<EPhrasePositionalInput>(NodeName, InChildNodes)
00089      {}
```

**4.26.2.3 FPhrasePositionalInputNode() [3/5]**

```
FPhrasePositionalInputNode::FPhrasePositionalInputNode (
    const TCHAR * NodeName,
    TDelegate< void(FParseRecord &Record)> InOnPhraseParsed,
    TPhraseNodeArray InChildNodes ) [inline]
```

Definition at line 91 of file [PhraseDirectionalInputNode.h](#).

```
00092      : FPhraseEnumInputNode<EPhrasePositionalInput>(NodeName, InOnPhraseParsed, InChildNodes)
00093      {}
```

**4.26.2.4 FPhrasePositionalInputNode() [4/5]**

```
FPhrasePositionalInputNode::FPhrasePositionalInputNode (
    const TCHAR * NodeName,
    TPhraseNodeArray InChildNodes,
    TDelegate< void(int32 Input)> InOnInputRecieved ) [inline]
```

Definition at line 95 of file [PhraseDirectionalInputNode.h](#).

```
00096      : FPhraseEnumInputNode<EPhrasePositionalInput>(NodeName, InChildNodes, InOnInputRecieved)
00097      {}
```

**4.26.2.5 FPhrasePositionalInputNode() [5/5]**

```
FPhrasePositionalInputNode::FPhrasePositionalInputNode (
    const TCHAR * NodeName,
    TDelegate< void(FParseRecord &Record)> InOnPhraseParsed,
    TPhraseNodeArray InChildNodes,
    TDelegate< void(int32 Input)> InOnInputRecieved ) [inline]
```

Definition at line 99 of file [PhraseDirectionalInputNode.h](#).

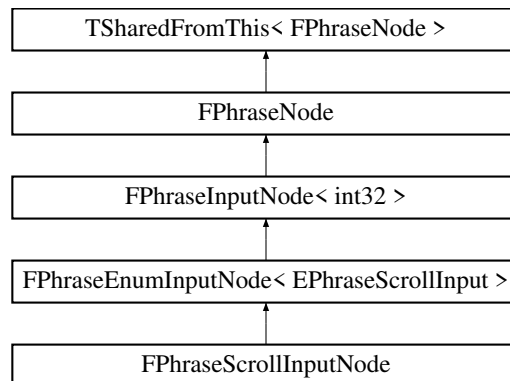
```
00100      : FPhraseEnumInputNode<EPhrasePositionalInput>(NodeName, InOnPhraseParsed, InChildNodes,
00101      InOnInputRecieved)
00101      {}
```

The documentation for this class was generated from the following file:

- [Source/OpenAccessibilityCommunication/Public/PhraseTree/PhraseDirectionalInputNode.h](#)

## 4.27 FPhraseScrollInputNode Class Reference

Inheritance diagram for FPhraseScrollInputNode:



### Public Member Functions

- [FPhraseScrollInputNode](#) (const TCHAR \*NodeName)
- [FPhraseScrollInputNode](#) (const TCHAR \*NodeName, TPhraseNodeArray InChildNodes)
- [FPhraseScrollInputNode](#) (const TCHAR \*NodeName, TDelegate< void([FParseRecord](#) &Record)> InOnPhraseParsed, TPhraseNodeArray InChildNodes)
- [FPhraseScrollInputNode](#) (const TCHAR \*NodeName, TPhraseNodeArray InChildNodes, TDelegate< void(int32 Input)> InOnInputRecieved)
- [FPhraseScrollInputNode](#) (const TCHAR \*NodeName, TDelegate< void([FParseRecord](#) &Record)> InOnPhraseParsed, TPhraseNodeArray InChildNodes, TDelegate< void(int32 Input)> InOnInputRecieved)

### Additional Inherited Members

#### 4.27.1 Detailed Description

Definition at line 56 of file [PhraseDirectionalInputNode.h](#).

#### 4.27.2 Constructor & Destructor Documentation

##### 4.27.2.1 FPhraseScrollInputNode() [1/5]

```
FPhraseScrollInputNode::FPhraseScrollInputNode (
    const TCHAR * NodeName ) [inline]
```

Definition at line 59 of file [PhraseDirectionalInputNode.h](#).

```
00060     : FPhraseEnumInputNode<EPhraseScrollInput> (NodeName)
00061     {}
```

**4.27.2.2 FPhraseScrollInputNode() [2/5]**

```
FPhraseScrollInputNode::FPhraseScrollInputNode (
    const TCHAR * NodeName,
    TPhraseNodeArray InChildNodes ) [inline]
```

Definition at line 63 of file [PhraseDirectionalInputNode.h](#).

```
00064      : FPhraseEnumInputNode<EPhraseScrollInput>(NodeName, InChildNodes)
00065      {}
```

**4.27.2.3 FPhraseScrollInputNode() [3/5]**

```
FPhraseScrollInputNode::FPhraseScrollInputNode (
    const TCHAR * NodeName,
    TDelegate< void(FParseRecord &Record)> InOnPhraseParsed,
    TPhraseNodeArray InChildNodes ) [inline]
```

Definition at line 67 of file [PhraseDirectionalInputNode.h](#).

```
00068      : FPhraseEnumInputNode<EPhraseScrollInput>(NodeName, InOnPhraseParsed, InChildNodes)
00069      {}
```

**4.27.2.4 FPhraseScrollInputNode() [4/5]**

```
FPhraseScrollInputNode::FPhraseScrollInputNode (
    const TCHAR * NodeName,
    TPhraseNodeArray InChildNodes,
    TDelegate< void(int32 Input)> InOnInputRecieved ) [inline]
```

Definition at line 71 of file [PhraseDirectionalInputNode.h](#).

```
00072      : FPhraseEnumInputNode<EPhraseScrollInput>(NodeName, InChildNodes, InOnInputRecieved)
00073      {}
```

**4.27.2.5 FPhraseScrollInputNode() [5/5]**

```
FPhraseScrollInputNode::FPhraseScrollInputNode (
    const TCHAR * NodeName,
    TDelegate< void(FParseRecord &Record)> InOnPhraseParsed,
    TPhraseNodeArray InChildNodes,
    TDelegate< void(int32 Input)> InOnInputRecieved ) [inline]
```

Definition at line 75 of file [PhraseDirectionalInputNode.h](#).

```
00076      : FPhraseEnumInputNode<EPhraseScrollInput>(NodeName, InOnPhraseParsed, InChildNodes,
    InOnInputRecieved)
00077      {}
```

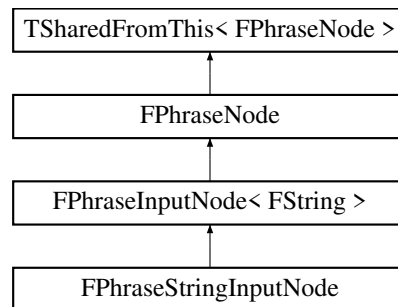
The documentation for this class was generated from the following file:

- [Source/OpenAccessibilityCommunication/Public/PhraseTree/PhraseDirectionalInputNode.h](#)



## 4.28 FPhraseStringInputNode Class Reference

Inheritance diagram for FPhraseStringInputNode:



### Public Member Functions

- [FPhraseStringInputNode](#) (const TCHAR \*InInputString)
- [FPhraseStringInputNode](#) (const TCHAR \*InInputString, TPhraseNodeArray InChildNodes)
- [FPhraseStringInputNode](#) (const TCHAR \*InInputString, TDelegate< void([FParseRecord](#) &Record)> InOnPhraseParsed, TPhraseNodeArray InChildNodes)
- [FPhraseStringInputNode](#) (const TCHAR \*InInputString, TPhraseNodeArray InChildNodes, TDelegate< void(FString Input)> InOnInputRecieved)
- **FPhraseStringInputNode** (const TCHAR \*InInputString, TDelegate< void([FParseRecord](#) &Record)> InOnPhraseParsed, TPhraseNodeArray InChildNodes, TDelegate< void(FString Input)> InOnInputRecieved)

### Protected Member Functions

- virtual bool [MeetsInputRequirements](#) (const FString &InPhrase) override  
*Checks if the Given Phrase Meets Requirements for usage as Input. In Correlation to this Nodes Input Specifications.*
- virtual bool [RecordInput](#) (const FString &InInput, [FParseRecord](#) &OutParseRecord) override  
*Records the Input onto the Parse Record.*

### Additional Inherited Members

#### 4.28.1 Detailed Description

Definition at line 11 of file [PhraseStringInputNode.h](#).

#### 4.28.2 Constructor & Destructor Documentation

**4.28.2.1 FPhraseStringInputNode() [1/4]**

```
FPhraseStringInputNode::FPhraseStringInputNode (
    const TCHAR * InInputString )
```

Definition at line 7 of file [PhraseStringInputNode.cpp](#).

```
00008      : FPhraseInputNode(InInputString)
00009 {
00010
00011 };
```

**4.28.2.2 FPhraseStringInputNode() [2/4]**

```
FPhraseStringInputNode::FPhraseStringInputNode (
    const TCHAR * InInputString,
    TPhraseNodeArray InChildNodes )
```

Definition at line 13 of file [PhraseStringInputNode.cpp](#).

```
00014      : FPhraseInputNode(InInputString, InChildNodes)
00015 {
00016
00017 }
```

**4.28.2.3 FPhraseStringInputNode() [3/4]**

```
FPhraseStringInputNode::FPhraseStringInputNode (
    const TCHAR * InInputString,
    TDelegate< void(FParseRecord &Record)> InOnPhraseParsed,
    TPhraseNodeArray InChildNodes )
```

Definition at line 19 of file [PhraseStringInputNode.cpp](#).

```
00020      : FPhraseInputNode(InInputString, InOnPhraseParse, InChildNodes)
00021 {
00022
00023 }
```

**4.28.2.4 FPhraseStringInputNode() [4/4]**

```
FPhraseStringInputNode::FPhraseStringInputNode (
    const TCHAR * InInputString,
    TPhraseNodeArray InChildNodes,
    TDelegate< void(FString Input)> InOnInputRecieved )
```

Definition at line 25 of file [PhraseStringInputNode.cpp](#).

```
00026      : FPhraseInputNode(InInputString, InChildNodes, InOnInputRecieved)
00027 {
00028
00029 }
```

#### 4.28.2.5 ~FPhraseStringInputNode()

```
FPhraseStringInputNode::~FPhraseStringInputNode ( ) [virtual]
```

Definition at line 31 of file [PhraseStringInputNode.cpp](#).

```
00032 {
00033
00034 }
```

### 4.28.3 Member Function Documentation

#### 4.28.3.1 MeetsInputRequirements()

```
bool FPhraseStringInputNode::MeetsInputRequirements (
    const FString & InPhrase ) [override], [protected], [virtual]
```

Checks if the Given Phrase Meets Requirements for usage as Input. In Correlation to this Nodes Input Specifications.

##### Parameters

|                 |   |
|-----------------|---|
| <i>InPhrase</i> | - The Phrase To Check If It Meets Requirements. |
|-----------------|---|

##### Returns

True, if the Phrase Meets Requirements. Otherwise False.

Reimplemented from [FPhraseInputNode< FString >](#).

Definition at line 36 of file [PhraseStringInputNode.cpp](#).

```
00037 {
00038     if (InPhrase.IsEmpty())
00039         return false;
00040     else return true;
00041 }
```

#### 4.28.3.2 RecordInput()

```
bool FPhraseStringInputNode::RecordInput (
    const FString & InInput,
    FParseRecord & OutParseRecord ) [override], [protected], [virtual]
```

Records the Input onto the Parse Record.

##### Parameters

|                       |   |
|-----------------------|---|
| <i>InInput</i>        | - The Phrase To Record onto the Parse Record. |
| <i>OutParseRecord</i> | - Returns the Updated ParseRecord.            |

**Returns**

True, if the Input Was Successful in Recording. Otherwise False.

Reimplemented from [FPhraseInputNode< FString >](#).

Definition at line 43 of file [PhraseStringInputNode.cpp](#).

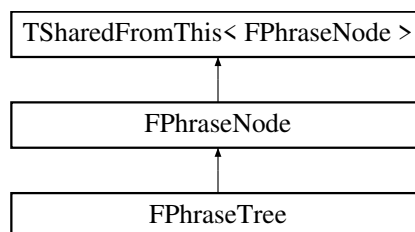
```
00044 {
00045     if (InInput.IsEmpty())
00046         return false;
00047
00048     UParseStringInput* ParseInput = MakeParseInput<UParseStringInput>();
00049     ParseInput->SetValue(InInput);
00050
00051     OutParseRecord.AddPhraseInput(BoundPhrase, ParseInput);
00052
00053     OnInputReceived.ExecuteIfBound(InInput);
00054
00055     return true;
00056 }
```

The documentation for this class was generated from the following files:

- Source/OpenAccessibilityCommunication/Public/PhraseTree/PhraseStringInputNode.h
- Source/OpenAccessibilityCommunication/Private/PhraseTree/PhraseStringInputNode.cpp

## 4.29 FPhraseTree Class Reference

Inheritance diagram for FPhraseTree:



### Public Member Functions

- [FPhraseTreeContextManager](#) & [GetContextManager](#) ()
- bool [Tick](#) (float DeltaTime)
- virtual [FParseResult](#) [ParsePhrase](#) (TArray< FString > &InPhraseWordArray, [FParseRecord](#) &InParseRecord) override  
*Parses The Phrase Down This Node, Propagating Down Any Child Nodes If Required.*
- void [BindBranch](#) (const TPhraseNode &InNode)  
*Bind a branch to the tree. Attaching to any overlapping nodes.*
- void [BindBranches](#) (const TPhraseNodeArray &InNodes)  
*Bind Multiple Branches to the Tree, that are not connected.*
- void [ParseTranscription](#) (TArray< FString > InTranscriptionSegments)  
*Parses and Propagates the given Transcription Segments down the tree.*

## Additional Inherited Members

### 4.29.1 Detailed Description

Definition at line 227 of file [PhraseTree.h](#).

### 4.29.2 Constructor & Destructor Documentation

#### 4.29.2.1 FPhraseTree()

```
FPhraseTree::FPhraseTree ( )
```

Definition at line 12 of file [PhraseTree.cpp](#).

```
00012             : FPhraseNode(TEXT("ROOT_NODE"))
00013 {
00014     ContextManager = FPhraseTreeContextManager();
00015
00016     FTickerDelegate TickDelegate = FTickerDelegate::CreateRaw(this, &FPhraseTree::Tick);
00017     TickDelegateHandle = FTSTicker::GetCoreTicker().AddTicker(TickDelegate);
00018 }
```

#### 4.29.2.2 ~FPhraseTree()

```
FPhraseTree::~FPhraseTree ( ) [virtual]
```

Definition at line 20 of file [PhraseTree.cpp](#).

```
00021 {
00022     FTSTicker::GetCoreTicker().RemoveTicker(TickDelegateHandle);
00023 }
```

### 4.29.3 Member Function Documentation

#### 4.29.3.1 BindBranch()

```
void FPhraseTree::BindBranch (
    const TPhraseNode & InNode )
```

Bind a branch to the tree. Attaching to any overlapping nodes.

##### Parameters

|               |   |
|---------------|---|
| <i>InNode</i> | The constructed branch to attach to the tree. |
|---------------|---|

Definition at line 182 of file [PhraseTree.cpp](#).

```

00183 {
00184     TArray<FPhraseTreeBranchBind> ToBindArray = TArray<FPhraseTreeBranchBind>();
00185
00186     ToBindArray.Add(FPhraseTreeBranchBind(AsShared(), InNode));
00187
00188     while (!ToBindArray.IsEmpty())
00189     {
00190         FPhraseTreeBranchBind BranchToBind = ToBindArray.Pop();
00191
00192         // Check all ChildNodes to see if they are similar in purpose.
00193         for (auto& ChildNode : BranchToBind.StartNode->ChildNodes)
00194         {
00195             // If a ChildNode meets the same requirements as the BranchRoot,
00196             // then Split Bind Process to the ChildNodes.
00197             if (ChildNode->RequiresPhrase(BranchToBind.BranchRoot->BoundPhrase))
00198             {
00199                 for (auto& BranchChildNode : BranchToBind.BranchRoot->ChildNodes)
00200                 {
00201                     ToBindArray.Add(FPhraseTreeBranchBind(ChildNode, BranchChildNode));
00202                 }
00203
00204                 continue;
00205             }
00206         }
00207
00208         // If the Start Node has no similar children, then bind the branch to the start node.
00209         // Can force bind, as previous checks show no child is similar.
00210         BranchToBind.StartNode->BindChildNodeForce(BranchToBind.BranchRoot);
00211     }
00212 }

```

#### 4.29.3.2 BindBranches()

```

void FPhraseTree::BindBranches (
    const TPhraseNodeArray & InNodes )

```

Bind Multiple Branches to the Tree, that are not connected.

Definition at line 214 of file [PhraseTree.cpp](#).

```

00215 {
00216     for (const TSharedPtr<FPhraseNode>& Node : InNodes)
00217     {
00218         BindBranch(Node);
00219     }
00220 }

```

#### 4.29.3.3 GetContextManager()

[FPhraseTreeContextManager](#) & FPhraseTree::GetContextManager ( ) [inline]

Definition at line 233 of file [PhraseTree.h](#).

```

00233                                     {
00234     return ContextManager;
00235 }

```

#### 4.29.3.4 ParsePhrase()

```

FParseResult FPhraseTree::ParsePhrase (
    TArray< FString > & InPhraseWordArray,
    FParseRecord & InParseRecord ) [override], [virtual]

```

Parses The Phrase Down This Node, Propagating Down Any Child Nodes If Required.

## Parameters

|                          |  |
|--------------------------|--|
| <i>InPhraseWordArray</i> | - The Current Array of Transcription Phrases.  |
| <i>InParseRecord</i>     | - The Parse Record of the Current Propagation. |

## Returns

The Result of the Parsing of the Phrase, and any Propagation.

Reimplemented from [FPhraseNode](#).

Definition at line 141 of file [PhraseTree.cpp](#).

```

00142 {
00143     if (InPhraseWordArray.IsEmpty())
00144     {
00145         UE_LOG(LogOpenAccessibilityCom, Warning, TEXT("|| Phrase Tree || Provided Transcription
Segment is Empty ||"));
00146
00147         return FParseResult(PHRASE_NOT_PARSED);
00148     }
00149
00150     // First give the last visited node a chance to parse the phrase.
00151     // due to the possibility of connecting phrases over different transcription segments.
00152     if (LastVistedNode != nullptr && LastVistedNode.IsValid())
00153     {
00154         TArray<FString> PhraseWordArrayCopy = TArray(InPhraseWordArray);
00155
00156         FParseResult ParseResult = LastVistedNode->ParseChildren(PhraseWordArrayCopy,
LastVistedParseRecord);
00157         if (ParseResult.Result == PHRASE_PARSED)
00158         {
00159             LastVistedNode.Reset();
00160             InParseRecord = LastVistedParseRecord;
00161             LastVistedParseRecord = FParseRecord();
00162
00163             return ParseResult;
00164         }
00165         else if (ParseResult.Result != PHRASE_UNABLE_TO_PARSE)
00166         {
00167             return ParseResult;
00168         }
00169     }
00170
00171     // Check if the Context Stack has Objects, if so propagation from the Context Root.
00172     if (ContextManager.HasContextObjects())
00173     {
00174         // Propagate from the Context Root, that is the Top of the Context Stack.
00175         return
ContextManager.PeekContextObject()->GetContextRoot()->ParsePhraseAsContext(InPhraseWordArray,
InParseRecord);
00176     }
00177
00178     // Otherwise, start a new propagation entirely from the Tree Root.
00179     return ParseChildren(InPhraseWordArray, InParseRecord);
00180 }

```

## 4.29.3.5 ParseTranscription()

```

void FPhraseTree::ParseTranscription (
    TArray< FString > InTranscriptionSegments )

```

Parses and Propogates the given Transcription Segments down the tree.

## Parameters

|                                |  |
|--------------------------------|--|
| <i>InTranscriptionSegments</i> |  |
|--------------------------------|--|

Definition at line 33 of file [PhraseTree.cpp](#).

```

00034 {
00035     if (InTranscriptionSegments.IsEmpty())
00036     {
00037         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Phrase Tree || Provided Transcription is Empty
||"))
00038         return;
00039     }
00040
00041     TArray<FString> SegmentWordArray = TArray<FString>();
00042     int SegmentCount = 0;
00043
00044     // Loop over any Transcription Segments.
00045     for (FString& TranscriptionSegment : InTranscriptionSegments)
00046     {
00047         if (TranscriptionSegment.IsEmpty())
00048         {
00049             UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Phrase Tree || Transcription Segment is
Empty ||"))
00050             continue;
00051         }
00052
00053         // Filter the Transcription Segment, to remove any unwanted characters.
00054         TranscriptionSegment.TrimStartAndEndInline();
00055         TranscriptionSegment.ReplaceInline(TEXT("."), TEXT(""), ESearchCase::IgnoreCase);
00056         TranscriptionSegment.ReplaceInline(TEXT(","), TEXT(""), ESearchCase::IgnoreCase);
00057         TranscriptionSegment.ToUpperInline();
00058
00059         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Phrase Tree || Filtered Transcription Segment: {
%s } ||"), *TranscriptionSegment)
00060
00061         // Parse the Transcription Segment into an Array of Words, removing any white space.
00062         TranscriptionSegment.ParseIntoArrayWS(SegmentWordArray);
00063         if (SegmentWordArray.Num() == 0)
00064         {
00065             UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Phrase Tree || Transcription Segment has no
Word Content ||"))
00066             continue;
00067         }
00068
00069         Algo::Reverse(SegmentWordArray);
00070
00071         // Loop until the Segment is Empty
00072         while (!SegmentWordArray.IsEmpty())
00073         {
00074
00075             FParseRecord ParseRecord = FParseRecord(ContextManager.GetContextStack());
00076             FParseResult ParseResult = ParsePhrase(SegmentWordArray, ParseRecord);
00077
00078             ContextManager.UpdateContextStack(ParseRecord.ContextObjectStack);
00079
00080             UE_LOGFMT(LogOpenAccessibilityCom, Log, "|| Phrase Tree || Segment: {0} | Result: {1} ||",
SegmentCount, ParseResult.Result);
00081
00082             switch (ParseResult.Result)
00083             {
00084                 case PHRASE_PARSED:
00085                 case PHRASE_PARSED_AND_EXECUTED:
00086                 {
00087                     OA_LOG(LogOpenAccessibilityCom, Log, TEXT("PhraseTree Propagation"),
TEXT("{Success} Phrase Tree Parsed Correctly (%s)"),
*ParseRecord.GetPhraseString())
00088
00089                     LastVistedNode.Reset();
00090                     LastVistedParseRecord = FParseRecord();
00091
00092                     break;
00093                 }
00094
00095                 case PHRASE_REQUIRES_MORE:
00096                 {
00097                     OA_LOG(LogOpenAccessibilityCom, Log, TEXT("PhraseTree Propagation"),
TEXT("{Failed} Phrase Tree Propagation Requires More Segments. (%s)"),
*ParseRecord.GetPhraseString());
00098
00099                     // Store Reach Nodes, and the ParseRecord for future propagation attempts.
00100                     LastVistedNode = ParseResult.ReachedNode;
00101                     LastVistedParseRecord = ParseRecord;
00102                 }
00103
00104                 case PHRASE_REQUIRES_MORE_CORRECT_PHRASES:
00105                 {
00106                     OA_LOG(LogOpenAccessibilityCom, Log, TEXT("PhraseTree Propagation"),
TEXT("{Failed} Phrase Tree Propagation Requires More Correct Segments. (%s)"),
*ParseRecord.GetPhraseString())
00107
00108                     LastVistedNode = ParseResult.ReachedNode;
00109
00110                     LastVistedNode = ParseResult.ReachedNode;
00111

```



```

00112             LastVistedParseRecord = ParseRecord;
00113
00114             // Dirty Way of Ensuring all Segments in Transcription are Attempted.
00115             if (!SegmentWordArray.IsEmpty())
00116                 SegmentWordArray.Pop();
00117
00118             break;
00119         }
00120
00121         default:
00122             case PHRASE_UNABLE_TO_PARSE:
00123             {
00124                 OA_LOG(LogOpenAccessibilityCom, Log, TEXT("PhraseTree Propagation"),
00125                     TEXT("{Failed} Phrase Tree Propagation Failed. (%s)"),
00126                     *ParseRecord.GetPhraseString())
00127
00128                 // Dirty Way of Ensuring all Segments in Transcription are Attempted.
00129                 if (!SegmentWordArray.IsEmpty())
00130                     SegmentWordArray.Pop();
00131
00132                 break;
00133             }
00134         }
00135     }
00136     SegmentCount++;
00137     SegmentWordArray.Reset();
00138 }
00139 }

```

#### 4.29.3.6 Tick()

```

bool FPhraseTree::Tick (
    float DeltaTime )

```

Definition at line 25 of file [PhraseTree.cpp](#).

```

00026 {
00027     // Filter InActive Context Objects out of the stack.
00028     ContextManager.FilterContextStack();
00029
00030     return true;
00031 }

```

The documentation for this class was generated from the following files:

- Source/OpenAccessibilityCommunication/Public/PhraseTree.h
- Source/OpenAccessibilityCommunication/Private/PhraseTree.cpp

## 4.30 FPhraseTreeBranchBind Struct Reference

### Public Member Functions

- [FPhraseTreeBranchBind](#) (TPhraseNode InRootNode, TPhraseNode InBranchRoot)

### Public Attributes

- TPhraseNode [StartNode](#)  
*The Node to start the binding of this branch root.*
- TPhraseNode [BranchRoot](#)  
*The Root Node of the Branch that needs to be bound.*

### 4.30.1 Detailed Description

Definition at line 25 of file [PhraseTree.h](#).

### 4.30.2 Constructor & Destructor Documentation

#### 4.30.2.1 FPhraseTreeBranchBind() [1/2]

```
FPhraseTreeBranchBind::FPhraseTreeBranchBind ( ) [inline]
```

Definition at line 27 of file [PhraseTree.h](#).

```
00028     {  
00029  
00030     }
```

#### 4.30.2.2 FPhraseTreeBranchBind() [2/2]

```
FPhraseTreeBranchBind::FPhraseTreeBranchBind (  
    TPhraseNode InRootNode,  
    TPhraseNode InBranchRoot ) [inline]
```

Definition at line 32 of file [PhraseTree.h](#).

```
00033     {  
00034         StartNode = InRootNode;  
00035         BranchRoot = InBranchRoot;  
00036     }
```

#### 4.30.2.3 ~FPhraseTreeBranchBind()

```
FPhraseTreeBranchBind::~~FPhraseTreeBranchBind ( ) [inline]
```

Definition at line 38 of file [PhraseTree.h](#).

```
00039     {  
00040         StartNode.Reset ();  
00041         BranchRoot.Reset ();  
00042     }
```

### 4.30.3 Member Data Documentation

#### 4.30.3.1 BranchRoot

`TPhraseNode FPhraseTreeBranchBind::BranchRoot`

The Root Node of the Branch that needs to be bound.

Definition at line 52 of file [PhraseTree.h](#).

#### 4.30.3.2 StartNode

`TPhraseNode FPhraseTreeBranchBind::StartNode`

The Node to start the binding of this branch root.

Definition at line 47 of file [PhraseTree.h](#).

The documentation for this struct was generated from the following file:

- [Source/OpenAccessibilityCommunication/Public/PhraseTree.h](#)

## 4.31 FPhraseTreeContextManager Struct Reference

### Public Member Functions

- `void IsEmpty ()`  
*Is the Context Stack Empty.*
- `bool HasContextObjects ()`  
*Does the Context Stack Contain Any Context Objects.*
- `bool HasContextObject (UPhraseTreeContextObject *InContextObject)`  
*Does the Context Stack Contain The Given Context Object.*
- `TArray< UPhraseTreeContextObject * > GetContextStack ()`  
*Gets the Entire Context Stack.*
- `void PeekContextObject (UPhraseTreeContextObject *OutContextObject)`  
*Peeks the Top Context Object On The Stack.*
- `UPhraseTreeContextObject * PeekContextObject ()`  
*Peeks the Top Context Object On The Stack.*
- `void PushContextObject (UPhraseTreeContextObject *InContextObject)`  
*Pushes a Context Object onto the Stack.*
- `void PopContextObject ()`  
*Pops the Top Context Object From The Stack.*
- `template<class CastToContextType > void PopContextObject (CastToContextType *OutContextObject)`  
*Pops the Top Context Object From The Stack.*
- `void PopContextObject (UPhraseTreeContextObject *OutContextObject)`  
*Pops the Top Context Object From The Stack.*

## Friends

- class [FPhraseTree](#)

### 4.31.1 Detailed Description

Definition at line 55 of file [PhraseTree.h](#).

### 4.31.2 Constructor & Destructor Documentation

#### 4.31.2.1 FPhraseTreeContextManager()

```
FPhraseTreeContextManager::FPhraseTreeContextManager ( ) [inline]
```

Definition at line 61 of file [PhraseTree.h](#).

```
00062     {
00063
00064     }
```

#### 4.31.2.2 ~FPhraseTreeContextManager()

```
FPhraseTreeContextManager::~~FPhraseTreeContextManager ( ) [inline]
```

Definition at line 66 of file [PhraseTree.h](#).

```
00067     {
00068
00069     }
```

### 4.31.3 Member Function Documentation

#### 4.31.3.1 GetContextStack()

```
TArray< UPhraseTreeContextObject * > FPhraseTreeContextManager::GetContextStack ( ) [inline]
```

Gets the Entire Context Stack.

##### Returns

An Array Containing the Current Context Stack.

Definition at line 104 of file [PhraseTree.h](#).

```
00105     {
00106         return this->ContextObjectStack;
00107     }
```

#### 4.31.3.2 HasContextObject()

```
bool FPhraseTreeContextManager::HasContextObject (
    UPhraseTreeContextObject * InContextObject ) [inline]
```

Does the Context Stack Contain The Given Context Object.

**Parameters**

|                        |  |
|------------------------|--|
| <i>InContextObject</i> | - The Context Object To Check if On The Stack. |
|------------------------|--|

**Returns**

True, if the Context Object is Contained on the Stack.

Definition at line 95 of file [PhraseTree.h](#).

```
00096     {
00097         return this->ContextObjectStack.Contains(InContextObject);
00098     }
```

**4.31.3.3 HasContextObjects()**

```
bool FPhraseTreeContextManager::HasContextObjects ( ) [inline]
```

Does the Context Stack Contain Any Context Objects.

**Returns**

True, if Context Objects are on the stack. Otherwise False.

Definition at line 85 of file [PhraseTree.h](#).

```
00086     {
00087         return this->ContextObjectStack.Num() > 0;
00088     }
```

**4.31.3.4 IsEmpty()**

```
void FPhraseTreeContextManager::IsEmpty ( ) [inline]
```

Is the Context Stack Empty.

Definition at line 76 of file [PhraseTree.h](#).

```
00077     {
00078         this->ContextObjectStack.IsEmpty();
00079     }
```

**4.31.3.5 PeekContextObject() [1/2]**

```
UPhraseTreeContextObject * FPhraseTreeContextManager::PeekContextObject ( ) [inline]
```

Peeks the Top Context Object On The Stack.

**Returns**

The Top Context Object on the Stack.

Definition at line 124 of file [PhraseTree.h](#).

```
00125     {
00126         return this->ContextObjectStack.Top();
00127     }
```

**4.31.3.6 PeekContextObject() [2/2]**

```
void FPhraseTreeContextManager::PeekContextObject (
    UPhraseTreeContextObject * OutContextObject ) [inline]
```

Peeks the Top Context Object On The Stack.

**Parameters**

|                         |                                   |
|-------------------------|-----------------------------------|
| <i>OutContextObject</i> | - Returns the Top Context Object. |
|-------------------------|-----------------------------------|

Definition at line 115 of file [PhraseTree.h](#).

```
00116     {
00117         OutContextObject = this->ContextObjectStack.Top();
00118     }
```

**4.31.3.7 PopContextObject() [1/3]**

```
void FPhraseTreeContextManager::PopContextObject ( ) [inline]
```

Pops the Top Context Object From The Stack.

Definition at line 141 of file [PhraseTree.h](#).

```
00142     {
00143         this->ContextObjectStack.Pop();
00144     }
```

**4.31.3.8 PopContextObject() [2/3]**

```
template<class CastToContextType >
void FPhraseTreeContextManager::PopContextObject (
    CastToContextType * OutContextObject ) [inline]
```

Pops the Top Context Object From The Stack.

**Template Parameters**

|                          |  |
|--------------------------|--|
| <i>CastToContextType</i> | DownCast Type for the Popped Context Object. (Must be Derived From <a href="#">UPhraseTreeContextObject</a> ). |
|--------------------------|--|

**Parameters**

|                         |  |
|-------------------------|--|
| <i>OutContextObject</i> | - Returns the Popped Downcasted Context Object From the Stack. |
|-------------------------|--|

Definition at line 152 of file [PhraseTree.h](#).

```
00153     {
00154         OutContextObject = Cast<CastToContextType>(this->ContextObjectStack.Pop());
00155     }
```

**4.31.3.9 PopContextObject() [3/3]**

```
void FPhraseTreeContextManager::PopContextObject (
    UPhraseTreeContextObject * OutContextObject ) [inline]
```

Pops the Top Context Object From The Stack.

**Parameters**

|                         |   |
|-------------------------|---|
| <i>OutContextObject</i> | - Returns the Popped Context Object From the Stack. |
|-------------------------|---|

Definition at line 161 of file [PhraseTree.h](#).

```
00162     {
00163         OutContextObject = this->ContextObjectStack.Pop();
00164     }
```

**4.31.3.10 PushContextObject()**

```
void FPhraseTreeContextManager::PushContextObject (
    UPhraseTreeContextObject * InContextObject ) [inline]
```

Pushes a Context Object onto the Stack.

**Parameters**

|                        |  |
|------------------------|--|
| <i>InContextObject</i> | - The Context Object To Push Onto The Stack. |
|------------------------|--|

Definition at line 133 of file [PhraseTree.h](#).

```
00134     {
00135         this->ContextObjectStack.Push(InContextObject);
00136     }
```

**4.31.4 Friends And Related Function Documentation****4.31.4.1 FPhraseTree**

```
friend class FPhraseTree [friend]
```

Definition at line 57 of file [PhraseTree.h](#).

The documentation for this struct was generated from the following file:

- Source/OpenAccessibilityCommunication/Public/PhraseTree.h

**4.32 FSocketCommunicationServer Class Reference****Public Member Functions**

- [FSocketCommunicationServer](#) (const std::string SendAddress="tcp://127.0.0.1:5555", const std::string RecvAddress="tcp://127.0.0.1:5556", const int PollTimeout=10)
- bool [EventOccured](#) ()

*Notifies when an Event Has Occured In the Socket.*

- bool [SendArrayBuffer](#) (const float \*MessageData, size\_t Size, ComSendFlags SendFlags=ComSendFlags↵::none)  
*Sends an Array of Data over the Socket, using a Buffer.*
- bool [SendArrayBuffer](#) (const float MessageData[], ComSendFlags SendFlags=ComSendFlags::none)  
*Sends an Array of Data over the Socket, using a Buffer.*
- bool [SendArrayBuffer](#) (const TArray< float > &ArrayMessage, ComSendFlags SendFlags=ComSendFlags↵::none)  
*Sends an Array of Data over the Socket, using a Buffer.*
- bool [SendArrayMessage](#) (const float \*MessageData, size\_t Size, ComSendFlags SendFlags=ComSend↵Flags::none)  
*Sends an Array of Data over the Socket, using a message.*
- bool [SendArrayMessage](#) (const float MessageData[], ComSendFlags SendFlags=ComSendFlags::none)  
*Sends an Array of Data over the Socket, using a message.*
- bool [SendArrayMessage](#) (const TArray< float > &ArrayMessage, ComSendFlags SendFlags=ComSend↵Flags::none)  
*Sends an Array of Data over the Socket, using a message.*
- bool [SendArrayMessageWithMeta](#) (const float \*MessageData, size\_t Size, const TSharedRef< FJsonObject > &Metadata, ComSendFlags SendFlags=ComSendFlags::none)  
*Sends an Array of Data over the Socket, using a message.*
- bool [SendArrayMessageWithMeta](#) (const float MessageData[], const TSharedRef< FJsonObject > &Meta-  
data, ComSendFlags SendFlags=ComSendFlags::none)  
*Sends an Array of Data over the Socket, using a message.*
- bool [SendArrayMessageWithMeta](#) (const TArray< float > &ArrayMessage, const TSharedRef< FJsonObject > &Meta-  
data, ComSendFlags SendFlags=ComSendFlags::none)  
*Sends an Array of Data over the Socket, using a message.*
- bool [SendStringBuffer](#) (const std::string StringMessage, ComSendFlags SendFlags=ComSendFlags::none)  
*Sends a String Buffer over the Socket.*
- bool [SendJsonBuffer](#) (const std::string JsonMessage, ComSendFlags SendFlags=ComSendFlags::none)  
*Sends a JSON Buffer over the Socket.*
- template<typename T >  
bool [RecvArray](#) (TArray< T > &OutArrayData, size\_t Size, ComRecvFlags RecvFlag=ComRecvFlags::none)  
*Recives an Array of Data from the Socket.*
- bool [RecvString](#) (FString &OutStringMessage, ComRecvFlags RecvFlag=ComRecvFlags::none)  
*Recives String Data From the Socket.*
- bool [RecvJson](#) (FString &OutJsonMessage, ComRecvFlags RecvFlag=ComRecvFlags::none)  
*Recieves JSON Data From The Socket.*
- bool [RecvStringMultipart](#) (TArray< FString > &OutMessages, ComRecvFlags RecvFlag=ComRecvFlags↵::none)  
*Receives An Array of String Data From The Socket.*
- bool [RecvStringMultipartWithMeta](#) (TArray< FString > &OutMessages, TSharedPtr< FJsonObject > &Out↵Metadata, ComRecvFlags RecvFlag=ComRecvFlags::none)  
*Receives An Array of String Data From The Socket, With JSON Metadata.*

## Protected Member Functions

- bool [RecvMultipartWithMeta](#) (std::vector< zmq::message\_t > &OutMultipartMessages, TSharedPtr< FJsonObject > &OutMetadata, ComRecvFlags RecvFlags)  
*Recieves a Multipart Message From The Socket, and a Metadata Object.*
- bool [SerializeJSON](#) (const TSharedRef< FJsonObject > &InJsonObject, FString &OutJsonString)  
*Serializes the JSON Object into a JSON String.*
- bool [DeserializeJSON](#) (const FString &InJsonString, TSharedPtr< FJsonObject > &OutJsonObject)  
*Deserializes the JSON String into a JSON Object.*



## Protected Attributes

- `zmq::context_t * Context`  
*The Context Used for the Socket Communication.*
- `zmq::socket_t * SendSocket`  
*The Socket Used For Sending Data.*
- `zmq::socket_t * RecvSocket`  
*The Socket Used For Receiving Data.*
- `zmq::poller_t<int> * Poller`  
*The Poller used for Polling for Events on the Receiving Socket.*
- `std::string SendAddress`
- `std::string RecvAddress`
- `int PollTimeout`  
*The Time Taken By The Poller To Look For Events.*

### 4.32.1 Detailed Description

Definition at line 22 of file [SocketCommunicationServer.h](#).

### 4.32.2 Constructor & Destructor Documentation

#### 4.32.2.1 FSocketCommunicationServer()

```
FSocketCommunicationServer::FSocketCommunicationServer (
    const std::string SendAddress = "tcp://127.0.0.1:5555",
    const std::string RecvAddress = "tcp://127.0.0.1:5556",
    const int PollTimeout = 10 )
```

Definition at line 8 of file [SocketCommunicationServer.cpp](#).

```
00009 : SendAddress(SendAddress), RecvAddress(RecvAddress), PollTimeout(PollTimeout)
00010 {
00011     Context = new zmq::context_t(1);
00012     if (Context == nullptr)
00013     {
00014         UE_LOG(LogOpenAccessibilityCom, Error, TEXT("Failed to create ZMQ context"));
00015         return;
00016     }
00017
00018     SendSocket = new zmq::socket_t(*Context, ZMQ_PUSH);
00019     if (SendSocket == nullptr)
00020     {
00021         UE_LOG(LogOpenAccessibilityCom, Error, TEXT("Failed to create ZMQ socket"));
00022         return;
00023     }
00024
00025     RecvSocket = new zmq::socket_t(*Context, ZMQ_PULL);
00026     if (RecvSocket == nullptr)
00027     {
00028         UE_LOG(LogOpenAccessibilityCom, Error, TEXT("Failed to create ZMQ socket"));
00029         return;
00030     }
00031
00032     Poller = new zmq::poller_t<int>();
00033     if (Poller == nullptr)
00034     {
00035         UE_LOG(LogOpenAccessibilityCom, Error, TEXT("Failed to create ZMQ poller"));
00036         return;
00037     }
00038
00039     SendSocket->connect(SendAddress);
00040     RecvSocket->bind(RecvAddress);
00041
00042     Poller->add(*RecvSocket, zmq::event_flags::pollin);
00043 }
```

#### 4.32.2.2 ~FSocketCommunicationServer()

FSocketCommunicationServer::~~FSocketCommunicationServer ( )

Definition at line 45 of file [SocketCommunicationServer.cpp](#).

```
00046 {
00047     Poller->remove(*RecvSocket);
00048     delete Poller; Poller = nullptr;
00049
00050     SendSocket->disconnect(SendAddress);
00051     SendSocket->close();
00052     delete SendSocket; SendSocket = nullptr;
00053
00054     RecvSocket->unbind(RecvAddress);
00055     RecvSocket->close();
00056     delete RecvSocket; RecvSocket = nullptr;
00057
00058     Context->shutdown();
00059     Context->close();
00060     delete Context; Context = nullptr;
00061 }
```

### 4.32.3 Member Function Documentation

#### 4.32.3.1 DeserializeJSON()

```
bool FSocketCommunicationServer::DeserializeJSON (
    const FString & InJsonString,
    TSharedPtr< FJsonObject > & OutJsonObject ) [protected]
```

Deserializes the JSON String into a JSON Object.

##### Parameters

|                      |  |
|----------------------|--|
| <i>InJsonString</i>  | - The JSON String To Deserialize.                |
| <i>OutJsonObject</i> | - The Returned JSON Object from Deserialization. |

##### Returns

True, if the JSON was Successfully in Deserialization. Otherwise False.

Definition at line 444 of file [SocketCommunicationServer.cpp](#).

```
00445 {
00446     return FJsonSerializer::Deserialize(TJsonReaderFactory<TCHAR>::Create(InJsonString),
        OutJsonObject);
00447 }
```

#### 4.32.3.2 EventOccured()

```
bool FSocketCommunicationServer::EventOccured ( )
```

Notifies when an Event Has Occured In the Socket.

**Returns**

True, When an Event was Recived from the Socket. Otherwise False.

Definition at line 63 of file [SocketCommunicationServer.cpp](#).

```
00064 {
00065     std::vector<zmq::poller_event<int>> PollEvents(1);
00066     if (Poller->wait_all(PollEvents, std::chrono::milliseconds(PollTimeout)) > 0)
00067     {
00068         PollEvents.clear();
00069         return true;
00070     }
00071
00072     PollEvents.clear();
00073     return false;
00074 }
```

**4.32.3.3 RecvArray()**

```
template<typename T >
bool FSocketCommunicationServer::RecvArray (
    TArray< T > & OutArrayData,
    size_t Size,
    ComRecvFlags RecvFlag = ComRecvFlags::none )
```

Recives an Array of Data from the Socket.

**Template Parameters**

|          |  |
|----------|--|
| <i>T</i> | The Type To Cast The Recived Data In The Array To. |
|----------|--|

**Parameters**

|                     |  |
|---------------------|--|
| <i>OutArrayData</i> | - The Returned Array of Data From The Socket.    |
| <i>Size</i>         | - The Size of the Data Recieved From The Array.  |
| <i>RecvFlag</i>     | - The Recv Flags To Use When Recieving The Data. |

**Returns**

True, if the Data was Recived from the Socket Successfully. False, if an error occurs in receiving.

Definition at line 303 of file [SocketCommunicationServer.cpp](#).

```
00304 {
00305     zmq::message_t RecvMessage;
00306
00307     auto Result = RecvSocket->recv(RecvMessage, RecvFlags);
00308     if (Result.has_value())
00309     {
00310         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Com Server: Recv Array || Recv %d bytes",
Result.value()));
00311
00312         OutArrayData.Append(RecvMessage.data<T>(), Result.value());
00313         return true;
00314     }
00315     else if (zmq_errno() == EAGAIN)
00316     {
00317         UE_LOG(LogOpenAccessibilityCom, Warning, TEXT("|| Com Server: Recv Array || EAGAIN Error
Occured ||"));
00318         return true;
00319     }
```

```

00320     }
00321
00322     return false;
00323 }

```

#### 4.32.3.4 RecvJson()

```

bool FSocketCommunicationServer::RecvJson (
    FString & OutJsonMessage,
    ComRecvFlags RecvFlag = ComRecvFlags::none )

```

Recieves JSON Data From The Socket.

##### Parameters

|                       |  |
|-----------------------|--|
| <i>OutJsonMessage</i> | - Returns the JSON String Data Received From the Socket. |
| <i>RecvFlag</i>       | - The Recv Flags To Use When Recieving The Data.         |

##### Returns

True, if the Data was Recived from the Socket Successfully. False, if an error occurd in receiving.

Definition at line 348 of file [SocketCommunicationServer.cpp](#).

```

00349 {
00350     zmq::message_t RecvMessage;
00351
00352     auto Result = RecvSocket->recv(RecvMessage, RecvFlags);
00353     if (Result.has_value())
00354     {
00355         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Com Server: Recv JSON || Recv %d bytes"),
Result.value());
00356
00357         OutJsonMessage = FString(Result.value(), UTF8_TO_TCHAR(RecvMessage.data()));
00358
00359         return true;
00360     }
00361     else if (zmq_errno() == EAGAIN)
00362     {
00363         UE_LOG(LogOpenAccessibilityCom, Warning, TEXT("|| Com Server: Recv JSON || EAGAIN Error
Occured ||"));
00364         return true;
00365     }
00366
00367     return false;
00368 }

```

#### 4.32.3.5 RecvMultipartWithMeta()

```

bool FSocketCommunicationServer::RecvMultipartWithMeta (
    std::vector< zmq::message_t > & OutMultipartMessages,
    TSharedPtr< FJsonObject > & OutMetadata,
    ComRecvFlags RecvFlags ) [protected]

```

Recieves a Multipart Message From The Socket, and a Metadata Object.

## Parameters

|                             |   |
|-----------------------------|---|
| <i>OutMultipartMessages</i> | - Returns the Array of Messages Contained in The Multipart. |
| <i>OutMetadata</i>          | - Returns the Metadata JSON Object from the Multipart.      |
| <i>RecvFlags</i>            | - The Recv Flags To Use When Recieving The Data.            |

## Returns

True, if the Multipart was Recieved Successfully. False, if an error ocurred in receiving.

Definition at line 409 of file [SocketCommunicationServer.cpp](#).

```

00410 {
00411     auto Result = zmq::recv_multipart(*RecvSocket, std::back_inserter(OutMultipartMessages),
    RecvFlags);
00412     if (Result.has_value())
00413     {
00414         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Com Server: Recv Multipart || Recv %d
messages"), Result.value());
00415
00416         // Pop Metadata Messages from the Front of Array.
00417         zmq::message_t MetadataMessage = MoveTempIfPossible(OutMultipartMessages[0]);
00418         OutMultipartMessages.erase(OutMultipartMessages.begin());
00419
00420         if (DeserializeJSON(FString(UTF8_TO_TCHAR(MetadataMessage.data())), MetadataMessage.size()),
    OutMetadata))
00421         {
00422             return true;
00423         }
00424         else
00425         {
00426             UE_LOG(LogOpenAccessibilityCom, Error, TEXT("|| Com Server: Recv Multipart || Failed to
deserialize metadata ||"));
00427             return false;
00428         }
00429     }
00430     else if (zmq_errno() == EAGAIN)
00431     {
00432         UE_LOG(LogOpenAccessibilityCom, Warning, TEXT("|| Com Server: Recv Multipart || EAGAIN Error
Occured ||"));
00433         return true;
00434     }
00435
00436     return false;
00437 }
```

## 4.32.3.6 RecvString()

```

bool FSocketCommunicationServer::RecvString (
    FString & OutStringMessage,
    ComRecvFlags RecvFlag = ComRecvFlags::none )
```

Recives String Data From the Socket.

## Parameters

|                         |  |
|-------------------------|--|
| <i>OutStringMessage</i> | - Returns the String Data Recived From the Socket. |
| <i>RecvFlag</i>         | - The Recv Flags To Use When Recieving The Data.   |

## Returns

True, if the Data was Recived from the Socket Successfully. False, if an error occurs in receiving.

Definition at line 325 of file [SocketCommunicationServer.cpp](#).

```

00326 {
00327     zmq::message_t RecvMessage;
00328
00329     auto Result = RecvSocket->recv(RecvMessage, RecvFlags);
00330     if (Result.has_value())
00331     {
00332         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Com Server: Recv String || Recv %d bytes"),
Result.value());
00333
00334         FStringMessage = FString(Result.value(), UTF8_TO_TCHAR(RecvMessage.data()));
00335
00336         return true;
00337     }
00338     else if (zmq_errno() == EAGAIN)
00339     {
00340
00341         UE_LOG(LogOpenAccessibilityCom, Warning, TEXT("|| Com Server: Recv String || EAGAIN Error
Occured ||"));
00342         return true;
00343     }
00344
00345     return false;
00346 }
```

#### 4.32.3.7 RecvStringMultipart()

```

bool FSocketCommunicationServer::RecvStringMultipart (
    TArray< FString > & OutMessages,
    ComRecvFlags RecvFlag = ComRecvFlags::none )
```

Receives An Array of String Data From The Socket.

##### Parameters

|                    |  |
|--------------------|--|
| <i>OutMessages</i> | - Returns the Multipart of String Data Received From the Socket. |
| <i>RecvFlag</i>    | - The Recv Flags To Use When Recieving The Data.                 |

##### Returns

True, if the Data was Received from the Socket Successfully. False, if an error occurred in receiving.

Definition at line 370 of file [SocketCommunicationServer.cpp](#).

```

00371 {
00372     std::vector<zmq::message_t> RecvMessages;
00373
00374     auto Result = zmq::recv_multipart(*RecvSocket, std::back_inserter(RecvMessages), RecvFlags);
00375     if (Result.has_value())
00376     {
00377         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Com Server: Recv Multipart || Recv %d
messages"), Result.value());
00378
00379         for (auto& Message : RecvMessages)
00380         {
00381             OutMessages.Add(FString(Message.size(), UTF8_TO_TCHAR(Message.data())));
00382         }
00383
00384         return true;
00385     }
00386     else if (zmq_errno() == EAGAIN)
00387     {
00388         UE_LOG(LogOpenAccessibilityCom, Warning, TEXT("|| Com Server: Recv Multipart || EAGAIN Error
Occured ||"));
00389         return true;
00390     }
00391
00392     return false;
00393 }
```

#### 4.32.3.8 RecvStringMultipartWithMeta()

```
bool FSocketCommunicationServer::RecvStringMultipartWithMeta (
    TArray< FString > & OutMessages,
    TSharedPtr< FJsonObject > & OutMetadata,
    ComRecvFlags RecvFlag = ComRecvFlags::none )
```

Receives An Array of String Data From The Socket, With JSON Metadata.

##### Parameters

|                    |  |
|--------------------|--|
| <i>OutMessages</i> | - Returns the Received Array of String Data.     |
| <i>OutMetadata</i> | - Returns a JSON Object containing Metadata.     |
| <i>RecvFlag</i>    | - The Recv Flags To Use When Recieving The Data. |

##### Returns

True, if the Multipart was Received Successfully. False, if an error occured in receiving.

Definition at line 395 of file [SocketCommunicationServer.cpp](#).

```
00396 {
00397     std::vector<zmq::message_t> RecvMessages;
00398     if (!RecvMultipartWithMeta(RecvMessages, OutMetadata, RecvFlag))
00399         return false;
00400
00401     for (auto& Message : RecvMessages)
00402     {
00403         OutMessages.Add(FString(Message.size(), UTF8_TO_TCHAR(Message.data())));
00404     }
00405
00406     return true;
00407 }
```

#### 4.32.3.9 SendArrayBuffer() [1/3]

```
bool FSocketCommunicationServer::SendArrayBuffer (
    const float * MessageData,
    size_t Size,
    ComSendFlags SendFlags = ComSendFlags::none )
```

Sends an Array of Data over the Socket, using a Buffer.

##### Parameters

|                    |  |
|--------------------|--|
| <i>MessageData</i> | - The Array of Message Data To Send.               |
| <i>Size</i>        | - The Size of the Provided Data Array.             |
| <i>SendFlags</i>   | - The Send Flags for when sending over the socket. |

##### Returns

True, if the Buffer was Sent Successfully. False, if an error occurs in sending.

Definition at line 76 of file [SocketCommunicationServer.cpp](#).

```

00077 {
00078     auto Result = SendSocket->send(zmq::const_buffer(MessageData, Size * sizeof(float)), SendFlags);
00079     if (Result.has_value())
00080     {
00081         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Com Server: Sent Array || Sent %d of %d bytes"),
Result.value(), Size * sizeof(float));
00082         return true;
00083     }
00084     else if (zmq_errno() == EAGAIN)
00085     {
00086         UE_LOG(LogOpenAccessibilityCom, Warning, TEXT("|| Com Server: Sent Array || EAGAIN Error
Occured ||"));
00087         return true;
00088     }
00089
00090     return false;
00091 }

```

#### 4.32.3.10 SendArrayBuffer() [2/3]

```

bool FSocketCommunicationServer::SendArrayBuffer (
    const float MessageData[],
    ComSendFlags SendFlags = ComSendFlags::none )

```

Sends an Array of Data over the Socket, using a Buffer.

##### Parameters

|                    |  |
|--------------------|--|
| <i>MessageData</i> | - The Array of Message Data To Send.               |
| <i>SendFlags</i>   | - The Send Flags for when sending over the socket. |

##### Returns

True, if the Buffer was Sent Successfully. False, if an error occurs in sending.

Definition at line 93 of file [SocketCommunicationServer.cpp](#).

```

00094 {
00095     auto Result = SendSocket->send(zmq::const_buffer(MessageData, sizeof MessageData), SendFlags);
00096     if (Result.has_value())
00097     {
00098         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Com Server: Sent Array || Sent %d of %d bytes"),
Result.value(), int(sizeof MessageData));
00099         return true;
00100     }
00101     else if (zmq_errno() == EAGAIN)
00102     {
00103         UE_LOG(LogOpenAccessibilityCom, Warning, TEXT("|| Com Server: Sent Array || EAGAIN Error
Occured ||"));
00104         return true;
00105     }
00106
00107     return false;
00108 }

```

#### 4.32.3.11 SendArrayBuffer() [3/3]

```

bool FSocketCommunicationServer::SendArrayBuffer (
    const TArray< float > & ArrayMessage,
    ComSendFlags SendFlags = ComSendFlags::none )

```

Sends an Array of Data over the Socket, using a Buffer.



## Parameters

|                     |  |
|---------------------|--|
| <i>ArrayMessage</i> | - The Array of Message Data To Send.               |
| <i>SendFlags</i>    | - The Send Flags for when sending over the socket. |

## Returns

True, if the Buffer was Sent Successfully. False, if an error occurs in sending.

Definition at line 110 of file [SocketCommunicationServer.cpp](#).

```

00111 {
00112     auto Result = SendSocket->send(zmq::const_buffer(ArrayMessage.GetData(), ArrayMessage.Num() *
sizeof(float)), SendFlag);
00113     if (Result.has_value())
00114     {
00115         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Com Server: Sent Array || Sent %d of %d bytes"),
Result.value(), int(ArrayMessage.Num() * sizeof(float)));
00116         return true;
00117     }
00118     else if (zmq_errno() == EAGAIN)
00119     {
00120         UE_LOG(LogOpenAccessibilityCom, Warning, TEXT("|| Com Server: Sent Array || EAGAIN Error
Occured ||"));
00121         return true;
00122     }
00123
00124     return false;
00125 }
```

## 4.32.3.12 SendArrayMessage() [1/3]

```

bool FSocketCommunicationServer::SendArrayMessage (
    const float * MessageData,
    size_t Size,
    ComSendFlags SendFlags = ComSendFlags::none )
```

Sends an Array of Data over the Socket, using a message.

## Parameters

|                    |  |
|--------------------|--|
| <i>MessageData</i> | - The Array of Data To Send.                       |
| <i>Size</i>        | - The Size of the Data in the Array.               |
| <i>SendFlags</i>   | - The Send Flags for when sending over the socket. |

## Returns

True, if the Message was Sent Successfully. False, if an error occurs in sending.

Definition at line 127 of file [SocketCommunicationServer.cpp](#).

```

00128 {
00129     auto Result = SendSocket->send(zmq::message_t(MessageData, Size * sizeof(float)), SendFlags);
00130     if (Result.has_value())
00131     {
00132         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Com Server: Sent Array || Sent %d of %d bytes"),
Result.value(), Size * sizeof(float));
00133         return true;
00134     }
00135     else if (zmq_errno() == EAGAIN)
00136     {
```

```

00137         UE_LOG(LogOpenAccessibilityCom, Warning, TEXT("|| Com Server: Sent Array || EAGAIN Error
Occured ||"));
00138         return true;
00139     }
00140
00141     return false;
00142 }

```

#### 4.32.3.13 SendArrayMessage() [2/3]

```

bool FSocketCommunicationServer::SendArrayMessage (
    const float MessageData[],
    ComSendFlags SendFlags = ComSendFlags::none )

```

Sends an Array of Data over the Socket, using a message.

##### Parameters

|                    |  |
|--------------------|--|
| <i>MessageData</i> | - The Array of Data To Send.                   |
| <i>SendFlags</i>   | - The Send Flags To Use When Sending The Data. |

##### Returns

True, if the Message was Sent Successfully. False, if an error occurs in sending.

Definition at line 144 of file [SocketCommunicationServer.cpp](#).

```

00145 {
00146     auto Result = SendSocket->send(zmq::message_t(MessageData, sizeof MessageData), SendFlags);
00147     if (Result.has_value())
00148     {
00149         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Com Server: Sent Array || Sent %d of %d bytes"),
Result.value(), int(sizeof MessageData));
00150         return true;
00151     }
00152     else if (zmq_errno() == EAGAIN)
00153     {
00154         UE_LOG(LogOpenAccessibilityCom, Warning, TEXT("|| Com Server: Sent Array || EAGAIN Error
Occured ||"));
00155         return true;
00156     }
00157     return false;
00158 }
00159 }

```

#### 4.32.3.14 SendArrayMessage() [3/3]

```

bool FSocketCommunicationServer::SendArrayMessage (
    const TArray< float > & ArrayMessage,
    ComSendFlags SendFlags = ComSendFlags::none )

```

Sends an Array of Data over the Socket, using a message.

##### Parameters

|                     |  |
|---------------------|--|
| <i>ArrayMessage</i> | - The Array of Data To Send.                   |
| <i>SendFlags</i>    | - The Send Flags To Use When Sending The Data. |

**Returns**

True, if the Message was Sent Successfully. False, if an error occurs in sending.

Definition at line 161 of file [SocketCommunicationServer.cpp](#).

```
00162 {
00163     auto Result = SendSocket->send(zmq::message_t(ArrayMessage.GetData(), ArrayMessage.Num() *
sizeof(float)), SendFlags);
00164     if (Result.has_value())
00165     {
00166         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Com Server: Sent Array || Sent %d of %d bytes"),
Result.value(), int(ArrayMessage.Num() * sizeof(float)));
00167         return true;
00168     }
00169     else if (zmq_errno() == EAGAIN)
00170     {
00171         UE_LOG(LogOpenAccessibilityCom, Warning, TEXT("|| Com Server: Sent Array || EAGAIN Error
Occured ||"));
00172         return true;
00173     }
00174     return false;
00175 }
00176 }
```

**4.32.3.15 SendArrayMessageWithMeta() [1/3]**

```
bool FSocketCommunicationServer::SendArrayMessageWithMeta (
    const float * MessageData,
    size_t Size,
    const TSharedRef< FJsonObject > & Metadata,
    ComSendFlags SendFlags = ComSendFlags::none )
```

Sends an Array of Data over the Socket, using a message.

**Parameters**

|                    |  |
|--------------------|--|
| <i>MessageData</i> | - The Array of Data To Send.                   |
| <i>Size</i>        | - The Size of The Data Array.                  |
| <i>Metadata</i>    | - The JSON Metadata to Send With The Message.  |
| <i>SendFlags</i>   | - The Send Flags To Use When Sending The Data. |

**Returns**

True, if the Message was Sent Successfully. False, if an error occurs in sending.

Definition at line 178 of file [SocketCommunicationServer.cpp](#).

```
00179 {
00180     FString MetaDataString;
00181     if (!SerializeJSON(Metadata, MetaDataString))
00182     {
00183         UE_LOG(LogOpenAccessibilityCom, Error, TEXT("|| Com Server: Sent Array || Failed to serialize
metadata ||"));
00184         return false;
00185     }
00186     std::vector<zmq::message_t> Messages;
00187     Messages.push_back(zmq::message_t(*MetaDataString, MetaDataString.Len() * sizeof(TCHAR)));
00188     Messages.push_back(zmq::message_t(MessageData, Size * sizeof(float)));
00189     auto Result = zmq::send_multipart(*SendSocket, Messages, SendFlags);
00190     if (Result.has_value())
00191     {
00192     }
```

```

00195         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Com Server: Sent Array || Sent %d of %d bytes"),
Result.value(), Size * sizeof(float));
00196         return true;
00197     }
00198     else if (zmq_errno() == EAGAIN)
00199     {
00200         UE_LOG(LogOpenAccessibilityCom, Warning, TEXT("|| Com Server: Sent Array || EAGAIN Error
Occured ||"));
00201         return true;
00202     }
00203
00204     return false;
00205 }

```

#### 4.32.3.16 SendArrayMessageWithMeta() [2/3]

```

bool FSocketCommunicationServer::SendArrayMessageWithMeta (
    const float MessageData[],
    const TSharedRef< FJsonObject > & Metadata,
    ComSendFlags SendFlags = ComSendFlags::none )

```

Sends an Array of Data over the Socket, using a message.

##### Parameters

|                    |  |
|--------------------|--|
| <i>MessageData</i> | - The Array of Data To Send.                   |
| <i>Metadata</i>    | - The JSON Metadata to Send With The Message.  |
| <i>SendFlags</i>   | - The Send Flags To Use When Sending The Data. |

##### Returns

True, if the Message was Sent Successfully. False, if an error occurs in sending.

Definition at line 207 of file [SocketCommunicationServer.cpp](#).

```

00208 {
00209     FString MetaDataString;
00210     if (!SerializeJSON(Metadata, MetaDataString))
00211     {
00212         UE_LOG(LogOpenAccessibilityCom, Error, TEXT("|| Com Server: Sent Array || Failed to serialize
metadata ||"));
00213         return false;
00214     }
00215
00216     std::vector<zmq::message_t> Messages;
00217     Messages.push_back(zmq::message_t(*MetaDataString, MetaDataString.Len() * sizeof(TCHAR)));
00218     Messages.push_back(zmq::message_t(MessageData, sizeof MessageData));
00219
00220     auto Result = zmq::send_multipart(*SendSocket, Messages, SendFlags);
00221     if (Result.has_value())
00222     {
00223         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Com Server: Sent Array || Sent %d of %d bytes"),
Result.value(), int(sizeof MessageData));
00224
00225         return true;
00226     }
00227     else if (zmq_errno() == EAGAIN)
00228     {
00229         UE_LOG(LogOpenAccessibilityCom, Warning, TEXT("|| Com Server: Sent Array || EAGAIN Error
Occured ||"));
00230         return true;
00231     }
00232
00233     return false;
00234 }

```

**4.32.3.17 SendArrayMessageWithMeta()** [3/3]

```
bool FSocketCommunicationServer::SendArrayMessageWithMeta (
    const TArray< float > & ArrayMessage,
    const TSharedRef< FJsonObject > & Metadata,
    ComSendFlags SendFlags = ComSendFlags::none )
```

Sends an Array of Data over the Socket, using a message.

**Parameters**

|                     |  |
|---------------------|--|
| <i>ArrayMessage</i> | - The Array of Data To Send.                   |
| <i>Metadata</i>     | - The JSON Metadata to Send With The Message.  |
| <i>SendFlags</i>    | - The Send Flags To Use When Sending The Data. |

**Returns**

True, if the Message was Sent Successfully. False, if an error occurs in sending.

Definition at line 236 of file [SocketCommunicationServer.cpp](#).

```
00237 {
00238     FString MetaDataString;
00239     if (!SerializeJSON(Metadata, MetaDataString))
00240     {
00241         UE_LOG(LogOpenAccessibilityCom, Error, TEXT("|| Com Server: Sent Array || Failed to serialize
00242         metadata ||"));
00243         return false;
00244     }
00245     std::vector<zmq::message_t> Messages;
00246     Messages.push_back(zmq::message_t(*MetaDataString, MetaDataString.Len() * sizeof(TCHAR)));
00247     Messages.push_back(zmq::message_t(ArrayMessage.GetData(), ArrayMessage.Num() * sizeof(float)));
00248     auto Result = zmq::send_multipart(*SendSocket, Messages, SendFlags);
00249     if (Result.has_value())
00250     {
00251         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Com Server: Sent Array || Sent %d of %d
00252         Messages"), Result.value(), Messages.size());
00253         return true;
00254     }
00255     else if (zmq_errno() == EAGAIN)
00256     {
00257         UE_LOG(LogOpenAccessibilityCom, Warning, TEXT("|| Com Server: Sent Array || EAGAIN Error
00258         Occured ||"));
00259         return true;
00260     }
00261     return false;
00262 }
00263
00264 }
```

**4.32.3.18 SendJsonBuffer()**

```
bool FSocketCommunicationServer::SendJsonBuffer (
    const std::string JsonMessage,
    ComSendFlags SendFlags = ComSendFlags::none )
```

Sends a JSON Buffer over the Socket.

**Parameters**

|                    |  |
|--------------------|--|
| <i>JsonMessage</i> | - The JSOn String Data To Send.                |
| <i>SendFlags</i>   | - The Send Flags To Use When Sending The Data. |

**Returns**

True, if the Buffer was Sent Successfully. False, if an error occurs in sending.

Definition at line 283 of file [SocketCommunicationServer.cpp](#).

```

00284 {
00285     auto Result = SendSocket->send(zmq::const_buffer(JsonMessage.c_str(), JsonMessage.size()),
    SendFlags);
00286     if (Result.has_value())
00287     {
00288         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Com Server: Sent JSON || Sent %d of %d bytes"),
    Result.value(), JsonMessage.size());
00289         return true;
00290     }
00291     else if (zmq_errno() == EAGAIN)
00292     {
00293         UE_LOG(LogOpenAccessibilityCom, Warning, TEXT("|| Com Server: Sent JSON || EAGAIN Error
    Occured ||"));
00294         return true;
00295     }
00296
00297     return false;
00298 }
```

**4.32.3.19 SendStringBuffer()**

```

bool FSocketCommunicationServer::SendStringBuffer (
    const std::string StringMessage,
    ComSendFlags SendFlags = ComSendFlags::none )
```

Sends a String Buffer over the Socket.

**Parameters**

|                      |  |
|----------------------|--|
| <i>StringMessage</i> | - The String Data To Send.                     |
| <i>SendFlags</i>     | - The Send Flags To Use When Sending The Data. |

**Returns**

True, if the Buffer was Sent Successfully. False, if an error occurs in sending.

Definition at line 266 of file [SocketCommunicationServer.cpp](#).

```

00267 {
00268     auto Result = SendSocket->send(zmq::const_buffer(StringMessage.c_str(), StringMessage.size()),
    SendFlags);
00269     if (Result.has_value())
00270     {
00271         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Com Server: Sent String || Sent %d of %d
    bytes"), Result.value(), StringMessage.size());
00272         return true;
00273     }
00274     else if (zmq_errno() == EAGAIN)
00275     {
00276         UE_LOG(LogOpenAccessibilityCom, Warning, TEXT("|| Com Server: Sent String || EAGAIN Error
    Occured ||"));
```

```

00277         return true;
00278     }
00279
00280     return false;
00281 }

```

#### 4.32.3.20 SerializeJSON()

```

bool FSocketCommunicationServer::SerializeJSON (
    const TSharedRef< FJsonObject > & InJsonObject,
    FString & OutJsonString ) [protected]

```

Serializes the JSON Object into a JSON String.

##### Parameters

|                      |   |
|----------------------|---|
| <i>InJsonObject</i>  | - The JSON Object To Serialize.                           |
| <i>OutJsonString</i> | - The Returned Serialized JSON String from Serialization. |

##### Returns

True, if the JSON Object was Successful in Serialization. Otherwise False.

Definition at line 439 of file [SocketCommunicationServer.cpp](#).

```

00440 {
00441     return FJsonSerializer::Serialize(InJsonObject,
        TJsonWriterFactory<TCHAR>::Create(&OutJsonString));
00442 }

```

### 4.32.4 Member Data Documentation

#### 4.32.4.1 Context

```

zmq::context_t* FSocketCommunicationServer::Context [protected]

```

The Context Used for the Socket Communication.

Definition at line 205 of file [SocketCommunicationServer.h](#).

#### 4.32.4.2 Poller

```

zmq::poller_t<int>* FSocketCommunicationServer::Poller [protected]

```

The Poller used for Polling for Events on the Receiving Socket.

Definition at line 220 of file [SocketCommunicationServer.h](#).

#### 4.32.4.3 PollTimeout

```
int FSocketCommunicationServer::PollTimeout [protected]
```

The Time Taken By The Poller To Look For Events.

Definition at line 228 of file [SocketCommunicationServer.h](#).

#### 4.32.4.4 RecvAddress

```
std::string FSocketCommunicationServer::RecvAddress [protected]
```

Definition at line 223 of file [SocketCommunicationServer.h](#).

#### 4.32.4.5 RecvSocket

```
zmq::socket_t* FSocketCommunicationServer::RecvSocket [protected]
```

The Socket Used For Receiving Data.

Definition at line 215 of file [SocketCommunicationServer.h](#).

#### 4.32.4.6 SendAddress

```
std::string FSocketCommunicationServer::SendAddress [protected]
```

Definition at line 222 of file [SocketCommunicationServer.h](#).

#### 4.32.4.7 SendSocket

```
zmq::socket_t* FSocketCommunicationServer::SendSocket [protected]
```

The Socket Used For Sending Data.

Definition at line 210 of file [SocketCommunicationServer.h](#).

The documentation for this class was generated from the following files:

- Source/OpenAccessibilityCommunication/Public/SocketCommunicationServer.h
- Source/OpenAccessibilityCommunication/Private/SocketCommunicationServer.cpp



## 4.33 FTranscriptionVisualizer Class Reference

### Public Member Functions

- virtual bool [Tick](#) (float DeltaTime)
- void [ConstructVisualizer](#) ()  
*Constructs the Visualizer Window, and Its Content.*
- void [UpdateVisualizer](#) ()  
*Updates the Visualizer Window, If Active.*
- void [ReparentWindow](#) ()  
*Represents the Visualizer Window to the Active Window.*
- void [MoveVisualizer](#) ()  
*Moves the Visualizer Window to the Active Window Position.*
- void [OnTranscriptionRecieved](#) (TArray< FString > InTranscription)  
*Callback for when Transcriptions are Recieved From Transcribed Audio.*

### Protected Member Functions

- bool [GetTopScreenVisualizerPosition](#) (FVector2D &OutPosition)  
*Gets the Position of the Visualizer for the Top Active Screen.*
- bool [GetDisplayVisualizerPosition](#) (FVector2D &OutPosition)  
*Gets the Position of the Visualizer for the Last Active Display.*
- void [RegisterTicker](#) ()  
*Registers the Ticker for the Visualizer.*
- void [UnregisterTicker](#) ()  
*Unregisters the Ticker for the Visualizer.*

### Protected Attributes

- FTSTicker::FDelegateHandle [TickDelegateHandle](#)
- TWeakPtr< SWindow > [VisWindow](#)  
*The Visualizers Containing Window.*
- TWeakPtr< class [SAccessibilityTranscriptionVis](#) > [VisContent](#)  
*The Content of the Visualizer Window.*

#### 4.33.1 Detailed Description

Definition at line 7 of file [TranscriptionVisualizer.h](#).

#### 4.33.2 Constructor & Destructor Documentation

#### 4.33.2.1 FTranscriptionVisualizer()

FTranscriptionVisualizer::FTranscriptionVisualizer ( )

Definition at line 7 of file [TranscriptionVisualizer.cpp](#).

```
00008 {
00009     RegisterTicker();
00010 }
```

#### 4.33.2.2 ~FTranscriptionVisualizer()

FTranscriptionVisualizer::~~FTranscriptionVisualizer ( )

Definition at line 12 of file [TranscriptionVisualizer.cpp](#).

```
00013 {
00014     UnregisterTicker();
00015 }
```

### 4.33.3 Member Function Documentation

#### 4.33.3.1 ConstructVisualizer()

void FTranscriptionVisualizer::ConstructVisualizer ( )

Constructs the Visualizer Window, and Its Content.

Definition at line 31 of file [TranscriptionVisualizer.cpp](#).

```
00032 {
00033     TSharedPtr<SAccessibilityTranscriptionVis> MenuContent = SNew(SAccessibilityTranscriptionVis)
00034         .VisAmount(2);
00035
00036     MenuContent->ForceVolatile(true);
00037
00038     FDisplayMetrics DisplayMetrics;
00039     FSlateApplication::Get().GetDisplayMetrics(DisplayMetrics);
00040
00041     FVector2D VisPosition = FVector2D();
00042
00043     if (FSlateApplication::Get().GetActiveTopLevelRegularWindow().IsValid())
00044     {
00045         VisPosition =
00046             FSlateApplication::Get().GetActiveTopLevelRegularWindow()->GetPositionInScreen();
00047
00048         VisPosition.X = DisplayMetrics.PrimaryDisplayWidth;
00049         VisPosition.Y = DisplayMetrics.PrimaryDisplayHeight;
00050
00051         TSharedRef<SWindow> MenuWindow = SNew(SWindow)
00052             .Type(EWindowType::Normal)
00053             .SizingRule(ESizingRule::Autosized)
00054             .ScreenPosition(VisPosition)
00055             .ClientSize(FVector2D(10, 10))
00056             .IsPopupWindow(true)
00057             .InitialOpacity(0.5f)
00058             .SupportsTransparency(EWindowTransparency::PerWindow)
00059             .ActivationPolicy(EWindowActivationPolicy::Always)
00060             .AdjustInitialSizeAndPositionForDPIScale(true)
00061             [
00062                 MenuContent.ToSharedRef()
00063             ];
00064
00065         TSharedPtr<SWindow> TopLevelWindow = FSlateApplication::Get().GetActiveTopLevelRegularWindow();
00066
00067         MenuWindow->AssignParentWidget(TopLevelWindow);
00068         FSlateApplication::Get().AddWindowAsNativeChild(MenuWindow, TopLevelWindow.ToSharedRef(), true);
00069
00070         VisWindow = MenuWindow.ToWeakPtr();
00071         VisContent = MenuContent.ToWeakPtr();
00072 }
```

### 4.33.3.2 GetDisplayVisualizerPosition()

```
bool FTranscriptionVisualizer::GetDisplayVisualizerPosition (
    FVector2D & OutPosition ) [protected]
```

Gets the Position of the Visualizer for the Last Active Display.

#### Parameters

|                    |  |
|--------------------|--|
| <i>OutPosition</i> |  |
|--------------------|--|

Definition at line 145 of file [TranscriptionVisualizer.cpp](#).

```
00146 {
00147     FDisplayMetrics DisplayMetrics;
00148     FSlateApplication::Get().GetDisplayMetrics(DisplayMetrics);
00149
00150     OutPosition.X = DisplayMetrics.PrimaryDisplayWidth;
00151     OutPosition.Y = DisplayMetrics.PrimaryDisplayHeight;
00152
00153     return true;
00154 }
```

### 4.33.3.3 GetTopScreenVisualizerPosition()

```
bool FTranscriptionVisualizer::GetTopScreenVisualizerPosition (
    FVector2D & OutPosition ) [protected]
```

Gets the Position of the Visualizer for the Top Active Screen.

#### Parameters

|                    |  |
|--------------------|--|
| <i>OutPosition</i> |  |
|--------------------|--|

Definition at line 128 of file [TranscriptionVisualizer.cpp](#).

```
00129 {
00130     TSharedPtr<SWindow> TopLevelWindow = FSlateApplication::Get().GetActiveTopLevelRegularWindow();
00131     if (!TopLevelWindow.IsValid())
00132         return false;
00133
00134     FVector2D ActiveWindowPosition = TopLevelWindow->GetPositionInScreen();
00135     FVector2D ActiveWindowBounds = TopLevelWindow->GetClientSizeInScreen();
00136
00137     TSharedPtr<SWindow> VisWindowPtr = VisWindow.Pin();
00138
00139     OutPosition.X = (ActiveWindowPosition.X + ActiveWindowBounds.X / 2) -
00140         (VisWindowPtr->GetClientSizeInScreen().X / 2);
00141     OutPosition.Y = (ActiveWindowPosition.Y + ActiveWindowBounds.Y - 50) -
00142         VisWindowPtr->GetClientSizeInScreen().Y;
00143
00144     return true;
00145 }
```

### 4.33.3.4 MoveVisualizer()

```
void FTranscriptionVisualizer::MoveVisualizer ( )
```

Moves the Visualizer Window to the Active Window Position.

Definition at line 108 of file [TranscriptionVisualizer.cpp](#).

```
00109 {
00110     FVector2D NewPosition = FVector2D();
00111
00112     if (!GetTopScreenVisualizerPosition(NewPosition))
00113     {
00114         GetDisplayVisualizerPosition(NewPosition);
00115     }
00116
00117     VisWindow.Pin() -> MoveWindowTo(NewPosition);
00118 }
```

#### 4.33.3.5 OnTranscriptionRecieved()

```
void FTranscriptionVisualizer::OnTranscriptionRecieved (
    TArray< FString > InTranscription )
```

Callback for when Transcriptions are Recieved From Transcribed Audio.

##### Parameters

|                        |  |
|------------------------|--|
| <i>InTranscription</i> | Incoming Array of Transcription Strings. |
|------------------------|--|

Definition at line 120 of file [TranscriptionVisualizer.cpp](#).

```
00121 {
00122     for (int i = 0; i < InTranscription.Num(); i++)
00123     {
00124         VisContent.Pin() -> UpdateTopTranscription(InTranscription[i]);
00125     }
00126 }
```

#### 4.33.3.6 RegisterTicker()

```
void FTranscriptionVisualizer::RegisterTicker ( ) [protected]
```

Registers the Ticker for the Visualizer.

Definition at line 156 of file [TranscriptionVisualizer.cpp](#).

```
00157 {
00158     FTickerDelegate TickDelegate = FTickerDelegate::CreateRaw(this, &FTranscriptionVisualizer::Tick);
00159
00160     TickDelegateHandle = FTicker::GetCoreTicker().AddTicker(TickDelegate);
00161 }
```

#### 4.33.3.7 ReparentWindow()

```
void FTranscriptionVisualizer::ReparentWindow ( )
```

Reparents the Visualizer Window to the Active Window.

Definition at line 86 of file [TranscriptionVisualizer.cpp](#).

```
00087 {
00088     TSharedPtr<SWindow> TopLevelActiveWindow =
        FSlateApplication::Get().GetActiveTopLevelRegularWindow();
00089     if (!TopLevelActiveWindow.IsValid())
00090         return;
00091
00092     TSharedPtr<SWindow> VisWindowPtr = VisWindow.Pin();
00093
00094     if (TopLevelActiveWindow == VisWindow.Pin() ||
00095         TopLevelActiveWindow->GetContent() == VisWindowPtr->GetParentWidget())
00096         return;
00097
00098     TSharedPtr<SWindow> PrevParentWindow = VisWindowPtr->GetParentWindow();
00099     if (PrevParentWindow.IsValid())
00100     {
00101         PrevParentWindow->RemoveDescendantWindow(VisWindowPtr.ToSharedRef());
00102     }
00103
00104     VisWindowPtr->AssignParentWidget(TopLevelActiveWindow);
00105     TopLevelActiveWindow->AddChildWindow(VisWindowPtr.ToSharedRef());
00106 }
```

#### 4.33.3.8 Tick()

```
bool FTranscriptionVisualizer::Tick (
    float DeltaTime ) [virtual]
```

Definition at line 17 of file [TranscriptionVisualizer.cpp](#).

```
00018 {
00019     if (VisWindow.IsValid())
00020     {
00021         UpdateVisualizer();
00022     }
00023     else if (FSlateApplication::Get().GetActiveTopLevelRegularWindow().IsValid() &&
        FSlateApplication::Get().IsActive())
00024     {
00025         ConstructVisualizer();
00026     }
00027
00028     return true;
00029 }
```

#### 4.33.3.9 UnregisterTicker()

```
void FTranscriptionVisualizer::UnregisterTicker ( ) [protected]
```

Unregisters the Ticker for the Visualizer.

Definition at line 163 of file [TranscriptionVisualizer.cpp](#).

```
00164 {
00165     FTSTicker::GetCoreTicker().RemoveTicker(TickDelegateHandle);
00166 }
```

#### 4.33.3.10 UpdateVisualizer()

```
void FTranscriptionVisualizer::UpdateVisualizer ( )
```

Updates the Visualizer Window, If Active.

Definition at line 73 of file [TranscriptionVisualizer.cpp](#).

```
00074 {
00075     if (FSlateApplication::Get().IsActive())
00076     {
00077         VisWindow.Pin()->ShowWindow();
00078
00079         // ReparentWindow();
00080
00081         MoveVisualizer();
00082     }
00083     else VisWindow.Pin()->HideWindow();
00084 }
```

### 4.33.4 Member Data Documentation

#### 4.33.4.1 TickDelegateHandle

```
FTSTicker::FDelegateHandle FTranscriptionVisualizer::TickDelegateHandle [protected]
```

Definition at line 75 of file [TranscriptionVisualizer.h](#).

#### 4.33.4.2 VisContent

```
TWeakPtr<class SAccessibilityTranscriptionVis> FTranscriptionVisualizer::VisContent [protected]
```

The Content of the Visualizer Window.

Definition at line 87 of file [TranscriptionVisualizer.h](#).

#### 4.33.4.3 VisWindow

```
TWeakPtr<SWindow> FTranscriptionVisualizer::VisWindow [protected]
```

The Visualizers Containing Window.

Definition at line 82 of file [TranscriptionVisualizer.h](#).

The documentation for this class was generated from the following files:

- Source/OpenAccessibility/Public/TranscriptionVisualizer.h
- Source/OpenAccessibility/Private/TranscriptionVisualizer.cpp

## 4.34 UAccessibilityGraphEditorContext::FTreeViewTickRequirements Struct Reference

### Public Attributes

- FString [PrevSearchText](#)
- int32 [PrevNumItemsBeingObserved](#)
- int32 [PrevNumGeneratedChildren](#)
- double [PrevScrollDistance](#)

### 4.34.1 Detailed Description

Definition at line 146 of file [AccessibilityGraphEditorContext.h](#).

### 4.34.2 Constructor & Destructor Documentation

#### 4.34.2.1 FTreeViewTickRequirements()

```
UAccessibilityGraphEditorContext::FTreeViewTickRequirements::FTreeViewTickRequirements ( )  
[inline]
```

Definition at line 150 of file [AccessibilityGraphEditorContext.h](#).

```
00151         : PrevSearchText (FString())  
00152         , PrevNumItemsBeingObserved (-1)  
00153         , PrevNumGeneratedChildren (-1)  
00154         , PrevScrollDistance (-1.00)  
00155     { }
```

### 4.34.3 Member Data Documentation

#### 4.34.3.1 PrevNumGeneratedChildren

```
int32 UAccessibilityGraphEditorContext::FTreeViewTickRequirements::PrevNumGeneratedChildren
```

Definition at line 159 of file [AccessibilityGraphEditorContext.h](#).

#### 4.34.3.2 PrevNumItemsBeingObserved

```
int32 UAccessibilityGraphEditorContext::FTreeViewTickRequirements::PrevNumItemsBeingObserved
```

Definition at line 158 of file [AccessibilityGraphEditorContext.h](#).

#### 4.34.3.3 PrevScrollDistance

```
double UAccessibilityGraphEditorContext::FTreeViewTickRequirements::PrevScrollDistance
```

Definition at line 160 of file [AccessibilityGraphEditorContext.h](#).

#### 4.34.3.4 PrevSearchText

```
FString UAccessibilityGraphEditorContext::FTreeViewTickRequirements::PrevSearchText
```

Definition at line 157 of file [AccessibilityGraphEditorContext.h](#).

The documentation for this struct was generated from the following file:

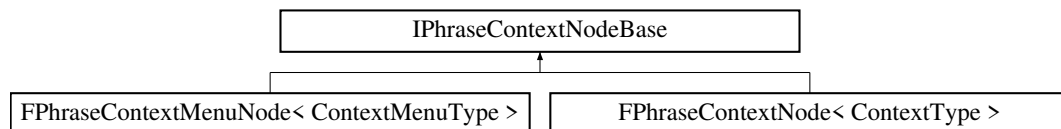
- Source/OpenAccessibility/Public/AccessibilityWrappers/AccessibilityGraphEditorContext.h

### 4.35 IPhraseContextNodeBase Class Reference

Base Abstract Class For Phrase Context Nodes, that are required to have a Context Node.

```
#include <IPhraseContextNode.h>
```

Inheritance diagram for IPhraseContextNodeBase:



#### Protected Member Functions

- virtual bool [HasContextObject](#) (TArray< [UPhraseTreeContextObject](#) \* > InContextObjects) const =0  
*Checks if the Given Context Array Contains Context Objects.*
- virtual [UPhraseTreeContextObject](#) \* [CreateContextObject](#) (FParseRecord &Record)=0  
*Creates a Context Object, using Record Inputs.*
- virtual void [ConstructContextChildren](#) (TArray< TSharedPtr< class [FPhraseNode](#) > > &InChildNodes)=0  
*Constructs the Context Nodes Children, from Given Child Nodes. Allowing for Inclusion of Utility Nodes in relation to the Context.*

#### 4.35.1 Detailed Description

Base Abstract Class For Phrase Context Nodes, that are required to have a Context Node.

Definition at line 12 of file [IPhraseContextNode.h](#).



## 4.35.2 Member Function Documentation

### 4.35.2.1 ConstructContextChildren()

```
virtual void IPhraseContextNodeBase::ConstructContextChildren (
    TArray< TSharedPtr< class FPhraseNode > > & InChildNodes ) [protected], [pure
virtual]
```

Constructs the Context Nodes Children, from Given Child Nodes. Allowing for Inclusion of Utility Nodes in relation to the Context.

#### Parameters

|                     |                                   |
|---------------------|-----------------------------------|
| <i>InChildNodes</i> | - An Array of the Nodes Children. |
|---------------------|-----------------------------------|

### 4.35.2.2 CreateContextObject()

```
virtual UPhraseTreeContextObject * IPhraseContextNodeBase::CreateContextObject (
    FParseRecord & Record ) [protected], [pure virtual]
```

Creates a Context Object, using Record Inputs.

#### Returns

The Created Context Object, otherwise nullptr

Implemented in [FPhraseContextMenuNode< ContextMenuType >](#), and [FPhraseContextNode< ContextType >](#).

### 4.35.2.3 HasContextObject()

```
virtual bool IPhraseContextNodeBase::HasContextObject (
    TArray< UPhraseTreeContextObject * > InContextObjects ) const [protected], [pure
virtual]
```

Checks if the Given Context Array Contains Context Objects.

#### Parameters

|                         |   |
|-------------------------|---|
| <i>InContextObjects</i> | - The Array To Check For Context Objects. |
|-------------------------|---|

**Returns**

True, if their is Context Objects in the Given Array.

Implemented in [FPhraseContextMenuNode< ContextMenuType >](#), and [FPhraseContextNode< ContextType >](#).

The documentation for this class was generated from the following file:

- Source/OpenAccessibilityCommunication/Public/PhraseTree/IPhraseContextNode.h

## 4.36 IPhraseNodeBase Class Reference

### Public Member Functions

- virtual bool [IsLeafNode](#) () const =0  
*States if the Phrase Node is a LeafNode.*
- virtual bool [HasLeafChild](#) () const =0  
*States if the the Single Child Node is a Leaf Node, if it exists.*
- virtual bool [RequiresPhrase](#) (const FString InPhrase)=0  
*Checks if the Given Phrase is Bound to the Node.*
- virtual [FParseResult](#) [ParsePhrase](#) (TArray< FString > &InPhraseWordArray, [FParseRecord](#) &InParseRecord)=0  
*Parses the phrase down the given Node, propagating down child nodes if required.*
- virtual [FParseResult](#) [ParsePhraseAsContext](#) (TArray< FString > &InPhraseWordArray, [FParseRecord](#) &InParseRecord)=0  
*Parses the phrase down the given node, propagating down child nodes if required. Missed Pop of the Phrase Array from this Node.*

### 4.36.1 Detailed Description

Definition at line 10 of file [PhraseNode.h](#).

### 4.36.2 Member Function Documentation

#### 4.36.2.1 HasLeafChild()

```
virtual bool IPhraseNodeBase::HasLeafChild ( ) const [pure virtual]
```

States if the the Single Child Node is a Leaf Node, if it exists.

**Returns**

#### 4.36.2.2 IsLeafNode()

```
virtual bool IPhraseNodeBase::IsLeafNode ( ) const [pure virtual]
```

States if the Phrase Node is a LeafNode.

##### Returns

true, if the Node is a Leaf Node otherwise false.

#### 4.36.2.3 ParsePhrase()

```
virtual FParseResult IPhraseNodeBase::ParsePhrase (
    TArray< FString > & InPhraseWordArray,
    FParseRecord & InParseRecord ) [pure virtual]
```

Parses the phrase down the given Node, propagating down child nodes if required.

##### Parameters

|                          |  |
|--------------------------|--|
| <i>InPhraseWordArray</i> | The Array of Phrase Strings to Propagate against.            |
| <i>InParseRecord</i>     | The Record of Propagation of collected context's and inputs. |

##### Returns

Returns the Result of the propogation, including any key nodes met.

#### 4.36.2.4 ParsePhraseAsContext()

```
virtual FParseResult IPhraseNodeBase::ParsePhraseAsContext (
    TArray< FString > & InPhraseWordArray,
    FParseRecord & InParseRecord ) [pure virtual]
```

Parses the phrase down the given node, propagating down child nodes if required. Missed Pop of the Phrase Array from this Node.

##### Parameters

|                          |  |
|--------------------------|--|
| <i>InPhraseWordArray</i> |  |
| <i>InParseRecord</i>     |  |

##### Returns

Returns the Result of the propogation, including any key nodes met.

#### 4.36.2.5 RequiresPhrase()

```
virtual bool IPhraseNodeBase::RequiresPhrase (
    const FString InPhrase ) [pure virtual]
```

Checks if the Given Phrase is Bound to the Node.

##### Parameters

|                 |                                       |
|-----------------|---------------------------------------|
| <i>InPhrase</i> | The Phrase String to Compare Against. |
|-----------------|---------------------------------------|

##### Returns

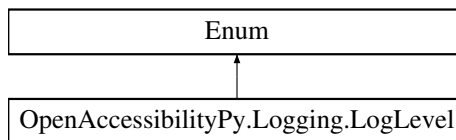
True, if the Node requires the given phrase string otherwise false.

The documentation for this class was generated from the following file:

- Source/OpenAccessibilityCommunication/Public/PhraseTree/PhraseNode.h

## 4.37 OpenAccessibilityPy.Logging.LogLevel Class Reference

Inheritance diagram for OpenAccessibilityPy.Logging.LogLevel:



### Static Public Attributes

- int [INFO](#) = 0
- int [WARNING](#) = 1
- int [ERROR](#) = 2

#### 4.37.1 Detailed Description

Definition at line 4 of file [Logging.py](#).

#### 4.37.2 Member Data Documentation

#### 4.37.2.1 ERROR

```
int OpenAccessibilityPy.Logging.LogLevel.ERROR = 2 [static]
```

Definition at line 7 of file [Logging.py](#).

#### 4.37.2.2 INFO

```
int OpenAccessibilityPy.Logging.LogLevel.INFO = 0 [static]
```

Definition at line 5 of file [Logging.py](#).

#### 4.37.2.3 WARNING

```
int OpenAccessibilityPy.Logging.LogLevel.WARNING = 1 [static]
```

Definition at line 6 of file [Logging.py](#).

The documentation for this class was generated from the following file:

- Content/Python/OpenAccessibilityPy/Logging.py

## 4.38 NumericParser Class Reference

### Static Public Member Functions

- static bool [IsValidNumeric](#) (const FString &StringToCheck, bool ConvertToUpper=true)  
*Checks if the String is a Valid Numeric in Comparison to its String Permutations.*
- static void [StringToNumeric](#) (FString &NumericString, bool ConvertToUpper=true)  
*Converts a String to its Numeric Permutation.*

#### 4.38.1 Detailed Description

Definition at line 7 of file [Utils.h](#).

#### 4.38.2 Member Function Documentation

##### 4.38.2.1 IsValidNumeric()

```
bool NumericParser::IsValidNumeric (  
    const FString & StringToCheck,  
    bool ConvertToUpper = true ) [static]
```

Checks if the String is a Valid Numeric in Comparison to its String Permutations.

**Parameters**

|                       |  |
|-----------------------|--|
| <i>StringToCheck</i>  | - The String To Check if it is a Numeric.                    |
| <i>ConvertToUpper</i> | - Should The String Be Converted To Upper before Comparison. |

**Returns**

Definition at line 7 of file [Utils.cpp](#).

```
00008 {  
00009     return StringMappings.Contains(ConvertToUpper ? StringToCheck.ToUpper() : StringToCheck);  
00010 }
```

**4.38.2.2 StringToNumeric()**

```
void NumericParser::StringToNumeric (  
    FString & NumericString,  
    bool ConvertToUpper = true ) [static]
```

Converts a String to its Numeric Permutation.

**Parameters**

|                       |  |
|-----------------------|--|
| <i>NumericString</i>  | - The String To Convert To Numeric.                          |
| <i>ConvertToUpper</i> | - Should The String Be Converted To Upper before Conversion. |

Definition at line 12 of file [Utils.cpp](#).

```
00013 {  
00014     if (const FString* FoundMapping = StringMappings.Find(NumericString))  
00015     {  
00016         NumericString = ConvertToUpper ? *FoundMapping->ToUpper() : *FoundMapping;  
00017     }  
00018     else UE_LOG(LogOpenAccessibilityCom, Warning, TEXT("|| Numeric Parser || No Mapping Found for  
String: %s ||"), *NumericString);  
00019 }
```

The documentation for this class was generated from the following files:

- Source/OpenAccessibilityCommunication/Public/PhraseTree/Utils.h
- Source/OpenAccessibilityCommunication/Private/PhraseTree/Utils.cpp

## 4.39 OAEEditorAccessibilityManager Class Reference

### 4.39.1 Detailed Description

Definition at line 10 of file [OAEEditorAccessibilityManager.h](#).

## 4.39.2 Constructor & Destructor Documentation

### 4.39.2.1 OAEditorAccessibilityManager()

OAEditorAccessibilityManager::OAEditorAccessibilityManager ( )

Definition at line 6 of file [OAEditorAccessibilityManager.cpp](#).

```
00007 {  
00008 }
```

### 4.39.2.2 ~OAEditorAccessibilityManager()

OAEditorAccessibilityManager::~~OAEditorAccessibilityManager ( )

Definition at line 10 of file [OAEditorAccessibilityManager.cpp](#).

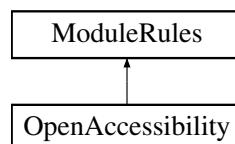
```
00011 {  
00012 }
```

The documentation for this class was generated from the following files:

- Source/OpenAccessibility/Public/OAEditorAccessibilityManager.h
- Source/OpenAccessibility/Private/OAEditorAccessibilityManager.cpp

## 4.40 OpenAccessibility Class Reference

Inheritance diagram for OpenAccessibility:



### Public Member Functions

- [OpenAccessibility](#) (ReadOnlyTargetRules Target)

### 4.40.1 Detailed Description

Definition at line 6 of file [OpenAccessibility.Build.cs](#).

### 4.40.2 Constructor & Destructor Documentation

#### 4.40.2.1 OpenAccessibility()

OpenAccessibility.OpenAccessibility (   
 ReadOnlyTargetRules Target ) [inline]

Definition at line 8 of file [OpenAccessibility.Build.cs](#).

```

00008                                     : base(Target)
00009 {
00010     PCHUsage = ModuleRules.PCHUsageMode.UseExplicitOrSharedPCHs;
00011
00012     PublicIncludePaths.AddRange (
00013         new string[] {
00014             // ... add public include paths required here ...
00015         }
00016     );
00017
00018     PrivateIncludePaths.AddRange(
00019         new string[] {
00020             // ... add other private include paths required here ...
00021         }
00022     );
00023
00024
00025     PublicDependencyModuleNames.AddRange (
00026         new string[]
00027         {
00028             "Core",
00029             // ... add other public dependencies that you statically link with here ...
00030         }
00031     );
00032
00033
00034     PrivateDependencyModuleNames.AddRange (
00035         new string[]
00036         {
00037             // Internal Plugin Modules
00038             "OpenAccessibilityCommunication",
00039
00040             // Core Modules
00041             "CoreUObject",
00042             "Engine",
00043             "Json",
00044
00045             // Editor Modules
00046             "UnrealEd",
00047             "GraphEditor",
00048             "Kismet",
00049             "AIModule",
00050
00051             // Slate UI
00052             "Slate",
00053             "SlateCore",
00054             "EditorStyle",
00055         }
00056     );
00057
00058
00059     DynamicallyLoadedModuleNames.AddRange (
00060         new string[]
00061         {
00062             // ... add any modules that your module loads dynamically here ...
00063         }
00064     );
00065
00066     CircularlyReferencedDependentModules.AddRange(
00067         new string[]
00068         {
00069
00070         }
00071     );
00072 }
```

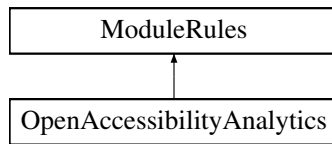
The documentation for this class was generated from the following file:

- [Source/OpenAccessibility/OpenAccessibility.Build.cs](#)



## 4.41 OpenAccessibilityAnalytics Class Reference

Inheritance diagram for OpenAccessibilityAnalytics:



### Public Member Functions

- [OpenAccessibilityAnalytics](#) (ReadOnlyTargetRules Target)

#### 4.41.1 Detailed Description

Definition at line 6 of file [OpenAccessibilityAnalytics.Build.cs](#).

#### 4.41.2 Constructor & Destructor Documentation

##### 4.41.2.1 OpenAccessibilityAnalytics()

`OpenAccessibilityAnalytics.OpenAccessibilityAnalytics (`  
     `ReadOnlyTargetRules Target ) [inline]`

Definition at line 8 of file [OpenAccessibilityAnalytics.Build.cs](#).

```

00008                                     : base(Target)
00009     {
00010         PCHUsage = ModuleRules.PCHUsageMode.UseExplicitOrSharedPCHs;
00011
00012         PublicIncludePaths.AddRange (
00013             new string[] {
00014                 // ... add public include paths required here ...
00015             }
00016         );
00017
00018         PrivateIncludePaths.AddRange (
00019             new string[] {
00020                 // ... add other private include paths required here ...
00021             }
00022         );
00023
00024
00025         PublicDependencyModuleNames.AddRange (
00026             new string[]
00027             {
00028                 "Core",
00029                 // ... add other public dependencies that you statically link with here ...
00030             }
00031         );
00032
00033
00034         PrivateDependencyModuleNames.AddRange (
00035             new string[]
00036             {
00037                 "Engine",
00038             }
00039         );
00040

```

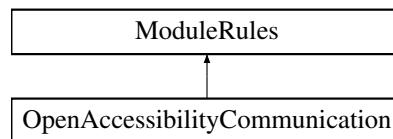
```
00041
00042     DynamicallyLoadedModuleNames.AddRange(
00043         new string[]
00044         {
00045             // ... add any modules that your module loads dynamically here ...
00046         }
00047     );
00048
00049     CircularlyReferencedDependentModules.AddRange(
00050         new string[]
00051         {
00052         }
00053     );
00054 }
00055 }
```

The documentation for this class was generated from the following file:

- [Source/OpenAccessibilityAnalytics/OpenAccessibilityAnalytics.Build.cs](#)

## 4.42 OpenAccessibilityCommunication Class Reference

Inheritance diagram for OpenAccessibilityCommunication:



### Public Member Functions

- [OpenAccessibilityCommunication](#) (ReadOnlyTargetRules Target)

#### 4.42.1 Detailed Description

Definition at line 7 of file [OpenAccessibilityCommunication.Build.cs](#).

#### 4.42.2 Constructor & Destructor Documentation

## 4.42.2.1 OpenAccessibilityCommunication()

OpenAccessibilityCommunication.OpenAccessibilityCommunication (   
 ReadOnlyTargetRules Target ) [inline]

Definition at line 9 of file [OpenAccessibilityCommunication.Build.cs](#).

```

00009                                     : base(Target)
00010     {
00011         PCHUsage = ModuleRules.PCHUsageMode.UseExplicitOrSharedPCHs;
00012
00013         PublicIncludePaths.AddRange (
00014             new string[] {
00015                 // ... add public include paths required here ...
00016             }
00017         );
00018
00019         PrivateIncludePaths.AddRange (
00020             new string[] {
00021                 // ... add other private include paths required here ...
00022             }
00023         );
00024
00025
00026         PublicDependencyModuleNames.AddRange (
00027             new string[]
00028             {
00029                 "Core",
00030                 // ... add other public dependencies that you statically link with here ...
00031             }
00032         );
00033
00034         PrivateDependencyModuleNames.AddRange (
00035             new string[]
00036             {
00037                 // Internal Plugin Dependencies
00038                 "OpenAccessibilityAnalytics",
00039
00040                 // Internal ThirdParty Dependencies
00041                 "ZeroMQ",
00042
00043                 // Core Modules
00044                 "CoreUObject",
00045                 "Engine",
00046                 "Json",
00047
00048                 // Editor Modules
00049                 "UnrealEd",
00050                 "Projects",
00051
00052                 // Slate UI Modules
00053                 "Slate",
00054                 "SlateCore",
00055
00056                 // Audio Modules
00057                 "AudioMixer",
00058                 "AudioCaptureCore",
00059                 "AudioCapture",
00060                 "InputCore",
00061             }
00062         );
00063
00064
00065         DynamicallyLoadedModuleNames.AddRange (
00066             new string[]
00067             {
00068                 // ... add any modules that your module loads dynamically here ...
00069             }
00070         );
00071
00072         CircularlyReferencedDependentModules.AddRange (
00073             new string[]
00074             {
00075
00076             }
00077         );
00078     }

```

The documentation for this class was generated from the following file:

- [Source/OpenAccessibilityCommunication/OpenAccessibilityCommunication.Build.cs](#)

## 4.43 OpenAccessibilityPy.OpenAccessibilityPy Class Reference

### Public Member Functions

- def `__init__` (self, int worker\_count=2, str whisper\_model="distil-small.en", str device="auto", str compute\_type="default", int poll\_timeout=10)
- def `__del__` (self)
- def `Tick` (self, float delta\_time)
- def `HandleTranscriptionRequest` (self, np.ndarray recv\_message, dict metadata=None)
- def `Shutdown` (self)

### Public Attributes

- `worker_pool`
- `whisper_interface`
- `com_server`
- `audio_resampler`
- `tick_handle`
- `pyshutdown_handle`

#### 4.43.1 Detailed Description

Python Runtime Class for Open Accessibility Plugin

Definition at line 34 of file `__init__.py`.

#### 4.43.2 Constructor & Destructor Documentation

##### 4.43.2.1 `__init__()`

```
def OpenAccessibilityPy.OpenAccessibilityPy.__init__ (
    self,
    int worker_count = 2,
    str whisper_model = "distil-small.en",
    str device = "auto",
    str compute_type = "default",
    int poll_timeout = 10 )
```

Constructor of Python Runtime Class for Open Accessibility Plugin

Args:

`worker_count` (int, optional): Amount of Thread Workers for Audio Transcription. Defaults to 2.  
`whisper_model` (str, optional): Hugging-Face Model Specifier for CTranslate Compatible Models. Defaults to "aut  
`device` (str, optional): Device host for the Whisper Model (Can be "auto", "cpu", "cuda"). Defaults to "aut  
`compute_type` (str, optional): Data Structure for Whisper Compute. Defaults to "default".  
`poll_timeout` (int, optional): Amount of time (ms) for event polling on the Transcription Socket. Defaults

Definition at line 37 of file `__init__.py`.

```

00047     ):
00048         """Constructor of Python Runtime Class for Open Accessibility Plugin
00049
00050         Args:
00051             worker_count (int, optional): Amount of Thread Workers for Audio Transcription. Defaults
to 2.
00052             whisper_model (str, optional): Hugging-Face Model Specifier for CTranslate Compatible
Models. Defaults to "distil-small.en".
00053             device (str, optional): Device host for the Whisper Model (Can be "auto", "cpu", "cuda").
Defaults to "auto".
00054             compute_type (str, optional): Data Structure for Whisper Compute. Defaults to "default".
00055             poll_timeout (int, optional): Amount of time (ms) for event polling on the Transcription
Socket. Defaults to 10.
00056         """
00057
00058         self.worker_pool = ThreadPool(
00059             max_workers=worker_count, thread_name_prefix="TranscriptionWorker"
00060         )
00061
00062         self.whisper_interface = WhisperInterface(
00063             model_name=whisper_model,
00064             device=device,
00065             compute_type=compute_type,
00066             transcribe_workers=worker_count,
00067         )
00068         self.com_server = CommunicationServer(
00069             send_socket_type=zmq.PUSH,
00070             recv_socket_type=zmq.PULL,
00071             poll_timeout=poll_timeout,
00072         )
00073         self.audio_resampler = AudioResampler(target_sample_rate=16000)
00074
00075         self.tick_handle = ue.register_slate_post_tick_callback(self.Tick)
00076
00077         self.pyshutdown_handle = ue.register_python_shutdown_callback(self.Shutdown)
00078

```

#### 4.43.2.2 `__del__()`

```

def OpenAccessibilityPy.OpenAccessibilityPy.__del__ (
    self )

```

Destructor of Python Runtime Class for Open Accessibility Plugin

Definition at line 79 of file `__init__.py`.

```

00079     def __del__(self):
00080         """Destructor of Python Runtime Class for Open Accessibility Plugin"""
00081
00082         self.Shutdown()
00083

```

### 4.43.3 Member Function Documentation

#### 4.43.3.1 `HandleTranscriptionRequest()`

```

def OpenAccessibilityPy.OpenAccessibilityPy.HandleTranscriptionRequest (
    self,
    np.ndarray recv_message,
    dict metadata = None )

```

Handles Incoming Transcription Requests

Takes the Incoming AudioBuffer, Resamples it to 16kHz and Transcribes it using Whisper.

Args:

recv\_message (np.ndarray): ndarray of the incoming audio buffer.  
 metadata (dict, optional): metadata of the incoming audio buffer, if any is recieved. Defaults to None.

Definition at line 100 of file `__init__.py`.

```

00102     ):
00103         """Handles Incoming Transcription Requests
00104
00105         Takes the Incoming AudioBuffer, Resamples it to 16kHz and Transcribes it using Whisper.
00106
00107         Args:
00108             recv_message (np.ndarray): ndarray of the incoming audio buffer.
00109             metadata (dict, optional): metadata of the incoming audio buffer, if any is recieved.
00110             Defaults to None.
00111         """
00112         Log(
00113             f"Handling Transcription Request | Message: {recv_message} | Size: {recv_message.size} |
00114             Shape: {recv_message.shape}"
00115         )
00116         sample_rate = metadata.get("sample_rate", 48000)
00117         num_channels = metadata.get("num_channels", 2)
00118         message_ndarray = self.audio_resampler.resample(
00119             recv_message, sample_rate, num_channels
00120         )
00121
00122         trans_segments, trans_metadata = self.whisper_interface.process_audio_buffer(
00123             message_ndarray
00124         )
00125
00126         encoded_segments = [
00127             transcription.text.encode() for transcription in trans_segments
00128         ]
00129
00130         Log(f"Encoded Segments: {encoded_segments}")
00131
00132         if len(encoded_segments) > 0:
00133             try:
00134                 self.com_server.SendMultipartWithMeta(
00135                     message=encoded_segments, meta=trans_metadata
00136                 )
00137             except:
00138                 Log("Error Sending Encoded Transcription Segments", LogLevel.ERROR)
00139
00140         else:
00141             Log("No Transcription Segments Returned", LogLevel.WARNING)
00142
00143
00144

```

#### 4.43.3.2 Shutdown()

```

def OpenAccessibilityPy.OpenAccessibilityPy.Shutdown (
    self )

```

Shutsdown the Python Runtime Components, and Forces a Garbage Collection.

Definition at line 145 of file `__init__.py`.

```

00145     def Shutdown(self):
00146         """Shutsdown the Python Runtime Components, and Forces a Garbage Collection."""
00147
00148         if self.tick_handle:
00149             ue.unregister_slate_post_tick_callback(self.tick_handle)
00150             del self.tick_handle
00151
00152         if self.worker_pool:
00153             self.worker_pool.shutdown(wait=False, cancel_futures=True)

```

```

00154         del self.worker_pool
00155
00156         if self.audio_resampler:
00157             del self.audio_resampler
00158
00159         if self.com_server:
00160             del self.com_server
00161
00162         if self.whisper_interface:
00163             del self.whisper_interface
00164
00165         # Force a Garbage Collection
00166         gc_collect()

```

#### 4.43.3.3 Tick()

```

def OpenAccessibilityPy.OpenAccessibilityPy.Tick (
    self,
    float delta_time )

```

Tick Callback for Unreal Engine Slate Post Tick.

Detecting Incoming Transcription Requests and Handling them, through the Worker Pool.

Args:

delta\_time (float): Time since last tick

Definition at line 84 of file [\\_\\_init\\_\\_.py](#).

```

00084     def Tick(self, delta_time: float):
00085         """Tick Callback for Unreal Engine Slate Post Tick.
00086
00087         Detecting Incoming Transcription Requests and Handling them, through the Worker Pool.
00088
00089         Args:
00090             delta_time (float): Time since last tick
00091         """
00092
00093         if self.com_server.EventOccured():
00094             Log("Event Occured")
00095
00096             message, metadata = self.com_server.ReceiveNDArrayWithMeta(dtype=np.float32)
00097
00098             self.worker_pool.submit(self.HandleTranscriptionRequest, message, metadata)
00099

```

### 4.43.4 Member Data Documentation

#### 4.43.4.1 audio\_resampler

OpenAccessibilityPy.OpenAccessibilityPy.audio\_resampler

Definition at line 73 of file [\\_\\_init\\_\\_.py](#).

#### 4.43.4.2 com\_server

OpenAccessibilityPy.OpenAccessibilityPy.com\_server

Definition at line 68 of file [\\_\\_init\\_\\_.py](#).

#### 4.43.4.3 pyshutdown\_handle

OpenAccessibilityPy.OpenAccessibilityPy.pyshutdown\_handle

Definition at line 77 of file [\\_\\_init\\_\\_.py](#).

#### 4.43.4.4 tick\_handle

OpenAccessibilityPy.OpenAccessibilityPy.tick\_handle

Definition at line 75 of file [\\_\\_init\\_\\_.py](#).

#### 4.43.4.5 whisper\_interface

OpenAccessibilityPy.OpenAccessibilityPy.whisper\_interface

Definition at line 62 of file [\\_\\_init\\_\\_.py](#).

#### 4.43.4.6 worker\_pool

OpenAccessibilityPy.OpenAccessibilityPy.worker\_pool

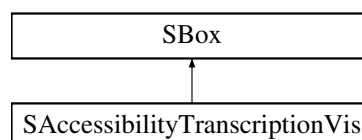
Definition at line 58 of file [\\_\\_init\\_\\_.py](#).

The documentation for this class was generated from the following file:

- Content/Python/OpenAccessibilityPy/\_\_init\_\_.py

## 4.44 SAccessibilityTranscriptionVis Class Reference

Inheritance diagram for SAccessibilityTranscriptionVis:





## Public Member Functions

- [SLATE\\_BEGIN\\_ARGS](#) ([SAccessibilityTranscriptionVis](#))
- void [Construct](#) (const FArguments &InArgs)
- virtual void [Tick](#) (const FGeometry &AllottedGeometry, const double InCurrentTime, const float InDeltaTime) override
- void [UpdateTopTranscription](#) (const FString &InTopTranscription)

*Updates the Top Transcription Text, shifting all current transcriptions down.*

## Protected Attributes

- TWeakPtr< SVerticalBox > [TranscriptionContainer](#)  
*The Container of the Transcription Slots.*
- TArray< TWeakPtr< STextBlock > > [TranscriptionSlots](#)  
*Array of the created Transcription Slots, displaying recieved transcriptions.*

### 4.44.1 Detailed Description

Definition at line 9 of file [SAccessibilityTranscriptionVis.h](#).

### 4.44.2 Constructor & Destructor Documentation

#### 4.44.2.1 ~SAccessibilityTranscriptionVis()

```
SAccessibilityTranscriptionVis::~SAccessibilityTranscriptionVis ( )
```

Definition at line 5 of file [SAccessibilityTranscriptionVis.cpp](#).

```
00006 {  
00007 }
```

### 4.44.3 Member Function Documentation

#### 4.44.3.1 Construct()

```
void SAccessibilityTranscriptionVis::Construct (
    const FArguments & InArgs )
```

Definition at line 9 of file [SAccessibilityTranscriptionVis.cpp](#).

```
00010 {
00011     // Transcription Holder
00012     TSharedPtr<SVerticalBox> TranscriptionHolder = SNew(SVerticalBox)
00013         + SVerticalBox::Slot()
00014         .Padding(4.0f)
00015         .AutoHeight();
00016
00017     // Verify a least one slot will be constructed
00018     int TranscriptionSlotAmount = FMath::Max(1, InArgs._VisAmount);
00019
00020     FSlateFontInfo FontInfo = FAppStyle::GetFontStyle("NormalText");
00021     FontInfo.Size = 12;
00022
00023     TSharedPtr<STextBlock> CurrentTranscriptionSlot;
00024     for (int i = 0; i < TranscriptionSlotAmount; i++)
00025     {
00026         TranscriptionHolder->AddSlot()
00027             .HAlign(HAlign_Center)
00028             .Padding(4.0f)
00029             .AutoHeight()
00030             [
00031                 SAssignNew(CurrentTranscriptionSlot, STextBlock)
00032                 .Text(FText::GetEmpty())
00033                 .Font(FontInfo)
00034                 .SimpleTextMode(true)
00035                 .ColorAndOpacity(i == 0 ? FSlateColor(FLinearColor(1.0f, 1.0f, 0, 1.0f)) :
00036                     FSlateColor(FLinearColor(0.5f, 0.5f, 0.5f, 1.0f)))
00037             ];
00038         TranscriptionSlots.Add(CurrentTranscriptionSlot);
00039     }
00040
00041     // Construct the Main Component
00042
00043     ChildSlot
00044     .Padding(FMargin(5.0f))
00045     [
00046         SNew(SOverlay)
00047         + SOverlay::Slot()
00048         .ZOrder(1)
00049         [
00050             SNew(SBorder)
00051             .BorderBackgroundColor(FSlateColor(FLinearColor::Gray))
00052             [
00053                 SNew(SBox)
00054                 .MinDesiredWidth(250.0f)
00055                 .MinDesiredHeight(60.0f)
00056                 [
00057                     TranscriptionHolder.ToSharedRef()
00058                 ]
00059             ]
00060         ]
00061     ];
00062     this->TranscriptionContainer = TranscriptionHolder;
00064 }
```

#### 4.44.3.2 SLATE\_BEGIN\_ARGS()

```
SAccessibilityTranscriptionVis::SLATE_BEGIN_ARGS (
    SAccessibilityTranscriptionVis ) [inline]
```

Definition at line 13 of file [SAccessibilityTranscriptionVis.h](#).

```
00014     : _VisAmount(1)
00015     {}
```

#### 4.44.3.3 Tick()

```
void SAccessibilityTranscriptionVis::Tick (
    const FGeometry & AllottedGeometry,
    const double InCurrentTime,
    const float InDeltaTime ) [override], [virtual]
```

Definition at line 66 of file [SAccessibilityTranscriptionVis.cpp](#).

```
00067 {
00068     SBox::Tick(AllottedGeometry, InCurrentTime, InDeltaTime);
00069 }
```

#### 4.44.3.4 UpdateTopTranscription()

```
void SAccessibilityTranscriptionVis::UpdateTopTranscription (
    const FString & InTopTranscription )
```

Updates the Top Transcription Text, shifting all current transcriptions down.

Definition at line 71 of file [SAccessibilityTranscriptionVis.cpp](#).

```
00072 {
00073     FString LastTopText = InTopTranscription;
00074     FString TempText;
00075
00076     TSharedPtr<STextBlock> CurrentTranscriptionSlot;
00077     for (TWeakPtr<STextBlock>& TranscriptionSlot : TranscriptionSlots)
00078     {
00079         CurrentTranscriptionSlot = TranscriptionSlot.Pin();
00080
00081         TempText = FString(CurrentTranscriptionSlot->GetText().ToString());
00082         CurrentTranscriptionSlot->SetText(FText::FromString(LastTopText));
00083
00084         CurrentTranscriptionSlot->Invalidate(EInvalidateWidgetReason::PaintAndVolatility);
00085
00086         LastTopText = TempText;
00087     }
00088
00089     TranscriptionContainer.Pin()->Invalidate(EInvalidateWidget::Layout);
00090 }
```

### 4.44.4 Member Data Documentation

#### 4.44.4.1 TranscriptionContainer

```
TWeakPtr<SVerticalBox> SAccessibilityTranscriptionVis::TranscriptionContainer [protected]
```

The Container of the Transcription Slots.

Definition at line 39 of file [SAccessibilityTranscriptionVis.h](#).

#### 4.44.4.2 TranscriptionSlots

```
TArray<TWeakPtr<STextBlock> > SAccessibilityTranscriptionVis::TranscriptionSlots [protected]
```

Array of the created Transcription Slots, displaying recieved transcriptions.

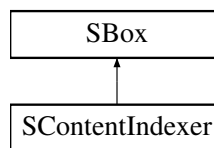
Definition at line 44 of file [SAccessibilityTranscriptionVis.h](#).

The documentation for this class was generated from the following files:

- Source/OpenAccessibility/Public/AccessibilityWidgets/SAccessibilityTranscriptionVis.h
- Source/OpenAccessibility/Private/AccessibilityWidgets/SAccessibilityTranscriptionVis.cpp

### 4.45 SContentIndexer Class Reference

Inheritance diagram for SContentIndexer:



#### Public Member Functions

- [SLATE\\_BEGIN\\_ARGS](#) ([SContentIndexer](#))
- **SLATE\_PRIVATE\_ATTRIBUTE\_VARIABLE** (EVisibility, IndexVisibility)
- void [Construct](#) (const FArguments &InArgs)
- virtual void [Tick](#) (const FGeometry &AllottedGeometry, const double InCurrentTime, const float InDeltaTime) override
- void [UpdateIndex](#) (const int32 IndexValue)  
*Updates the Index Value Displayed on the Indexer Text Widget.*
- TSharedRef< SWidget > [GetContent](#) () const  
*Gets the Current Content Being Indexed.*
- template<typename CastType >  
TSharedRef< CastType > [GetContent](#) () const  
*Gets the Current Content Being Indexed and Casts it to the Provided Type.*

#### Protected Member Functions

- TSharedPtr< SWidget > [ConstructTopIndexer](#) (const FArguments &InArgs)  
*Constructs the Indexer Widget with the Index on Top of the Content.*
- TSharedPtr< SWidget > [ConstructBottomIndexer](#) (const FArguments &InArgs)  
*Constructs the Indexer Widget with the Index Below the Content.*
- TSharedPtr< SWidget > [ConstructLeftIndexer](#) (const FArguments &InArgs)  
*Constructs the Indexer Widget with the Index to the Left of the Content.*
- TSharedPtr< SWidget > [ConstructRightIndexer](#) (const FArguments &InArgs)  
*Constructs the Indexer Widget with the Index to the Right of the Content.*
- TSharedPtr< SWidget > [ConstructContentContainer](#) (TSharedRef< SWidget > ContentToIndex)  
*Constructs the Container for the Indexer with the provided Content.*
- TSharedPtr< SWidget > [ConstructIndexContainer](#) (const FArguments &InArgs, FLinearColor Text← Color=FLinearColor::White)  
*Constructs the Indexer Widget with the provided Index Value.*
- FText [ConstructIndexText](#) (int32 Index)  
*Creates the Text Element of the Provided Index Value.*

## Protected Attributes

- TWeakPtr< SWidget > [IndexedContent](#)  
*The Content That The Indexer Is Currently Indexing.*
- TWeakPtr< class [SIndexer](#) > [IndexerWidget](#)  
*The Text Block that Displays the Index Value.*

### 4.45.1 Detailed Description

Definition at line 16 of file [SContentIndexer.h](#).

### 4.45.2 Constructor & Destructor Documentation

#### 4.45.2.1 ~SContentIndexer()

```
SContentIndexer::~SContentIndexer ( )
```

Definition at line 6 of file [SContentIndexer.cpp](#).

```
00007 {
00008 }
```

### 4.45.3 Member Function Documentation

#### 4.45.3.1 Construct()

```
void SContentIndexer::Construct (
    const FArguments & InArgs )
```

Definition at line 10 of file [SContentIndexer.cpp](#).

```
00011 {
00012     TSharedPtr<SWidget> Content;
00013
00014     switch (InArgs._IndexPositionToContent)
00015     {
00016         case EIndexerPosition::Top:
00017             Content = ConstructTopIndexer(InArgs);
00018             break;
00019
00020         case EIndexerPosition::Bottom:
00021             Content = ConstructBottomIndexer(InArgs);
00022             break;
00023
00024         default:
00025             case EIndexerPosition::Left:
00026                 Content = ConstructLeftIndexer(InArgs);
00027                 break;
00028
00029             case EIndexerPosition::Right:
00030                 Content = ConstructRightIndexer(InArgs);
00031                 break;
00032     }
00033
00034     ChildSlot
00035     [
00036         Content.ToSharedRef()
00037     ];
00038 }
```

#### 4.45.3.2 ConstructBottomIndexer()

```
TSharedPtr< SWidget > SContentIndexer::ConstructBottomIndexer (
    const FArguments & InArgs ) [protected]
```

Constructs the Indexer Widget with the Index Below the Content.

##### Parameters

|                       |   |
|-----------------------|---|
| <i>IndexValue</i>     | The Index Value to Index.                 |
| <i>ContentToIndex</i> | The Content that the Indexer is Wrapping. |

##### Returns

Definition at line 74 of file [SContentIndexer.cpp](#).

```
00075 {
00076     return SNew(SVerticalBox)
00077         .Visibility(AccessWidgetVisibilityAttribute(InArgs._ContentToIndex.ToSharedRef()))
00078
00079         + SVerticalBox::Slot()
00080         .HAlign(HAlign_Center)
00081         .VAlign(VAlign_Center)
00082         .AutoHeight()
00083         [
00084             ConstructContentContainer(InArgs._ContentToIndex.ToSharedRef()).ToSharedRef()
00085         ]
00086
00087         + SVerticalBox::Slot()
00088         .HAlign(HAlign_Center)
00089         .VAlign(VAlign_Center)
00090         .AutoHeight()
00091         .Padding(.1f, .25f)
00092         [
00093             ConstructIndexContainer(InArgs).ToSharedRef()
00094         ];
00095 }
```

#### 4.45.3.3 ConstructContentContainer()

```
TSharedPtr< SWidget > SContentIndexer::ConstructContentContainer (
    TSharedPtr< SWidget > ContentToIndex ) [protected]
```

Constructs the Container for the Indexer with the provided Content.

##### Parameters

|                       |  |
|-----------------------|--|
| <i>ContentToIndex</i> | The Content that needs to be wrapped with an Indexer Widget. |
|-----------------------|--|

##### Returns

Definition at line 143 of file [SContentIndexer.cpp](#).

```
00144 {
00145     IndexedContent = ContentToIndex;
```

```
00146     return ContentToIndex;
00147 }
```

#### 4.45.3.4 ConstructIndexContainer()

```
TSharedPtr< SWidget > SContentIndexer::ConstructIndexContainer (
    const FArguments & InArgs,
    FLinearColor TextColor = FLinearColor::White ) [protected]
```

Constructs the Indexer Widget with the provided Index Value.

##### Parameters

|                   |  |
|-------------------|--|
| <i>IndexValue</i> | The Index Value to be displayed in the Indexer Widget. |
| <i>TextColor</i>  | The Color of the Text displaying the Index.            |

##### Returns

Definition at line 149 of file [SContentIndexer.cpp](#).

```
00150 {
00151     return SAssignNew(IndexerWidget, SIndexer)
00152         .TextColor(TextColor)
00153         .BorderColor(FLinearColor::Gray)
00154         .IndexValue(InArgs._IndexValue)
00155         .IndexVisibility(InArgs._IndexVisibility);
00156 }
```

#### 4.45.3.5 ConstructIndexText()

```
FText SContentIndexer::ConstructIndexText (
    int32 Index ) [protected]
```

Creates the Text Element of the Provided Index Value.

##### Parameters

|              |                                 |
|--------------|---------------------------------|
| <i>Index</i> | The Index to convert into Text. |
|--------------|---------------------------------|

##### Returns

Definition at line 158 of file [SContentIndexer.cpp](#).

```
00159 {
00160     return FText::FromString(FString::FromInt(Index));
00161 }
```

#### 4.45.3.6 ConstructLeftIndexer()

```
TSharedPtr< SWidget > SContentIndexer::ConstructLeftIndexer (
    const FArguments & InArgs ) [protected]
```

Constructs the Indexer Widget with the Index to the Left of the Content.

##### Parameters

|                       |   |
|-----------------------|---|
| <i>IndexValue</i>     | The Index Value to Index.                 |
| <i>ContentToIndex</i> | The Content that the Indexer is Wrapping. |

##### Returns

Definition at line 97 of file [SContentIndexer.cpp](#).

```
00098 {
00099     return SNew(SHorizontalBox)
00100         .Visibility(AccessWidgetVisibilityAttribute(InArgs._ContentToIndex.ToSharedRef()))
00101
00102         + SHorizontalBox::Slot()
00103         .VAlign(VAlign_Center)
00104         .HAlign(HAlign_Center)
00105         .AutoWidth()
00106         .Padding(.25f, .1f)
00107         [
00108             ConstructIndexContainer(InArgs).ToSharedRef()
00109         ]
00110
00111         + SHorizontalBox::Slot()
00112         .VAlign(VAlign_Center)
00113         .HAlign(HAlign_Center)
00114         .AutoWidth()
00115         [
00116             ConstructContentContainer(InArgs._ContentToIndex.ToSharedRef()).ToSharedRef()
00117         ];
00118 }
```

#### 4.45.3.7 ConstructRightIndexer()

```
TSharedPtr< SWidget > SContentIndexer::ConstructRightIndexer (
    const FArguments & InArgs ) [protected]
```

Constructs the Indexer Widget with the Index to the Right of the Content.

##### Parameters

|                       |   |
|-----------------------|---|
| <i>IndexValue</i>     | The Index Value to Index.                 |
| <i>ContentToIndex</i> | The Content that the Indexer is Wrapping. |

##### Returns

Definition at line 120 of file [SContentIndexer.cpp](#).



```

00121 {
00122     return SNew(SHorizontalBox)
00123     .Visibility(AccessWidgetVisibilityAttribute(InArgs._ContentToIndex.ToSharedRef()))
00124
00125     + SHorizontalBox::Slot()
00126     .VAlign(VAlign_Center)
00127     .HAlign(HAlign_Center)
00128     .AutoWidth()
00129     [
00130         ConstructContentContainer(InArgs._ContentToIndex.ToSharedRef()).ToSharedRef()
00131     ]
00132
00133     + SHorizontalBox::Slot()
00134     .VAlign(VAlign_Center)
00135     .HAlign(HAlign_Center)
00136     .AutoWidth()
00137     .Padding(.25f, .1f)
00138     [
00139         ConstructIndexContainer(InArgs).ToSharedRef()
00140     ];
00141 }

```

#### 4.45.3.8 ConstructTopIndexer()

```

TSharedPtr< SWidget > SContentIndexer::ConstructTopIndexer (
    const FArguments & InArgs ) [protected]

```

Constructs the Indexer Widget with the Index on Top of the Content.

##### Parameters

|                       |   |
|-----------------------|---|
| <i>IndexValue</i>     | The Index Value to Index.                 |
| <i>ContentToIndex</i> | The Content that the Indexer is Wrapping. |

##### Returns

Definition at line 51 of file [SContentIndexer.cpp](#).

```

00052 {
00053     return SNew(SVerticalBox)
00054     .Visibility(AccessWidgetVisibilityAttribute(InArgs._ContentToIndex.ToSharedRef()))
00055
00056     + SVerticalBox::Slot()
00057     .HAlign(HAlign_Center)
00058     .VAlign(VAlign_Center)
00059     .AutoHeight()
00060     .Padding(.1f, .25f)
00061     [
00062         ConstructIndexContainer(InArgs).ToSharedRef()
00063     ]
00064
00065     + SVerticalBox::Slot()
00066     .HAlign(HAlign_Center)
00067     .VAlign(VAlign_Center)
00068     .AutoHeight()
00069     [
00070         ConstructContentContainer(InArgs._ContentToIndex.ToSharedRef()).ToSharedRef()
00071     ];
00072 }

```

**4.45.3.9 GetContent() [1/2]**

```
TSharedRef< SWidget > SContentIndexer::GetContent ( ) const [inline]
```

Gets the Current Content Being Indexed.

**Returns**

A Shared Ptr of the Indexed Content

Definition at line 54 of file [SContentIndexer.h](#).

```
00055     {
00056         return IndexedContent.Pin().ToSharedRef();
00057     }
```

**4.45.3.10 GetContent() [2/2]**

```
template<typename CastType >
TSharedRef< CastType > SContentIndexer::GetContent ( ) const [inline]
```

Gets the Current Content Being Indexed and Casts it to the Provided Type.

**Template Parameters**

|                 |                                       |
|-----------------|---------------------------------------|
| <i>CastType</i> | The Type To Cast The Stored Value To. |
|-----------------|---------------------------------------|

**Returns**

The Casted SharedReference.

Definition at line 66 of file [SContentIndexer.h](#).

```
00067     {
00068         return CastStaticSharedPtr<CastType>(IndexedContent.Pin());
00069     }
```

**4.45.3.11 SLATE\_BEGIN\_ARGS()**

```
SContentIndexer::SLATE_BEGIN_ARGS (
    SContentIndexer ) [inline]
```

Definition at line 20 of file [SContentIndexer.h](#).

```
00021     : _IndexValue(0)
00022     , _IndexPositionToContent(EIndexerPosition::Left)
00023     , _ContentToIndex(SNullWidget::NullWidget)
00024     {}
```

#### 4.45.3.12 Tick()

```
void SContentIndexer::Tick (
    const FGeometry & AllottedGeometry,
    const double InCurrentTime,
    const float InDeltaTime ) [override], [virtual]
```

Definition at line 40 of file [SContentIndexer.cpp](#).

```
00041 {
00042     SBox::Tick(AllottedGeometry, InCurrentTime, InDeltaTime);
00043 }
```

#### 4.45.3.13 UpdateIndex()

```
void SContentIndexer::UpdateIndex (
    const int32 IndexValue )
```

Updates the Index Value Displayed on the Indexer Text Widget.

##### Parameters

|                   |                                 |
|-------------------|---------------------------------|
| <i>IndexValue</i> | The New Interger Index to Show. |
|-------------------|---------------------------------|

Definition at line 45 of file [SContentIndexer.cpp](#).

```
00046 {
00047     if (IndexerWidget.IsValid())
00048         IndexerWidget.Pin()->UpdateIndex( IndexValue );
00049 }
```

### 4.45.4 Member Data Documentation

#### 4.45.4.1 IndexedContent

```
TWeakPtr<SWidget> SContentIndexer::IndexedContent [protected]
```

The Content That The Indexer Is Currently Indexing.

Definition at line 132 of file [SContentIndexer.h](#).

#### 4.45.4.2 IndexerWidget

```
TWeakPtr<class SIndexer> SContentIndexer::IndexerWidget [protected]
```

The Text Block that Displays the Index Value.

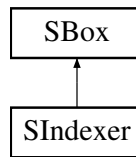
Definition at line 137 of file [SContentIndexer.h](#).

The documentation for this class was generated from the following files:

- Source/OpenAccessibility/Public/AccessibilityWidgets/SContentIndexer.h
- Source/OpenAccessibility/Private/AccessibilityWidgets/SContentIndexer.cpp

## 4.46 SIndexer Class Reference

Inheritance diagram for SIndexer:



### Public Member Functions

- [SLATE\\_BEGIN\\_ARGS](#) ([SIndexer](#))
- **SLATE\_PRIVATE\_ARGUMENT\_VARIABLE** (int32, IndexValue)
- **SLATE\_PRIVATE\_ATTRIBUTE\_VARIABLE** (EVisibility, IndexVisibility)
- virtual void [Tick](#) (const FGeometry &AllotedGeometry, const double InCurrentTime, const float InDeltaTime) override
- void [Construct](#) (const FArguments &InArgs)
- void [UpdateIndex](#) (const int32 NewIndex)  
*Updates the Index Widget with the New Index Value.*
- void [UpdateIndex](#) (const FString &NewIndex)  
*Updates the Index Widget with the New String Index Value.*
- void [UpdateIndex](#) (const FText &NewIndex)  
*Updates the Index Widget with the New Text Index Value.*
- TSharedPtr< STextBlock > [GetIndexText](#) () const  
*Gets the Index TextBlock Widget.*

### Protected Attributes

- TWeakPtr< STextBlock > [IndexTextBlock](#)  
*Weak Pointer to the Main TextBlock Widget.*

#### 4.46.1 Detailed Description

Definition at line 7 of file [SIndexer.h](#).

#### 4.46.2 Constructor & Destructor Documentation

##### 4.46.2.1 ~SIndexer()

```
SIndexer::~SIndexer ( )
```

Definition at line 5 of file [SIndexer.cpp](#).

```
00006 {
00007
00008 }
```

### 4.46.3 Member Function Documentation

#### 4.46.3.1 Construct()

```
void SIndexer::Construct (
    const FArguments & InArgs )
```

Definition at line 15 of file [SIndexer.cpp](#).

```
00016 {
00017     ChildSlot
00018     [
00019         SNew(SBorder)
00020         .HAlign(HAlign_Center)
00021         .VAlign(VAlign_Center)
00022         .Visibility(InArgs._IndexVisibility)
00023         .Padding(FMargin(4.f, 2.f))
00024         .BorderBackgroundColor( FSlateColor(InArgs._BorderColor) )
00025     [
00026         SAssignNew(IndexTextBlock, STextBlock)
00027         .Text( FText::FromString(FString::FromInt(InArgs._IndexValue)) )
00028         .TextShapingMethod( ETextShapingMethod::KerningOnly )
00029         .ColorAndOpacity( FSlateColor(InArgs._TextColor) )
00030     ]
00031 ];
00032 }
```

#### 4.46.3.2 GetIndexText()

```
TSharedPtr< STextBlock > SIndexer::GetIndexText ( ) const [inline]
```

Gets the Index TextBlock Widget.

##### Returns

A Valid TextBlock Widget, if it is still found. Otherwise InValid SharedPtr.

Definition at line 55 of file [SIndexer.h](#).

```
00056 {
00057     return IndexTextBlock.IsValid() ? IndexTextBlock.Pin() : TSharedPtr<STextBlock>();
00058 }
```

#### 4.46.3.3 SLATE\_BEGIN\_ARGS()

```
SIndexer::SLATE_BEGIN_ARGS (
    SIndexer ) [inline]
```

Definition at line 10 of file [SIndexer.h](#).

```
00011 : _TextColor(FLinearColor::White)
00012 , _BorderColor(FLinearColor::Black)
00013 {}
```

#### 4.46.3.4 Tick()

```
void SIndexer::Tick (
    const FGeometry & AllotedGeometry,
    const double InCurrentTime,
    const float InDeltaTime ) [override], [virtual]
```

Definition at line 10 of file [SIndexer.cpp](#).

```
00011 {
00012     SBox::Tick(AllotedGeometry, InCurrentTime, InDeltaTime);
00013 }
```

#### 4.46.3.5 UpdateIndex() [1/3]

```
void SIndexer::UpdateIndex (
    const FString & NewIndex )
```

Updates the Index Widget with the New String Index Value.

##### Parameters

|                 |  |
|-----------------|--|
| <i>NewIndex</i> | - The New Index Value, in String Form. |
|-----------------|--|

Definition at line 44 of file [SIndexer.cpp](#).

```
00045 {
00046     if (!IndexTextBlock.IsValid())
00047         return;
00048
00049     IndexTextBlock.Pin()->SetText (
00050         FText::FromString(NewIndex)
00051     );
00052 }
```

#### 4.46.3.6 UpdateIndex() [2/3]

```
void SIndexer::UpdateIndex (
    const FText & NewIndex )
```

Updates the Index Widget with the New Text Index Value.

##### Parameters

|                 |                                      |
|-----------------|--------------------------------------|
| <i>NewIndex</i> | - The New Index Value, in Text Form. |
|-----------------|--------------------------------------|

Definition at line 54 of file [SIndexer.cpp](#).

```
00055 {
00056     if (!IndexTextBlock.IsValid())
00057         return;
00058
00059     IndexTextBlock.Pin()->SetText (NewIndex);
00060 }
```

## 4.46.3.7 UpdateIndex() [3/3]

```
void SIndexer::UpdateIndex (
    const int32 NewIndex )
```

Updates the Index Widget with the New Index Value.

## Parameters

|                 |                        |
|-----------------|------------------------|
| <i>NewIndex</i> | - The New Index Value. |
|-----------------|------------------------|

Definition at line 34 of file [SIndexer.cpp](#).

```
00035 {
00036     if (!IndexTextBlock.IsValid())
00037         return;
00038
00039     IndexTextBlock.Pin()->SetText(
00040         FText::FromString( FString::FromInt(NewIndex) )
00041     );
00042 }
```

## 4.46.4 Member Data Documentation

## 4.46.4.1 IndexTextBlock

```
TWeakPtr<STextBlock> SIndexer::IndexTextBlock [protected]
```

Weak Pointer to the Main TextBlock Widget.

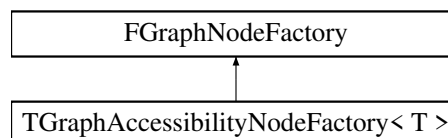
Definition at line 65 of file [SIndexer.h](#).

The documentation for this class was generated from the following files:

- Source/OpenAccessibility/Public/AccessibilityWidgets/SIndexer.h
- Source/OpenAccessibility/Private/AccessibilityWidgets/SIndexer.cpp

## 4.47 TGraphAccessibilityNodeFactory&lt; T &gt; Class Template Reference

Inheritance diagram for TGraphAccessibilityNodeFactory< T >:



## Public Member Functions

- [TGraphAccessibilityNodeFactory](#) (TSharedRef< [FAssetAccessibilityRegistry](#) > InAccessibilityRegistry)
- virtual TSharedPtr< class SGraphNode > [CreateNodeWidget](#) (UEdGraphNode \*InNode) override
- virtual TSharedPtr< class SGraphPin > [CreatePinWidget](#) (UEdGraphPin \*InPin) override

## Protected Attributes

- TSharedRef< [FAssetAccessibilityRegistry](#) > [AccessibilityRegistry](#)
- TSharedPtr< T > [Implementation](#)

### 4.47.1 Detailed Description

```
template<class T>
class TGraphAccessibilityNodeFactory< T >
```

Definition at line 17 of file [AccessibilityNodeFactory.h](#).

### 4.47.2 Constructor & Destructor Documentation

#### 4.47.2.1 TGraphAccessibilityNodeFactory() [1/2]

```
template<class T >
TGraphAccessibilityNodeFactory< T >::TGraphAccessibilityNodeFactory ( ) [inline]
```

Definition at line 23 of file [AccessibilityNodeFactory.h](#).

```
00024     {
00025         Implementation = TSharedPtr<T>();
00026
00027         AccessibilityRegistry =
00028         FOpenAccessibilityModule::Get().AssetAccessibilityRegistry.ToSharedRef();
00028     }
```

#### 4.47.2.2 TGraphAccessibilityNodeFactory() [2/2]

```
template<class T >
TGraphAccessibilityNodeFactory< T >::TGraphAccessibilityNodeFactory (
    TSharedRef< FAssetAccessibilityRegistry > InAccessibilityRegistry ) [inline]
```

Definition at line 30 of file [AccessibilityNodeFactory.h](#).

```
00031     : AccessibilityRegistry(InAccessibilityRegistry)
00032     {
00033         Implementation = TSharedPtr<T>();
00034     }
```



## 4.47.2.3 ~TGraphAccessibilityNodeFactory()

```
template<class T >
virtual TGraphAccessibilityNodeFactory< T >::~~TGraphAccessibilityNodeFactory ( ) [inline],
[virtual]
```

Definition at line 36 of file [AccessibilityNodeFactory.h](#).

```
00037 {
00038
00039 }
```

## 4.47.3 Member Function Documentation

## 4.47.3.1 CreateNodeWidget()

```
template<class T >
TSharedPtr< class SGraphNode > TGraphAccessibilityNodeFactory< T >::CreateNodeWidget (
    UEdGraphNode * InNode ) [override], [virtual]
```

Creates a Visual Node Widget from the Provided Node Object.

## Parameters

|               |  |
|---------------|--|
| <i>InNode</i> | The Node To Create a Node Widget From. |
|---------------|--|

## Returns

Definition at line 70 of file [AccessibilityNodeFactory.h](#).

```
00071 {
00072     check(InNode != nullptr);
00073
00074     TSharedPtr<SGraphNode> OutNode = Implementation->CreateNodeWidget(InNode);
00075
00076     // Apply Accessibility Visuals to the Node.
00077
00078     TSharedRef<FGraphIndexer> GraphIndexer =
        AccessibilityRegistry->GetGraphIndexer(InNode->GetGraph());
00079
00080     int NodeIndex = -1;
00081     GraphIndexer->GetOrAddNode(InNode);
00082
00083     TSharedRef<SWidget> WidgetToWrap = OutNode->GetSlot(ENodeZone::Center)->GetWidget();
00084
00085     check(WidgetToWrap != SNullWidget::NullWidget);
00086
00087     OutNode->GetOrAddSlot(ENodeZone::Center)
00088         .HAlign(HAlign_Fill)
00089         [
00090             SNew(SVerticalBox)
00091
00092             + SVerticalBox::Slot()
00093             .HAlign(HAlign_Fill)
00094             .AutoHeight()
00095             .Padding(FMargin(1.5f, 0.25f))
00096             [
00097                 SNew(SOverlay)
00098
00099                 + SOverlay::Slot()
00100                 [
```

```

00101             SNew(SImage)
00102                 .Image(FAppStyle::Get().GetBrush("Graph.Node.Body"))
00103         ]
00104     + SOverlay::Slot()
00105     .Padding(FMargin(4.0f, 0.0f))
00106     [
00107         SNew(SHorizontalBox)
00108             + SHorizontalBox::Slot()
00109             .HAlign(HAlign_Right)
00110             .VAlign(VAlign_Center)
00111             .Padding(1.f)
00112             [
00113                 SNew(SOverlay)
00114                     + SOverlay::Slot()
00115                     [
00116                         SNew(SIndexer)
00117                             .IndexValue(NodeIndex)
00118                             .TextColor(FLinearColor::White)
00119                             .BorderColor(FLinearColor::Gray)
00120                     ]
00121                 ]
00122             ]
00123     ]
00124 ]
00125 + SVerticalBox::Slot()
00126 .HAlign(HAlign_Fill)
00127 .AutoHeight()
00128 [
00129     WidgetToWrap
00130 ]
00131 ];
00132 };
00133
00134 return OutNode;
00135 }

```

#### 4.47.3.2 CreatePinWidget()

```

template<class T>
TSharedPtr< class SGraphPin> TGraphAccessibilityNodeFactory< T>::CreatePinWidget (
    UEdGraphPin* InPin ) [override], [virtual]

```

Creates a Visual Pin Widget from the Provided Pin Object.

##### Parameters

|              |                                      |
|--------------|--------------------------------------|
| <i>InPin</i> | The Pin to Create a Pin Widget From. |
|--------------|--------------------------------------|

##### Returns

Definition at line 138 of file [AccessibilityNodeFactory.h](#).

```

00139 {
00140     check(InPin != nullptr);
00141
00142     TSharedPtr<SGraphPin> OutPin = Implementation->CreatePinWidget(InPin);
00143     SGraphPin* OutPinPtr = OutPin.Get();
00144
00145     TSharedRef<FGraphIndexer> GraphIndexer =
00146         AccessibilityRegistry->GetGraphIndexer(InPin->GetOwningNode()->GetGraph());
00147
00148     int PinIndex = -1;
00149     PinIndex = InPin->GetOwningNode()->GetPinIndex(InPin);
00150
00151     TSharedRef<SWidget> AccessibilityWidget = SNew(SOverlay)
00152         .Visibility_Lambda([OutPinPtr]() -> EVisibility {
00153             if (OutPinPtr->HasAnyUserFocusOrFocusedDescendants() || OutPinPtr->IsHovered())
00154                 return EVisibility::Visible;
00155         });
00156 }

```

```

00154
00155         return EVisibility::Hidden;
00156     })
00157     + SOverlay::Slot()
00158     [
00159         SNew(STextBlock)
00160             .ColorAndOpacity(FLinearColor::White)
00161             .ShadowColorAndOpacity(FLinearColor::Black)
00162             .ShadowOffset(FIntPoint(-1, 1))
00163             .Font(FAppStyle::Get().GetFontStyle("Graph.Node.Pin.Font"))
00164             .Text(FText::FromString("[ " + FString::FromInt(PinIndex) + " ]"))
00165     ];
00166
00167     // Get Pin Widget Content, before modifying it.
00168     TSharedRef<SWidget> PinWidgetContent = OutPin->GetContent();
00169
00170     // Modify the Pin Widget Content, based on the Pin's Direction.
00171     switch (OutPin->GetDirection())
00172     {
00173     case EEdGraphPinDirection::EGPD_Input:
00174     {
00175         OutPin->SetContent(
00176             SNew(SHorizontalBox)
00177                 + SHorizontalBox::Slot()
00178                 [
00179                     PinWidgetContent
00180                 ]
00181                 + SHorizontalBox::Slot()
00182                 [
00183                     AccessibilityWidget
00184                 ]
00185             );
00186
00187         break;
00188     }
00189
00190     case EEdGraphPinDirection::EGPD_Output:
00191     {
00192         OutPin->SetContent(
00193             SNew(SHorizontalBox)
00194                 + SHorizontalBox::Slot()
00195                 .AutoWidth()
00196                 [
00197                     AccessibilityWidget
00198                 ]
00199                 + SHorizontalBox::Slot()
00200                 .AutoWidth()
00201                 [
00202                     PinWidgetContent
00203                 ]
00204             );
00205
00206         break;
00207     }
00208     }
00209
00210     return OutPin;
00211 }

```

## 4.47.4 Member Data Documentation

### 4.47.4.1 AccessibilityRegistry

```

template<class T>
TSharedRef<FAssetAccessibilityRegistry> TGraphAccessibilityNodeFactory< T >::AccessibilityRegistry [protected]

```

The Asset Registry of the Open Accessibility Plugin.

Definition at line 64 of file [AccessibilityNodeFactory.h](#).

#### 4.47.4.2 Implementation

```
template<class T >
TSharedPtr<T> TGraphAccessibilityNodeFactory< T >::Implementation [protected]
```

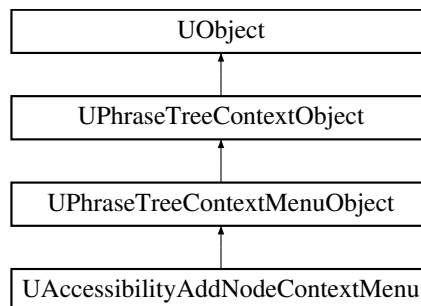
Definition at line 66 of file [AccessibilityNodeFactory.h](#).

The documentation for this class was generated from the following file:

- Source/OpenAccessibility/Public/AccessibilityNodeFactory.h

## 4.48 UAccessibilityAddNodeContextMenu Class Reference

Inheritance diagram for UAccessibilityAddNodeContextMenu:



### Public Member Functions

- [UAccessibilityAddNodeContextMenu](#) (TSharedRef< IMenu > [Menu](#))
- [UAccessibilityAddNodeContextMenu](#) (TSharedRef< IMenu > [Menu](#), TSharedRef< SGraphActionMenu > [GraphMenu](#))
- [UAccessibilityAddNodeContextMenu](#) (TSharedRef< IMenu > [Menu](#), TSharedRef< SGraphActionMenu > [GraphMenu](#), TSharedRef< STreeView< TSharedPtr< FGraphActionNode > > > [TreeView](#))
- virtual void [Init](#) (TSharedRef< IMenu > InMenu, TSharedRef< [FPhraseNode](#) > InContextRoot) override  
*Initializes the Context Menu.*
- void [Init](#) (TSharedRef< IMenu > InMenu, TSharedRef< SGraphActionMenu > InGraphMenu, TSharedRef< STreeView< TSharedPtr< FGraphActionNode > > > InTreeView)  
*Initializes the Context Menu from the given components.*
- virtual void [Init](#) (TSharedRef< IMenu > InMenu) override  
*Initializes the Context Menu from the given components.*
- virtual bool [Tick](#) (float DeltaTime) override
- virtual bool [Close](#) () override  
*Closes the Context Menu.*
- virtual void [ScaleMenu](#) (const float ScaleFactor=1.5f) override  
*Scaled the Context Menu's Core Components based on the provided ScaleFactor.*
- bool [DoesItemsRequireRefresh](#) ()  
*Does the Context Menu's TreeView Require a Refresh of Accessibility Widgets.*
- void [RefreshAccessibilityWidgets](#) ()  
*Performs a Refresh of the TreeView's Accessibility Widgets.*
- void [GetGraphActionFromIndex](#) (const int32 InIndex, FGraphActionNode \*OutGraphAction)

- Gets the GraphActionNode from the given Index.*
- FGraphNode \* [GetGraphActionFromIndex](#) (const int32 InIndex)  
*Gets the GraphActionNode from the given Index.*
- TSharedPtr< FGraphNode > [GetGraphActionFromIndexSP](#) (const int32 InIndex)  
*Gets the GraphActionNode from the given Index.*
- void [SelectGraphAction](#) (const int32 InIndex)  
*Performs a Selection in the TreeView, based on the given Index.*
- void [PerformGraphAction](#) (const int32 InIndex)  
*Performs the Action to the Linked GraphActionNode, based on the given Index.*
- FString [GetFilterText](#) ()  
*Gets the Current Filter Text in the Search Bar.*
- void [SetFilterText](#) (const FString &InFilterText)  
*Overrides the Current Filter Text with the given string.*
- void [AppendFilterText](#) (const FString &InFilterText)  
*Append the given string to the End of the Current Filter Text.*
- void [ResetFilterText](#) ()  
*Clears the Current Filter Text.*
- void [SetScrollDistance](#) (const float InScrollDistance)  
*Sets the Scroll Distance of the TreeView.*
- void [AppendScrollDistance](#) (const float InScrollDistance)  
*Adds the provided value to the Current Scroll Distance.*
- void [SetScrollDistanceTop](#) ()  
*Sets the Scroll Distance to the Top of the TreeView. Taking the View to the First Item in the TreeView.*
- void [SetScrollDistanceBottom](#) ()  
*Sets the Scroll Distance to the Bottom of the TreeView. Taking the View to the Last Item in the TreeView.*
- void [ToggleContextAwareness](#) ()  
*Toggles the Context Awareness of the Node List.*

## Public Attributes

- TWeakPtr< SGraphActionMenu > [GraphMenu](#)  
*The SGraphActionMenu for the Context Menu.*
- TWeakPtr< STreeView< TSharedPtr< FGraphNode > > > [TreeView](#)  
*The STreeView for the Context Menu.*
- TWeakPtr< SEditableTextBox > [FilterTextBox](#)  
*The SEditableTextBox for the Context Menu. Used for Filtering through GraphNodes.*
- TWeakPtr< SCheckBox > [ContextAwarenessCheckBox](#)  
*The Context Awareness CheckBox for the Context Menu. Used for toggling Context Awareness, in searching for GraphNodes.*

## Protected Member Functions

- void [ApplyAccessibilityWidget](#) (TSharedRef< STableRow< TSharedPtr< FGraphNode > > > Item↵ Widget)  
*Applies the Accessibility Widget to the given Item's TableRow Widget.*
- void [UpdateAccessibilityWidget](#) (TSharedRef< STableRow< TSharedPtr< FGraphNode > > > ItemWidget)  
*Updates the previously applied Accessibility Widget, with the new index.*

## Protected Attributes

- FString [PrevFilterString](#)
- int32 [PrevNumItemsBeingObserved](#)
- int32 [PrevNumGeneratedChildren](#)
- double [PrevScrollDistance](#)

### 4.48.1 Detailed Description

Definition at line 17 of file [AccessibilityAddNodeContextMenu.h](#).

### 4.48.2 Constructor & Destructor Documentation

#### 4.48.2.1 UAccessibilityAddNodeContextMenu() [1/4]

UAccessibilityAddNodeContextMenu::UAccessibilityAddNodeContextMenu ( )

Definition at line 13 of file [AccessibilityAddNodeContextMenu.cpp](#).

```
00014 : UPhraseTreeContextMenuObject()
00015 {
00016
00017 }
```

#### 4.48.2.2 UAccessibilityAddNodeContextMenu() [2/4]

UAccessibilityAddNodeContextMenu::UAccessibilityAddNodeContextMenu (   
 TSharedRef< IMenu > Menu )

Definition at line 19 of file [AccessibilityAddNodeContextMenu.cpp](#).

```
00020 : UPhraseTreeContextMenuObject(Menu)
00021 {
00022
00023 }
```

#### 4.48.2.3 UAccessibilityAddNodeContextMenu() [3/4]

UAccessibilityAddNodeContextMenu::UAccessibilityAddNodeContextMenu (   
 TSharedRef< IMenu > Menu,   
 TSharedRef< SGraphActionMenu > GraphMenu )

Definition at line 25 of file [AccessibilityAddNodeContextMenu.cpp](#).

```
00026 : UPhraseTreeContextMenuObject(Menu)
00027 {
00028     this->GraphMenu = GraphMenu;
00029     this->FilterTextBox = GraphMenu->GetFilterTextBox();
00030 }
```

**4.48.2.4 UAccessibilityAddNodeContextMenu()** [4/4]

```
UAccessibilityAddNodeContextMenu::UAccessibilityAddNodeContextMenu (
    TSharedRef< IMenu > Menu,
    TSharedRef< SGraphActionMenu > GraphMenu,
    TSharedRef< STreeView< TSharedPtr< FGraphActionNode > > > TreeView )
```

Definition at line 32 of file [AccessibilityAddNodeContextMenu.cpp](#).

```
00033 : UPhraseTreeContextMenuObject (Menu)
00034 {
00035     this->GraphMenu = GraphMenu;
00036     this->TreeView = TreeView;
00037     this->FilterTextBox = GraphMenu->GetFilterTextBox();
00038 }
```

**4.48.2.5 ~UAccessibilityAddNodeContextMenu()**

```
UAccessibilityAddNodeContextMenu::~UAccessibilityAddNodeContextMenu ( )
```

Definition at line 40 of file [AccessibilityAddNodeContextMenu.cpp](#).

```
00041 {
00042
00043 }
```

**4.48.3 Member Function Documentation****4.48.3.1 AppendFilterText()**

```
void UAccessibilityAddNodeContextMenu::AppendFilterText (
    const FString & InFilterText )
```

Append the given string to the End of the Current Filter Text.

**Parameters**

|                     |                                |
|---------------------|--------------------------------|
| <i>InFilterText</i> | The Text To Append to the End. |
|---------------------|--------------------------------|

Definition at line 282 of file [AccessibilityAddNodeContextMenu.cpp](#).

```
00283 {
00284     FilterTextBox.Pin()->SetText(
00285         FText::FromString(
00286             FilterTextBox.Pin()->GetText().ToString() + TEXT(" ") + InFilterText
00287         )
00288     );
00289 }
```

**4.48.3.2 AppendScrollDistance()**

```
void UAccessibilityAddNodeContextMenu::AppendScrollDistance (
    const float InScrollDistance )
```

Adds the provided value to the Current Scroll Distance.

#### Parameters

|                         |  |
|-------------------------|--|
| <i>InScrollDistance</i> | The Scroll Distance to Add the Current Distance. Positive Values are down, with Negative being up. |
|-------------------------|--|

Definition at line 301 of file [AccessibilityAddNodeContextMenu.cpp](#).

```
00302 {
00303     if (TreeView.Pin()->GetScrollOffset() + InScrollDistance < 0.0f)
00304     {
00305         TreeView.Pin()->SetScrollOffset(0.0f);
00306         return;
00307     }
00308
00309     TreeView.Pin()->AddScrollOffset(InScrollDistance, true);
00310 }
```

#### 4.48.3.3 ApplyAccessibilityWidget()

```
void UAccessibilityAddNodeContextMenu::ApplyAccessibilityWidget (
    TSharedRef< STableRow< TSharedPtr< FGraphActionNode > > > ItemWidget ) [protected]
```

Applies the Accessibility Widget to the given Item's TableRow Widget.

#### Parameters

|                   |                          |
|-------------------|--------------------------|
| <i>Item</i>       | The Item to apply to.    |
| <i>ItemWidget</i> | The Items linked widget. |

Definition at line 327 of file [AccessibilityAddNodeContextMenu.cpp](#).

```
00328 {
00329     TSharedPtr<SWidget> ItemContent = ItemWidget->GetContent();
00330
00331     ItemWidget->SetContent(
00332         SNew(SContentIndexer)
00333             .IndexValue(ItemWidget->GetIndexInList())
00334             .IndexPositionToContent(EIndexerPosition::Left)
00335             .ContentToIndex(ItemContent)
00336     );
00337 }
```

#### 4.48.3.4 Close()

```
bool UAccessibilityAddNodeContextMenu::Close ( ) [override], [virtual]
```

Closes the Context Menu.



**Returns**

Returns True if the Menu was Successfully closed.

Reimplemented from [UPhraseTreeContextMenuObject](#).

Definition at line 131 of file [AccessibilityAddNodeContextMenu.cpp](#).

```
00132 {
00133     RemoveTickDelegate();
00134     Menu.Pin()->Dismiss();
00135
00136     return true;
00137 }
```

**4.48.3.5 DoesItemsRequireRefresh()**

```
bool UAccessibilityAddNodeContextMenu::DoesItemsRequireRefresh ( )
```

Does the Context Menu's TreeView Require a Refresh of Accessibility Widgets.

**Returns**

Returns True if the Context Menu requires change.

Definition at line 157 of file [AccessibilityAddNodeContextMenu.cpp](#).

```
00158 {
00159     TSharedPtr<STreeView<TSharedPtr<FGraphNode>>> TreeViewPtr = TreeView.Pin();
00160
00161     return (
00162         FilterTextBox.Pin()->GetText().ToString() != PrevFilterString ||
00163         TreeViewPtr->GetNumItemsBeingObserved() != PrevNumItemsBeingObserved ||
00164         TreeViewPtr->GetNumGeneratedChildren() != PrevNumGeneratedChildren ||
00165         TreeViewPtr->GetScrollDistance().Y != PrevScrollDistance
00166     );
00167 }
```

**4.48.3.6 GetFilterText()**

```
FString UAccessibilityAddNodeContextMenu::GetFilterText ( )
```

Gets the Current Filter Text in the Search Bar.

**Returns**

The Current Filter Text in the Search Bar.

Definition at line 272 of file [AccessibilityAddNodeContextMenu.cpp](#).

```
00273 {
00274     return FilterTextBox.Pin()->GetText().ToString();
00275 }
```

**4.48.3.7 GetGraphNodeFromIndex() [1/2]**

```
FGraphNode * UAccessibilityAddNodeContextMenu::GetGraphNodeFromIndex (
    const int32 InIndex )
```

Gets the GraphActionNode from the given Index.

**Parameters**

|                |                                |
|----------------|--------------------------------|
| <i>InIndex</i> | The Index of the Node to Find. |
|----------------|--------------------------------|

**Returns**

The Found GraphActionNode for the Index, or nullptr.

Definition at line 207 of file [AccessibilityAddNodeContextMenu.cpp](#).

```
00208 {
00209     TArrayView<const TSharedPtr<FGraphActionNode> Items = TreeView.Pin()->GetItems();
00210
00211     if (Items.Num() > InIndex)
00212         return Items[InIndex].Get();
00213
00214     else return nullptr;
00215 }
```

**4.48.3.8 GetGraphActionFromIndex() [2/2]**

```
void UAccessibilityAddNodeContextMenu::GetGraphActionFromIndex (
    const int32 InIndex,
    FGraphActionNode * OutGraphAction )
```

Gets the GraphActionNode from the given Index.

**Parameters**

|                       |  |
|-----------------------|--|
| <i>InIndex</i>        | The Index of the Node to Find.                       |
| <i>OutGraphAction</i> | The Found GraphActionNode for the Index, or nullptr. |

Definition at line 217 of file [AccessibilityAddNodeContextMenu.cpp](#).

```
00218 {
00219     TArrayView<const TSharedPtr<FGraphActionNode> Items = TreeView.Pin()->GetItems();
00220
00221     if (Items.Num() > InIndex)
00222         OutGraphAction = Items[InIndex].Get();
00223
00224     else OutGraphAction = nullptr;
00225 }
```

**4.48.3.9 GetGraphActionFromIndexSP()**

```
TSharedPtr< FGraphActionNode > UAccessibilityAddNodeContextMenu::GetGraphActionFromIndexSP (
    const int32 InIndex )
```

Gets the GraphActionNode from the given Index.

**Parameters**

|                |                                |
|----------------|--------------------------------|
| <i>InIndex</i> | The Index of the Node to Find. |
|----------------|--------------------------------|

**Returns**

The Found `GraphNode` for the Index, or `nullptr`.

Definition at line 227 of file [AccessibilityAddNodeContextMenu.cpp](#).

```
00228 {
00229     if (TreeView.Pin()->GetItems().Num() <= InIndex)
00230     {
00231         UE_LOG(LogOpenAccessibility, Warning, TEXT("GetGraphNodeFromIndexSP: Provided Index is Out
of Range."));
00232         return nullptr;
00233     }
00234     return TreeView.Pin()->GetItems()[InIndex];
00235 }
```

**4.48.3.10 Init() [1/3]**

```
void UAccessibilityAddNodeContextMenu::Init (
    TSharedRef< IMenu > InMenu ) [override], [virtual]
```

Initializes the Context Menu from the given components.

**Parameters**

|               |  |
|---------------|--|
| <i>InMenu</i> |  |
|---------------|--|

Reimplemented from [UPhraseTreeContextMenuObject](#).

Definition at line 52 of file [AccessibilityAddNodeContextMenu.cpp](#).

```
00053 {
00054     UPhraseTreeContextMenuObject::Init(InMenu);
00055
00056     // This is a Mess but half the Menu Containers are private, so have to move myself to key aspects
of the Menu.
00057
00058     TSharedPtr<SWidget> KeyboardFocusedWidget = StaticCastSharedPtr<SEditableText>(
00059         FSlateApplication::Get().GetKeyboardFocusedWidget()
00060     );
00061     if (!KeyboardFocusedWidget.IsValid())
00062     {
00063         UE_LOG(LogOpenAccessibility, Warning, TEXT("AddNodeContextMenuWrapper::Init: KeyboardFocusedWidget
is Invalid."));
00064         return;
00065     }
00066
00067     this->GraphMenu = StaticCastSharedPtr<SGraphNodeMenu>(
00068         KeyboardFocusedWidget
00069         ->GetParentWidget()
00070         ->GetParentWidget()
00071         ->GetParentWidget()
00072         ->GetParentWidget()
00073         ->GetParentWidget()
00074     );
00075
00076     {
00077         TSharedPtr<SSearchBar> SearchBox = StaticCastSharedPtr<SSearchBar>(
00078             KeyboardFocusedWidget
00079             ->GetParentWidget()
00080             ->GetParentWidget()
00081             ->GetParentWidget()
00082         );
00083
00084         TSharedRef<SWidget> SearchBoxSibling =
SearchBox->GetParentWidget()->GetChildren()->GetChildAt(1);
00085         this->TreeView = StaticCastSharedRef<STreeView<TSharedPtr<FGraphNode>>>(
00086             SearchBoxSibling->GetChildren()->GetChildAt(0)->GetChildren()->GetChildAt(0)
00087         );
00088     }
00089
00090     {
```

```

00091         TSharedRef<SCheckBox> CheckBox = StaticCastSharedRef<SCheckBox>(
00092             this->GraphMenu.Pin()->GetParentWidget()->GetChildren()->GetChildAt(0)->GetChildren()->GetChildAt(2)
00093         );
00094
00095         this->ContextAwarenessCheckBox = CheckBox;
00096     }
00097
00098     this->FilterTextBox = this->GraphMenu.Pin()->GetFilterTextBox();
00099
00100     FSlateApplication::Get().SetKeyboardFocus(this->TreeView.Pin());
00101 }

```

#### 4.48.3.11 Init() [2/3]

```

void UAccessibilityAddNodeContextMenu::Init (
    TSharedRef< IMenu > InMenu,
    TSharedRef< FPhraseNode > InContextRoot ) [override], [virtual]

```

Initializes the Context Menu.

##### Parameters

|                      |   |
|----------------------|---|
| <i>InMenu</i>        | The Menu to Initialize from and obtain key components.                    |
| <i>InContextRoot</i> | The Context Root in the PhraseTree that this ContextMenu Originates from. |

Reimplemented from [UPhraseTreeContextMenuObject](#).

Definition at line 45 of file [AccessibilityAddNodeContextMenu.cpp](#).

```

00046 {
00047     Init(InMenu);
00048
00049     this->ContextRoot = InContextRoot;
00050 }

```

#### 4.48.3.12 Init() [3/3]

```

void UAccessibilityAddNodeContextMenu::Init (
    TSharedRef< IMenu > InMenu,
    TSharedRef< SGraphActionMenu > InGraphMenu,
    TSharedRef< STreeView< TSharedPtr< FGraphNode > > > InTreeView )

```

Initializes the Context Menu from the given components.

##### Parameters

|                    |  |
|--------------------|--|
| <i>InMenu</i>      | The Menu Item, for the targetContext Menu.             |
| <i>InGraphMenu</i> | The GraphActionMenu, for the target Context Menu.      |
| <i>InTreeView</i>  | The GraphAction TreeView, for the target Context Menu. |

Definition at line 103 of file [AccessibilityAddNodeContextMenu.cpp](#).

```

00104 {
00105     UPhraseTreeContextMenuObject::Init(InMenu);

```

```

00106
00107     this->GraphMenu = InGraphMenu;
00108     this->TreeView = InTreeView;
00109     this->FilterTextBox = InGraphMenu->GetFilterTextBox();
00110 }

```

#### 4.48.3.13 PerformGraphAction()

```

void UAccessibilityAddNodeContextMenu::PerformGraphAction (
    const int32 InIndex )

```

Performs the Action to the Linked GraphActionNode, based on the given Index.

##### Parameters

|                |  |
|----------------|--|
| <i>InIndex</i> |  |
|----------------|--|

Definition at line 251 of file [AccessibilityAddNodeContextMenu.cpp](#).

```

00252 {
00253     TSharedPtr<FGraphActionNode> GraphAction = GetGraphActionFromIndexSP(InIndex);
00254
00255     if (!GraphAction.IsValid())
00256     {
00257         UE_LOG(LogOpenAccessibility, Warning, TEXT("PerformGraphAction: Provided GraphAction is
Invalid."));
00258     }
00259
00260     if (GraphAction->IsActionNode())
00261     {
00262         TreeView.Pin()->Private_ClearSelection();
00263         TreeView.Pin()->Private_SetItemSelection(GraphAction, true, true);
00264         TreeView.Pin()->Private_SignalSelectionChanged(ESelectInfo::OnMouseClicked);
00265     }
00266     else
00267     {
00268         TreeView.Pin()->Private_OnItemDoubleClicked(GraphAction);
00269     }
00270 }

```

#### 4.48.3.14 RefreshAccessibilityWidgets()

```

void UAccessibilityAddNodeContextMenu::RefreshAccessibilityWidgets ( )

```

Performs a Refresh of the TreeView's Accessibility Widgets.

Definition at line 169 of file [AccessibilityAddNodeContextMenu.cpp](#).

```

00170 {
00171
00172     TSharedPtr<STreeView<TSharedPtr<FGraphActionNode>> TreeViewPtr = TreeView.Pin();
00173
00174     TArray<TSharedPtr<FGraphActionNode> Items =
TArray<TSharedPtr<FGraphActionNode> (TreeViewPtr->GetRootItems());
00175
00176     {
00177         TSharedPtr<STableRow<TSharedPtr<FGraphActionNode>> ItemWidget = nullptr;
00178
00179         while (Items.Num() > 0)
00180         {
00181             const TSharedPtr<FGraphActionNode> Item = Items[0];
00182             Items.RemoveAt(0);
00183
00184             if (TreeViewPtr->IsItemExpanded(Item))
00185                 Items.Append(Item->Children);

```

```

00186
00187     ItemWidget = StaticCastSharedPtr<STableRow<TSharedPtr<FGraphActionNode>>>(
00188         TreeViewPtr->WidgetFromItem(Item)
00189     );
00190
00191     if (!ItemWidget.IsValid())
00192         continue;
00193
00194     // TO-DO: Change To Non-HardCoded Type Comparison.
00195     if (ItemWidget->GetContent()->GetType() == "SContentIndexer")
00196     {
00197         UpdateAccessibilityWidget(ItemWidget.ToSharedRef());
00198     }
00199     else
00200     {
00201         ApplyAccessibilityWidget(ItemWidget.ToSharedRef());
00202     }
00203 }
00204 }
00205 }

```

#### 4.48.3.15 ResetFilterText()

```
void UAccessibilityAddNodeContextMenu::ResetFilterText ( )
```

Clears the Current Filter Text.

Definition at line 291 of file [AccessibilityAddNodeContextMenu.cpp](#).

```

00292 {
00293     FilterTextBox.Pin()->SetText(FText::FromString(TEXT("")));
00294 }

```

#### 4.48.3.16 ScaleMenu()

```
void UAccessibilityAddNodeContextMenu::ScaleMenu (
    const float ScaleFactor = 1.5f ) [override], [virtual]
```

Scaled the Context Menu's Core Components based on the provided ScaleFactor.

##### Parameters

|                    |  |
|--------------------|--|
| <i>ScaleFactor</i> | The Factor for Scaling the Context Menu. |
|--------------------|--|

Reimplemented from [UPhraseTreeContextMenuObject](#).

Definition at line 139 of file [AccessibilityAddNodeContextMenu.cpp](#).

```

00140 {
00141     // Scale TreeView Element
00142     {
00143         TSharedPtr<STreeView<TSharedPtr<FGraphActionNode>>> TreeViewPtr = TreeView.Pin();
00144
00145         TreeViewPtr->SetItemHeight(16 * ScaleFactor);
00146     }
00147
00148     // Scale Window Element
00149     {
00150         TSharedPtr<SWindow> WindowPtr = Window.Pin();
00151
00152         WindowPtr->SetSizingRule(ESizingRule::UserSized);
00153         WindowPtr->Resize(WindowPtr->GetSizeInScreen() * ScaleFactor);
00154     }

```

```
00155 }
```

#### 4.48.3.17 SelectGraphAction()

```
void UAccessibilityAddNodeContextMenu::SelectGraphAction (
    const int32 InIndex )
```

Performs a Selection in the TreeView, based on the given Index.

##### Parameters

|                |  |
|----------------|--|
| <i>InIndex</i> |  |
|----------------|--|

Definition at line 237 of file [AccessibilityAddNodeContextMenu.cpp](#).

```
00238 {
00239     TSharedPtr<FGraphNode> GraphAction = GetGraphActionFromIndexSP(InIndex);
00240
00241     if (GraphAction.IsValid())
00242     {
00243         TreeView.Pin()->Private_OnItemClicked(GraphAction);
00244     }
00245     else
00246     {
00247         UE_LOG(LogOpenAccessibility, Warning, TEXT("SelectGraphAction: Provided GraphAction is
Invalid."));
00248     }
00249 }
```

#### 4.48.3.18 SetFilterText()

```
void UAccessibilityAddNodeContextMenu::SetFilterText (
    const FString & InFilterText )
```

Overrides the Current Filter Text with the given string.

##### Parameters

|                     |                              |
|---------------------|------------------------------|
| <i>InFilterText</i> | The String to Override with. |
|---------------------|------------------------------|

Definition at line 277 of file [AccessibilityAddNodeContextMenu.cpp](#).

```
00278 {
00279     FilterTextBox.Pin()->SetText(FText::FromString(InFilterText));
00280 }
```

#### 4.48.3.19 SetScrollDistance()

```
void UAccessibilityAddNodeContextMenu::SetScrollDistance (
    const float InScrollDistance )
```

Sets the Scroll Distance of the TreeView.

## Parameters

|                         |  |
|-------------------------|--|
| <i>InScrollDistance</i> | The Value to Set the Scroll Distance to. |
|-------------------------|--|

Definition at line 296 of file [AccessibilityAddNodeContextMenu.cpp](#).

```
00297 {
00298     TreeView.Pin()->SetScrollOffset(InScrollDistance);
00299 }
```

**4.48.3.20 SetScrollDistanceBottom()**

```
void UAccessibilityAddNodeContextMenu::SetScrollDistanceBottom ( )
```

Sets the Scroll Distance to the Bottom of the TreeView. Taking the View to the Last Item in the TreeView.

Definition at line 317 of file [AccessibilityAddNodeContextMenu.cpp](#).

```
00318 {
00319     TreeView.Pin()->ScrollToBottom();
00320 }
```

**4.48.3.21 SetScrollDistanceTop()**

```
void UAccessibilityAddNodeContextMenu::SetScrollDistanceTop ( )
```

Sets the Scroll Distance to the Top of the TreeView. Taking the View to the First Item in the TreeView.

Definition at line 312 of file [AccessibilityAddNodeContextMenu.cpp](#).

```
00313 {
00314     TreeView.Pin()->ScrollToTop();
00315 }
```

**4.48.3.22 Tick()**

```
bool UAccessibilityAddNodeContextMenu::Tick (
    float DeltaTime ) [override], [virtual]
```

Reimplemented from [UPhraseTreeContextMenuObject](#).

Definition at line 112 of file [AccessibilityAddNodeContextMenu.cpp](#).

```
00113 {
00114     if (!GraphMenu.IsValid() || !Menu.IsValid())
00115         return false;
00116
00117     if (DoesItemsRequireRefresh())
00118         RefreshAccessibilityWidgets();
00119
00120     TSharedPtr<TTreeView<TSharedPtr<FGraphNode>>> TreeViewPtr = TreeView.Pin();
00121
00122     // Set Previous Vars For Next Tick
00123     PrevFilterString = FilterTextBox.Pin()->GetText().ToString();
00124     PrevNumItemsBeingObserved = TreeViewPtr->GetNumItemsBeingObserved();
00125     PrevNumGeneratedChildren = TreeViewPtr->GetNumGeneratedChildren();
00126     PrevScrollDistance = TreeViewPtr->GetScrollDistance().Y;
00127
00128     return true;
00129 }
```



### 4.48.3.23 ToggleContextAwareness()

```
void UAccessibilityAddNodeContextMenu::ToggleContextAwareness ( )
```

Toggles the Context Awareness of the Node List.

Definition at line 322 of file [AccessibilityAddNodeContextMenu.cpp](#).

```
00323 {
00324     ContextAwarenessCheckBox.Pin()->ToggleCheckedState();
00325 }
```

### 4.48.3.24 UpdateAccessibilityWidget()

```
void UAccessibilityAddNodeContextMenu::UpdateAccessibilityWidget (
    TSharedRef< STableRow< TSharedPtr< FGraphNode > > > ItemWidget ) [protected]
```

Updates the previously applied Accessibility Widget, with the new index.

#### Parameters

|                   |                     |
|-------------------|---------------------|
| <i>ItemWidget</i> | The Item to update. |
|-------------------|---------------------|

Definition at line 339 of file [AccessibilityAddNodeContextMenu.cpp](#).

```
00340 {
00341     TSharedPtr<SContentIndexer> ItemContent =
        StaticCastSharedPtr<SContentIndexer>(ItemWidget->GetContent());
00342
00343     ItemContent->UpdateIndex(ItemWidget->GetIndexInList());
00344 }
```

## 4.48.4 Member Data Documentation

### 4.48.4.1 ContextAwarenessCheckBox

```
TWeakPtr<SCheckBox> UAccessibilityAddNodeContextMenu::ContextAwarenessCheckBox
```

The Context Awareness CheckBox for the Context Menu. Used for toggling Context Awareness, in searching for GraphNodes.

Definition at line 203 of file [AccessibilityAddNodeContextMenu.h](#).

### 4.48.4.2 FilterTextBox

```
TWeakPtr<SEditableTextBox> UAccessibilityAddNodeContextMenu::FilterTextBox
```

The SEditableTextBox for the Context Menu. Used for Filtering through GraphNodes.

Definition at line 198 of file [AccessibilityAddNodeContextMenu.h](#).

#### 4.48.4.3 GraphMenu

```
TWeakPtr<SGraphActionMenu> UAccessibilityAddNodeContextMenu::GraphMenu
```

The SGraphActionMenu for the Context Menu.

Definition at line 188 of file [AccessibilityAddNodeContextMenu.h](#).

#### 4.48.4.4 PrevFilterString

```
FString UAccessibilityAddNodeContextMenu::PrevFilterString [protected]
```

Definition at line 207 of file [AccessibilityAddNodeContextMenu.h](#).

#### 4.48.4.5 PrevNumGeneratedChildren

```
int32 UAccessibilityAddNodeContextMenu::PrevNumGeneratedChildren [protected]
```

Definition at line 209 of file [AccessibilityAddNodeContextMenu.h](#).

#### 4.48.4.6 PrevNumItemsBeingObserved

```
int32 UAccessibilityAddNodeContextMenu::PrevNumItemsBeingObserved [protected]
```

Definition at line 208 of file [AccessibilityAddNodeContextMenu.h](#).

#### 4.48.4.7 PrevScrollDistance

```
double UAccessibilityAddNodeContextMenu::PrevScrollDistance [protected]
```

Definition at line 210 of file [AccessibilityAddNodeContextMenu.h](#).

#### 4.48.4.8 TreeView

```
TWeakPtr<STreeView<TSharedPtr<FGraphNode> > > UAccessibilityAddNodeContextMenu::Tree↔  
View
```

The STreeView for the Context Menu.

Definition at line 193 of file [AccessibilityAddNodeContextMenu.h](#).

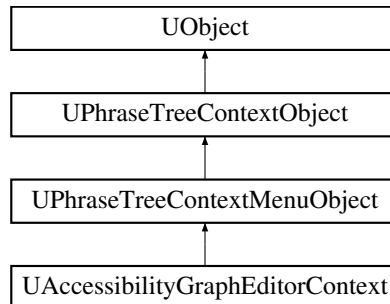
The documentation for this class was generated from the following files:

- Source/OpenAccessibility/Public/AccessibilityWrappers/AccessibilityAddNodeContextMenu.h
- Source/OpenAccessibility/Private/AccessibilityWrappers/AccessibilityAddNodeContextMenu.cpp

## 4.49 UAccessibilityGraphEditorContext Class Reference

```
#include <AccessibilityGraphEditorContext.h>
```

Inheritance diagram for UAccessibilityGraphEditorContext:



### Classes

- struct [FTreeViewTickRequirements](#)

### Public Member Functions

- virtual void [Init](#) (TSharedRef< IMenu > InMenu, TSharedRef< [FPhraseNode](#) > InContextRoot) override
- virtual bool [Tick](#) (float DeltaTime) override
- virtual bool [Close](#) () override
- virtual void [ScaleMenu](#) (const float ScaleFactor=1.5f) override
- TSharedPtr< FGraphActionNode > [GetTreeViewAction](#) (const int32 &InIndex)
- void [SelectAction](#) (const int32 &InIndex)
- FString [GetFilterText](#) ()
- void [SetFilterText](#) (const FString &NewString)
- void [AppendFilterText](#) (const FString &StringToAdd)
- void [SetScrollDistance](#) (const float NewDistance)
- void [AppendScrollDistance](#) (const float DistanceToAdd)
- void [SetScrollDistanceTop](#) ()
- void [SetScrollDistanceBottom](#) ()

### Protected Member Functions

- const int32 [GetStaticIndexOffset](#) ()
- bool [FindGraphActionMenu](#) (const TSharedRef< SWidget > &SearchRoot)
- bool [FindTreeView](#) (const TSharedRef< SWidget > &SearchRoot)
- bool [FindStaticComponents](#) (const TSharedRef< SWidget > &SearchRoot)
- bool [TreeViewCanTick](#) ()
- bool [TreeViewRequiresTick](#) ()
- void [TickTreeViewAccessibility](#) ()
- void [UpdateAccessibilityWidget](#) (const TSharedRef< [SContentIndexer](#) > &ContextIndexer, const int32 &NewIndex)
- const TSharedRef< [SContentIndexer](#) > [CreateAccessibilityWrapper](#) (const TSharedRef< SWidget > &ContentToWrap, const int32 &Index)

## Protected Attributes

- [FTreeViewTickRequirements](#) [TreeViewTickRequirements](#)
- TWeakPtr< SGraphActionMenu > [GraphMenu](#) = TWeakPtr<SGraphActionMenu>()
- TWeakPtr< SEditableTextBox > [FilterTextBox](#) = TWeakPtr<SEditableTextBox>()
- TWeakPtr< STreeView< TSharedPtr< FGraphActionNode > > > [TreeView](#) = TWeakPtr<STreeView<TSharedPtr<FGraphActionNode>>>()
- TArray< TWeakPtr< SCheckBox > > [CheckBoxes](#) = TArray<TWeakPtr<SCheckBox>>()

## Additional Inherited Members

### 4.49.1 Detailed Description

A Dynamic Phrase Tree Context Object for Most Node Editor Based Context Menus.

Definition at line 20 of file [AccessibilityGraphEditorContext.h](#).

### 4.49.2 Constructor & Destructor Documentation

#### 4.49.2.1 UAccessibilityGraphEditorContext()

```
UAccessibilityGraphEditorContext::UAccessibilityGraphEditorContext ( )
```

Definition at line 13 of file [AccessibilityGraphEditorContext.cpp](#).

```
00014     : Super()
00015 {
00016
00017 }
```

### 4.49.3 Member Function Documentation

#### 4.49.3.1 AppendFilterText()

```
void UAccessibilityGraphEditorContext::AppendFilterText (
    const FString & StringToAdd )
```

Appends the provided string to the Context Menus SearchBar, if it contains one.

#### Parameters

|                    |  |
|--------------------|--|
| <i>StringToAdd</i> | The Text to Append to the End of the Active SearchBar. |
|--------------------|--|

Definition at line 152 of file [AccessibilityGraphEditorContext.cpp](#).

```

00153 {
00154     if (!FilterTextBox.IsValid())
00155         return;
00156
00157     TSharedPtr<SEditableTextBox> FilterTextBoxPtr = FilterTextBox.Pin();
00158
00159     FilterTextBoxPtr->SetText (
00160         FText::FromString( FilterTextBoxPtr->GetText().ToString() + TEXT(" ") + StringToAdd )
00161     );
00162 }

```

#### 4.49.3.2 AppendScrollDistance()

```

void UAccessibilityGraphEditorContext::AppendScrollDistance (
    const float DistanceToAdd )

```

Adds the Scroll Distance of the Context Menus TreeView, if it contains one.

##### Parameters

|                      |  |
|----------------------|--|
| <i>DistanceToAdd</i> | The distance to append to the Scroll Area. |
|----------------------|--|

Definition at line 172 of file [AccessibilityGraphEditorContext.cpp](#).

```

00173 {
00174     auto TreeViewPtr = TreeView.Pin();
00175
00176     if (TreeViewPtr->GetScrollOffset() + DistanceToAdd < 0.0f)
00177     {
00178         TreeViewPtr->SetScrollOffset(0.0f);
00179         return;
00180     }
00181
00182     TreeViewPtr->AddScrollOffset(DistanceToAdd);
00183 }

```

#### 4.49.3.3 Close()

```

bool UAccessibilityGraphEditorContext::Close ( ) [override], [virtual]

```

Closes the Graph Editor Context Wrapper Instance.

##### Returns

True on successful Closing of the Context Menu, False on Failure.

Reimplemented from [UPhraseTreeContextMenuObject](#).

Definition at line 64 of file [AccessibilityGraphEditorContext.cpp](#).

```

00065 {
00066     Super::Close();
00067
00068     return true;
00069 }

```

#### 4.49.3.4 CreateAccessibilityWrapper()

```
const TSharedRef< SContentIndexer > UAccessibilityGraphEditorContext::CreateAccessibility↵
Wrapper (
    const TSharedRef< SWidget > & ContentToWrap,
    const int32 & Index ) [protected]
```

Creates a Content Indexer wrapping the provided Widget.

##### Parameters

|                      |                                      |
|----------------------|--------------------------------------|
| <i>ContentToWrap</i> | The Content to Wrap with an Indexer. |
| <i>Index</i>         | The Index of the Provided Content.   |

##### Returns

A Shared Reference of the created Content Indexer, wrapping the provided Content.

Definition at line 349 of file [AccessibilityGraphEditorContext.cpp](#).

```
00350 {
00351     return SNew(SContentIndexer)
00352         .IndexValue(Index)
00353         .IndexPositionToContent(EIndexerPosition::Left)
00354         .ContentToIndex(ContentToWrap);
00355 }
```

#### 4.49.3.5 FindGraphActionMenu()

```
bool UAccessibilityGraphEditorContext::FindGraphActionMenu (
    const TSharedRef< SWidget > & SearchRoot ) [protected]
```

Finds the SGraphActionMenu Widget descending from the provided widget.

##### Parameters

|                   |   |
|-------------------|---|
| <i>SearchRoot</i> | The Starting Point for the Widget Search. |
|-------------------|---|

##### Returns

True if a GraphActionMenu Widget was Found, otherwise False.

Definition at line 200 of file [AccessibilityGraphEditorContext.cpp](#).

```
00201 {
00202     TSharedPtr<SGraphActionMenu> GraphActionMenu = GetWidgetDescendant<SGraphActionMenu>(SearchRoot,
TEXT("SGraphActionMenu"));
00203     if (GraphActionMenu.IsValid())
00204     {
00205         GraphMenu = GraphActionMenu;
00206         FilterTextBox = GraphActionMenu->GetFilterTextBox();
00207
00208         return true;
00209     }
00210
00211     return false;
00212 }
```

### 4.49.3.6 FindStaticComponents()

```
bool UAccessibilityGraphEditorContext::FindStaticComponents (
    const TSharedRef< SWidget > & SearchRoot ) [protected]
```

Finds any Static Components of the Context Menu and sorts them into the necessary arrays.

#### Parameters

|                   |   |
|-------------------|---|
| <i>SearchRoot</i> | The Starting Point for the Widget Search. |
|-------------------|---|

#### Returns

True if Static Components were Found, otherwise False.

Definition at line 230 of file [AccessibilityGraphEditorContext.cpp](#).

```
00231 {
00232     TArray<FSlotBase*> FoundComponentSlots = GetWidgetSlotsByType(
00233         SearchRoot,
00234         TSet<FString> {
00235             TEXT("SCheckBox")
00236         }
00237     );
00238
00239     if (!FoundComponentSlots.IsEmpty())
00240     {
00241         // Sort and Index the Static Components.
00242         for (int i = 0; i < FoundComponentSlots.Num(); i++)
00243         {
00244             FSlotBase* FoundComponentSlot = FoundComponentSlots[i];
00245
00246             TSharedPtr<SWidget> DetachedWidget = FoundComponentSlot->DetachWidget();
00247             if (!DetachedWidget.IsValid())
00248                 continue;
00249
00250             int32 ComponentIndex = -1;
00251             FString ComponentType = DetachedWidget->GetTypeAsString();
00252
00253             if (ComponentType == "SCheckBox")
00254             {
00255                 ComponentIndex = CheckBoxes.Num();
00256                 CheckBoxes.Add(StaticCastSharedPtr<SCheckBox>(DetachedWidget));
00257             }
00258
00259             FoundComponentSlot->AttachWidget (
00260                 SNew(SContentIndexer)
00261                     .IndexValue(ComponentIndex)
00262                     .IndexPositionToContent(EIndexerPosition::Left)
00263                     .ContentToIndex(DetachedWidget)
00264             );
00265         }
00266
00267         return true;
00268     }
00269
00270     return false;
00271 }
```

### 4.49.3.7 FindTreeView()

```
bool UAccessibilityGraphEditorContext::FindTreeView (
    const TSharedRef< SWidget > & SearchRoot ) [protected]
```

Finds the STreeView Widget descending from the provided widget.

**Parameters**

|                   |   |
|-------------------|---|
| <i>SearchRoot</i> | The Starting Point for the Widget Search. |
|-------------------|---|

**Returns**

True if a TreeView Widget was Found, otherwise False.

Definition at line 214 of file [AccessibilityGraphEditorContext.cpp](#).

```
00215 {
00216     TSharedPtr<STreeView<TSharedPtr<FGraphNode>>> ContextTreeView =
        GetWidgetDescendant<STreeView<TSharedPtr<FGraphNode>>>(
00217         SearchRoot,
00218         TEXT("STreeView<TSharedPtr<FGraphNode>>")
00219     );
00220     if (ContextTreeView.IsValid())
00221     {
00222         TreeView = ContextTreeView;
00223
00224         return true;
00225     }
00226
00227     return false;
00228 }
```

**4.49.3.8 GetFilterText()**

```
FString UAccessibilityGraphEditorContext::GetFilterText ( )
```

Gets Filter Text of the Context Menus SearchBar, if it contains one.

**Returns**

The Current Filter Text of the Context Menus SearchBar, an Empty String on Failure.

Definition at line 137 of file [AccessibilityGraphEditorContext.cpp](#).

```
00138 {
00139     return FilterTextBox.IsValid() ? FilterTextBox.Pin()->GetText().ToString() : FString();
00140 }
```

**4.49.3.9 GetStaticIndexOffset()**

```
const int32 UAccessibilityGraphEditorContext::GetStaticIndexOffset ( ) [protected]
```

Gets the Offset in Indexes of Found Static Components of the Context Menu.

**Returns**

The Offset of the Static Components Indexes.

Definition at line 195 of file [AccessibilityGraphEditorContext.cpp](#).

```
00196 {
00197     return CheckBoxes.Num();
00198 }
```

**4.49.3.10 GetTreeViewAction()**

```
TSharedPtr< FGraphNode > UAccessibilityGraphEditorContext::GetTreeViewAction (
    const int32 & InIndex )
```

Gets an Action on the Active TreeView, based on the provided Index.



## Parameters

|                |   |
|----------------|---|
| <i>InIndex</i> | The Index of the TreeView Action to Find. |
|----------------|---|

## Returns

A Valid Shared Pointer of the Found Action, an Invalid Shared Pointer on Failure.

Definition at line 93 of file [AccessibilityGraphEditorContext.cpp](#).

```
00094 {
00095     TArrayView<const TSharedPtr<FGraphNode> > Items = TreeView.Pin()->GetItems();
00096
00097     if (TreeView.IsValid() && Items.Num() > InIndex && InIndex >= 0)
00098         return TreeView.Pin()->GetItems()[InIndex];
00099
00100     return TSharedPtr<FGraphNode>();
00101 }
```

## 4.49.3.11 Init()

```
void UAccessibilityGraphEditorContext::Init (
    TSharedPtr< IMenu > InMenu,
    TSharedPtr< FPhraseNode > InContextRoot ) [override], [virtual]
```

Initializes the Graph Editor Context Wrapper.

## Parameters

|                      |   |
|----------------------|---|
| <i>InMenu</i>        | The Interface of the Graph Editor Context Menu.                   |
| <i>InContextRoot</i> | A Reference to the Originating PhraseNode of this Context Object. |

Reimplemented from [UPhraseTreeContextMenuObject](#).

Definition at line 19 of file [AccessibilityGraphEditorContext.cpp](#).

```
00020 {
00021     Super::Init(InMenu, InContextRoot);
00022
00023     TSharedPtr<SWindow> WindowRef = Window.Pin().ToSharedRef();
00024
00025     if (!FindGraphActionMenu(WindowRef))
00026     {
00027         UE_LOG(LogOpenAccessibility, Warning, TEXT("GraphEditorContext: Cannot Find a SGraphActionMenu
Widget"));
00028     }
00029
00030     if (!FindStaticComponents(WindowRef))
00031     {
00032         UE_LOG(LogOpenAccessibility, Warning, TEXT("GraphEditorContext: Cannot Find Any Static
Components"));
00033     }
00034
00035     if (!FindTreeView(WindowRef))
00036     {
00037         UE_LOG(LogOpenAccessibility, Warning, TEXT("GraphEditorContext: Cannot Find a STreeView
Widget"));
00038     }
00039     else
00040     {
00041         TreeViewTickRequirements = FTreeViewTickRequirements();
00042     }
00043 }
```

#### 4.49.3.12 ScaleMenu()

```
void UAccessibilityGraphEditorContext::ScaleMenu (
    const float ScaleFactor = 1.5f ) [override], [virtual]
```

Scales Elements of the Context Menu, by the provided Scalar.

##### Parameters

|                    |  |
|--------------------|--|
| <i>ScaleFactor</i> | The Scalar to Scale Menu Elements By. (1.5 by Default) |
|--------------------|--|

Reimplemented from [UPhraseTreeContextMenuObject](#).

Definition at line 71 of file [AccessibilityGraphEditorContext.cpp](#).

```
00072 {
00073     Super::ScaleMenu(ScaleFactor);
00074
00075     // Scale TreeView
00076     if (TreeView.IsValid())
00077     {
00078         TSharedPtr<STreeView<TSharedPtr<FGraphNode>>> TreeViewPtr = TreeView.Pin();
00079
00080         TreeViewPtr->SetItemHeight(16 * ScaleFactor);
00081     }
00082
00083     // Scale Window Element
00084     if (Window.IsValid())
00085     {
00086         TSharedPtr<SWindow> WindowPtr = Window.Pin();
00087
00088         WindowPtr->SetSizingRule(ESizingRule::UserSized);
00089         WindowPtr->Resize(WindowPtr->GetSizeInScreen() * ScaleFactor);
00090     }
00091 }
```

#### 4.49.3.13 SelectAction()

```
void UAccessibilityGraphEditorContext::SelectAction (
    const int32 & InIndex )
```

Selects the Action on the Graph Editor Context Menu, based on the given index.

##### Parameters

|                |                                     |
|----------------|-------------------------------------|
| <i>InIndex</i> | The Index of the Action To Perform. |
|----------------|-------------------------------------|

Definition at line 103 of file [AccessibilityGraphEditorContext.cpp](#).

```
00104 {
00105     if (InIndex < 0)
00106         return;
00107
00108     if (!CheckBoxes.IsEmpty() && InIndex < CheckBoxes.Num())
00109     {
00110         if (CheckBoxes[InIndex].IsValid())
00111         {
00112             CheckBoxes[InIndex].Pin()->ToggleCheckedState();
00113             return;
00114         }
00115     }
00116
00117     TSharedPtr<FGraphNode> ChosenTreeViewAction = GetTreeViewAction(InIndex -
    GetStaticIndexOffset());
00118     if (!ChosenTreeViewAction.IsValid())
```

```

00119     {
00120         UE_LOG(LogOpenAccessibility, Warning, TEXT("SelectGraphAction: Provided TreeView Action is
Invalid"))
00121         return;
00122     }
00123
00124     auto TreeViewPtr = TreeView.Pin();
00125     if (ChosenTreeViewAction->IsActionNode())
00126     {
00127         TreeViewPtr->Private_ClearSelection();
00128         TreeViewPtr->Private_SetItemSelection(ChosenTreeViewAction, true, true);
00129         TreeViewPtr->Private_SignalSelectionChanged(ESelectInfo::Type::OnMouseClicked);
00130     }
00131     else
00132     {
00133         TreeViewPtr->Private_OnItemDoubleClicked(ChosenTreeViewAction);
00134     }
00135 }

```

#### 4.49.3.14 SetFilterText()

```

void UAccessibilityGraphEditorContext::SetFilterText (
    const FString & NewString )

```

Sets the Filter Text of the Context Menus SearchBar, if it contains one.

##### Parameters

|                  |                                |
|------------------|--------------------------------|
| <i>NewString</i> | The New Text of the SearchBar. |
|------------------|--------------------------------|

Definition at line 142 of file [AccessibilityGraphEditorContext.cpp](#).

```

00143 {
00144     if (!FilterTextBox.IsValid())
00145         return;
00146
00147     FilterTextBox.Pin()->SetText (
00148         FText::FromString(NewString)
00149     );
00150 }

```

#### 4.49.3.15 SetScrollDistance()

```

void UAccessibilityGraphEditorContext::SetScrollDistance (
    const float NewDistance )

```

Sets the Scroll Distance of the Context Menus TreeView, if it contains one.

##### Parameters

|                    |                                      |
|--------------------|--------------------------------------|
| <i>NewDistance</i> | The New Distance of the Scroll Area. |
|--------------------|--------------------------------------|

Definition at line 164 of file [AccessibilityGraphEditorContext.cpp](#).

```

00165 {
00166     if (TreeView.IsValid())
00167         return;
00168
00169     TreeView.Pin()->SetScrollOffset(NewDistance);
00170 }

```

#### 4.49.3.16 SetScrollDistanceBottom()

```
void UAccessibilityGraphEditorContext::SetScrollDistanceBottom ( )
```

Sets the Scroll Distance of the Context Menus TreeView to the Bottom, if it contains one.

Definition at line 190 of file [AccessibilityGraphEditorContext.cpp](#).

```
00191 {
00192     TreeView.Pin()->ScrollToBottom();
00193 }
```

#### 4.49.3.17 SetScrollDistanceTop()

```
void UAccessibilityGraphEditorContext::SetScrollDistanceTop ( )
```

Sets the Scroll Distance of the Context Menus TreeView to the Top, if it contains one.

Definition at line 185 of file [AccessibilityGraphEditorContext.cpp](#).

```
00186 {
00187     TreeView.Pin()->ScrollToTop();
00188 }
```

#### 4.49.3.18 Tick()

```
bool UAccessibilityGraphEditorContext::Tick (
    float DeltaTime ) [override], [virtual]
```

Reimplemented from [UPhraseTreeContextMenuObject](#).

Definition at line 45 of file [AccessibilityGraphEditorContext.cpp](#).

```
00046 {
00047     Super::Tick(DeltaTime);
00048
00049     if (TreeViewCanTick())
00050     {
00051         TickTreeViewAccessibility();
00052
00053         TSharedPtr<STreeView<TSharedPtr<FGraphNode>>> TreeViewPtr = TreeView.Pin();
00054
00055         TreeViewTickRequirements.PrevSearchText = FilterTextBox.Pin()->GetText().ToString();
00056         TreeViewTickRequirements.PrevNumGeneratedChildren = TreeViewPtr->GetNumGeneratedChildren();
00057         TreeViewTickRequirements.PrevNumItemsBeingObserved = TreeViewPtr->GetNumItemsBeingObserved();
00058         TreeViewTickRequirements.PrevScrollDistance = TreeViewPtr->GetScrollDistance().Y;
00059     }
00060
00061     return true;
00062 }
```

### 4.49.3.19 TickTreeViewAccessibility()

```
void UAccessibilityGraphEditorContext::TickTreeViewAccessibility ( ) [protected]
```

Updates the TreeView Accessibility Components.

Definition at line 297 of file [AccessibilityGraphEditorContext.cpp](#).

```
00298 {
00299     if (!TreeViewRequiresTick())
00300         return;
00301
00302     TSharedPtr<STreeView<TSharedPtr<FGraphNode>> TreeViewPtr = TreeView.Pin();
00303
00304     TArray<TSharedPtr<FGraphNode> Items = TArray<TSharedPtr<FGraphNode> (
00305         TreeViewPtr->GetRootItems()
00306     );
00307
00308
00309     TSharedPtr<STableRow<TSharedPtr<FGraphNode>> ItemWidget = nullptr;
00310     const int32 IndexOffset = GetStaticIndexOffset();
00311
00312     while (Items.Num() > 0)
00313     {
00314         const TSharedPtr<FGraphNode> Item = Items[0];
00315         Items.RemoveAt(0);
00316
00317         if (TreeViewPtr->IsItemExpanded(Item))
00318             Items.Append(Item->Children);
00319
00320         ItemWidget = StaticCastSharedPtr<STableRow<TSharedPtr<FGraphNode>> (
00321             TreeViewPtr->WidgetFromItem(Item)
00322         );
00323         if (!ItemWidget.IsValid())
00324             continue;
00325
00326         TSharedPtr<SWidget> ItemContent = ItemWidget->GetContent();
00327
00328         if (ItemContent->GetType() == "SContentIndexer")
00329         {
00330             UpdateAccessibilityWidget (
00331                 StaticCastSharedPtr<SContentIndexer>(ItemContent.ToSharedRef()),
00332                 IndexOffset + ItemWidget->GetIndexInList()
00333             );
00334         }
00335         else
00336         {
00337             ItemWidget->SetContent (
00338                 CreateAccessibilityWrapper(ItemContent.ToSharedRef(), IndexOffset +
00339                 ItemWidget->GetIndexInList())
00340             );
00341         }
00342     }
```

### 4.49.3.20 TreeViewCanTick()

```
bool UAccessibilityGraphEditorContext::TreeViewCanTick ( ) [protected]
```

Checks if all required components for ticking the TreeView are available.

#### Returns

True if all required components are found for TreeView Ticking, otherwise False.

Definition at line 273 of file [AccessibilityGraphEditorContext.cpp](#).

```
00274 {
00275     return TreeView.IsValid() && GraphMenu.IsValid();
00276 }
```

#### 4.49.3.21 TreeViewRequiresTick()

```
bool UAccessibilityGraphEditorContext::TreeViewRequiresTick ( ) [protected]
```

Checks if the Dynamic TreeView Accessibility Components Require a Refresh.

##### Returns

True if the TreeView Accessibility Assets Require a Refresh.

Definition at line 278 of file [AccessibilityGraphEditorContext.cpp](#).

```
00279 {
00280     if (!TreeView.IsValid() || !GraphMenu.IsValid())
00281         return false;
00282
00283     bool bFilterTextChange = FilterTextBox.IsValid()
00284         ? FilterTextBox.Pin()->GetText().ToString() != TreeViewTickRequirements.PrevSearchText
00285         : false;
00286
00287     TSharedPtr<STreeView<TSharedPtr<FGraphActionNode>> TreeViewPtr = TreeView.Pin();
00288
00289     return (
00290         bFilterTextChange ||
00291         TreeViewPtr->GetNumItemsBeingObserved() != TreeViewTickRequirements.PrevNumItemsBeingObserved
00292     ||
00293         TreeViewPtr->GetNumGeneratedChildren() != TreeViewTickRequirements.PrevNumGeneratedChildren ||
00294         TreeViewPtr->GetScrollDistance().Y != TreeViewTickRequirements.PrevScrollDistance
00295     );
00296 }
```

#### 4.49.3.22 UpdateAccessibilityWidget()

```
void UAccessibilityGraphEditorContext::UpdateAccessibilityWidget (
    const TSharedPtr< SContentIndexer > & ContextIndexer,
    const int32 & NewIndex ) [protected]
```

Updates the provided Content Indexer Widget with the given Index.

##### Parameters

|                       |   |
|-----------------------|---|
| <i>ContextIndexer</i> | The Context Indexer Widget to Update.         |
| <i>NewIndex</i>       | The Index to update the Context Indexer With. |

Definition at line 344 of file [AccessibilityGraphEditorContext.cpp](#).

```
00345 {
00346     ContextIndexer->UpdateIndex(NewIndex);
00347 }
```

### 4.49.4 Member Data Documentation

#### 4.49.4.1 CheckBoxes

```
TArray<TWeakPtr<SCheckBox> > UAccessibilityGraphEditorContext::CheckBoxes = TArray<TWeakPtr<SCheckBox>>() [protected]
```

Definition at line 206 of file [AccessibilityGraphEditorContext.h](#).

#### 4.49.4.2 FilterTextBox

```
TWeakPtr<SEditableTextBox> UAccessibilityGraphEditorContext::FilterTextBox = TWeakPtr<SEditable↔
TextBox>() [protected]
```

Definition at line 202 of file [AccessibilityGraphEditorContext.h](#).

#### 4.49.4.3 GraphMenu

```
TWeakPtr<SGraphActionMenu> UAccessibilityGraphEditorContext::GraphMenu = TWeakPtr<SGraph↔
ActionMenu>() [protected]
```

Definition at line 201 of file [AccessibilityGraphEditorContext.h](#).

#### 4.49.4.4 TreeView

```
TWeakPtr<STreeView<TSharedPtr<FGraphActionNode> > > UAccessibilityGraphEditorContext::Tree↔
View = TWeakPtr<STreeView<TSharedPtr<FGraphActionNode>>>() [protected]
```

Definition at line 204 of file [AccessibilityGraphEditorContext.h](#).

#### 4.49.4.5 TreeViewTickRequirements

```
FTreeViewTickRequirements UAccessibilityGraphEditorContext::TreeViewTickRequirements [protected]
```

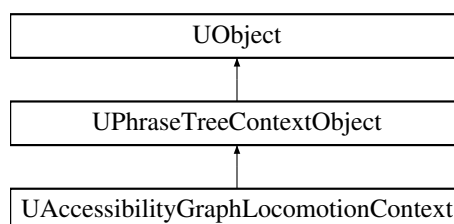
Definition at line 199 of file [AccessibilityGraphEditorContext.h](#).

The documentation for this class was generated from the following files:

- Source/OpenAccessibility/Public/AccessibilityWrappers/AccessibilityGraphEditorContext.h
- Source/OpenAccessibility/Private/AccessibilityWrappers/AccessibilityGraphEditorContext.cpp

## 4.50 UAccessibilityGraphLocomotionContext Class Reference

Inheritance diagram for UAccessibilityGraphLocomotionContext:



## Public Member Functions

- [UAccessibilityGraphLocomotionContext](#) (const FObjectInitializer &ObjectInitializer)
- void [Init](#) ()
- void [Init](#) (TSharedRef< SGraphEditor > InGraphEditor)
- bool [SelectChunk](#) (const int32 &Index)
- bool [RevertToPreviousView](#) ()
- void [ConfirmSelection](#) ()
- void [CancelLocomotion](#) ()
- virtual bool [Close](#) () override

## Protected Member Functions

- bool [MoveViewport](#) (const FVector2D &InTopLeft, const FVector2D &InBottomRight) const
- bool [MoveViewport](#) (const [FPanelViewPosition](#) &NewViewPosition) const
- void [ChangeChunkVis](#) (const int32 &Index, const FLinearColor &NewColor=FLinearColor::Yellow)
- void [CreateVisualGrid](#) (const TSharedRef< SGraphEditor > InGraphEditor)
- void [GenerateVisualChunks](#) (const TSharedRef< SGraphEditor > InGraphEditor, FIntVector2 InVisualChunkSize=FIntVector2(10))
- void [CalculateVisualChunksBounds](#) ()
- void [RemoveVisualGrid](#) ()
- void [HideNativeVisuals](#) ()
- void [UnHideNativeVisuals](#) ()
- void [OnFocusChanged](#) (const FFocusEvent &FocusEvent, const FWeakWidgetPath &OldFocusedWidgetPath, const TSharedPtr< SWidget > &OldFocusedWidget, const FWidgetPath &NewFocusedWidgetPath, const TSharedPtr< SWidget > &NewFocusedWidget)
- void [BindFocusChangedEvent](#) ()
- void [UnbindFocusChangedEvent](#) ()

## Protected Attributes

- FVector2D [StartViewPosition](#)
- float [StartViewZoom](#)
- [FPanelViewPosition](#) [CurrentViewPosition](#)
- TArray< [FPanelViewPosition](#) > [PreviousPositions](#)
- TArray< [FGraphLocomotionChunk](#) > [ChunkArray](#)
- FIntVector2 [ChunkSize](#)
- TWeakPtr< SUniformGridPanel > [GridContainer](#)
- TWeakPtr< SOverlay > [GridParent](#)
- TWeakPtr< SGraphEditor > [LinkedEditor](#)

### 4.50.1 Detailed Description

Definition at line 99 of file [AccessibilityGraphLocomotionContext.h](#).

### 4.50.2 Constructor & Destructor Documentation



### 4.50.2.1 UAccessibilityGraphLocomotionContext()

```
UAccessibilityGraphLocomotionContext::UAccessibilityGraphLocomotionContext (
    const FObjectInitializer & ObjectInitializer )
```

Definition at line 9 of file [AccessibilityGraphLocomotionContext.cpp](#).

```
00010 : UPhraseTreeContextObject()
00011 {
00012     LinkedEditor = TWeakPtr<SGraphEditor>();
00013 }
```

### 4.50.2.2 ~UAccessibilityGraphLocomotionContext()

```
UAccessibilityGraphLocomotionContext::~UAccessibilityGraphLocomotionContext ( ) [virtual]
```

Definition at line 15 of file [AccessibilityGraphLocomotionContext.cpp](#).

```
00016 {
00017     Close();
00018
00019     UE_LOG(LogOpenAccessibility, Warning, TEXT("GraphLocomotion: CONTEXT DESTROYED.));
00020 }
```

## 4.50.3 Member Function Documentation

### 4.50.3.1 BindFocusChangedEvent()

```
void UAccessibilityGraphLocomotionContext::BindFocusChangedEvent ( ) [protected]
```

Definition at line 364 of file [AccessibilityGraphLocomotionContext.cpp](#).

```
00365 {
00366     FocusChangedHandle = FSlateApplication::Get().OnFocusChanging()
00367         .AddUObject(this, &UAccessibilityGraphLocomotionContext::OnFocusChanged);
00368 }
```

### 4.50.3.2 CalculateVisualChunksBounds()

```
void UAccessibilityGraphLocomotionContext::CalculateVisualChunksBounds ( ) [protected]
```

Definition at line 248 of file [AccessibilityGraphLocomotionContext.cpp](#).

```
00249 {
00250     if (!LinkedEditor.IsValid())
00251         return;
00252
00253     SGraphPanel* LinkedPanel = LinkedEditor.Pin()->GetGraphPanel();
00254     FVector2D PanelGeoSize = LinkedPanel->GetTickSpaceGeometry().GetLocalSize();
00255
00256     double ChunkWidgetSizeX = PanelGeoSize.X / ChunkSize.X;
00257     double ChunkWidgetSizeY = PanelGeoSize.Y / ChunkSize.Y;
00258
00259     FGraphLocomotionChunk Chunk;
00260     double ChunkX, ChunkY;
00261
00262     int32 ArrIndex;
00263     for (int Y = 0; Y < ChunkSize.Y; Y++)
```

```

00264     {
00265         for (int X = 0; X < ChunkSize.X; X++)
00266         {
00267             ArrIndex = (Y * ChunkSize.X) + X;
00268
00269             Chunk = ChunkArray[ArrIndex];
00270
00271             ChunkX = X * ChunkWidgetSizeX;
00272             ChunkY = Y * ChunkWidgetSizeY;
00273
00274             Chunk.SetChunkBounds (
00275                 FVector2D (ChunkX, ChunkY),
00276                 FVector2D (ChunkWidgetSizeX + ChunkX, ChunkWidgetSizeY + ChunkY)
00277             );
00278
00279             ChunkArray[ArrIndex] = Chunk;
00280         }
00281     }
00282 }

```

#### 4.50.3.3 CancelLocomotion()

```
void UAccessibilityGraphLocomotionContext::CancelLocomotion ( )
```

Definition at line 121 of file [AccessibilityGraphLocomotionContext.cpp](#).

```

00122 {
00123     if (LinkedEditor.IsValid())
00124     {
00125         LinkedEditor.Pin()->SetViewLocation(StartViewPosition, StartViewZoom);
00126
00127         Close();
00128     }
00129 }

```

#### 4.50.3.4 ChangeChunkVis()

```
void UAccessibilityGraphLocomotionContext::ChangeChunkVis (
    const int32 & Index,
    const FLinearColor & NewColor = FLinearColor::Yellow ) [protected]
```

Definition at line 172 of file [AccessibilityGraphLocomotionContext.cpp](#).

```

00173 {
00174     check(Index < ChunkArray.Num() && Index >= 0)
00175
00176     ChunkArray[Index].SetVisColor(NewColor);
00177 }

```

#### 4.50.3.5 Close()

```
bool UAccessibilityGraphLocomotionContext::Close ( ) [override], [virtual]
```

Reimplemented from [UPhraseTreeContextObject](#).

Definition at line 131 of file [AccessibilityGraphLocomotionContext.cpp](#).

```

00132 {
00133     UnbindFocusChangedEvent();
00134
00135     if (SelectionTimerHandle.IsValid())
00136         GEditor->GetTimerManager()->ClearTimer(SelectionTimerHandle);
00137 }

```

```

00138     RemoveVisualGrid();
00139     UnHideNativeVisuals();
00140
00141     bIsActive = false;
00142
00143     RemoveFromRoot();
00144     MarkAsGarbage();
00145
00146     UE_LOG(LogOpenAccessibility, Warning, TEXT("GraphLocomotion: CONTEXT CLOSED.));
00147
00148     return true;
00149 }

```

#### 4.50.3.6 ConfirmSelection()

```
void UAccessibilityGraphLocomotionContext::ConfirmSelection ( )
```

Definition at line 116 of file [AccessibilityGraphLocomotionContext.cpp](#).

```

00117 {
00118     Close();
00119 }

```

#### 4.50.3.7 CreateVisualGrid()

```
void UAccessibilityGraphLocomotionContext::CreateVisualGrid (
    const TSharedRef< SGraphEditor > InGraphEditor ) [protected]
```

Definition at line 179 of file [AccessibilityGraphLocomotionContext.cpp](#).

```

00180 {
00181     TSharedPtr<SOverlay> GraphViewport =
        StaticCastSharedPtr<SOverlay>(InGraphEditor->GetGraphPanel()->GetParentWidget());
00182     if (!GraphViewport.IsValid())
00183     {
00184         UE_LOG(LogOpenAccessibility, Warning, TEXT("GraphLocomotion: NO GRAPH VIEWPORT FOUND.));
00185         return;
00186     }
00187
00188     GridParent = GraphViewport;
00189
00190     GraphViewport->AddSlot()
00191         .ZOrder(1)
00192         .VAlign(VAlign_Fill)
00193         .HAlign(HAlign_Fill)
00194     [
00195         SAssignNew(GridContainer, SUniformGridPanel)
00196     ];
00197 }

```

#### 4.50.3.8 GenerateVisualChunks()

```
void UAccessibilityGraphLocomotionContext::GenerateVisualChunks (
    const TSharedRef< SGraphEditor > InGraphEditor,
    FIntVector2 InVisualChunkSize = FIntVector2(10) ) [protected]
```

Definition at line 199 of file [AccessibilityGraphLocomotionContext.cpp](#).

```

00200 {
00201     ChunkArray.Reset(InVisualChunkSize.X * InVisualChunkSize.Y);
00202     ChunkSize = InVisualChunkSize;
00203
00204     TSharedPtr<SUniformGridPanel> GridContainerPtr = GridContainer.Pin();

```

```

00205
00206     int32 ChunkIndex = -1;
00207     TSharedPtr<SBox> ChunkWidget;
00208     TSharedPtr<SBorder> ChunkVisWidget;
00209     TSharedPtr<SIndexer> ChunkIndexer;
00210
00211     for (int32 Y = 0; Y < InVisualChunkSize.Y; Y++)
00212     {
00213         for (int32 X = 0; X < InVisualChunkSize.X; X++)
00214         {
00215             ChunkIndex = X + (Y * InVisualChunkSize.X);
00216             FGraphLocomotionChunk& GraphChunk = ChunkArray.EmplaceAt_GetRef(ChunkIndex);
00217
00218             GridContainerPtr->AddSlot(X, Y)
00219             [
00220                 SAssignNew(ChunkWidget, SBox)
00221                 [
00222                     SAssignNew(ChunkVisWidget, SBorder)
00223                     .Padding(0.5f)
00224                     .BorderBackgroundColor(FLinearColor::Yellow)
00225                     [
00226                         SNew(SBorder)
00227                         .HAlign(HAlign_Center)
00228                         .VAlign(VAlign_Center)
00229                         .BorderBackgroundColor(FLinearColor::Yellow)
00230                         [
00231                             SAssignNew(ChunkIndexer, SIndexer)
00232                             .TextColor(FLinearColor::Yellow)
00233                             .IndexValue(ChunkIndex)
00234                         ]
00235                     ]
00236                 ]
00237             ];
00238
00239             GraphChunk.ChunkWidget = ChunkWidget;
00240             GraphChunk.ChunkVisWidget = ChunkVisWidget;
00241             GraphChunk.ChunkIndexer = ChunkIndexer;
00242         }
00243     }
00244
00245     CalculateVisualChunksBounds();
00246 }

```

#### 4.50.3.9 HideNativeVisuals()

```
void UAccessibilityGraphLocomotionContext::HideNativeVisuals ( ) [protected]
```

Definition at line 302 of file [AccessibilityGraphLocomotionContext.cpp](#).

```

00303 {
00304     NativeWidgetVisibility.Empty();
00305
00306     TSharedPtr<SOverlay> GraphViewport = GridParent.Pin();
00307     TSharedPtr<SUniformGridPanel> VisualGrid = GridContainer.Pin();
00308     SGraphPanel* GraphPanel = LinkedEditor.Pin()->GetGraphPanel();
00309
00310     FChildren* ViewportChildren = GraphViewport->GetChildren();
00311
00312     TSharedPtr<SWidget> ChildWidget;
00313     for (int32 i = 0; i < ViewportChildren->Num(); i++)
00314     {
00315         ChildWidget = ViewportChildren->GetChildAt(i);
00316
00317         if (ChildWidget != VisualGrid && ChildWidget.Get() != GraphPanel)
00318         {
00319             NativeWidgetVisibility.Add(ChildWidget.Get(), ChildWidget->GetVisibility());
00320
00321             ChildWidget->SetVisibility(EVisibility::Hidden);
00322         }
00323     }
00324 }

```

**4.50.3.10 Init() [1/2]**

```
void UAccessibilityGraphLocomotionContext::Init ( )
```

Definition at line 22 of file [AccessibilityGraphLocomotionContext.cpp](#).

```
00023 {
00024     {
00025         TSharedPtr<SDockTab> ActiveTab = FGlobalTabmanager::Get()->GetActiveTab();
00026         if (!ActiveTab.IsValid())
00027         {
00028             UE_LOG(LogOpenAccessibility, Warning, TEXT("GraphLocomotion: NO ACTIVE TAB FOUND.));
00029             return;
00030         }
00031
00032         LinkedEditor = StaticCastSharedRef<SGraphEditor>(ActiveTab->GetContent());
00033         if (!LinkedEditor.IsValid())
00034         {
00035             UE_LOG(LogOpenAccessibility, Warning, TEXT("GraphLocomotion: CURRENT ACTIVE TAB IS NOT OF
TYPE - SGraphEditor"));
00036             return;
00037         }
00038     }
00039
00040     TSharedPtr<SGraphEditor> LinkedEditorPtr = LinkedEditor.Pin();
00041
00042     Init(LinkedEditorPtr.ToSharedRef());
00043 }
```

**4.50.3.11 Init() [2/2]**

```
void UAccessibilityGraphLocomotionContext::Init (
    TSharedRef< SGraphEditor > InGraphEditor )
```

Definition at line 45 of file [AccessibilityGraphLocomotionContext.cpp](#).

```
00046 {
00047     LinkedEditor = InGraphEditor;
00048
00049     InGraphEditor->GetViewLocation(StartViewPosition, StartViewZoom);
00050     InGraphEditor->ZoomToFit(false);
00051
00052     CreateVisualGrid(InGraphEditor);
00053     GenerateVisualChunks(InGraphEditor, FIntVector2(6, 4));
00054
00055     HideNativeVisuals();
00056
00057     BindFocusChangedEvent();
00058 }
```

**4.50.3.12 MoveViewport() [1/2]**

```
bool UAccessibilityGraphLocomotionContext::MoveViewport (
    const FPanelViewPosition & NewViewPosition ) const [protected]
```

Definition at line 162 of file [AccessibilityGraphLocomotionContext.cpp](#).

```
00163 {
00164     if (!LinkedEditor.IsValid())
00165         return false;
00166
00167     SGraphPanel* LinkedPanel = LinkedEditor.Pin()->GetGraphPanel();
00168
00169     return LinkedPanel->JumpToRect(NewViewPosition.TopLeft, NewViewPosition.BotRight);
00170 }
```

#### 4.50.3.13 MoveViewport() [2/2]

```
bool UAccessibilityGraphLocomotionContext::MoveViewport (
    const FVector2D & InTopLeft,
    const FVector2D & InBottomRight ) const [protected]
```

Definition at line 151 of file [AccessibilityGraphLocomotionContext.cpp](#).

```
00152 {
00153     if (!LinkedEditor.IsValid())
00154         return false;
00155
00156     TSharedPtr<SGraphEditor> LinkedEditorPtr = LinkedEditor.Pin();
00157     SGraphPanel* LinkedPanel = LinkedEditorPtr->GetGraphPanel();
00158
00159     return LinkedPanel->JumpToRect(InTopLeft, InBottomRight);
00160 }
```

#### 4.50.3.14 OnFocusChanged()

```
void UAccessibilityGraphLocomotionContext::OnFocusChanged (
    const FFocusEvent & FocusEvent,
    const FWeakWidgetPath & OldFocusedWidgetPath,
    const TSharedPtr< SWidget > & OldFocusedWidget,
    const FWidgetPath & NewFocusedWidgetPath,
    const TSharedPtr< SWidget > & NewFocusedWidget ) [protected]
```

Definition at line 344 of file [AccessibilityGraphLocomotionContext.cpp](#).

```
00349 {
00350     if (!bIsActive)
00351         return;
00352
00353     UE_LOG(LogOpenAccessibility, Warning, TEXT("GraphLocomotion: FOCUS CHANGED."));
00354
00355     TSharedPtr<SGraphEditor> LinkedEditorPtr = LinkedEditor.Pin();
00356
00357     if (!NewFocusedWidgetPath.ContainsWidget(LinkedEditorPtr.ToSharedRef()))
00358     {
00359         bIsActive = false;
00360         Close();
00361     }
00362 }
```

#### 4.50.3.15 RemoveVisualGrid()

```
void UAccessibilityGraphLocomotionContext::RemoveVisualGrid ( ) [protected]
```

Definition at line 284 of file [AccessibilityGraphLocomotionContext.cpp](#).

```
00285 {
00286     TSharedPtr<SUniformGridPanel> GridContainerPtr = GridContainer.Pin();
00287     if (GridContainerPtr.IsValid())
00288     {
00289         TSharedPtr<SOverlay> ParentWidget = StaticCastSharedPtr<SOverlay>(
00290             GridContainerPtr->GetParentWidget()
00291         );
00292
00293         if (ParentWidget.IsValid()) {
00294             ParentWidget->RemoveSlot(GridContainerPtr.ToSharedRef());
00295
00296             GridParent = ParentWidget;
00297         }
00298         else UE_LOG(LogOpenAccessibility, Warning, TEXT("GraphLocomotion: PARENT WIDGET NOT FOUND,
00299             CANNOT REMOVE LOCOMOTION WIDGETS.))
00300     }
```

## 4.50.3.16 RevertToPreviousView()

```
bool UAccessibilityGraphLocomotionContext::RevertToPreviousView ( )
```

Definition at line 100 of file [AccessibilityGraphLocomotionContext.cpp](#).

```
00101 {
00102     if (PreviousPositions.IsEmpty())
00103     {
00104         LinkedEditor.Pin()->ZoomToFit(false);
00105         return true;
00106     }
00107
00108     if (!MoveViewport(PreviousPositions.Pop()))
00109     {
00110         return false;
00111     }
00112
00113     return true;
00114 }
```

## 4.50.3.17 SelectChunk()

```
bool UAccessibilityGraphLocomotionContext::SelectChunk (
    const int32 & Index )
```

Definition at line 60 of file [AccessibilityGraphLocomotionContext.cpp](#).

```
00061 {
00062     if (Index > ChunkArray.Num() || Index < 0)
00063         return false;
00064
00065     const FGraphLocomotionChunk SelectedChunk = ChunkArray[Index];
00066
00067     const SGraphPanel* LinkedPanel = LinkedEditor.Pin()->GetGraphPanel();
00068
00069     const FVector2D GraphTopLeftCoord =
00070     LinkedPanel->PanelCoordToGraphCoord(SelectedChunk.GetChunkTopLeft());
00071     const FVector2D GraphBottomRightCoord =
00072     LinkedPanel->PanelCoordToGraphCoord(SelectedChunk.GetChunkBottomRight());
00073
00074     ChangeChunkVis(Index, FLinearColor::Red);
00075
00076     GEditor->GetTimerManager()->SetTimer(
00077     SelectionTimerHandle,
00078     [this, Index, GraphTopLeftCoord, GraphBottomRightCoord]()
00079     {
00080         ChangeChunkVis(Index);
00081
00082         if (MoveViewport(GraphTopLeftCoord, GraphBottomRightCoord))
00083         {
00084             if (CurrentViewPosition != FVector2D::ZeroVector)
00085                 PreviousPositions.Push(CurrentViewPosition);
00086
00087             CurrentViewPosition = FPanelViewPosition(GraphTopLeftCoord, GraphBottomRightCoord);
00088         }
00089         else
00090         {
00091             UE_LOG(LogOpenAccessibility, Log, TEXT("Failed To Jump To Viewport Coords (TopLeft: %s
00092             | BottomRight: %s)",
00093             *GraphTopLeftCoord.ToString(), *GraphBottomRightCoord.ToString()));
00094         }
00095     },
00096     0.5f,
00097     false
00098 );
00099
00100     return true;
00101 }
```

#### 4.50.3.18 UnbindFocusChangedEvent()

```
void UAccessibilityGraphLocomotionContext::UnbindFocusChangedEvent ( ) [protected]
```

Definition at line 370 of file [AccessibilityGraphLocomotionContext.cpp](#).

```
00371 {
00372     if (FocusChangedHandle.IsValid())
00373     {
00374         FSlateApplication::Get().OnFocusChanging().Remove(FocusChangedHandle);
00375     }
00376 }
```

#### 4.50.3.19 UnHideNativeVisuals()

```
void UAccessibilityGraphLocomotionContext::UnHideNativeVisuals ( ) [protected]
```

Definition at line 326 of file [AccessibilityGraphLocomotionContext.cpp](#).

```
00327 {
00328     FChildren* ViewportChildren = GridParent.Pin()->GetChildren();
00329
00330     TSharedPtr<SWidget> ChildWidget;
00331     for (int32 i = 0; i < ViewportChildren->Num(); i++)
00332     {
00333         ChildWidget = ViewportChildren->GetChildAt(i);
00334
00335         if (NativeWidgetVisibility.Contains(ChildWidget.Get()))
00336         {
00337             ChildWidget->SetVisibility(NativeWidgetVisibility[ChildWidget.Get()]);
00338         }
00339     }
00340
00341     NativeWidgetVisibility.Empty();
00342 }
```

### 4.50.4 Member Data Documentation

#### 4.50.4.1 ChunkArray

```
TArray<FGraphLocomotionChunk> UAccessibilityGraphLocomotionContext::ChunkArray [protected]
```

Definition at line 160 of file [AccessibilityGraphLocomotionContext.h](#).

#### 4.50.4.2 ChunkSize

```
FIntVector2 UAccessibilityGraphLocomotionContext::ChunkSize [protected]
```

Definition at line 162 of file [AccessibilityGraphLocomotionContext.h](#).



#### 4.50.4.3 CurrentViewPosition

`FPanelViewPosition` `UAccessibilityGraphLocomotionContext::CurrentViewPosition` [protected]

Definition at line 155 of file [AccessibilityGraphLocomotionContext.h](#).

#### 4.50.4.4 GridContainer

`TWeakPtr<SUniformGridPanel>` `UAccessibilityGraphLocomotionContext::GridContainer` [protected]

Definition at line 167 of file [AccessibilityGraphLocomotionContext.h](#).

#### 4.50.4.5 GridParent

`TWeakPtr<SOverlay>` `UAccessibilityGraphLocomotionContext::GridParent` [protected]

Definition at line 169 of file [AccessibilityGraphLocomotionContext.h](#).

#### 4.50.4.6 LinkedEditor

`TWeakPtr<SGraphEditor>` `UAccessibilityGraphLocomotionContext::LinkedEditor` [protected]

Definition at line 171 of file [AccessibilityGraphLocomotionContext.h](#).

#### 4.50.4.7 PreviousPositions

`TArray<FPanelViewPosition>` `UAccessibilityGraphLocomotionContext::PreviousPositions` [protected]

Definition at line 156 of file [AccessibilityGraphLocomotionContext.h](#).

#### 4.50.4.8 StartViewPosition

`FVector2D` `UAccessibilityGraphLocomotionContext::StartViewPosition` [protected]

Definition at line 153 of file [AccessibilityGraphLocomotionContext.h](#).

#### 4.50.4.9 StartViewZoom

```
float UAccessibilityGraphLocomotionContext::StartViewZoom [protected]
```

Definition at line 153 of file [AccessibilityGraphLocomotionContext.h](#).

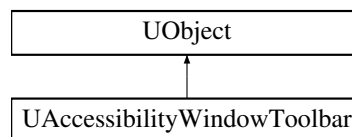
The documentation for this class was generated from the following files:

- Source/OpenAccessibility/Public/AccessibilityWrappers/AccessibilityGraphLocomotionContext.h
- Source/OpenAccessibility/Private/AccessibilityWrappers/AccessibilityGraphLocomotionContext.cpp

## 4.51 UAccessibilityWindowToolbar Class Reference

```
#include <AccessibilityWindowToolbar.h>
```

Inheritance diagram for UAccessibilityWindowToolbar:



### Public Member Functions

- bool [Tick](#) (float DeltaTime)
- void [SelectToolbarItem](#) (int32 Index)
- bool [IsActiveToolbar](#) (const TSharedRef< SWidget > &ToolkitWidget)
- TSharedPtr< SWidget > [GetActiveToolkitWidget](#) () const

#### 4.51.1 Detailed Description

Accessibility Wrapper for Window ToolBar Elements.

Definition at line 15 of file [AccessibilityWindowToolbar.h](#).

#### 4.51.2 Constructor & Destructor Documentation

### 4.51.2.1 UAccessibilityWindowToolbar()

UAccessibilityWindowToolbar::UAccessibilityWindowToolbar ( )

Definition at line 9 of file [AccessibilityWindowToolbar.cpp](#).

```
00009                                     : UObject()
00010 {
00011     LastToolkit = TWeakPtr<SWidget>();
00012     LastTopWindow = TWeakPtr<SWindow>();
00013     LastToolkitParent = TWeakPtr<SBorder>();
00014
00015     ConsoleCommands.Add(IconsoleManager::Get().RegisterConsoleCommand(
00016         TEXT("OpenAccessibiliy.ToolBar.ShowIndexerStats"),
00017         TEXT("Displays the Indexer Stats for the Toolbar."),
00018
00019         FConsoleCommandDelegate::CreateLambda([this]() {
00020             UE_LOG(LogOpenAccessibility, Display, TEXT("| ToolBar Indexer Stats | Indexed Amount: %d |
00021             "), ToolbarIndex.Num())
00022         }));
00023
00024     BindTicker();
00025 }
```

### 4.51.2.2 ~UAccessibilityWindowToolbar()

UAccessibilityWindowToolbar::~UAccessibilityWindowToolbar ( ) [virtual]

Definition at line 27 of file [AccessibilityWindowToolbar.cpp](#).

```
00028 {
00029     UE_LOG(LogOpenAccessibility, Log, TEXT("AccessibilityToolBar: Destroyed.));
00030
00031     UnbindTicker();
00032 }
```

## 4.51.3 Member Function Documentation

### 4.51.3.1 GetActiveToolkitWidget()

TSharedPtr< SWidget > UAccessibilityWindowToolbar::GetActiveToolkitWidget ( ) const

Gets the Stored Active Toolkit Widget.

#### Returns

Shared Pointer to the Active Toolkit Widget, otherwise Invalid Pointer.

Definition at line 240 of file [AccessibilityWindowToolbar.cpp](#).

```
00241 {
00242     if (LastToolkit.IsValid())
00243         return LastToolkit.Pin();
00244
00245     return TSharedPtr<SWidget>();
00246 }
```

### 4.51.3.2 IsActiveToolbar()

bool UAccessibilityWindowToolbar::IsActiveToolbar (
 const TSharedRef< SWidget > & ToolkitWidget )

Checks to see if the Active Toolkit being Indexed is the provided Toolkit Widget.

**Parameters**

|                      |  |
|----------------------|--|
| <i>ToolkitWidget</i> | Toolkit Widget to Check if it is the active toolkit being Indexed. |
|----------------------|--|

**Returns**

True if the provided toolkit is the active widget, otherwise False.

Definition at line 233 of file [AccessibilityWindowToolbar.cpp](#).

```
00234 {
00235     return LastToolkit.IsValid()
00236         ? LastToolkit.Pin() == ToolkitWidget
00237         : false;
00238 }
```

**4.51.3.3 SelectToolbarItem()**

```
void UAccessibilityWindowToolbar::SelectToolbarItem (
    int32 Index )
```

Selects the Active ToolBars Element, based on the provided Index.

**Parameters**

|              |   |
|--------------|---|
| <i>Index</i> | The Index of the ToolBar Element To Select. |
|--------------|---|

Definition at line 197 of file [AccessibilityWindowToolbar.cpp](#).

```
00198 {
00199     if (ToolbarIndex.IsEmpty())
00200     {
00201         UE_LOG(LogOpenAccessibility, Warning, TEXT("ToolBar Index is Empty.))
00202         return;
00203     }
00204
00205     SMultiBlockBaseWidget* LinkedButton;
00206     if (!ToolbarIndex.GetValue(Index, LinkedButton))
00207     {
00208         UE_LOG(LogOpenAccessibility, Warning, TEXT("Provided Index is Not Linked to a ToolBar
Button.))
00209         return;
00210     }
00211
00212     TSharedPtr<const FMultiBlock> MultiBlock = LinkedButton->GetBlock();
00213     if (!MultiBlock.IsValid())
00214     {
00215         UE_LOG(LogOpenAccessibility, Warning, TEXT("Provided ToolBar MultiBlock is Not Valid.))
00216     }
00217
00218     TSharedPtr<const FUICommandList> ActionList = MultiBlock->GetActionList();
00219     TSharedPtr<const FUICommandInfo> Action = MultiBlock->GetAction();
00220
00221     if (ActionList.IsValid() && Action.IsValid())
00222     {
00223         ActionList->ExecuteAction( Action.ToSharedRef() );
00224     }
00225     else
00226     {
00227         const FUIAction& DirectAction = MultiBlock->GetDirectActions();
00228
00229         DirectAction.Execute();
00230     }
00231 }
```

## 4.51.3.4 Tick()

```
bool UAccessibilityWindowToolbar::Tick (
    float DeltaTime )
```

Definition at line 34 of file [AccessibilityWindowToolbar.cpp](#).

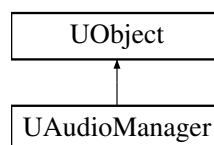
```
00035 {
00036     TSharedPtr<SWindow> TopWindow = FSlateApplication::Get().GetActiveTopLevelRegularWindow();
00037     if (!TopWindow.IsValid())
00038     {
00039         return true;
00040     }
00041
00042     TSharedPtr<SBorder> ContentContainer;
00043     if (TopWindow != LastTopWindow)
00044         ContentContainer = GetWindowContentContainer(TopWindow.ToSharedRef());
00045     else ContentContainer = LastToolkitParent.Pin();
00046
00047     if (!ContentContainer.IsValid())
00048     {
00049         return true;
00050     }
00051
00052
00053     TSharedPtr<SWidget> Toolkit = ContentContainer->GetContent();
00054     if (!Toolkit.IsValid())
00055     {
00056         return true;
00057     }
00058
00059     if (ApplyToolbarIndexing(Toolkit.ToSharedRef(), TopWindow.ToSharedRef()))
00060     {
00061         LastToolkit = Toolkit;
00062         //UE_LOG(LogOpenAccessibility, Log, TEXT("AccessibilityToolBar: Toolkit Indexing Applied To
00063         %s"), *Toolkit->GetTypeAsString());
00064     }
00065
00066     LastTopWindow = TopWindow;
00067     LastToolkitParent = ContentContainer;
00068     return true;
00069 }
```

The documentation for this class was generated from the following files:

- Source/OpenAccessibility/Public/AccessibilityWrappers/AccessibilityWindowToolbar.h
- Source/OpenAccessibility/Private/AccessibilityWrappers/AccessibilityWindowToolbar.cpp

## 4.52 UAudioManager Class Reference

Inheritance diagram for UAudioManager:



## Public Member Functions

- void [StartCapturingAudio](#) ()  
*Starts The Capturing of Audio onto the Buffer.*
- void [StopCapturingAudio](#) ()  
*Stops the Capturing of Audio onto the Buffer, and sends the audio to the transcription service.*

- void [PRIVATE\\_OnAudioGenerate](#) (const float \*InAudio, int32 NumSamples)  
*Callback For When Audio is Generated by The Audio Stream.*
- void [SaveAudioBufferToWAV](#) (const FString &FilePath)  
*Saves the Audio Buffer to a WAV File.*
- bool [IsCapturingAudio](#) () const  
*Is the Audio Manager Currently Capturing Audio.*
- int32 [GetAudioCaptureSampleRate](#) () const  
*Gets the Sample Rate of the Audio Capture.*
- int32 [GetAudioCaptureNumChannels](#) () const  
*Gets the Number of Channels of the Audio Capture.*
- void [OnDefaultDeviceChanged](#) (EAudioDeviceChangedRole ChangedRole, FString DeviceID)  
*Callback for when the Default Audio Device Changes. Allowing for dynamic re-registration of the Audio Generator, to make sure the new device is being used.*

## Public Attributes

- [FAudioManagerSettings Settings](#)  
*The Settings of the Audio Manager.*
- TDelegate< void(const TArray< float >)> [OnAudioReadyForTranscription](#)  
*Delegate for when the AudioBuffer is Ready To Be Sent For Transcription.*

## 4.52.1 Detailed Description

Definition at line 50 of file [AudioManager.h](#).

## 4.52.2 Constructor & Destructor Documentation

### 4.52.2.1 UAudioManager()

`UAudioManager::UAudioManager ( )`

Definition at line 12 of file [AudioManager.cpp](#).

```
00013 {
00014     Settings = FAudioManagerSettings();
00015
00016     // Create Audio Capture Object and Initialize Audio Stream
00017     bIsCapturingAudio = false;
00018     AudioCapture = NewObject<UAudioCapture>();
00019     AudioCapture->OpenDefaultAudioStream();
00020     AudioCapture->StartCapturingAudio();
00021
00022     RegisterAudioGenerator();
00023
00024     // Create FileIO Objects
00025     FileWriter = new Audio::FSoundWavePCMWriter();
00026 }
```

#### 4.52.2.2 ~UAudioManager()

UAudioManager::~UAudioManager ( ) [virtual]

Definition at line 28 of file [AudioManager.cpp](#).

```
00029 {  
00030     UnregisterAudioGenerator();  
00031  
00032     AudioCapture->StopCapturingAudio();  
00033     AudioCapture->RemoveFromRoot();  
00034  
00035     delete AudioCapture; AudioCapture = nullptr;  
00036     delete FileWriter; FileWriter = nullptr;  
00037 }
```

### 4.52.3 Member Function Documentation

#### 4.52.3.1 GetAudioCaptureNumChannels()

int32 UAudioManager::GetAudioCaptureNumChannels ( ) const [inline]

Gets the Number of Channels of the Audio Capture.

##### Returns

The Number of Channels used in the Audiocapture.

Definition at line 97 of file [AudioManager.h](#).

```
00097 { return AudioCapture->GetNumChannels(); }
```

#### 4.52.3.2 GetAudioCaptureSampleRate()

int32 UAudioManager::GetAudioCaptureSampleRate ( ) const [inline]

Gets the Sample Rate of the Audio Capture.

##### Returns

The Sample Rate of the Audiocapture.

Definition at line 91 of file [AudioManager.h](#).

```
00091 { return AudioCapture->GetSampleRate(); }
```

#### 4.52.3.3 IsCapturingAudio()

```
bool UAudioManager::IsCapturingAudio ( ) const [inline]
```

Is the Audio Manager Currently Capturing Audio.

##### Returns

True, if Audio is being Captured. False, if Audio is being ignored.

Definition at line 85 of file [AudioManager.h](#).

```
00085 { return bIsCapturingAudio; }
```

#### 4.52.3.4 OnDefaultDeviceChanged()

```
void UAudioManager::OnDefaultDeviceChanged (
    EAudioDeviceChangedRole ChangedRole,
    FString DeviceID )
```

Callback for when the Default Audio Device Changes. Allowing for dynamic re-registration of the Audio Generator, to make sure the new device is being used.

##### Parameters

|                    |  |
|--------------------|--|
| <i>ChangedRole</i> |  |
| <i>DeviceID</i>    |  |

Definition at line 88 of file [AudioManager.cpp](#).

```
00089 {
00090     UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Default Device Changed || Role: %d || DeviceID: %s
||"), ChangedRole, *DeviceID);
00091
00092     this->UnregisterAudioGenerator();
00093     this->RegisterAudioGenerator();
00094 }
```

#### 4.52.3.5 PRIVATE\_OnAudioGenerate()

```
void UAudioManager::PRIVATE_OnAudioGenerate (
    const float * InAudio,
    int32 NumSamples )
```

Callback For When Audio is Generated by The Audio Stream.

##### Parameters

|                   |  |
|-------------------|--|
| <i>InAudio</i>    | - The Incoming Audiobuffer Array.                  |
| <i>NumSamples</i> | - The Size of the Incoming Audiobuffer in Samples. |



Definition at line 67 of file [AudioManager.cpp](#).

```
00068 {
00069     if (bIsCapturingAudio == false)
00070         return;
00071
00072     // Need to Check Samples are above threshold and ignore if their run length is too long.
00073
00074     AudioBuffer.Append(InAudio, NumSamples);
00075 }
```

#### 4.52.3.6 SaveAudioBufferToWAV()

```
void UAudioManager::SaveAudioBufferToWAV (
    const FString & FilePath )
```

Saves the Audio Buffer to a WAV File.

##### Parameters

|                 |   |
|-----------------|---|
| <i>FilePath</i> | - The Path To Save the Audiobuffers WAV File. |
|-----------------|---|

Definition at line 77 of file [AudioManager.cpp](#).

```
00078 {
00079     UE_LOG(LogOpenAccessibilityCom, Log, TEXT("Starting to Save Audio Buffer to WAV"));
00080
00081     Audio::FSampleBuffer SampleBuffer = Audio::FSampleBuffer(AudioBuffer.GetData(), AudioBuffer.Num(),
00082         AudioCapture->GetNumChannels(), AudioCapture->GetSampleRate());
00083     FileWriter->BeginWriteToWavFile(SampleBuffer, Settings.SaveName, const_cast<FString*>(FilePath),
00084     []() {
00085         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("Audio Buffer Saved to WAV"));
00086     });
00086 }
```

#### 4.52.3.7 StartCapturingAudio()

```
void UAudioManager::StartCapturingAudio ( )
```

Starts The Capturing of Audio onto the Buffer.

Definition at line 39 of file [AudioManager.cpp](#).

```
00040 {
00041     AudioBuffer.Empty();
00042
00043     bIsCapturingAudio = true;
00044 }
```

#### 4.52.3.8 StopCapturingAudio()

```
void UAudioManager::StopCapturingAudio ( )
```

Stops the Capturing of Audio onto the Buffer, and sends the audio to the transcription service.

Definition at line 46 of file [AudioManager.cpp](#).

```

00047 {
00048     bIsCapturingAudio = false;
00049
00050     if (AudioBuffer.Num() == 0)
00051         return;
00052
00053     SaveAudioBufferToWAV(Settings.SavePath);
00054
00055     if (OnAudioReadyForTranscription.ExecuteIfBound(AudioBuffer))
00056     {
00057         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Executing Audio Ready For Transcription
00058         Delegate. ||"));
00059     }
00059     else
00060     {
00061         UE_LOG(LogOpenAccessibilityCom, Warning, TEXT("|| No Delegates Bound to Audio Ready For
00062         Transcription Delegate. ||"));
00063     }
00064     AudioBuffer.Empty();
00065 }

```

## 4.52.4 Member Data Documentation

### 4.52.4.1 OnAudioReadyForTranscription

TDelegate<void(const TArray<float>)> UAudioManager::OnAudioReadyForTranscription

Delegate for when the AudioBuffer is Ready To Be Sent For Transcription.

Definition at line 124 of file [AudioManager.h](#).

### 4.52.4.2 Settings

FAudioManagerSettings UAudioManager::Settings

The Settings of the Audio Manager.

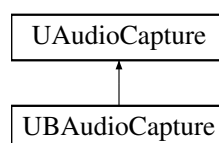
Definition at line 119 of file [AudioManager.h](#).

The documentation for this class was generated from the following files:

- Source/OpenAccessibilityCommunication/Public/AudioManager.h
- Source/OpenAccessibilityCommunication/Private/AudioManager.cpp

## 4.53 UBAudioCapture Class Reference

Inheritance diagram for UBAudioCapture:



## Public Member Functions

- bool [OpenDefaultAudioStream](#) (int32 OverrideSampleRate, int32 OverrideInputChannels)  
*Opens the default audio stream.*

### 4.53.1 Detailed Description

Definition at line 11 of file [UBAudioCapture.h](#).

### 4.53.2 Constructor & Destructor Documentation

#### 4.53.2.1 UBAudioCapture()

```
UBAudioCapture::UBAudioCapture ( )
```

Definition at line 6 of file [UBAudioCapture.cpp](#).

```
00006         : UAudioCapture()  
00007 {  
00008  
00009 }
```

#### 4.53.2.2 ~UBAudioCapture()

```
UBAudioCapture::~UBAudioCapture ( ) [virtual]
```

Definition at line 11 of file [UBAudioCapture.cpp](#).

```
00012 {  
00013 }
```

### 4.53.3 Member Function Documentation

#### 4.53.3.1 OpenDefaultAudioStream()

```
bool UBAudioCapture::OpenDefaultAudioStream (  
    int32 OverrideSampleRate,  
    int32 OverrideInputChannels )
```

Opens the default audio stream.

#### Parameters

|                              |  |
|------------------------------|--|
| <i>OverrideSampleRate</i>    | Override for the Audiobuffers Sample Rate.       |
| <i>OverrideInputChannels</i> | Override for the Amount of Input Channel Amount. |

Generated by Doxygen

**Returns**

True, if the Audiostream was opened correctly. False, if the Audio Stream could not be opened.

Definition at line 15 of file [UBAudioCapture.cpp](#).

```

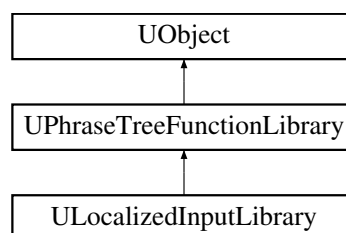
00016 {
00017     if (!AudioCapture.IsStreamOpen())
00018     {
00019         if (!AudioCapture.IsStreamOpen())
00020         {
00021             Audio::FOnAudioCaptureFunction OnCapture = [this](const void* AudioData, int32 NumFrames,
int32 InNumChannels, int32 InSampleRate, double StreamTime, bool bOverflow)
00022             {
00023                 OnGeneratedAudio((const float*)AudioData, NumFrames * InNumChannels);
00024             };
00025
00026             // Start the stream here to avoid hitching the audio render thread.
00027             Audio::FAudioCaptureDeviceParams Params;
00028             if (OverrideSampleRate != NULL)
00029                 Params.SampleRate = OverrideSampleRate;
00030             if (OverrideInputChannels != NULL)
00031                 Params.NumInputChannels = OverrideInputChannels;
00032
00033             if (AudioCapture.OpenAudioCaptureStream(Params, MoveTemp(OnCapture), 1024))
00034             {
00035                 // If we opened the capture stream succesfully, get the capture device info and
00036                 initialize the UAudioGenerator
00037                 Audio::FCaptureDeviceInfo Info;
00038                 if (AudioCapture.GetCaptureDeviceInfo(Info))
00039                 {
00040                     Init(
00041                         OverrideSampleRate != NULL ? OverrideSampleRate : Info.PreferredSampleRate ,
00042                         OverrideInputChannels != NULL ? OverrideInputChannels : Info.InputChannels
00043                     );
00044
00045                     return true;
00046                 }
00047             }
00048         }
00049         return false;
00050     }
00051     return false;
00052 }
00053
00054 }
```

The documentation for this class was generated from the following files:

- Source/OpenAccessibilityCommunication/Public/UBAudioCapture.h
- Source/OpenAccessibilityCommunication/Private/UBAudioCapture.cpp

## 4.54 ULocalizedInputLibrary Class Reference

Inheritance diagram for ULocalizedInputLibrary:



## Public Member Functions

- [ULocalizedStringLibrary](#) (const FObjectInitializer &ObjectInitializer)
- virtual void [BindBranches](#) (TSharedRef< [FPhraseTree](#) > PhraseTree) override
- void [KeyboardInputAdd](#) (FParseRecord &Record)
- void [KeyboardInputRemove](#) (FParseRecord &Record)
- void [KeyboardInputReset](#) (FParseRecord &Record)
- void [KeyboardInputConfirm](#) (FParseRecord &Record)
- void [KeyboardInputExit](#) (FParseRecord &Record)

### 4.54.1 Detailed Description

Definition at line 12 of file [LocalizedStringLibrary.h](#).

### 4.54.2 Constructor & Destructor Documentation

#### 4.54.2.1 ULocalizedStringLibrary()

```
ULocalizedStringLibrary::ULocalizedString (
    const FObjectInitializer & ObjectInitializer )
```

Definition at line 13 of file [LocalizedStringLibrary.cpp](#).

```
00014 {
00015
00016 }
```

#### 4.54.2.2 ~ULocalizedStringLibrary()

```
ULocalizedStringLibrary::~ULocalizedString ( ) [virtual]
```

Definition at line 18 of file [LocalizedStringLibrary.cpp](#).

```
00019 {
00020
00021 }
```

### 4.54.3 Member Function Documentation

#### 4.54.3.1 BindBranches()

```
void ULocalizedStringLibrary::BindBranches (
    TSharedRef< FPhraseTree > PhraseTree ) [override], [virtual]
```

Binds Branches originating from this Library onto the provided Phrase Tree.

## Parameters

|                   |  |
|-------------------|--|
| <i>PhraseTree</i> | Reference to the PhraseTree to Bind this Library to. |
|-------------------|--|

Reimplemented from [UPhraseTreeFunctionLibrary](#).

Definition at line 23 of file [LocalizedInputLibrary.cpp](#).

```

00024 {
00025     PhraseTree->BindBranch(
00026         MakeShared<FPhraseNode>(TEXT("INPUT"),
00027             TPhraseNodeArray {
00028
00029                 MakeShared<FPhraseNode>(TEXT("ADD"),
00030                     TPhraseNodeArray {
00031
00032                         MakeShared<FPhraseStringInputNode>(TEXT("PHRASE_TO_ADD"),
00033                             TPhraseNodeArray {
00034
00035                                 MakeShared<FPhraseEventNode>(CreateParseDelegate(this,
00036                                     &ULocalizedInputLibrary::KeyboardInputAdd))
00037                             })
00038                         },
00039                     ),
00040
00041                 MakeShared<FPhraseNode>(TEXT("REMOVE"),
00042                     TPhraseNodeArray {
00043
00044                         MakeShared<FPhraseInputNode<int32>>(TEXT("AMOUNT"),
00045                             TPhraseNodeArray {
00046
00047                                 MakeShared<FPhraseEventNode>(CreateParseDelegate(this, &ULocalizedInputLibrary::KeyboardInputRemove))
00048                             })
00049                         },
00050                     ),
00051                 ),
00052
00053                 MakeShared<FPhraseNode>(TEXT("RESET"),
00054                     TPhraseNodeArray {
00055
00056                         MakeShared<FPhraseEventNode>(CreateParseDelegate(this,
00057                             &ULocalizedInputLibrary::KeyboardInputReset))
00058                     },
00059                 ),
00060                 /*
00061                 MakeShared<FPhraseNode>(TEXT("CONFIRM"),
00062                     TPhraseNodeArray {
00063
00064                         MakeShared<FPhraseEventNode>(CreateParseDelegate(this,
00065                             &ULocalizedInputLibrary::KeyboardInputConfirm))
00066                     },
00067                 */
00068
00069                 MakeShared<FPhraseNode>(TEXT("EXIT"),
00070                     TPhraseNodeArray {
00071
00072                         MakeShared<FPhraseEventNode>(CreateParseDelegate(this,
00073                             &ULocalizedInputLibrary::KeyboardInputExit))
00074                     })
00075                 },
00076             );
00077     };
00078 }
```

#### 4.54.3.2 KeyboardInputAdd()

```

void ULocalizedInputLibrary::KeyboardInputAdd (
    FParseRecord & Record )
```

Phrase Event for Adding String Words to the Active Keyboard Focus.

## Parameters

|               |   |
|---------------|---|
| <i>Record</i> | The ParseRecord accumulated until this Event. |
|---------------|---|

Definition at line 80 of file [LocalizedStringLibrary.cpp](#).

```

00080                                     {
00081     GET_ACTIVE_KEYBOARD_WIDGET(KeyboardFocusedWidget);
00082
00083     FString WidgetType = KeyboardFocusedWidget->GetTypeAsString();
00084
00085     UParseStringInput *PhraseInput = Record.GetPhraseInput<UParseStringInput>(TEXT("PHRASE_TO_ADD"));
00086     if (PhraseInput == nullptr)
00087         return;
00088
00089     if (WidgetType == "SEditableText")
00090     {
00091         TSharedPtr<SEditableText> EditableText =
00092             StaticCastSharedPtr<SEditableText>(KeyboardFocusedWidget);
00093         if (!EditableText.IsValid()) {
00094             UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("KeyboardInputAdd: CURRENT ACTIVE
00095 WIDGET IS NOT OF TYPE - SEditableText"));
00096             return;
00097         }
00098         FString CurrText = EditableText->GetText().ToString();
00099         EditableText->SetText(
00100             FText::FromString(CurrText.TrimStartAndEnd() + TEXT(" ") + PhraseInput->GetValue()));
00101     }
00102     else if (WidgetType == "SMultiLineEditableText")
00103     {
00104         TSharedPtr<SMultiLineEditableText> MultiLineEditableText =
00105             StaticCastSharedPtr<SMultiLineEditableText>(KeyboardFocusedWidget);
00106         if (!MultiLineEditableText.IsValid()) {
00107             UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("KeyboardInputAdd: CURRENT ACTIVE
00108 WIDGET IS NOT OF TYPE - SMultiLineEditableText"));
00109             return;
00110         }
00111         FString CurrText = MultiLineEditableText->GetText().ToString();
00112         MultiLineEditableText->SetText(
00113             FText::FromString(CurrText.TrimStartAndEnd() + TEXT(" ") + PhraseInput->GetValue()));
00114     }
00115     else UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("KeyboardInputAdd: CURRENT ACTIVE
00116 WIDGET IS NOT AN INTERFACEABLE TYPE"));
00117 }

```

### 4.54.3.3 KeyboardInputConfirm()

```

void ULocalizedStringLibrary::KeyboardInputConfirm (
    FParseRecord & Record )

```

Phrase Event for Submitting the Keyboard Input on the Active Keyboard Focus.

## Parameters

|               |  |
|---------------|--|
| <i>Record</i> | The Parse Record accumulated until this Event. |
|---------------|--|

Definition at line 192 of file [LocalizedStringLibrary.cpp](#).

```

00193 {
00194     GET_ACTIVE_KEYBOARD_WIDGET(KeyboardFocusedWidget);
00195
00196     FName WidgetType = KeyboardFocusedWidget->GetType();
00197
00198     if (WidgetType == SEditableText::StaticWidgetClass().GetWidgetType())
00199     {
00200         TSharedPtr<SEditableText> EditableText =
00201             StaticCastSharedPtr<SEditableText>(KeyboardFocusedWidget);

```

```

00201         if (!EditableText.IsValid())
00202         {
00203             UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("KeyboardInputConfirm: CURRENT
ACTIVE WIDGET IS NOT OF TYPE - SEditableText"))
00204             return;
00205         }
00206     }
00207 }
00208 else if (WidgetType == SMultiLineEditableText::StaticWidgetClass().GetWidgetType())
00209 {
00210     TSharedPtr<SMultiLineEditableText> MultiLineEditableText =
StaticCastSharedPtr<SMultiLineEditableText>(KeyboardFocusedWidget);
00211     if (!MultiLineEditableText.IsValid())
00212     {
00213         UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("KeyboardInputConfirm: CURRENT
ACTIVE WIDGET IS NOT OF TYPE - SMultiLineEditableText"))
00214         return;
00215     }
00216 }
00217 }
00218 else UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("KeyboardInputConfirm: CURRENT ACTIVE
WIDGET IS NOT AN INTERFACEABLE TYPE"))
00219 }

```

#### 4.54.3.4 KeyboardInputExit()

```

void ULocalizedInputLibrary::KeyboardInputExit (
    FParseRecord & Record )

```

Phrase Event for Exiting the Active Keyboard Focus, with no changes.

##### Parameters

|               |  |
|---------------|--|
| <i>Record</i> | The Parse Record accumulated until this Event. |
|---------------|--|

Definition at line 221 of file [LocalizedInputLibrary.cpp](#).

```

00222 {
00223     FSlateApplication& SlateApp = FSlateApplication::Get();
00224     if (!SlateApp.IsInitialized())
00225         return;
00226
00227     SlateApp.ClearKeyboardFocus();
00228 }

```

#### 4.54.3.5 KeyboardInputRemove()

```

void ULocalizedInputLibrary::KeyboardInputRemove (
    FParseRecord & Record )

```

Phrase Event for Removing String Chunks from the Active Keyboard Focus.

##### Parameters

|               |  |
|---------------|--|
| <i>Record</i> | The Parse Record accumulated until this Event. |
|---------------|--|

Definition at line 118 of file [LocalizedInputLibrary.cpp](#).

```

00119 {
00120     GET_ACTIVE_KEYBOARD_WIDGET(KeyboardFocusedWidget);

```



```

00121
00122     FString WidgetType = KeyboardFocusedWidget->GetTypeAsString();
00123
00124     UParseIntInput* RemoveInput = Record.GetPhraseInput<UParseIntInput>(TEXT("AMOUNT"));
00125     if (RemoveInput == nullptr)
00126         return;
00127
00128     if (WidgetType == "SEditableText")
00129     {
00130         TSharedPtr<SEditableText> EditableText =
00131             StaticCastSharedPtr<SEditableText>(KeyboardFocusedWidget);
00132         if (!EditableText.IsValid()) {
00133             UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("KeyboardInputRemove: CURRENT ACTIVE
00134 WIDGET IS NOT OF TYPE - SEditableText"));
00135             return;
00136         }
00137         EditableText->SetText (
00138             FText::FromString(
00139                 EventUtils::RemoveWordsFromEnd(EditableText->GetText().ToString(),
00140 RemoveInput->GetValue())
00141             );
00142     }
00143     else if (WidgetType == "SMultiLineEditableText")
00144     {
00145         TSharedPtr<SMultiLineEditableText> MultiLineEditableText =
00146             StaticCastSharedPtr<SMultiLineEditableText>(KeyboardFocusedWidget);
00147         if (!MultiLineEditableText.IsValid()) {
00148             UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("KeyboardInputRemove: CURRENT ACTIVE
00149 WIDGET IS NOT OF TYPE - SMultiLineEditableText"));
00150             return;
00151         }
00152         MultiLineEditableText->SetText (
00153             FText::FromString(
00154                 EventUtils::RemoveWordsFromEnd(MultiLineEditableText->GetText().ToString(),
00155 RemoveInput->GetValue())
00156             );
00157     }
00158     else UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("KeyboardInputRemove: CURRENT ACTIVE
00159 WIDGET IS NOT AN INTERFACEABLE TYPE"));
00160 }

```

#### 4.54.3.6 KeyboardInputReset()

```

void ULocalizedStringLibrary::KeyboardInputReset (
    FParseRecord & Record )

```

Phrase Event for Resetting the Active Keyboard Focus.

##### Parameters

|               |  |
|---------------|--|
| <i>Record</i> | The Parse Record accumulated until this Event. |
|---------------|--|

Definition at line 159 of file LocalizedInputLibrary.cpp.

```

00160 {
00161     GET_ACTIVE_KEYBOARD_WIDGET(KeyboardFocusedWidget);
00162
00163     FString WidgetType = KeyboardFocusedWidget->GetTypeAsString();
00164
00165     if (WidgetType == "SEditableText")
00166     {
00167         TSharedPtr<SEditableText> EditableText =
00168             StaticCastSharedPtr<SEditableText>(KeyboardFocusedWidget);
00169         if (!EditableText.IsValid()) {
00170             UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("KeyboardInputReset: CURRENT ACTIVE
00171 WIDGET IS NOT OF TYPE - SEditableText"));
00172             return;
00173         }
00174         EditableText->SetText (

```

```

00174         FText::FromString(TEXT(""))
00175     );
00176 }
00177 else if (WidgetType == "SMultiLineEditableText")
00178 {
00179     TSharedPtr<SMultiLineEditableText> MultiLineEditableText =
00180     StaticCastSharedPtr<SMultiLineEditableText>(KeyboardFocusedWidget);
00181     if (!MultiLineEditableText.IsValid()) {
00182         UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("KeyboardInputReset: CURRENT ACTIVE
00183 WIDGET IS NOT OF TYPE - SMultiLineEditableText"));
00184         return;
00185     }
00186     MultiLineEditableText->SetText(
00187         FText::FromString(TEXT(""))
00188     );
00189 }
00190 else UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("KeyboardInputReset: CURRENT ACTIVE
00191 WIDGET IS NOT AN INTERFACEABLE TYPE"));
00192 }

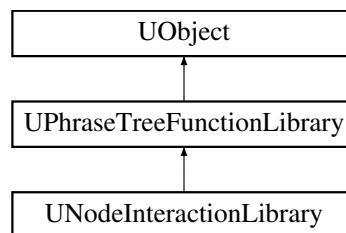
```

The documentation for this class was generated from the following files:

- Source/OpenAccessibility/Public/PhraseEvents/LocalizedInputLibrary.h
- Source/OpenAccessibility/Private/PhraseEvents/LocalizedInputLibrary.cpp

## 4.55 UNodeInteractionLibrary Class Reference

Inheritance diagram for UNodeInteractionLibrary:



### Public Member Functions

- [UNodeInteractionLibrary](#) (const FObjectInitializer &ObjectInitializer)
- virtual void [BindBranches](#) (TSharedRef< [FPhraseTree](#) > PhraseTree) override
- void [MoveNode](#) (FParseRecord &Record)
- void [DeleteNode](#) (FParseRecord &Record)
- void [NodeIndexFocus](#) (int32 Index)
- void [PinConnect](#) (FParseRecord &Record)
- void [PinDisconnect](#) (FParseRecord &Record)
- TSharedPtr< IMenu > [NodeAddMenu](#) (FParseRecord &Record)
- TSharedPtr< IMenu > [NodeAddPinMenu](#) (FParseRecord &Record)
- void [NodeAddSelect](#) (FParseRecord &Record)
- void [NodeAddSearchAdd](#) (FParseRecord &Record)
- void [NodeAddSearchRemove](#) (FParseRecord &Record)
- void [NodeAddSearchReset](#) (FParseRecord &Record)
- void [NodeAddScroll](#) (FParseRecord &Record)
- void [SelectionNodeToggle](#) (FParseRecord &Record)
- void [SelectionReset](#) (FParseRecord &Record)
- void [SelectionMove](#) (FParseRecord &Record)
- void [SelectionAlignment](#) (FParseRecord &Record)

- void [SelectionStraighten](#) (FParseRecord &Record)
- void [SelectionComment](#) (FParseRecord &Record)
- void [LocomotionSelect](#) (FParseRecord &Record)
- void [LocomotionRevert](#) (FParseRecord &Record)
- void [LocomotionConfirm](#) (FParseRecord &Record)
- void [LocomotionCancel](#) (FParseRecord &Record)
- void [BlueprintCompile](#) (FParseRecord &Record)

### 4.55.1 Detailed Description

Definition at line 12 of file [NodeInteractionLibrary.h](#).

### 4.55.2 Constructor & Destructor Documentation

#### 4.55.2.1 UNodeInteractionLibrary()

```
UNodeInteractionLibrary::UNodeInteractionLibrary (
    const FObjectInitializer & ObjectInitializer )
```

Definition at line 21 of file [NodeInteractionLibrary.cpp](#).

```
00022     : Super(ObjectInitializer)
00023 {
00024
00025 }
```

#### 4.55.2.2 ~UNodeInteractionLibrary()

```
UNodeInteractionLibrary::~~UNodeInteractionLibrary ( ) [virtual]
```

Definition at line 27 of file [NodeInteractionLibrary.cpp](#).

```
00028 {
00029
00030 }
```

### 4.55.3 Member Function Documentation

#### 4.55.3.1 BindBranches()

```
void UNodeInteractionLibrary::BindBranches (
    TSharedRef< FPhraseTree > PhraseTree ) [override], [virtual]
```

Binds Branches originating from this Library onto the provided Phrase Tree.

## Parameters

|                   |  |
|-------------------|--|
| <i>PhraseTree</i> | Reference to the PhraseTree to Bind this Library to. |
|-------------------|--|

Reimplemented from [UPhraseTreeFunctionLibrary](#).

Definition at line 32 of file [NodeInteractionLibrary.cpp](#).

```

00033 {
00034     // Events
00035     TDelegate<void(int32)> NodeIndexFocusDelegate = CreateInputDelegate(this,
&UNodeInteractionLibrary::NodeIndexFocus);
00036
00037
00038     // Add Node Children Branch
00039     TPhraseNodeArray AddNodeContextChildren = TPhraseNodeArray {
00040
00041         MakeShared<FPhraseNode>(TEXT("SELECT"),
00042             TPhraseNodeArray {
00043
00044                 MakeShared<FPhraseInputNode<int32>>(TEXT("SELECTION_INDEX"),
00045                     TPhraseNodeArray {
00046
00047                         MakeShared<FPhraseEventNode>(CreateParseDelegate(this,
&UNodeInteractionLibrary::NodeAddSelect))
00048                     })
00049                 },
00050             },
00051         MakeShared<FPhraseNode>(TEXT("SEARCH"),
00052             TPhraseNodeArray{
00053
00054                 MakeShared<FPhraseNode>(TEXT("ADD"),
00055                     TPhraseNodeArray {
00056
00057                         MakeShared<FPhraseStringInputNode>(TEXT("SEARCH_PHRASE"),
00058                             TPhraseNodeArray{
00059
00060                                 MakeShared<FPhraseEventNode>(CreateParseDelegate(this,
&UNodeInteractionLibrary::NodeAddSearchAdd))
00061                             })
00062                         },
00063                     },
00064                 MakeShared<FPhraseNode>(TEXT("REMOVE"),
00065                     TPhraseNodeArray {
00066
00067                         MakeShared<FPhraseInputNode<int32>>(TEXT("AMOUNT"),
00068                             TPhraseNodeArray {
00069
00070                                 MakeShared<FPhraseEventNode>(CreateParseDelegate(this,
&UNodeInteractionLibrary::NodeAddSearchRemove))
00071                             })
00072                         },
00073                     },
00074                 MakeShared<FPhraseNode>(TEXT("RESET"),
00075                     TPhraseNodeArray {
00076
00077                         MakeShared<FPhraseEventNode>(CreateParseDelegate(this,
&UNodeInteractionLibrary::NodeAddSearchReset))
00078                     })
00079                 },
00080             MakeShared<FPhraseNode>(TEXT("SCROLL"),
00081                 TPhraseNodeArray {
00082
00083                     MakeShared<FPhraseScrollInputNode>(TEXT("DIRECTION"),
00084                         TPhraseNodeArray {
00085
00086                             MakeShared<FPhraseInputNode<int32>>(TEXT("AMOUNT"),
00087                                 TPhraseNodeArray {
00088
00089                                     MakeShared<FPhraseEventNode>(CreateParseDelegate(this,
&UNodeInteractionLibrary::NodeAddScroll))
00090                                 })
00091                             },
00092                         },
00093                     },
00094                 },
00095             },
00096         },
00097     },
00098 }
00099
00100
00101

```

```

00102         }},
00103     },
00104     },
00105 };
00106
00107 PhraseTree->BindBranches(
00108     TPhraseNodeArray
00109     {
00110         MakeShared<FPhraseNode>(TEXT("NODE"),
00111             TPhraseNodeArray {
00112                 MakeShared<FPhraseInputNode<int32>>(TEXT("NODE_INDEX"),
00113                     TPhraseNodeArray {
00114                         MakeShared<FPhraseNode>(TEXT("MOVE"),
00115                             TPhraseNodeArray {
00116                                 MakeShared<FPhrase2DDirectionalInputNode>(TEXT("DIRECTION"),
00117                                     TPhraseNodeArray {
00118                                         MakeShared<FPhraseInputNode<int32>>(TEXT("AMOUNT"),
00119                                             TPhraseNodeArray {
00120                                                 MakeShared<FPhraseEventNode>(CreateParseDelegate(this,
00121 &UNodeInteractionLibrary::MoveNode))
00122                                             }
00123                                         }
00124                                     }
00125                                 }
00126                             }
00127                         }
00128                     }
00129                 }
00130             }
00131         },
00132         MakeShared<FPhraseNode>(TEXT("REMOVE"),
00133             TPhraseNodeArray {
00134                 MakeShared<FPhraseEventNode>(CreateParseDelegate(this,
00135 &UNodeInteractionLibrary::DeleteNode))
00136             }
00137         },
00138         MakeShared<FPhraseInputNode<int32>>(TEXT("PIN_INDEX"),
00139             TPhraseNodeArray {
00140                 MakeShared<FPhraseNode>(TEXT("CONNECT"),
00141                     TPhraseNodeArray {
00142                         MakeShared<FPhraseContextMenuNode<UAccessibilityGraphEditorContext>>(
00143                             TEXT("ADD"),
00144                             1.5f,
00145                             CreateMenuDelegate(this, &UNodeInteractionLibrary::NodeAddPinMenu),
00146                             AddNodeContextChildren
00147                         ),
00148                         MakeShared<FPhraseInputNode<int32>>(TEXT("NODE_INDEX"),
00149                             TPhraseNodeArray {
00150                                 MakeShared<FPhraseInputNode<int32>>(TEXT("PIN_INDEX"),
00151                                     TPhraseNodeArray {
00152                                         MakeShared<FPhraseEventNode>(CreateParseDelegate(this,
00153 &UNodeInteractionLibrary::PinConnect))
00154                                     }
00155                                 }
00156                             }, NodeIndexFocusDelegate)
00157                         }
00158                     },
00159                     MakeShared<FPhraseNode>(TEXT("DISCONNECT"),
00160                         TPhraseNodeArray {
00161                             MakeShared<FPhraseInputNode<int32>>(TEXT("NODE_INDEX"),
00162                                 TPhraseNodeArray {
00163                                     MakeShared<FPhraseInputNode<int32>>(TEXT("PIN_INDEX"),
00164                                         TPhraseNodeArray {
00165                                             MakeShared<FPhraseEventNode>(CreateParseDelegate(this,
00166 &UNodeInteractionLibrary::PinDisconnect))
00167                                         }
00168                                     }
00169                                 }
00170                             }
00171                         }
00172                     }
00173                 }
00174             }
00175         }
00176     }
00177 }
00178
00179 }
00180
00181 }
00182
00183 }
00184

```

```

00185         })
00186     },
00187     }, NodeIndexFocusDelegate),
00188     MakeShared<FPhraseNode>(TEXT("SELECT"),
00189     TPhraseNodeArray {
00190         MakeShared<FPhraseInputNode<int32>>(TEXT("NODE_INDEX"),
00191         TPhraseNodeArray {
00192             MakeShared<FPhraseEventNode>(CreateParseDelegate(this,
00193             &UNodeInteractionLibrary::SelectionNodeToggle))
00194         },
00195         },
00196         MakeShared<FPhraseNode>(TEXT("RESET"),
00197         TPhraseNodeArray {
00198             MakeShared<FPhraseEventNode>(CreateParseDelegate(this,
00199             &UNodeInteractionLibrary::SelectionReset))
00200         },
00201         },
00202         MakeShared<FPhraseNode>(TEXT("MOVE"),
00203         TPhraseNodeArray {
00204             MakeShared<FPhrase2DDirectionalInputNode>(TEXT("DIRECTION"),
00205             TPhraseNodeArray {
00206                 MakeShared<FPhraseInputNode<int32>>(TEXT("AMOUNT"),
00207                 TPhraseNodeArray {
00208                     MakeShared<FPhraseEventNode>(CreateParseDelegate(this,
00209                     &UNodeInteractionLibrary::SelectionMove))
00210                 },
00211                 },
00212                 },
00213                 },
00214                 },
00215                 },
00216                 },
00217                 },
00218                 },
00219                 },
00220                 },
00221                 },
00222                 },
00223                 },
00224                 },
00225                 },
00226                 },
00227                 },
00228                 },
00229                 },
00230                 },
00231                 },
00232                 },
00233                 },
00234                 },
00235                 },
00236                 },
00237                 },
00238                 },
00239                 },
00240                 },
00241                 },
00242                 },
00243                 },
00244                 },
00245                 },
00246                 },
00247                 },
00248                 },
00249                 },
00250                 },
00251                 },
00252                 },
00253                 },
00254                 },
00255                 },
00256                 },
00257                 },
00258                 },
00259                 },
00260                 },
00261                 },
00262                 },
00263                 },
00264                 },
00265                 },

```

```

00266         MakeShared<FPhraseEventNode> (CreateParseDelegate (this,
&UNodeInteractionLibrary::BlueprintCompile))
00267     },
00268     },
00269     MakeShared<FPhraseContextNode<UAccessibilityGraphLocomotionContext>> (TEXT ("MOVE"),
00270     TPhraseNodeArray {
00271         MakeShared<FPhraseNode> (TEXT ("SELECT"),
00272         TPhraseNodeArray {
00273             MakeShared<FPhraseInputNode<int32>> (TEXT ("INDEX"),
00274             TPhraseNodeArray {
00275                 MakeShared<FPhraseEventNode> (CreateParseDelegate (this,
00276                 &UNodeInteractionLibrary::LocomotionSelect))
00277             })
00278         })
00279     },
00280     },
00281     },
00282     },
00283     MakeShared<FPhraseNode> (TEXT ("REVERT"),
00284     TPhraseNodeArray {
00285         MakeShared<FPhraseEventNode> (CreateParseDelegate (this,
00286         &UNodeInteractionLibrary::LocomotionRevert))
00287     })
00288     },
00289     },
00290     },
00291     MakeShared<FPhraseNode> (TEXT ("CONFIRM"),
00292     TPhraseNodeArray {
00293         MakeShared<FPhraseEventNode> (CreateParseDelegate (this,
00294         &UNodeInteractionLibrary::LocomotionConfirm))
00295     })
00296     },
00297     },
00298     },
00299     MakeShared<FPhraseNode> (TEXT ("CANCEL"),
00300     TPhraseNodeArray {
00301         MakeShared<FPhraseEventNode> (CreateParseDelegate (this,
00302         &UNodeInteractionLibrary::LocomotionCancel))
00303     })
00304     },
00305     },
00306     },
00307     })
00308     }
00309     );
00310     };
00311     };

```

### 4.55.3.2 BlueprintCompile()

```

void UNodeInteractionLibrary::BlueprintCompile (
    FParseRecord & Record )

```

Phrase Event for Compiling Blueprint Linked to the Active Blueprint Editor.

#### Parameters

|               |  |
|---------------|--|
| <i>Record</i> | The Parse Record accumulated until this Event. |
|---------------|--|

Definition at line 880 of file [NodeInteractionLibrary.cpp](#).

```

00881 {
00882     GET_CAST_ACTIVE_TAB_CONTENT (ActiveGraphEditor, SGraphEditor)
00883
00884     UEdGraph* ActiveGraph = ActiveGraphEditor->GetCurrentGraph();
00885     if (ActiveGraph == nullptr)
00886     {
00887         UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("BlueprintCompile: Active Graph Not
Found"));
00888         return;

```

```

00889     }
00890
00891     UBlueprint* FoundBlueprint = FBlueprintEditorUtils::FindBlueprintForGraph(ActiveGraph);
00892     if (FoundBlueprint == nullptr)
00893     {
00894         UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("BlueprintCompile: Blueprint Not
Found"));
00895         return;
00896     }
00897
00898     TSharedPtr<FBlueprintEditor> BlueprintEditor =
StaticCastSharedPtr<FBlueprintEditor>(FKismetEditorUtilities::GetIBlueprintEditorForObject(FoundBlueprint,
false));
00899     if (!BlueprintEditor.IsValid())
00900     {
00901         UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("BlueprintCompile: BlueprintEditor Not
Found"));
00902         return;
00903     }
00904
00905     BlueprintEditor->Compile();
00906 }

```

#### 4.55.3.3 DeleteNode()

```

void UNodeInteractionLibrary::DeleteNode (
    FParseRecord & Record )

```

Phrase Event for Deleting a Node, on the Active Graph Editor.

##### Parameters

|               |  |
|---------------|--|
| <i>Record</i> | The Parse Record accumulated until this Event. |
|---------------|--|

Definition at line 395 of file [NodeInteractionLibrary.cpp](#).

```

00396 {
00397     GET_CAST_ACTIVE_TAB_CONTENT(ActiveGraphEditor, SGraphEditor)
00398
00399     UParseIntInput* IndexInput = Record.GetPhraseInput<UParseIntInput>(TEXT("NODE_INDEX"));
00400     if (IndexInput == nullptr)
00401         return;
00402
00403     TSharedRef<FAssetAccessibilityRegistry> AssetRegistry = GetAssetRegistry();
00404     TSharedRef<FGraphIndexer> Indexer =
AssetRegistry->GetGraphIndexer(ActiveGraphEditor->GetCurrentGraph());
00405
00406     UEdGraphNode* Node = Indexer->GetNode(IndexInput->GetValue());
00407     if (Node == nullptr)
00408     {
00409         UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("DeleteNode: Node Not Found"));
00410         return;
00411     }
00412
00413     Node->Modify();
00414     Node->DestroyNode();
00415 }

```

#### 4.55.3.4 LocomotionCancel()

```

void UNodeInteractionLibrary::LocomotionCancel (
    FParseRecord & Record )

```

Phrase Event for Canceling the Active Graph Editors Locomotion Mode, reverting to viewport state before.



## Parameters

|               |  |
|---------------|--|
| <i>Record</i> | The Parse Record accumulated until this Event. |
|---------------|--|

Definition at line 873 of file [NodeInteractionLibrary.cpp](#).

```
00874 {  
00875     GET_TOP_CONTEXT(Record, LocomotionContext, UAccessibilityGraphLocomotionContext);  
00876  
00877     LocomotionContext->CancelLocomotion();  
00878 }
```

#### 4.55.3.5 LocomotionConfirm()

```
void UNodeInteractionLibrary::LocomotionConfirm (  
    FParseRecord & Record )
```

Phrase Event for Confirming the Current Viewport, on the Active Graph Editors Locomotion Mode.

## Parameters

|               |  |
|---------------|--|
| <i>Record</i> | The Parse Record accumulated until this Event. |
|---------------|--|

Definition at line 866 of file [NodeInteractionLibrary.cpp](#).

```
00867 {  
00868     GET_TOP_CONTEXT(Record, LocomotionContext, UAccessibilityGraphLocomotionContext);  
00869  
00870     LocomotionContext->ConfirmSelection();  
00871 }
```

#### 4.55.3.6 LocomotionRevert()

```
void UNodeInteractionLibrary::LocomotionRevert (  
    FParseRecord & Record )
```

Phrase Event for Reverting the Viewport to the Previous Rect, on the Active Graph Editors Locomotion Mode.

## Parameters

|               |  |
|---------------|--|
| <i>Record</i> | The Parse Record accumulated until this Event. |
|---------------|--|

Definition at line 856 of file [NodeInteractionLibrary.cpp](#).

```
00857 {  
00858     GET_TOP_CONTEXT(Record, LocomotionContext, UAccessibilityGraphLocomotionContext);  
00859  
00860     if (!LocomotionContext->RevertToPreviousView())  
00861     {  
00862         UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("Locomotion Revert: Failed to Revert to  
Previous View."));  
00863     }  
00864 }
```

#### 4.55.3.7 LocomotionSelect()

```
void UNodeInteractionLibrary::LocomotionSelect (
    FParseRecord & Record )
```

Phrase Event for Selecting a Viewport Rect for Movement, on the Active Graph Editors Locomotion Mode.

##### Parameters

|               |  |
|---------------|--|
| <i>Record</i> | The Parse Record accumulated until this Event. |
|---------------|--|

Definition at line 842 of file [NodeInteractionLibrary.cpp](#).

```
00843 {
00844     GET_TOP_CONTEXT(Record, LocomotionContext, UAccessibilityGraphLocomotionContext);
00845
00846     UParseIntInput* ViewSelectionInput = Record.GetPhraseInput<UParseIntInput>(TEXT("INDEX"));
00847     if (ViewSelectionInput == nullptr)
00848         return;
00849
00850     if (!LocomotionContext->SelectChunk(ViewSelectionInput->GetValue()))
00851     {
00852         UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("Locomotion Select: Failed to Choose New
View."));
00853     }
00854 }
```

#### 4.55.3.8 MoveNode()

```
void UNodeInteractionLibrary::MoveNode (
    FParseRecord & Record )
```

Phrase Event for Moving a Node, on the Active Graph Editor.

##### Parameters

|               |  |
|---------------|--|
| <i>Record</i> | The Parse Record accumulated until this Event. |
|---------------|--|

Definition at line 314 of file [NodeInteractionLibrary.cpp](#).

```
00314 {
00315     GET_CAST_ACTIVE_TAB_CONTENT(ActiveGraphEditor, SGraphEditor)
00316
00317     UParseIntInput* IndexInput = Record.GetPhraseInput<UParseIntInput>(TEXT("NODE_INDEX"));
00318     UParseEnumInput* DirectionInput = Record.GetPhraseInput<UParseEnumInput>(TEXT("DIRECTION"));
00319     UParseIntInput* AmountInput = Record.GetPhraseInput<UParseIntInput>(TEXT("AMOUNT"));
00320     if (IndexInput == nullptr || DirectionInput == nullptr || AmountInput == nullptr)
00321         return;
00322
00323     TSharedRef<FAssetAccessibilityRegistry> AssetRegistry = GetAssetRegistry();
00324     TSharedRef<FGraphIndexer> Indexer =
AssetRegistry->GetGraphIndexer(ActiveGraphEditor->GetCurrentGraph());
00325
00326     UEdGraphNode* Node = Indexer->GetNode(IndexInput->GetValue());
00327     if (Node == nullptr)
00328     {
00329         UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("MoveNode: Node Not Found"));
00330         return;
00331     }
00332
00333     FVector2D PositionDelta = FVector2D::ZeroVector;
00334     switch (EPhrase2DDirectionalInput(DirectionInput->GetValue()))
00335     {
00336     case EPhrase2DDirectionalInput::UP:
00337         PositionDelta.Y -= AmountInput->GetValue();
00338         break;
```

```

00339
00340         case EPhrase2DDirectionalInput::DOWN:
00341             PositionDelta.Y += AmountInput->GetValue();
00342             break;
00343
00344         case EPhrase2DDirectionalInput::LEFT:
00345             PositionDelta.X -= AmountInput->GetValue();
00346             break;
00347
00348         case EPhrase2DDirectionalInput::RIGHT:
00349             PositionDelta.X += AmountInput->GetValue();
00350             break;
00351
00352         default:
00353             UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("MoveNode: Invalid Direction"));
00354             return;
00355     }
00356
00357     SGraphPanel* GraphPanel = ActiveGraphEditor->GetGraphPanel();
00358     if (GraphPanel == nullptr)
00359     {
00360         UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("MoveNode: Linked Graph Panel Not
Found"));
00361     }
00362
00363     TSharedPtr<SGraphNode> NodeWidget = GraphPanel ? GraphPanel->GetNodeWidgetFromGuid(Node->NodeGuid)
: TSharedPtr<SGraphNode>();
00364     if (NodeWidget.IsValid())
00365     {
00366         SNodePanel::SNode::FNodeSet NodeFilter;
00367         NodeWidget->MoveTo(FVector2D(Node->NodePosX, Node->NodePosY) + PositionDelta, NodeFilter);
00368     }
00369     else
00370     {
00371         Node->Modify();
00372         Node->NodePosX += PositionDelta.X;
00373         Node->NodePosY += PositionDelta.Y;
00374     }
00375
00376     // Move Comment Node Children
00377     // Note: This is a workaround for the MoveTo Function not calling the override in
UEdGraphNode_Comment
00378     if (UEdGraphNode_Comment* CommentNode = Cast<UEdGraphNode_Comment>(Node))
00379     {
00380         for (UObject* _CommentChildNode : CommentNode->GetNodesUnderComment())
00381         {
00382             if (UEdGraphNode* CommentChildNode = Cast<UEdGraphNode>(_CommentChildNode))
00383             {
00384                 if (!GraphPanel->SelectionManager.IsNodeSelected(CommentChildNode))
00385                 {
00386                     CommentChildNode->Modify();
00387                     CommentChildNode->NodePosX += PositionDelta.X;
00388                     CommentChildNode->NodePosY += PositionDelta.Y;
00389                 }
00390             }
00391         }
00392     }
00393 }

```

#### 4.55.3.9 NodeAddMenu()

```

TSharedPtr< IMenu > UNodeInteractionLibrary::NodeAddMenu (
    FParseRecord & Record )

```

Menu Event for Initializing the Node Add Context Menu, on the Active Graph Editor.

##### Parameters

|               |  |
|---------------|--|
| <i>Record</i> | The Parse Record accumulated until this Event. |
|---------------|--|

**Returns**

A Shared Pointer to the Initialized Menu, otherwise an Invalid Shared Pointer.

Definition at line 510 of file [NodeInteractionLibrary.cpp](#).

```

00511 {
00512     GET_CAST_ACTIVE_TAB_CONTENT_RETURN(ActiveGraphEditor, SGraphEditor, TSharedPtr<IMenu>())
00513
00514     SGraphPanel* GraphPanel = ActiveGraphEditor->GetGraphPanel();
00515
00516     FVector2D SpawnLocation;
00517     {
00518         TSharedPtr<SWindow> TopLevelWindow =
00519         FSlateApplication::Get().GetActiveTopLevelRegularWindow();
00520
00521         if (TopLevelWindow.IsValid())
00522         {
00523             SpawnLocation = TopLevelWindow->GetPositionInScreen();
00524             FVector2D WindowSize = TopLevelWindow->GetSizeInScreen();
00525
00526             SpawnLocation.X += WindowSize.X / 5;
00527             SpawnLocation.Y += WindowSize.Y / 5;
00528         }
00529         else
00530         {
00531             FDisplayMetrics DisplayMetrics;
00532             FSlateApplication::Get().GetDisplayMetrics(DisplayMetrics);
00533
00534             SpawnLocation = FVector2D(
00535                 DisplayMetrics.PrimaryDisplayWidth / 5,
00536                 DisplayMetrics.PrimaryDisplayHeight / 5
00537             );
00538         }
00539
00540         TSharedPtr<SWidget> ContextWidgetToFocus = GraphPanel->SummonContextMenu(
00541             SpawnLocation,
00542             GraphPanel->GetPastePosition(),
00543             nullptr,
00544             nullptr,
00545             TArray<UEdGraphPin*>()
00546         );
00547
00548         if (!ContextWidgetToFocus.IsValid())
00549         {
00550             UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("NodeAddMenu: Context Keyboard Focus
00551             Widget Not Found"));
00552             return TSharedPtr<IMenu>();
00553         }
00554
00555         FWidgetPath KeyboardFocusPath;
00556         if (FSlateApplication::Get().FindPathToWidget(ContextWidgetToFocus.ToSharedRef(),
00557             KeyboardFocusPath))
00558         {
00559             return FSlateApplication::Get().FindMenuInWidgetPath(KeyboardFocusPath);
00560         }
00561         else
00562         {
00563             UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("NodeAddMenu: IMenu Could Not Be
00564             Found In Widget Path"));
00565             return TSharedPtr<IMenu>();
00566         }
00567     }
00568 }

```

**4.55.3.10 NodeAddPinMenu()**

```

TSharedPtr< IMenu > UNodeInteractionLibrary::NodeAddPinMenu (
    FParseRecord & Record )

```

Menu Event for Initializing the Node Add Context Menu from a Pin Connection, on the Active Graph Editor.

**Parameters**

|               |  |
|---------------|--|
| <b>Record</b> | The Parse Record accumulated until this Event. |
|---------------|--|

## Returns

A Shared Pointer to the Initialized Menu, otherwise an Invalid Shared Pointer.

Definition at line 566 of file [NodeInteractionLibrary.cpp](#).

```

00567 {
00568     GET_CAST_ACTIVE_TAB_CONTENT_RETURN(ActiveGraphEditor, SGraphEditor, TSharedPtr<IMenu>())
00569
00570     SGraphPanel* GraphPanel = ActiveGraphEditor->GetGraphPanel();
00571
00572     FVector2D SpawnLocation;
00573     {
00574         TSharedPtr<SWindow> TopLevelWindow =
00575         FSlateApplication::Get().GetActiveTopLevelRegularWindow();
00576
00577         if (TopLevelWindow.IsValid())
00578         {
00579             SpawnLocation = TopLevelWindow->GetPositionInScreen();
00580             FVector2D WindowSize = TopLevelWindow->GetSizeInScreen();
00581
00582             SpawnLocation.X += WindowSize.X / 5;
00583             SpawnLocation.Y += WindowSize.Y / 5;
00584         }
00585         else
00586         {
00587             FDisplayMetrics DisplayMetrics;
00588             FSlateApplication::Get().GetDisplayMetrics(DisplayMetrics);
00589
00590             SpawnLocation = FVector2D(
00591                 DisplayMetrics.PrimaryDisplayWidth / 5,
00592                 DisplayMetrics.PrimaryDisplayHeight / 5
00593             );
00594         }
00595
00596         TSharedRef<FGraphIndexer> Indexer =
00597         GetAssetRegistry()->GetGraphIndexer(ActiveGraphEditor->GetCurrentGraph());
00598
00599         UParseIntInput* NodeIndexInput = Record.GetPhraseInput<UParseIntInput>(TEXT("NODE_INDEX"));
00600         UParseIntInput* PinIndexInput = Record.GetPhraseInput<UParseIntInput>(TEXT("PIN_INDEX"));
00601
00602         if (NodeIndexInput == nullptr || PinIndexInput == nullptr)
00603         {
00604             UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("NodeAddMenu: Invalid Inputs"));
00605             return TSharedPtr<IMenu>();
00606         }
00607
00608         TSharedPtr<SWidget> ContextWidgetToFocus = GraphPanel->SummonContextMenu(
00609             SpawnLocation,
00610             GraphPanel->GetPastePosition(),
00611             nullptr,
00612             nullptr,
00613             TArray<UEdGraphPin*> {
00614                 Indexer->GetPin(
00615                     NodeIndexInput->GetValue(),
00616                     PinIndexInput->GetValue()
00617                 )
00618             }
00619         );
00620
00621         if (!ContextWidgetToFocus.IsValid())
00622         {
00623             UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("NodeAddMenu: Context Keyboard Focus
00624             Widget Not Found"));
00625             return TSharedPtr<IMenu>();
00626         }
00627
00628         FWidgetPath KeyboardFocusPath;
00629         if (FSlateApplication::Get().FindPathToWidget(ContextWidgetToFocus.ToSharedRef(),
00630             KeyboardFocusPath))
00631         {
00632             return FSlateApplication::Get().FindMenuInWidgetPath(KeyboardFocusPath);
00633         }
00634         else
00635         {
00636             UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("NodeAddMenu: IMenu Could Not Be
00637             Found In Widget Path"));
00638             return TSharedPtr<IMenu>();
00639         }
00640     }
00641 }

```

#### 4.55.3.11 NodeAddScroll()

```
void UNodeInteractionLibrary::NodeAddScroll (
    FParseRecord & Record )
```

Phrase Event for Applying Movement to the Scrollbar of the Active Graph Editors Node Add Context Menu.

##### Parameters

|               |  |
|---------------|--|
| <i>Record</i> | The Parse Record accumulated until this Event. |
|---------------|--|

Definition at line 681 of file [NodeInteractionLibrary.cpp](#).

```
00682 {
00683     GET_TOP_CONTEXT(Record, ContextMenu, UAccessibilityGraphEditorContext)
00684
00685     UParseEnumInput* DirectionInput = Record.GetPhraseInput<UParseEnumInput>(TEXT("DIRECTION"));
00686     UParseIntInput* AmountInput = Record.GetPhraseInput<UParseIntInput>(TEXT("AMOUNT"));
00687     if (DirectionInput == nullptr || AmountInput == nullptr)
00688         return;
00689
00690     switch (EPhraseScrollInput(DirectionInput->GetValue()))
00691     {
00692     case EPhraseScrollInput::UP:
00693         ContextMenu->AppendScrollDistance(-AmountInput->GetValue());
00694         break;
00695
00696     case EPhraseScrollInput::DOWN:
00697         ContextMenu->AppendScrollDistance(AmountInput->GetValue());
00698         break;
00699
00700     case EPhraseScrollInput::TOP:
00701         ContextMenu->SetScrollDistanceTop();
00702         break;
00703
00704     case EPhraseScrollInput::BOTTOM:
00705         ContextMenu->SetScrollDistanceBottom();
00706         break;
00707
00708     default:
00709         UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("NodeAddScroll: Invalid Scroll
00710             Position / Direction"));
00711         return;
00712     }
```

#### 4.55.3.12 NodeAddSearchAdd()

```
void UNodeInteractionLibrary::NodeAddSearchAdd (
    FParseRecord & Record )
```

Phrase Event for Appending Strings to the SearchBar on the Active Graph Editors Node Add Context Menu.

##### Parameters

|               |  |
|---------------|--|
| <i>Record</i> | The Parse Record accumulated until this Event. |
|---------------|--|

Definition at line 650 of file [NodeInteractionLibrary.cpp](#).

```
00651 {
00652     GET_TOP_CONTEXT(Record, ContextMenu, UAccessibilityGraphEditorContext)
00653
00654     UParseStringInput *SearchInput = Record.GetPhraseInput<UParseStringInput>(TEXT("SEARCH_PHRASE"));
00655     if (SearchInput == nullptr)
00656         return;
00657 }
```

```
00658     ContextMenu->AppendFilterText (SearchInput->GetValue());
00659 }
```

#### 4.55.3.13 NodeAddSearchRemove()

```
void UNodeInteractionLibrary::NodeAddSearchRemove (
    FParseRecord & Record )
```

Phrase Event for Removing String Chunks on the SearchBar of the Active Graph Editors Node Add Context Menu.

##### Parameters

|               |  |
|---------------|--|
| <i>Record</i> | The Parse Record accumulated until this Event. |
|---------------|--|

Definition at line 661 of file [NodeInteractionLibrary.cpp](#).

```
00662 {
00663     GET_TOP_CONTEXT(Record, ContextMenu, UAccessibilityGraphEditorContext);
00664
00665     UParseIntInput* RemoveAmountInput = Record.GetPhraseInput<UParseIntInput>(TEXT("AMOUNT"));
00666     if (RemoveAmountInput == nullptr)
00667         return;
00668
00669     ContextMenu->SetFilterText (
00670         EventUtils::RemoveWordsFromEnd(ContextMenu->GetFilterText(), RemoveAmountInput->GetValue())
00671     );
00672 }
```

#### 4.55.3.14 NodeAddSearchReset()

```
void UNodeInteractionLibrary::NodeAddSearchReset (
    FParseRecord & Record )
```

Phrase Event for Resetting the SearchBar of the Active Graph Editors Node Add Context Menu.

##### Parameters

|               |  |
|---------------|--|
| <i>Record</i> | The Parse Record accumulated until this Event. |
|---------------|--|

Definition at line 674 of file [NodeInteractionLibrary.cpp](#).

```
00675 {
00676     GET_TOP_CONTEXT(Record, ContextMenu, UAccessibilityGraphEditorContext)
00677
00678     ContextMenu->SetFilterText(TEXT(""));
00679 }
```

#### 4.55.3.15 NodeAddSelect()

```
void UNodeInteractionLibrary::NodeAddSelect (
    FParseRecord & Record )
```

Phrase Event for Selecting an Item on the Active Graph Editors Node Add Context Menu.

## Parameters

|               |  |
|---------------|--|
| <i>Record</i> | The Parse Record accumulated until this Event. |
|---------------|--|

Definition at line 639 of file [NodeInteractionLibrary.cpp](#).

```
00640 {
00641     GET_TOP_CONTEXT(Record, ContextMenu, UAccessibilityGraphEditorContext)
00642
00643     UParseIntInput* IndexInput = Record.GetPhraseInput<UParseIntInput>(TEXT("SELECTION_INDEX"));
00644     if (IndexInput == nullptr)
00645         return;
00646
00647     ContextMenu->SelectAction(IndexInput->GetValue());
00648 }
```

#### 4.55.3.16 NodeIndexFocus()

```
void UNodeInteractionLibrary::NodeIndexFocus (
    int32 Index )
```

Input Event for Adding the specified Node Index to the Active Selection Set.

## Parameters

|              |   |
|--------------|---|
| <i>Index</i> | The Index Provided Through Voice Input. |
|--------------|---|

Definition at line 417 of file [NodeInteractionLibrary.cpp](#).

```
00418 {
00419     GET_CAST_ACTIVE_TAB_CONTENT(ActiveGraphEditor, SGraphEditor)
00420
00421     TSharedRef<FGraphIndexer> Indexer = GetAssetRegistry()->GetGraphIndexer(
00422         ActiveGraphEditor->GetCurrentGraph()
00423     );
00424
00425     UEdGraphNode* Node = Indexer->GetNode(Index);
00426     if (Node == nullptr)
00427     {
00428         UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("NodeSelectionFocus: Node Not Found"));
00429         return;
00430     }
00431
00432     ActiveGraphEditor->SetNodeSelection(Node, true);
00433 }
```

#### 4.55.3.17 PinConnect()

```
void UNodeInteractionLibrary::PinConnect (
    FParseRecord & Record )
```

Phrase Event for Connecting Two Provided Pins, on the Active Graph Editor.

## Parameters

|               |  |
|---------------|--|
| <i>Record</i> | The Parse Record accumulated until this Event. |
|---------------|--|



Definition at line 435 of file [NodeInteractionLibrary.cpp](#).

```

00436 {
00437     GET_CAST_ACTIVE_TAB_CONTENT(ActiveGraphEditor, SGraphEditor)
00438
00439     UEdGraph* Graph = ActiveGraphEditor->GetCurrentGraph();
00440
00441     TArray<UParseInput*> NodeInputs = Record.GetPhraseInputs(TEXT("NODE_INDEX"));
00442     TArray<UParseInput*> PinInputs = Record.GetPhraseInputs(TEXT("PIN_INDEX"));
00443
00444     if (NodeInputs.Num() != 2 || PinInputs.Num() != 2)
00445     {
00446         UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("PinConnect: Invalid Inputs Amount"));
00447         return;
00448     }
00449
00450     TSharedRef<FGraphIndexer> Indexer = GetAssetRegistry()->GetGraphIndexer(Graph);
00451
00452     UEdGraphPin* SourcePin = Indexer->GetPin(
00453         Cast<UParseIntInput>(NodeInputs[0])->GetValue(),
00454         Cast<UParseIntInput>(PinInputs[0])->GetValue()
00455     );
00456
00457     UEdGraphPin* TargetPin = Indexer->GetPin(
00458         Cast<UParseIntInput>(NodeInputs[1])->GetValue(),
00459         Cast<UParseIntInput>(PinInputs[1])->GetValue()
00460     );
00461
00462     if (SourcePin == nullptr || TargetPin == nullptr)
00463     {
00464         UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("PinConnect: Pins Not Found"));
00465         return;
00466     }
00467
00468     if (!Graph->GetSchema()->TryCreateConnection(SourcePin, TargetPin))
00469     {
00470         UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("PinConnect: Pin Connection Failed"));
00471     }
00472 }

```

#### 4.55.3.18 PinDisconnect()

```

void UNodeInteractionLibrary::PinDisconnect (
    FParseRecord & Record )

```

Phrase Event for Disconnecting Two Provided Pins, on the Active Graph Editor.

##### Parameters

|               |  |
|---------------|--|
| <i>Record</i> | The Parse Record accumulated until this Event. |
|---------------|--|

Definition at line 474 of file [NodeInteractionLibrary.cpp](#).

```

00475 {
00476     GET_CAST_ACTIVE_TAB_CONTENT(ActiveGraphEditor, SGraphEditor)
00477
00478     UEdGraph* Graph = ActiveGraphEditor->GetCurrentGraph();
00479
00480     TArray<UParseInput*> NodeInputs = Record.GetPhraseInputs(TEXT("NODE_INDEX"));
00481     TArray<UParseInput*> PinInputs = Record.GetPhraseInputs(TEXT("PIN_INDEX"));
00482
00483     if (NodeInputs.Num() != 2 || PinInputs.Num() != 2)
00484     {
00485         UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("PinDisconnect: Invalid Inputs
Amount"));
00486         return;
00487     }
00488
00489     TSharedRef<FGraphIndexer> Indexer = GetAssetRegistry()->GetGraphIndexer(Graph);
00490
00491     UEdGraphPin* SourcePin = Indexer->GetPin(
00492         Cast<UParseIntInput>(NodeInputs[0])->GetValue(),
00493         Cast<UParseIntInput>(PinInputs[0])->GetValue()
00494     );
00495

```

```

00496     UEdGraphPin* TargetPin = Indexer->GetPin(
00497         Cast<UParseIntInput>(NodeInputs[1])->GetValue(),
00498         Cast<UParseIntInput>(PinInputs[1])->GetValue()
00499     );
00500
00501     if (SourcePin == nullptr || TargetPin == nullptr)
00502     {
00503         UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("PinDisconnect: Pins Not Found"));
00504         return;
00505     }
00506
00507     Graph->GetSchema()->BreakSinglePinLink(SourcePin, TargetPin);
00508 }

```

#### 4.55.3.19 SelectionAlignment()

```

void UNodeInteractionLibrary::SelectionAlignment (
    FParseRecord & Record )

```

Phrase Event for Aligning the Selection Sets Nodes, on the Active Graph Editor.

##### Parameters

|               |  |
|---------------|--|
| <i>Record</i> | The Parse Record accumulated until this Event. |
|---------------|--|

Definition at line 785 of file [NodeInteractionLibrary.cpp](#).

```

00786 {
00787     GET_CAST_ACTIVE_TAB_CONTENT(ActiveGraphEditor, SGraphEditor)
00788
00789     UParseEnumInput* PositionInput = Record.GetPhraseInput<UParseEnumInput>(TEXT("POSITION"));
00790     if (PositionInput == nullptr)
00791         return;
00792
00793     switch (EPhrasePositionalInput(PositionInput->GetValue()))
00794     {
00795     case EPhrasePositionalInput::TOP:
00796         ActiveGraphEditor->OnAlignTop();
00797         break;
00798
00799     case EPhrasePositionalInput::MIDDLE:
00800         ActiveGraphEditor->OnAlignMiddle();
00801         break;
00802
00803     case EPhrasePositionalInput::BOTTOM:
00804         ActiveGraphEditor->OnAlignBottom();
00805         break;
00806
00807     case EPhrasePositionalInput::LEFT:
00808         ActiveGraphEditor->OnAlignLeft();
00809         break;
00810
00811     case EPhrasePositionalInput::RIGHT:
00812         ActiveGraphEditor->OnAlignRight();
00813         break;
00814
00815     case EPhrasePositionalInput::CENTER:
00816         ActiveGraphEditor->OnAlignCenter();
00817         break;
00818     }
00819 }

```

#### 4.55.3.20 SelectionComment()

```

void UNodeInteractionLibrary::SelectionComment (
    FParseRecord & Record )

```

Phrase Event for Applying a Comment Node Around the Selection Set, on the Active Graph Editor.

## Parameters

|               |  |
|---------------|--|
| <i>Record</i> | The Parse Record accumulated until this Event. |
|---------------|--|

Definition at line 828 of file [NodeInteractionLibrary.cpp](#).

```
00829 {
00830     GET_CAST_ACTIVE_TAB_CONTENT(ActiveGraphEditor, SGraphEditor)
00831
00832     UEdGraph* Graph = ActiveGraphEditor->GetCurrentGraph();
00833
00834     TSharedPtr<FEdGraphSchemaAction> CommentCreateAction =
00835     Graph->GetSchema()->GetCreateCommentAction();
00836     if (CommentCreateAction.IsValid())
00837     {
00838         CommentCreateAction->PerformAction(Graph, nullptr, FVector2D(0, 0), true);
00839     }
00840     else UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("SelectionComment: Comment Creation
Failed"));
```

## 4.55.3.21 SelectionMove()

```
void UNodeInteractionLibrary::SelectionMove (
    FParseRecord & Record )
```

Phrase Event for Moving the Selection Set, on the Active Graph Editor.

## Parameters

|               |  |
|---------------|--|
| <i>Record</i> | The Parse Record accumulated until this Event. |
|---------------|--|

Definition at line 745 of file [NodeInteractionLibrary.cpp](#).

```
00746 {
00747     GET_CAST_ACTIVE_TAB_CONTENT(ActiveGraphEditor, SGraphEditor)
00748
00749     UParseEnumInput* Direction = Record.GetPhraseInput<UParseEnumInput>(TEXT("DIRECTION"));
00750     UParseIntInput* Amount = Record.GetPhraseInput<UParseIntInput>(TEXT("AMOUNT"));
00751     if (Direction == nullptr || Amount == nullptr)
00752         return;
00753
00754     for (UObject* NodeObj : ActiveGraphEditor->GetSelectedNodes())
00755     {
00756         UEdGraphNode* Node = Cast<UEdGraphNode>(NodeObj);
00757         if (Node == nullptr)
00758             continue;
00759
00760         switch (EPhrase2DDirectionalInput(Direction->GetValue()))
00761         {
00762             case EPhrase2DDirectionalInput::UP:
00763                 Node->NodePosY -= Amount->GetValue();
00764                 break;
00765
00766             case EPhrase2DDirectionalInput::DOWN:
00767                 Node->NodePosY += Amount->GetValue();
00768                 break;
00769
00770             case EPhrase2DDirectionalInput::LEFT:
00771                 Node->NodePosX -= Amount->GetValue();
00772                 break;
00773
00774             case EPhrase2DDirectionalInput::RIGHT:
00775                 Node->NodePosX += Amount->GetValue();
00776                 break;
00777
00778             default:
00779                 UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("SelectionMove: Invalid
Direction"));
00780                 return;
00781         }
```

```
00782     }
00783 }
```

#### 4.55.3.22 SelectionNodeToggle()

```
void UNodeInteractionLibrary::SelectionNodeToggle (
    FParseRecord & Record )
```

Phrase Event for Toggling the specified Nodes Selection State, on the Active Graph Editor.

##### Parameters

|               |  |
|---------------|--|
| <i>Record</i> | The Parse Record accumulated until this Event. |
|---------------|--|

Definition at line 714 of file [NodeInteractionLibrary.cpp](#).

```
00715 {
00716     GET_CAST_ACTIVE_TAB_CONTENT(ActiveGraphEditor, SGraphEditor);
00717
00718     UParseIntInput* IndexInput = Record.GetPhraseInput<UParseIntInput>(TEXT("NODE_INDEX"));
00719     if (IndexInput == nullptr)
00720         return;
00721
00722     TSharedRef<FGraphIndexer> Indexer = GetAssetRegistry()->GetGraphIndexer(
00723         ActiveGraphEditor->GetCurrentGraph()
00724     );
00725
00726     UEdGraphNode* Node = Indexer->GetNode(IndexInput->GetValue());
00727     if (Node == nullptr)
00728     {
00729         UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("SelectionToggle: Node Not Found"));
00730         return;
00731     }
00732
00733     ActiveGraphEditor->SetNodeSelection(
00734         Node,
00735         !ActiveGraphEditor->GetSelectedNodes().Contains(Node)
00736     );
00737 }
```

#### 4.55.3.23 SelectionReset()

```
void UNodeInteractionLibrary::SelectionReset (
    FParseRecord & Record )
```

Phrase Event for Resetting the Selection Set, on the Active Graph Editor.

##### Parameters

|               |  |
|---------------|--|
| <i>Record</i> | The Parse Record accumulated until this Event. |
|---------------|--|

Definition at line 739 of file [NodeInteractionLibrary.cpp](#).

```
00739 {
00740     GET_CAST_ACTIVE_TAB_CONTENT(ActiveGraphEditor, SGraphEditor)
00741
00742     ActiveGraphEditor->ClearSelectionSet();
00743 }
```

### 4.55.3.24 SelectionStraighten()

```
void UNodeInteractionLibrary::SelectionStraighten (
    FParseRecord & Record )
```

Phrase Event for Straightening the Selection Sets Connections, on the Active Graph Editor.

#### Parameters

|               |  |
|---------------|--|
| <i>Record</i> | The Parse Record accumulated until this Event. |
|---------------|--|

Definition at line 821 of file [NodeInteractionLibrary.cpp](#).

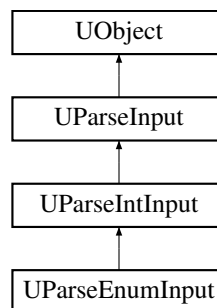
```
00822 {
00823     GET_CAST_ACTIVE_TAB_CONTENT(ActiveGraphEditor, SGraphEditor)
00824
00825     ActiveGraphEditor->OnStraightenConnections();
00826 }
```

The documentation for this class was generated from the following files:

- Source/OpenAccessibility/Public/PhraseEvents/NodeInteractionLibrary.h
- Source/OpenAccessibility/Private/PhraseEvents/NodeInteractionLibrary.cpp

## 4.56 UParseEnumInput Class Reference

Inheritance diagram for UParseEnumInput:



### Public Member Functions

- void [SetEnumType](#) (UEnum \*InEnumType)  
*Sets the Enum Type for the Input.*
- void [GetEnumType](#) (UEnum \*&OutEnumType)  
*Gets the EnumType Bound To This Input.*
- UEnum \* [GetEnumType](#) ()  
*Gets the EnumType Bound To This Input.*

### Protected Attributes

- UEnum \* [EnumType](#)

### 4.56.1 Detailed Description

Definition at line 11 of file [UParseEnumInput.h](#).

### 4.56.2 Constructor & Destructor Documentation

#### 4.56.2.1 ~UParseEnumInput()

```
virtual UParseEnumInput::~UParseEnumInput ( ) [inline], [virtual]
```

Definition at line 18 of file [UParseEnumInput.h](#).

```
00019     {
00020         delete EnumType;
00021     };
```

### 4.56.3 Member Function Documentation

#### 4.56.3.1 GetEnumType() [1/2]

```
UEnum * UParseEnumInput::GetEnumType ( ) [inline]
```

Gets the EnumType Bound To This Input.

##### Returns

The Bound EnumType of the Input.

Definition at line 45 of file [UParseEnumInput.h](#).

```
00046     {
00047         return EnumType;
00048     }
```

#### 4.56.3.2 GetEnumType() [2/2]

```
void UParseEnumInput::GetEnumType (
    UEnum *& OutEnumType ) [inline]
```

Gets the EnumType Bound To This Input.

##### Parameters

|                    |                            |
|--------------------|----------------------------|
| <i>OutEnumType</i> | The Bound EnumType To Set. |
|--------------------|----------------------------|

Definition at line 36 of file [UParseEnumInput.h](#).

```
00037     {
00038         OutEnumType = EnumType;
00039     }
```

#### 4.56.3.3 SetEnumType()

```
void UParseEnumInput::SetEnumType (
    UEnum * InEnumType ) [inline]
```

Sets the Enum Type for the Input.

##### Parameters

|                   |                                     |
|-------------------|-------------------------------------|
| <i>InEnumType</i> | The Enum Type to Set this Input To. |
|-------------------|-------------------------------------|

Definition at line 27 of file [UParseEnumInput.h](#).

```
00028     {
00029         EnumType = InEnumType;
00030     }
```

### 4.56.4 Member Data Documentation

#### 4.56.4.1 EnumType

```
UEnum* UParseEnumInput::EnumType [protected]
```

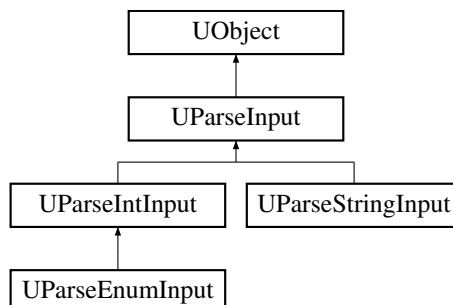
Definition at line 53 of file [UParseEnumInput.h](#).

The documentation for this class was generated from the following file:

- Source/OpenAccessibilityCommunication/Public/PhraseTree/Containers/Input/UParseEnumInput.h

## 4.57 UParseInput Class Reference

Inheritance diagram for UParseInput:



### 4.57.1 Detailed Description

Definition at line 11 of file [UParseInput.h](#).

### 4.57.2 Constructor & Destructor Documentation

#### 4.57.2.1 ~UParseInput()

```
virtual UParseInput::~~UParseInput ( ) [inline], [virtual]
```

Definition at line 18 of file [UParseInput.h](#).

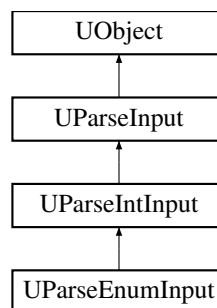
```
00019     {  
00020  
00021     };
```

The documentation for this class was generated from the following file:

- [Source/OpenAccessibilityCommunication/Public/PhraseTree/Containers/Input/UParseInput.h](#)

## 4.58 UParseIntInput Class Reference

Inheritance diagram for UParseIntInput:



### Public Member Functions

- void [SetValue](#) (int32 InValue)  
*Sets the Value of the Input.*
- void [GetValue](#) (int32 &OutValue)  
*Gets the Current Value of the Input.*
- int32 [GetValue](#) ()  
*Gets the Current Value of the Input.*

### Protected Attributes

- int32 [Value](#)



### 4.58.1 Detailed Description

Definition at line 11 of file [UParseIntInput.h](#).

### 4.58.2 Constructor & Destructor Documentation

#### 4.58.2.1 ~UParseIntInput()

```
virtual UParseIntInput::~UParseIntInput ( ) [inline], [virtual]
```

Definition at line 18 of file [UParseIntInput.h](#).

```
00019     {
00020
00021     };
```

### 4.58.3 Member Function Documentation

#### 4.58.3.1 GetValue() [1/2]

```
int32 UParseIntInput::GetValue ( ) [inline]
```

Gets the Current Value of the Input.

##### Returns

The Current Value of the Input.

Definition at line 45 of file [UParseIntInput.h](#).

```
00046     {
00047         return Value;
00048     }
```

#### 4.58.3.2 GetValue() [2/2]

```
void UParseIntInput::GetValue (
    int32 & OutValue ) [inline]
```

Gets the Current Value of the Input.

##### Parameters

|                 |   |
|-----------------|---|
| <i>OutValue</i> | - Returns the Current Value of the Input. |
|-----------------|---|

Definition at line 36 of file [UParseIntInput.h](#).

```
00037     {
00038         OutValue = Value;
00039     }
```

#### 4.58.3.3 SetValue()

```
void UParseIntInput::SetValue (
    int32 InValue ) [inline]
```

Sets the Value of the Input.

##### Parameters

|                |                                  |
|----------------|----------------------------------|
| <i>InValue</i> | - The Value to set the Input To. |
|----------------|----------------------------------|

Definition at line 27 of file [UParseIntInput.h](#).

```
00028     {
00029         Value = InValue;
00030     }
```

### 4.58.4 Member Data Documentation

#### 4.58.4.1 Value

```
int32 UParseIntInput::Value [protected]
```

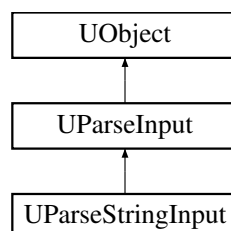
Definition at line 52 of file [UParseIntInput.h](#).

The documentation for this class was generated from the following file:

- Source/OpenAccessibilityCommunication/Public/PhraseTree/Containers/Input/UParseIntInput.h

## 4.59 UParseStringInput Class Reference

Inheritance diagram for UParseStringInput:



## Public Member Functions

- void [SetValue](#) (FString InValue)  
*Sets the Value of the Input.*
- void [GetValue](#) (FString &OutValue)  
*Gets the Value of the Input.*
- FString [GetValue](#) ()  
*Gets the Value of the Input.*

## Protected Attributes

- FString [Value](#)

### 4.59.1 Detailed Description

Definition at line 11 of file [UParseStringInput.h](#).

### 4.59.2 Constructor & Destructor Documentation

#### 4.59.2.1 ~UParseStringInput()

```
virtual UParseStringInput::~UParseStringInput ( ) [inline], [virtual]
```

Definition at line 18 of file [UParseStringInput.h](#).

```
00019 {
00020
00021 };
```

### 4.59.3 Member Function Documentation

#### 4.59.3.1 GetValue() [1/2]

```
FString UParseStringInput::GetValue ( ) [inline]
```

Gets the Value of the Input.

**Returns**

Definition at line 45 of file [UParseStringInput.h](#).

```
00046 {
00047     return Value;
00048 }
```

#### 4.59.3.2 GetValue() [2/2]

```
void UParseStringInput::GetValue (
    FString & OutValue ) [inline]
```

Gets the Value of the Input.

**Parameters**

|                 |   |
|-----------------|---|
| <i>OutValue</i> | - Returns the Current Value of the Input. |
|-----------------|---|

Definition at line 36 of file [UParseStringInput.h](#).

```
00037 {
00038     OutValue = Value;
00039 }
```

**4.59.3.3 SetValue()**

```
void UParseStringInput::SetValue (
    FString InValue ) [inline]
```

Sets the Value of the Input.

**Parameters**

|                |                                  |
|----------------|----------------------------------|
| <i>InValue</i> | - The Value to set the Input To. |
|----------------|----------------------------------|

Definition at line 27 of file [UParseStringInput.h](#).

```
00028 {
00029     Value = InValue;
00030 }
```

**4.59.4 Member Data Documentation****4.59.4.1 Value**

```
FString UParseStringInput::Value [protected]
```

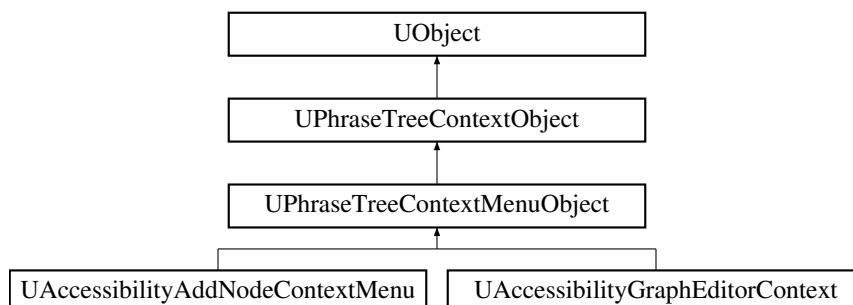
Definition at line 52 of file [UParseStringInput.h](#).

The documentation for this class was generated from the following file:

- Source/OpenAccessibilityCommunication/Public/PhraseTree/Containers/Input/UParseStringInput.h

**4.60 UPhraseTreeContextMenuObject Class Reference**

Inheritance diagram for UPhraseTreeContextMenuObject:



## Public Member Functions

- [UPhraseTreeContextMenuObject](#) (TSharedRef< IMenu > [Menu](#))
- virtual void [Init](#) (TSharedRef< IMenu > InMenu)  
*Initializes the Context Menu Object.*
- virtual void [Init](#) (TSharedRef< IMenu > InMenu, TSharedRef< [FPhraseNode](#) > InContextRoot)  
*Initializes the Context Menu Object.*
- virtual bool [Tick](#) (float DeltaTime)
- virtual bool [Close](#) () override  
*Closes the Context Menu Object.*
- void [BindTickDelegate](#) ()  
*Binds the Tick Delegate to the Core Ticker.*
- void [RemoveTickDelegate](#) ()  
*UnBinds the Tick Delegate from the Core Ticker.*
- void [BindMenuDismissed](#) (TSharedRef< IMenu > InMenu)  
*Binds the Menu Dismissed Callback to the Menu.*
- void [RemoveMenuDismissed](#) (TSharedRef< IMenu > InMenu)  
*UnBinds the Menu Dismissed Callback from the Menu.*
- virtual void [SetMenu](#) (TSharedRef< IMenu > InMenu)  
*Sets the Menu Object for this Context Menu.*
- virtual void [ScaleMenu](#) (const float ScaleFactor)  
*Scales the Provided Menu Object, and any Key Objects.*

## Public Attributes

- TWeakPtr< IMenu > [Menu](#)  
*The Menu Object.*
- TWeakPtr< SWindow > [Window](#)  
*The Menu's Window.*

## Protected Member Functions

- TSharedPtr< SWindow > [GetWindow](#) ()  
*Gets the Window Object for this Context Menu.*
- void [OnMenuDismissed](#) (TSharedRef< IMenu > [Menu](#))  
*Callback for the Dismissal of the Menu.*

## Additional Inherited Members

### 4.60.1 Detailed Description

Definition at line 14 of file [ContextMenuObject.h](#).

### 4.60.2 Constructor & Destructor Documentation

#### 4.60.2.1 UPhraseTreeContextMenuObject() [1/2]

UPhraseTreeContextMenuObject::UPhraseTreeContextMenuObject ( )

Definition at line 7 of file [ContextMenuObject.cpp](#).

```
00008      : UPhraseTreeContextObject ()
00009  {
00010
00011  }
```

#### 4.60.2.2 UPhraseTreeContextMenuObject() [2/2]

UPhraseTreeContextMenuObject::UPhraseTreeContextMenuObject (   
 TSharedRef< IMenu > Menu )

Definition at line 13 of file [ContextMenuObject.cpp](#).

```
00014      : UPhraseTreeContextObject ()
00015  {
00016
00017  }
```

#### 4.60.2.3 ~UPhraseTreeContextMenuObject()

UPhraseTreeContextMenuObject::~~UPhraseTreeContextMenuObject ( ) [virtual]

Definition at line 19 of file [ContextMenuObject.cpp](#).

```
00020  {
00021      // Unbind Tick Delegate
00022      RemoveTickDelegate();
00023
00024      if (Menu.IsValid())
00025          RemoveMenuDismissed(Menu.Pin().ToSharedRef());
00026
00027      UE_LOG(LogOpenAccessibilityCom, Warning, TEXT("|| Context Menu || Destroyed ||"))
00028  }
```

### 4.60.3 Member Function Documentation

#### 4.60.3.1 BindMenuDismissed()

void UPhraseTreeContextMenuObject::BindMenuDismissed (   
 TSharedRef< IMenu > InMenu )

Binds the Menu Dismissed Callback to the Menu.

##### Parameters

|               |  |
|---------------|--|
| <i>InMenu</i> |  |
|---------------|--|

Definition at line 66 of file [ContextMenuObject.cpp](#).

```
00067 {
00068     MenuDismissedHandle = InMenu->GetOnMenuDismissed()
00069     .AddUObject (this, &UPhraseTreeContextMenuObject::OnMenuDismissed);
00070 }
```

#### 4.60.3.2 BindTickDelegate()

```
void UPhraseTreeContextMenuObject::BindTickDelegate ( )
```

Binds the Tick Delegate to the Core Ticker.

Definition at line 54 of file [ContextMenuObject.cpp](#).

```
00055 {
00056     TickDelegate = FTickerDelegate::CreateUObject (this, &UPhraseTreeContextMenuObject::Tick);
00057     TickDelegateHandle = FTSTicker::GetCoreTicker().AddTicker (TickDelegate);
00058 }
```

#### 4.60.3.3 Close()

```
virtual bool UPhraseTreeContextMenuObject::Close ( ) [inline], [override], [virtual]
```

Closes the Context Menu Object.

**Returns**

Reimplemented from [UPhraseTreeContextObject](#).

Reimplemented in [UAccessibilityAddNodeContextMenu](#), and [UAccessibilityGraphEditorContext](#).

Definition at line 44 of file [ContextMenuObject.h](#).

```
00045 {
00046     RemoveTickDelegate();
00047     Menu.Pin()->Dismiss();
00048
00049     return true;
00050 };
```

#### 4.60.3.4 GetWindow()

```
TSharedPtr< SWindow > UPhraseTreeContextMenuObject::GetWindow ( ) [inline], [protected]
```

Gets the Window Object for this Context Menu.

**Returns**

Definition at line 95 of file [ContextMenuObject.h](#).

```
00096 {
00097     return Menu.Pin()->GetOwnedWindow();
00098 }
```

#### 4.60.3.5 Init() [1/2]

```
void UPhraseTreeContextMenuObject::Init (
    TSharedRef< IMenu > InMenu ) [virtual]
```

Initializes the Context Menu Object.

**Parameters**

|               |  |
|---------------|--|
| <i>InMenu</i> | - The Menu Object for this Context Menu. |
|---------------|--|

Reimplemented in [UAccessibilityAddNodeContextMenu](#).

Definition at line 30 of file [ContextMenuObject.cpp](#).

```
00031 {
00032     this->Menu = InMenu;
00033     this->Window = FSlateApplication::Get().FindWidgetWindow(
00034         InMenu->GetContent().ToSharedRef()
00035     );
00036
00037     BindMenuDismissed(InMenu);
00038     BindTickDelegate();
00039 }
```

**4.60.3.6 Init() [2/2]**

```
void UPhraseTreeContextMenuObject::Init (
    TSharedRef< IMenu > InMenu,
    TSharedRef< FPhraseNode > InContextRoot ) [virtual]
```

Initializes the Context Menu Object.

**Parameters**

|                      |  |
|----------------------|--|
| <i>InMenu</i>        | - The Menu Object For this Context Menu.               |
| <i>InContextRoot</i> | - The Context Root In The Phrase Tree For This Object. |

Reimplemented in [UAccessibilityAddNodeContextMenu](#), and [UAccessibilityGraphEditorContext](#).

Definition at line 41 of file [ContextMenuObject.cpp](#).

```
00042 {
00043     this->Menu = InMenu;
00044     this->Window = FSlateApplication::Get().FindWidgetWindow(
00045         InMenu->GetContent().ToSharedRef()
00046     );
00047
00048     this->ContextRoot = InContextRoot;
00049
00050     BindMenuDismissed(InMenu);
00051     BindTickDelegate();
00052 }
```

**4.60.3.7 OnMenuDismissed()**

```
void UPhraseTreeContextMenuObject::OnMenuDismissed (
    TSharedRef< IMenu > Menu ) [protected]
```

Callback for the Dismissal of the Menu.



## Parameters

|             |  |
|-------------|--|
| <i>Menu</i> |  |
|-------------|--|

Definition at line 77 of file [ContextMenuObject.cpp](#).

```
00078 {
00079     RemoveTickDelegate();
00080
00081     RemoveFromRoot();
00082     MarkAsGarbage();
00083
00084     bIsActive = false;
00085
00086     UE_LOG(LogOpenAccessibilityCom, Warning, TEXT("|| Context Menu || Dismissed ||"))
00087 }
```

#### 4.60.3.8 RemoveMenuDismissed()

```
void UPhraseTreeContextMenuObject::RemoveMenuDismissed (
    TSharedRef< IMenu > InMenu )
```

UnBinds the Menu Dismissed Callback from the Menu.

## Parameters

|               |  |
|---------------|--|
| <i>InMenu</i> |  |
|---------------|--|

Definition at line 72 of file [ContextMenuObject.cpp](#).

```
00073 {
00074     Menu.Pin()->GetOnMenuDismissed().Remove(MenuDismissedHandle);
00075 }
```

#### 4.60.3.9 RemoveTickDelegate()

```
void UPhraseTreeContextMenuObject::RemoveTickDelegate ( )
```

UnBinds the Tick Delegate from the Core Ticker.

Definition at line 60 of file [ContextMenuObject.cpp](#).

```
00061 {
00062     if (TickDelegateHandle != NULL)
00063         FTSTicker::GetCoreTicker().RemoveTicker(TickDelegateHandle);
00064 }
```

#### 4.60.3.10 ScaleMenu()

```
virtual void UPhraseTreeContextMenuObject::ScaleMenu (
    const float ScaleFactor ) [inline], [virtual]
```

Scales the Provided Menu Object, and any Key Objects.

## Parameters

|                    |  |
|--------------------|--|
| <i>ScaleFactor</i> |  |
|--------------------|--|

Reimplemented in [UAccessibilityAddNodeContextMenu](#), and [UAccessibilityGraphEditorContext](#).

Definition at line 87 of file [ContextMenuObject.h](#).

```
00087 {};
```

#### 4.60.3.11 SetMenu()

```
virtual void UPhraseTreeContextMenuObject::SetMenu (
    TSharedRef< IMenu > InMenu ) [inline], [virtual]
```

Sets the Menu Object for this Context Menu.

## Parameters

|               |  |
|---------------|--|
| <i>InMenu</i> |  |
|---------------|--|

Definition at line 78 of file [ContextMenuObject.h](#).

```
00079 {
00080     Menu = InMenu;
00081 }
```

#### 4.60.3.12 Tick()

```
virtual bool UPhraseTreeContextMenuObject::Tick (
    float DeltaTime ) [inline], [virtual]
```

Definition at line 38 of file [ContextMenuObject.h](#).

```
00038 { return true; };
```

### 4.60.4 Member Data Documentation

#### 4.60.4.1 Menu

```
TWeakPtr<IMenu> UPhraseTreeContextMenuObject::Menu
```

The Menu Object.

Definition at line 111 of file [ContextMenuObject.h](#).

#### 4.60.4.2 Window

```
TWeakPtr<SWindow> UPhraseTreeContextMenuObject::Window
```

The Menu's Window.

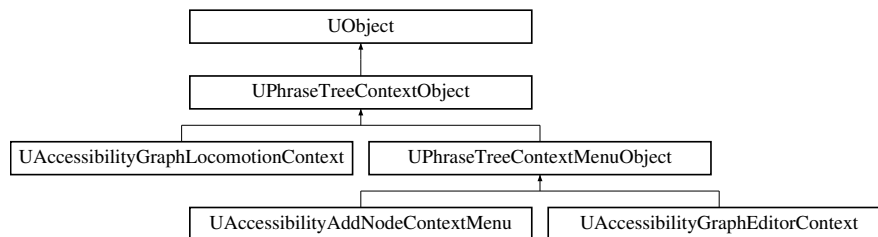
Definition at line 116 of file [ContextMenuObject.h](#).

The documentation for this class was generated from the following files:

- Source/OpenAccessibilityCommunication/Public/PhraseTree/Containers/ContextMenuObject.h
- Source/OpenAccessibilityCommunication/Private/PhraseTree/Containers/ContextMenuObject.cpp

## 4.61 UPhraseTreeContextObject Class Reference

Inheritance diagram for UPhraseTreeContextObject:



### Public Member Functions

- virtual bool [Close](#) ()
- void [SetContextRootNode](#) (TSharedRef< [FPhraseNode](#) > InRootNode)  
*Sets the Root Node In The Phrase Tree For This Context Objects.*
- TSharedPtr< [FPhraseNode](#) > [GetContextRoot](#) ()  
*Gets the Root Node For This Context.*
- const bool [GetIsActive](#) ()  
*Is the Context Still Active.*

### Protected Attributes

- bool [bIsActive](#) = true  
*Is the Context Object Still Active.*
- TWeakPtr< [FPhraseNode](#) > [ContextRoot](#)  
*The Root Node In The Phrase Tree (The Origin of the Context). Allowing for Propagation based on Context Root.*

#### 4.61.1 Detailed Description

Definition at line 12 of file [ContextObject.h](#).

## 4.61.2 Constructor & Destructor Documentation

### 4.61.2.1 UPhraseTreeContextObject()

UPhraseTreeContextObject::UPhraseTreeContextObject ( ) [inline]

Definition at line 18 of file [ContextObject.h](#).

```
00019         : UObject()
00020     {
00021
00022     }
```

### 4.61.2.2 ~UPhraseTreeContextObject()

virtual UPhraseTreeContextObject::~UPhraseTreeContextObject ( ) [inline], [virtual]

Definition at line 24 of file [ContextObject.h](#).

```
00025     {
00026
00027     }
```

## 4.61.3 Member Function Documentation

### 4.61.3.1 Close()

virtual bool UPhraseTreeContextObject::Close ( ) [inline], [virtual]

Reimplemented in [UAccessibilityAddNodeContextMenu](#), [UAccessibilityGraphEditorContext](#), and [UPhraseTreeContextMenuObject](#).

Definition at line 29 of file [ContextObject.h](#).

```
00029 { return true; }
```

### 4.61.3.2 GetContextRoot()

TSharedPtr< [FPhraseNode](#) > UPhraseTreeContextObject::GetContextRoot ( ) [inline]

Gets the Root Node For This Context.

**Returns**

Definition at line 44 of file [ContextObject.h](#).

```
00045     {
00046         return ContextRoot.Pin();
00047     }
```

### 4.61.3.3 GetIsActive()

```
const bool UPhraseTreeContextObject::GetIsActive ( ) [inline]
```

Is the Context Still Active.

**Returns**

Definition at line 53 of file [ContextObject.h](#).

```
00054     {
00055         return bIsActive;
00056     }
```

### 4.61.3.4 SetContextRootNode()

```
void UPhraseTreeContextObject::SetContextRootNode (
    TSharedRef< FPhraseNode > InRootNode ) [inline]
```

Sets the Root Node In The Phrase Tree For This Context Objects.

**Parameters**

|                   |  |
|-------------------|--|
| <i>InRootNode</i> |  |
|-------------------|--|

Definition at line 35 of file [ContextObject.h](#).

```
00036     {
00037         ContextRoot = InRootNode;
00038     }
```

## 4.61.4 Member Data Documentation

### 4.61.4.1 bIsActive

```
bool UPhraseTreeContextObject::bIsActive = true [protected]
```

Is the Context Object Still Active.

Definition at line 63 of file [ContextObject.h](#).

#### 4.61.4.2 ContextRoot

```
TWeakPtr<FPhraseNode> UPhraseTreeContextObject::ContextRoot [protected]
```

The Root Node In The Phrase Tree (The Origin of the Context). Allowing for Propagation based on Context Root.

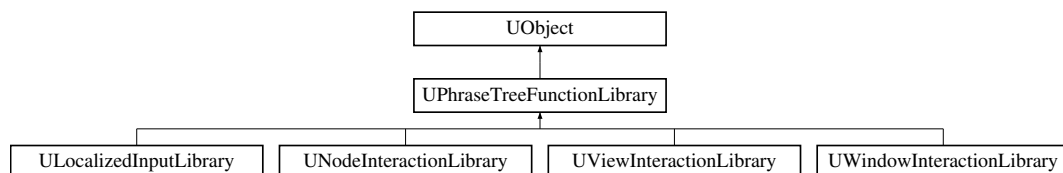
Definition at line 69 of file [ContextObject.h](#).

The documentation for this class was generated from the following file:

- [Source/OpenAccessibilityCommunication/Public/PhraseTree/Containers/ContextObject.h](#)

## 4.62 UPhraseTreeFunctionLibrary Class Reference

Inheritance diagram for UPhraseTreeFunctionLibrary:



### Public Member Functions

- virtual void [BindBranches](#) (TSharedRef< [FPhraseTree](#) > PhraseTree)

#### 4.62.1 Detailed Description

Definition at line 27 of file [PhraseTreeFunctionLibrary.h](#).

#### 4.62.2 Member Function Documentation

##### 4.62.2.1 BindBranches()

```
virtual void UPhraseTreeFunctionLibrary::BindBranches (
    TSharedRef< FPhraseTree > PhraseTree ) [inline], [virtual]
```

Reimplemented in [ULocalizedInputLibrary](#), [UNodeInteractionLibrary](#), [UViewInteractionLibrary](#), and [UWindowInteractionLibrary](#).

Definition at line 33 of file [PhraseTreeFunctionLibrary.h](#).

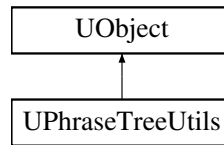
```
00033 {};
```

The documentation for this class was generated from the following file:

- [Source/OpenAccessibilityCommunication/Public/PhraseTree/PhraseTreeFunctionLibrary.h](#)

## 4.63 UPhraseTreeUtils Class Reference

Inheritance diagram for UPhraseTreeUtils:



### Public Member Functions

- void [RegisterFunctionLibrary](#) ([UPhraseTreeFunctionLibrary](#) \*LibraryToRegister)
- void [SetPhraseTree](#) (TSharedRef< [FPhraseTree](#) > NewPhraseTree)

### Protected Attributes

- TArray< [UPhraseTreeFunctionLibrary](#) \* > [RegisteredLibraries](#)
- TWeakPtr< [FPhraseTree](#) > [PhraseTree](#)

#### 4.63.1 Detailed Description

Definition at line 12 of file [PhraseTreeUtils.h](#).

#### 4.63.2 Constructor & Destructor Documentation

##### 4.63.2.1 UPhraseTreeUtils()

```
UPhraseTreeUtils::UPhraseTreeUtils ( )
```

Definition at line 5 of file [PhraseTreeUtils.cpp](#).

```
00006 {
00007
00008 }
```

##### 4.63.2.2 ~UPhraseTreeUtils()

```
UPhraseTreeUtils::~~UPhraseTreeUtils ( ) [virtual]
```

Definition at line 10 of file [PhraseTreeUtils.cpp](#).

```
00011 {
00012
00013 }
```

### 4.63.3 Member Function Documentation

#### 4.63.3.1 RegisterFunctionLibrary()

```
void UPhraseTreeUtils::RegisterFunctionLibrary (
    UPhraseTreeFunctionLibrary * LibraryToRegister )
```

Registers the provided Phrase Tree Function Library.



## Parameters

|                          |   |
|--------------------------|---|
| <i>LibraryToRegister</i> | The Phrase Tree Function Library to Register. |
|--------------------------|---|

Definition at line 15 of file [PhraseTreeUtils.cpp](#).

```

00016 {
00017     TSharedPtr<FPhraseTree> PhraseTreeSP = PhraseTree.Pin();
00018     if (!PhraseTreeSP.IsValid())
00019     {
00020         UE_LOG(LogOpenAccessibilityCom, Warning, TEXT("Cannot Register Phrase Tree Function Library
Due To Invalid Phrase Tree Reference.));
00021         return;
00022     }
00023
00024     // For some reason this needs to be told directly to be kept alive,
00025     // even though it is a UPROPERTY TArray and should be kept alive by the UObject system.
00026     LibraryToRegister->AddToRoot();
00027     LibraryToRegister->BindBranches(PhraseTreeSP.ToSharedRef());
00028 }
```

#### 4.63.3.2 SetPhraseTree()

```

void UPhraseTreeUtils::SetPhraseTree (
    TSharedPtr< FPhraseTree > NewPhraseTree ) [inline]
```

Sets the Phrase Tree Reference used for Registering Phrase Tree Function Libraries.

## Parameters

|                      |                                      |
|----------------------|--------------------------------------|
| <i>NewPhraseTree</i> | Reference to the Phrase Tree to use. |
|----------------------|--------------------------------------|

Definition at line 34 of file [PhraseTreeUtils.h](#).

```

00035 {
00036     this->PhraseTree = NewPhraseTree;
00037 }
```

### 4.63.4 Member Data Documentation

#### 4.63.4.1 PhraseTree

```
TWeakPtr<FPhraseTree> UPhraseTreeUtils::PhraseTree [protected]
```

Weak Pointer to the Current Phrase Tree Instance used in Registering the Phrase Tree Function Libraries.

Definition at line 51 of file [PhraseTreeUtils.h](#).

#### 4.63.4.2 RegisteredLibraries

```
TArray<UPhraseTreeFunctionLibrary*> UPhraseTreeUtils::RegisteredLibraries [protected]
```

An Array of all the Registered Phrase Tree Function Libraries.

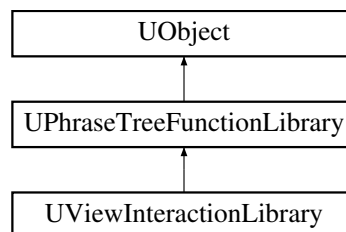
Definition at line 45 of file [PhraseTreeUtils.h](#).

The documentation for this class was generated from the following files:

- Source/OpenAccessibilityCommunication/Public/PhraseTreeUtils.h
- Source/OpenAccessibilityCommunication/Private/PhraseTreeUtils.cpp

## 4.64 UViewInteractionLibrary Class Reference

Inheritance diagram for UViewInteractionLibrary:



### Public Member Functions

- [UViewInteractionLibrary](#) (const FObjectInitializer &ObjectInitializer)
- void [BindBranches](#) (TSharedRef< [FPhraseTree](#) > PhraseTree) override
- void [MoveViewport](#) (FParseRecord &Record)
- void [ZoomViewport](#) (FParseRecord &Record)
- void [IndexFocus](#) (FParseRecord &Record)

#### 4.64.1 Detailed Description

Definition at line 12 of file [ViewInteractionLibrary.h](#).

#### 4.64.2 Constructor & Destructor Documentation

##### 4.64.2.1 UViewInteractionLibrary()

```
UViewInteractionLibrary::UViewInteractionLibrary (
    const FObjectInitializer & ObjectInitializer )
```

Definition at line 12 of file [ViewInteractionLibrary.cpp](#).

```
00013     : Super(ObjectInitializer)
00014 {
00015
00016 }
```

#### 4.64.2.2 ~UViewInteractionLibrary()

UViewInteractionLibrary::~UViewInteractionLibrary ( ) [virtual]

Definition at line 18 of file [ViewInteractionLibrary.cpp](#).

```
00019 {
00020
00021 }
```

### 4.64.3 Member Function Documentation

#### 4.64.3.1 BindBranches()

```
void UViewInteractionLibrary::BindBranches (
    TSharedRef< FPhraseTree > PhraseTree ) [override], [virtual]
```

Binds Branches originating from this Library onto the provided Phrase Tree.

##### Parameters

|                   |  |
|-------------------|--|
| <i>PhraseTree</i> | Reference to the PhraseTree to Bind this Library to. |
|-------------------|--|

Reimplemented from [UPhraseTreeFunctionLibrary](#).

Definition at line 23 of file [ViewInteractionLibrary.cpp](#).

```
00024 {
00025     PhraseTree->BindBranch(
00026         MakeShared<FPhraseNode>(TEXT("VIEW"),
00027             TPhraseNodeArray {
00028
00029                 MakeShared<FPhraseNode>(TEXT("MOVE"),
00030                     TPhraseNodeArray {
00031
00032                         MakeShared<FPhrase2DDirectionalInputNode>(TEXT("DIRECTION"),
00033                             TPhraseNodeArray {
00034
00035                                 MakeShared<FPhraseInputNode<int32>(TEXT("AMOUNT"),
00036                                     TPhraseNodeArray {
00037
00038                                         MakeShared<FPhraseEventNode>(CreateParseDelegate(this,
00039                                             &UViewInteractionLibrary::MoveViewport))
00039                                     })
00040                                 })
00041                             })
00042                         })
00043                     }),
00044                 ),
00045             MakeShared<FPhraseNode>(TEXT("ZOOM"),
00046                 TPhraseNodeArray {
00047
00048                     MakeShared<FPhrase2DDirectionalInputNode>(TEXT("DIRECTION"),
00049                         TPhraseNodeArray {
00050
00051                             MakeShared<FPhraseInputNode<int32>(TEXT("AMOUNT"),
00052                                 TPhraseNodeArray {
00053
00054                                     MakeShared<FPhraseEventNode>(CreateParseDelegate(this,
00055                                         &UViewInteractionLibrary::ZoomViewport))
00056                                 })
00057                             })
00058                         })
00059                     })
00060             })
00061         })
00062     }
```

```

00061         }},
00062
00063         MakeShared<FPhraseNode>(TEXT("FOCUS"),
00064         TPhraseNodeArray {
00065             MakeShared<FPhraseInputNode<int32>>(TEXT("INDEX"),
00066             TPhraseNodeArray {
00067                 MakeShared<FPhraseEventNode>(CreateParseDelegate(this,
00068                 &UViewInteractionLibrary::IndexFocus))
00069             })
00070         })
00071     })
00072
00073     })
00074
00075     })
00076 };
00077 }

```

#### 4.64.3.2 IndexFocus()

```

void UViewInteractionLibrary::IndexFocus (
    FParseRecord & Record )

```

Phrase Event for Focusing on the Active Viewports Indexed Item, if one is apparent.

##### Parameters

|               |  |
|---------------|--|
| <i>Record</i> | The Parse Record accumulated until this Event. |
|---------------|--|

Definition at line 165 of file [ViewInteractionLibrary.cpp](#).

```

00166 {
00167     GET_ACTIVE_TAB_CONTENT(ActiveTab)
00168
00169     FString TabType = ActiveTab->GetTypeAsString();
00170
00171     UParseIntInput* IndexInput = Record.GetPhraseInput<UParseIntInput>(TEXT("INDEX"));
00172     if (IndexInput == nullptr)
00173         return;
00174
00175     if (TabType == "SGraphEditor")
00176     {
00177         TSharedPtr<SGraphEditor> GraphEditor = StaticCastSharedPtr<SGraphEditor>(ActiveTab);
00178         if (!GraphEditor.IsValid())
00179             return;
00180
00181         TSharedRef<FAssetAccessibilityRegistry> AssetRegistry = GetAssetRegistry();
00182
00183         TSharedRef<FGraphIndexer> GraphIndexer =
00184             AssetRegistry->GetGraphIndexer(GraphEditor->GetCurrentGraph());
00185
00186         UEdGraphNode* Node = GraphIndexer->GetNode(IndexInput->GetValue());
00187         if (Node == nullptr)
00188         {
00189             UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("IndexFocus: INVALID INDEX INPUT"))
00190             return;
00191         }
00192
00193         GraphEditor->JumpToNode(Node);
00194     }
00195
00196     // Further ViewportS Specific Implementation Here
00197 }

```

#### 4.64.3.3 MoveViewport()

```

void UViewInteractionLibrary::MoveViewport (
    FParseRecord & Record )

```

Phrase Event for Moving the Active Viewport.

#### Parameters

|               |  |
|---------------|--|
| <i>Record</i> | The Parse Record accumulated until this Event. |
|---------------|--|

Definition at line 79 of file [ViewInteractionLibrary.cpp](#).

```

00079                                     {
00080     GET_ACTIVE_TAB_CONTENT (ActiveTab)
00081
00082     FString TabType = ActiveTab->GetTypeAsString();
00083
00084     UParseEnumInput* DirectionInput = Record.GetPhraseInput<UParseEnumInput>(TEXT("DIRECTION"));
00085     UParseIntInput* AmountInput = Record.GetPhraseInput<UParseIntInput>(TEXT("AMOUNT"));
00086     if (DirectionInput == nullptr || AmountInput == nullptr)
00087         return;
00088
00089     if (TabType == "SGraphEditor")
00090     {
00091         TSharedPtr<SGraphEditor> GraphEditor = StaticCastSharedPtr<SGraphEditor>(ActiveTab);
00092
00093         FVector2D ViewLocation;
00094         float ZoomAmount;
00095         GraphEditor->GetViewLocation(ViewLocation, ZoomAmount);
00096
00097         switch (EPhrase2DDirectionalInput (DirectionInput->GetValue()))
00098         {
00099             case EPhrase2DDirectionalInput::UP:
00100                 ViewLocation.Y -= AmountInput->GetValue();
00101                 break;
00102
00103             case EPhrase2DDirectionalInput::DOWN:
00104                 ViewLocation.Y += AmountInput->GetValue();
00105                 break;
00106
00107             case EPhrase2DDirectionalInput::LEFT:
00108                 ViewLocation.X -= AmountInput->GetValue();
00109                 break;
00110
00111             case EPhrase2DDirectionalInput::RIGHT:
00112                 ViewLocation.X += AmountInput->GetValue();
00113                 break;
00114
00115             default:
00116                 UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("MoveViewport: INVALID DIRECTION
00117 INPUT"));
00118                 return;
00119         }
00120         GraphEditor->SetViewLocation(ViewLocation, ZoomAmount);
00121     }
00122
00123     // Further Viewport Implementation Here
00124 }
```

#### 4.64.3.4 ZoomViewport()

```

void UViewInteractionLibrary::ZoomViewport (
    FParseRecord & Record )
```

Phrase Event for Zooming the Active Viewport.

#### Parameters

|               |  |
|---------------|--|
| <i>Record</i> | The Parse Record accumulated until this Event. |
|---------------|--|

Definition at line 126 of file [ViewInteractionLibrary.cpp](#).

```

00127 {
```

```

00128     GET_ACTIVE_TAB_CONTENT (ActiveTab)
00129
00130     FString TabType = ActiveTab->GetTypeAsString();
00131
00132     UParseEnumInput* DirectionInput = Record.GetPhraseInput<UParseEnumInput>(TEXT("DIRECTION"));
00133     UParseIntInput* AmountInput = Record.GetPhraseInput<UParseIntInput>(TEXT("AMOUNT"));
00134     if (DirectionInput == nullptr || AmountInput == nullptr)
00135         return;
00136
00137     if (TabType == "SGraphEditor")
00138     {
00139         TSharedPtr<SGraphEditor> GraphEditor = StaticCastSharedPtr<SGraphEditor>(ActiveTab);
00140
00141         FVector2D ViewLocation;
00142         float ZoomAmount;
00143         GraphEditor->GetViewLocation(ViewLocation, ZoomAmount);
00144
00145         switch (EPhrase2DDirectionalInput(DirectionInput->GetValue())) {
00146             case EPhrase2DDirectionalInput::UP:
00147                 ZoomAmount += AmountInput->GetValue();
00148                 break;
00149
00150             case EPhrase2DDirectionalInput::DOWN:
00151                 ZoomAmount -= AmountInput->GetValue();
00152                 break;
00153
00154             default:
00155                 UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("ZoomViewport: INVALID DIRECTION
00156 INPUT"));
00157                 return;
00158         }
00159         GraphEditor->SetViewLocation(ViewLocation, ZoomAmount);
00160     }
00161
00162     // Further Viewport Specific Implementation Here
00163 }

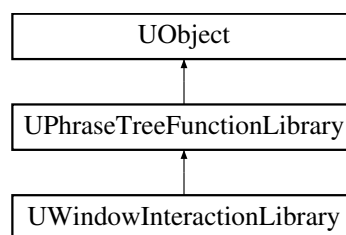
```

The documentation for this class was generated from the following files:

- Source/OpenAccessibility/Public/PhraseEvents/ViewInteractionLibrary.h
- Source/OpenAccessibility/Private/PhraseEvents/ViewInteractionLibrary.cpp

## 4.65 UWindowInteractionLibrary Class Reference

Inheritance diagram for UWindowInteractionLibrary:



### Public Member Functions

- [UWindowInteractionLibrary](#) (const FObjectInitializer &ObjectInitializer)
- void [BindBranches](#) (TSharedPtr< [FPhraseTree](#) > PhraseTree) override
- void [SwitchNextActiveWindow](#) (FPhraseRecord &Record)
- void [SwitchPrevActiveWindow](#) (FPhraseRecord &Record)
- void [CloseActiveWindow](#) (FPhraseRecord &Record)
- void [SelectToolBarItem](#) (FPhraseRecord &Record)
- void [SwitchNextTabInStack](#) (FPhraseRecord &Record)
- void [SwitchPrevTabInStack](#) (FPhraseRecord &Record)

## Protected Attributes

- class [UAccessibilityWindowToolBar](#) \* [WindowToolBar](#)

### 4.65.1 Detailed Description

Definition at line 12 of file [WindowInteractionLibrary.h](#).

### 4.65.2 Constructor & Destructor Documentation

#### 4.65.2.1 UWindowInteractionLibrary()

```
UWindowInteractionLibrary::UWindowInteractionLibrary (
    const FObjectInitializer & ObjectInitializer )
```

Definition at line 11 of file [WindowInteractionLibrary.cpp](#).

```
00012     : Super(ObjectInitializer)
00013 {
00014     WindowToolBar = NewObject<UAccessibilityWindowToolBar>();
00015 }
```

#### 4.65.2.2 ~UWindowInteractionLibrary()

```
UWindowInteractionLibrary::~UWindowInteractionLibrary ( ) [virtual]
```

Definition at line 17 of file [WindowInteractionLibrary.cpp](#).

```
00018 {
00019
00020 }
```

### 4.65.3 Member Function Documentation

#### 4.65.3.1 BindBranches()

```
void UWindowInteractionLibrary::BindBranches (
    TSharedRef< FPhraseTree > PhraseTree ) [override], [virtual]
```

Binds Branches originating from this Library onto the provided Phrase Tree.

#### Parameters

|                   |  |
|-------------------|--|
| <i>PhraseTree</i> | Reference to the PhraseTree to Bind this Library to. |
|-------------------|--|

Reimplemented from [UPhraseTreeFunctionLibrary](#).

Definition at line 22 of file [WindowInteractionLibrary.cpp](#).

```

00023 {
00024     PhraseTree->BindBranches (
00025         TPhraseNodeArray{
00026
00027             MakeShared<FPhraseNode> (TEXT ("WINDOW"),
00028                 TPhraseNodeArray{
00029
00030                 MakeShared<FPhraseNode> (TEXT ("NEXT"),
00031                     TPhraseNodeArray {
00032
00033                     MakeShared<FPhraseEventNode> (CreateParseDelegate (this,
00034                         &UWindowInteractionLibrary::SwitchNextActiveWindow))
00035
00036                     }},
00037
00038                 MakeShared<FPhraseNode> (TEXT ("PREVIOUS"),
00039                     TPhraseNodeArray {
00040
00041                     MakeShared<FPhraseEventNode> (CreateParseDelegate (this,
00042                         &UWindowInteractionLibrary::SwitchPrevActiveWindow))
00043
00044                     }},
00045
00046                 MakeShared<FPhraseNode> (TEXT ("CLOSE"),
00047                     TPhraseNodeArray {
00048
00049                     MakeShared<FPhraseEventNode> (CreateParseDelegate (this,
00050                         &UWindowInteractionLibrary::CloseActiveWindow))
00051
00052                     }},
00053
00054                 MakeShared<FPhraseNode> (TEXT ("TAB"),
00055                     TPhraseNodeArray {
00056
00057                     MakeShared<FPhraseNode> (TEXT ("NEXT"),
00058                         TPhraseNodeArray {
00059
00060                         MakeShared<FPhraseEventNode> (CreateParseDelegate (this,
00061                             &UWindowInteractionLibrary::SwitchNextTabInStack))
00062
00063                         }},
00064
00065                     MakeShared<FPhraseNode> (TEXT ("PREVIOUS"),
00066                         TPhraseNodeArray {
00067
00068                         MakeShared<FPhraseEventNode> (CreateParseDelegate (this,
00069                             &UWindowInteractionLibrary::SwitchPrevTabInStack))
00070
00071                         }},
00072
00073                     MakeShared<FPhraseNode> (TEXT ("TOOLBAR"),
00074                         TPhraseNodeArray {
00075
00076                         MakeShared<FPhraseInputNode<int32> (TEXT ("ITEM_INDEX"),
00077                             TPhraseNodeArray {
00078
00079                             MakeShared<FPhraseEventNode> (CreateParseDelegate (this,
00080                                 &UWindowInteractionLibrary::SelectToolBarItem))
00081
00082                             }},
00083
00084                         }},
00085
00086                     }},
00087
00088                 }
00089             );
00090     }

```

#### 4.65.3.2 CloseActiveWindow()

```

void UWindowInteractionLibrary::CloseActiveWindow (
    FParseRecord & Record )

```

Closes the Top Most Active Window, if it is not the Root Application Window.



## Parameters

|               |  |
|---------------|--|
| <i>Record</i> | The Parse Record accumulated until this Event. |
|---------------|--|

Definition at line 134 of file [WindowInteractionLibrary.cpp](#).

```

00134                                     {
00135     FSlateApplication& SlateApp = FSlateApplication::Get();
00136     if (!SlateApp.CanDisplayWindows())
00137     {
00138         UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("CloseActiveWindow: Slate Application
cannot display windows.));
00139         return;
00140     }
00141
00142     TSharedPtr<SWindow> ActiveWindow = SlateApp.GetActiveTopLevelWindow();
00143     if (!ActiveWindow.IsValid())
00144     {
00145         UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("CloseActiveWindow: No Active Window
Found.));
00146         return;
00147     }
00148
00149     TSharedPtr<SWindow> RootWindow = FGlobalTabmanager::Get()->GetRootWindow();
00150     if (ActiveWindow->IsVisible() && ActiveWindow != RootWindow)
00151     {
00152         ActiveWindow->RequestDestroyWindow();
00153     }
00154 }
```

## 4.65.3.3 SelectToolBarItem()

```

void UWindowInteractionLibrary::SelectToolBarItem (
    FParseRecord & Record )
```

Selects the Item from the Active Windows ToolBar.

## Parameters

|               |  |
|---------------|--|
| <i>Record</i> | The Parse Record accumulated until this Event. |
|---------------|--|

Definition at line 156 of file [WindowInteractionLibrary.cpp](#).

```

00157 {
00158     UParseIntInput* ItemIndexInput = Record.GetPhraseInput<UParseIntInput>(TEXT("ITEM_INDEX"));
00159     if (ItemIndexInput == nullptr)
00160     {
00161         UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("SelectToolBarItem: No Item Index
Found.));
00162         return;
00163     }
00164
00165     WindowToolBar->SelectToolbarItem(ItemIndexInput->GetValue());
00166 }
```

## 4.65.3.4 SwitchNextActiveWindow()

```

void UWindowInteractionLibrary::SwitchNextActiveWindow (
    FParseRecord & Record )
```

Selects the Next Window in the Slate Application, and switches the Focus to it.

## Parameters

|               |  |
|---------------|--|
| <i>Record</i> | The Parse Record accumulated until this Event. |
|---------------|--|

Definition at line 88 of file [WindowInteractionLibrary.cpp](#).

```

00089 {
00090     GET_ACTIVE_REGULAR_WINDOW(ActiveWindow)
00091
00092     TArray<TSharedRef<SWindow>> AllWindows;
00093     FSlateApplication::Get().GetAllVisibleWindowsOrdered(AllWindows);
00094
00095     int32 FoundIndex;
00096     if (!AllWindows.Find(ActiveWindow.ToSharedRef(), FoundIndex))
00097     {
00098         UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("SwitchNextActiveWindow: Cannot Find the
Current Active Window."))
00099         return;
00100     }
00101
00102     TSharedRef<SWindow> NextWindow = AllWindows[FoundIndex + 1 % AllWindows.Num()];
00103
00104     NextWindow->BringToFront(true);
00105
00106     // Set Window Major Tab Focus?
00107 }
```

#### 4.65.3.5 SwitchNextTabInStack()

```

void UWindowInteractionLibrary::SwitchNextTabInStack (
    FParseRecord & Record )
```

Selects the Next Tab In The Active Tab Stack, and switches the Focus to it.

## Parameters

|               |  |
|---------------|--|
| <i>Record</i> | The Parse Record accumulated until this Event. |
|---------------|--|

Definition at line 241 of file [WindowInteractionLibrary.cpp](#).

```

00242 {
00243     GET_ACTIVE_TAB(ActiveTab);
00244
00245     TSharedPtr<FTabManager> ActiveTabManager = ActiveTab->GetTabManagerPtr();
00246     if (!ActiveTabManager.IsValid())
00247     {
00248         UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("SwitchNextTabInStack: Cannot Find
Active Tab Manager"))
00249         return;
00250     }
00251
00252     TArray<FTabManager::FTab> FoundTabs =
TabUtils::CollectManagedTabs(ActiveTabManager.ToSharedRef());
00253
00254     const FTabId ActiveTabId = ActiveTab->GetLayoutIdentifier();
00255     for (int32 i = 0; i < FoundTabs.Num(); i++)
00256     {
00257         if (FoundTabs[i].TabId == ActiveTabId)
00258         {
00259             TSharedPtr<SDockTab> NextTabWidget = TSharedPtr<SDockTab>();
00260
00261             int CurrentTabIndex = i;
00262             while (!NextTabWidget.IsValid())
00263             {
00264                 CurrentTabIndex += 1;
00265                 CurrentTabIndex %= FoundTabs.Num();
00266
00267                 const FTabManager::FTab NextTab = FoundTabs[
CurrentTabIndex
00268             ];
00269
00270 }
```

```

00271         if (NextTab.TabId == ActiveTabId)
00272         {
00273             UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("SwitchNextTabInStack: No
Next Tab Found."))
00274             return;
00275         }
00276         NextTabWidget = ActiveTabManager->FindExistingLiveTab(
00277             NextTab.TabId
00278         );
00279     };
00280 }
00281
00282     FGlobalTabmanager::Get()->SetActiveTab(NextTabWidget);
00283     NextTabWidget->ActivateInParent(SetDirectly);
00284
00285     break;
00286 }
00287 }
00288
00289 /*
00290 // Most Straightforward Implementation, But Requires Private Core Tab Classes
00291
00292 TSharedPtr<SDockingTabStack> ActiveTabStack = ActiveTab->GetParentDockTabStack();
00293 if (!ActiveTabStack.IsValid())
00294 {
00295     UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("SwitchNextTabInStack: No Parent Tab
Stack Found."))
00296     return;
00297 }
00298
00299 TArray<TSharedPtr<SDockTab>> AllTabs = ActiveTabStack->GetAllChildTabs();
00300
00301 int32 FoundIndex;
00302 if (!AllTabs.Find(ActiveTab.ToSharedRef(), FoundIndex))
00303 {
00304     UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("SwitchNextTabInStack: Active Tab Not
Found In Tab Stack."))
00305     return;
00306 }
00307
00308 TSharedPtr<SDockTab> NextTab = AllTabs[FoundIndex + 1 % AllTabs.Num()];
00309
00310 FGlobalTabmanager::Get()->SetActiveTab(NextTab);
00311 NextTab->ActivateInParent(SetDirectly);
00312 */
00313 }

```

#### 4.65.3.6 SwitchPrevActiveWindow()

```

void UWindowInteractionLibrary::SwitchPrevActiveWindow (
    FParseRecord & Record )

```

Selects the Previous Window in the Slate Application, and switches the Focus to it.

##### Parameters

|               |  |
|---------------|--|
| <b>Record</b> | The Parse Record accumulated until this Event. |
|---------------|--|

Definition at line 109 of file [WindowInteractionLibrary.cpp](#).

```

00110 {
00111     GET_ACTIVE_REGULAR_WINDOW(ActiveWindow)
00112
00113     TArray<TSharedPtr<SWindow>> AllWindows;
00114     FSlateApplication::Get().GetAllVisibleWindowsOrdered(AllWindows);
00115
00116     int32 FoundIndex;
00117     if (!AllWindows.Find(ActiveWindow.ToSharedRef(), FoundIndex))
00118     {
00119         UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("SwitchPrevActiveWindow: Cannot Find the
Current Active Window."))
00120         return;
00121     }
00122 }

```

```

00123     TSharedRef<SWindow> PrevWindow = AllWindows[
00124         FoundIndex - 1 < 0
00125         ? AllWindows.Num() - 1
00126         : FoundIndex - 1
00127     ];
00128
00129     PrevWindow->BringToFront(true);
00130
00131     // Set Window Major Tab Focus?
00132 }

```

#### 4.65.3.7 SwitchPrevTabInStack()

```

void UWindowInteractionLibrary::SwitchPrevTabInStack (
    FParseRecord & Record )

```

Selects the Prev Tab In The Active Tab Stack, and switches the Focus to it.

##### Parameters

|               |  |
|---------------|--|
| <i>Record</i> | The Parse Record accumulated until this Event. |
|---------------|--|

Definition at line 315 of file [WindowInteractionLibrary.cpp](#).

```

00316 {
00317     GET_ACTIVE_TAB(ActiveTab);
00318
00319     TSharedPtr<FTabManager> ActiveTabManager = ActiveTab->GetTabManagerPtr();
00320     if (!ActiveTabManager.IsValid())
00321     {
00322         UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("SwitchPrevTabInStack: Cannot Find
Active Tab Manager"))
        return;
00323     }
00324
00325     TArray<FTabManager::FTab> FoundTabs =
00326     TabUtils::CollectManagedTabs(ActiveTabManager.ToSharedRef());
00327
00328     const FTabId ActiveTabId = ActiveTab->GetLayoutIdentifier();
00329     for (int32 i = 0; i < FoundTabs.Num(); i++)
00330     {
00331         if (FoundTabs[i].TabId == ActiveTabId)
00332         {
00333             TSharedPtr<SDockTab> PrevTabWidget = TSharedPtr<SDockTab>();
00334
00335             int CurrentTabIndex = i;
00336             while (!PrevTabWidget.IsValid())
00337             {
00338                 CurrentTabIndex -= 1;
00339                 if (CurrentTabIndex < 0)
00340                 {
00341                     CurrentTabIndex = FoundTabs.Num() - 1;
00342                 }
00343
00344                 const FTabManager::FTab NextTab = FoundTabs[
CurrentTabIndex
00345             ];
00346
00347             if (NextTab.TabId == ActiveTabId)
00348             {
00349                 UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("SwitchNextTabInStack: No
Next Tab Found.))
                return;
00350             }
00351
00352             PrevTabWidget = ActiveTabManager->FindExistingLiveTab(
NextTab.TabId
00353             );
00354
00355             FGlobalTabmanager::Get()->SetActiveTab(PrevTabWidget);
00356             PrevTabWidget->ActivateInParent(SetDirectly);
00357
00358             break;
00359
00360             break;
00361
00362

```

```

00363     }
00364 }
00365
00366 /*
00367 // Most Straightforward Implementation, But Requires Private Core Tab Classes
00368
00369 TSharedPtr<SDockingTabStack> ActiveTabStack = ActiveTab->GetParentDockTabStack();
00370 if (!ActiveTabStack.IsValid())
00371 {
00372     UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("SwitchPrevTabInStack: No Parent Tab
Stack Found."))
00373     return;
00374 }
00375
00376 TArray<TSharedPtr<SDockTab>> AllTabs = ActiveTabStack->GetAllChildTabs();
00377
00378 int32 FoundIndex;
00379 if (!AllTabs.Find(ActiveTab.ToSharedRef(), FoundIndex))
00380 {
00381     UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("SwitchPrevTabInStack: Active Tab Not
Found In Tab Stack."))
00382     return;
00383 }
00384
00385 TSharedPtr<SDockTab> PrevTab = AllTabs[
00386     FoundIndex - 1 < 0
00387     ? AllTabs.Num() - 1
00388     : FoundIndex - 1
00389 ];
00390
00391 FGlobalTabmanager::Get()->SetActiveTab(PrevTab);
00392 PrevTab->ActivateInParent(SetDirectly);
00393 */
00394 }

```

## 4.65.4 Member Data Documentation

### 4.65.4.1 WindowToolBar

class [UAccessibilityWindowToolBar](#)\* UWindowInteractionLibrary::WindowToolBar [protected]

Definition at line 86 of file [WindowInteractionLibrary.h](#).

The documentation for this class was generated from the following files:

- Source/OpenAccessibility/Public/PhraseEvents/WindowInteractionLibrary.h
- Source/OpenAccessibility/Private/PhraseEvents/WindowInteractionLibrary.cpp

## 4.66 OpenAccessibilityPy.WhisperInterface.WhisperInterface Class Reference

### Public Member Functions

- def [\\_\\_init\\_\\_](#) (self, str model\_name="distil-small.en", str device="auto", int cpu\_threads=4, int transcribe\_workers=2, str compute\_type="default") ↩
- def [\\_\\_del\\_\\_](#) (self)
- def [process\\_file\\_from\\_dir](#) (self, str filepath)
- tuple[list[Segment], dict] [process\\_audio\\_buffer](#) (self, np.ndarray audio\_buffer)

## Public Attributes

- [whisper\\_model](#)
- [beam\\_size](#)

### 4.66.1 Detailed Description

Interface Class for Interacting with the CTranslate2 Faster Whisper Model.

Definition at line 13 of file [WhisperInterface.py](#).

### 4.66.2 Constructor & Destructor Documentation

#### 4.66.2.1 \_\_init\_\_()

```
def OpenAccessibilityPy.WhisperInterface.WhisperInterface.__init__ (
    self,
    str  model_name = "distil-small.en",
    str  device = "auto",
    int  cpu_threads = 4,
    int  transcribe_workers = 2,
    str  compute_type = "default" )
```

Constructor of Whisper Interface Class

Args:

model\_name (str, optional): Hugging-Face Model Specifier for CTranslate Compatible Models. Defaults to "distil-small.en".  
 device (str, optional): Device host for the Whisper Model (Can be "auto", "cpu", "cuda"). Defaults to "auto".  
 cpu\_threads (int, optional): Amount of CPU Threads to use, if Hosting the Model on a CPU. Defaults to 4.  
 transcribe\_workers (int, optional): Amount of Thread Workers for Audio Transcription. Defaults to 2.  
 compute\_type (str, optional): Data Structure for Whisper Compute. Defaults to "default".

Definition at line 16 of file [WhisperInterface.py](#).

```
00023     ):
00024         """Constructor of Whisper Interface Class
00025
00026         Args:
00027             model_name (str, optional): Hugging-Face Model Specifier for CTranslate Compatible Models.
00028             Defaults to "distil-small.en".
00029             device (str, optional): Device host for the Whisper Model (Can be "auto", "cpu", "cuda").
00030             Defaults to "auto".
00031             cpu_threads (int, optional): Amount of CPU Threads to use, if Hosting the Model on a CPU.
00032             Defaults to 4.
00033             transcribe_workers (int, optional): Amount of Thread Workers for Audio Transcription.
00034             Defaults to 2.
00035             compute_type (str, optional): Data Structure for Whisper Compute. Defaults to "default".
00036
00037         """
00038         # Whisper Focused Variables
00039         self.whisper_model = WhisperModel(
00040             model_name,
00041             device=device,
00042             compute_type=compute_type,
00043             num_workers=transcribe_workers,
00044             cpu_threads=cpu_threads,
00045             local_files_only=True,
00046         )
00047         self.beam_size = 5
```

### 4.66.2.2 `__del__()`

```
def OpenAccessibilityPy.WhisperInterface.WhisperInterface.__del__ (
    self )
```

Destructor of Whisper Interface Class.

Definition at line 45 of file [WhisperInterface.py](#).

```
00045     def __del__(self):
00046         """Destructor of Whisper Interface Class."""
00047
00048         del self.whisper_model
00049
```

## 4.66.3 Member Function Documentation

### 4.66.3.1 `process_audio_buffer()`

```
tuple[list[Segment], dict] OpenAccessibilityPy.WhisperInterface.WhisperInterface.process_↵
audio_buffer (
    self,
    np.ndarray audio_buffer )
```

Transcribes an NDArray AudioBuffer.

Args:

audio\_buffer (np.ndarray): AudioBuffer to Transcribe.

Returns:

tuple[list[Segment], dict]: Tuple Containing a List of Transcription Segments and a Dictionary of Collected Metadata.

Definition at line 80 of file [WhisperInterface.py](#).

```
00082     ) -> tuple[list[Segment], dict]:
00083         """Transcribes an NDArray AudioBuffer.
00084
00085         Args:
00086             audio_buffer (np.ndarray): AudioBuffer to Transcribe.
00087
00088         Returns:
00089             tuple[list[Segment], dict]: Tuple Containing a List of Transcription Segments and a
00090             Dictionary of Collected Metadata.
00091
00092             segments, info = self.whisper_model.transcribe(
00093                 audio_buffer,
00094                 beam_size=self.beam_size,
00095                 language="en",
00096             )
00097
00098             Log(
00099                 f"WhisperInterface || Detected Language: {info.language} | Probability:
00100                 {info.language_probability} | Duration: {info.duration}"
00101             )
00102
00103             collected_metadata = {
00104                 "duration": info.duration,
00105                 "language": info.language,
00106                 "language_probability": info.language_probability,
00107             }
00108
00109             return list(segments), collected_metadata
```

### 4.66.3.2 process\_file\_from\_dir()

```
def OpenAccessibilityPy.WhisperInterface.WhisperInterface.process_file_from_dir (
    self,
    str filepath )
```

Transcribes an Audio Files from a Given WAV File Path.

Args:

filepath (str): Path to the Audio Files to Transcribe.

Returns:

A List of Segments containing the Transcribed Text and their Time Stamps.

Definition at line 50 of file [WhisperInterface.py](#).

```
00050     def process_file_from_dir(self, filepath: str):
00051         """Transcribes an Audio Files from a Given WAV File Path.
00052
00053         Args:
00054             filepath (str): Path to the Audio Files to Transcribe.
00055
00056         Returns:
00057             A List of Segments containing the Transcribed Text and their Time Stamps.
00058         """
00059
00060         segments, info = self.whisper_model.transcribe(
00061             filepath,
00062             beam_size=self.beam_size,
00063             language="en",
00064             prepend_punctuations="",
00065             append_punctuations="",
00066             vad_filter=True,
00067         )
00068
00069         Log(
00070             f"WhisperInterface | Detected Language: {info.language} | Probability:
00071             {info.language_probability} | Duration: {info.duration}"
00072         )
00073         for segment in segments:
00074             Log(
00075                 f"WhisperInterface | Segment : {segment.text} | Start: {segment.start} | End:
00076                 {segment.end}"
00077             )
00078         return list(segments)
00079
```

## 4.66.4 Member Data Documentation

### 4.66.4.1 beam\_size

OpenAccessibilityPy.WhisperInterface.WhisperInterface.beam\_size

Definition at line 43 of file [WhisperInterface.py](#).

### 4.66.4.2 whisper\_model

OpenAccessibilityPy.WhisperInterface.WhisperInterface.whisper\_model

Definition at line 35 of file [WhisperInterface.py](#).

The documentation for this class was generated from the following file:

- Content/Python/OpenAccessibilityPy/WhisperInterface.py



## Chapter 5

# File Documentation

### 5.1 init\_unreal.py

```
00001 import unreal
00002
00003 import subprocess
00004 import pkg_resources
00005 from os import path
00006
00007
00008 def get_requirements(requirements_dir: str) -> list[str]:
00009     with open(
00010         path.join(requirements_dir, "requirements.txt"), "r"
00011     ) as requirements_file:
00012         return [line.strip() for line in requirements_file.readlines()]
00013
00014
00015 def is_dependency_satisfied(dependency: str) -> bool:
00016     try:
00017         pkg_resources.require(dependency)
00018         return True
00019     except:
00020         return False
00021
00022
00023 def install_dependencies(deps_to_install: list):
00024
00025     unreal.log_warning(
00026         f"|| OpenAccessibility Python || Installing Dependencies: {deps_to_install} ||"
00027     )
00028
00029     with unreal.ScopedSlowTask(
00030         len(deps_to_install),
00031         "OpenAccessibility Installing Python Dependencies",
00032         enabled=True,
00033     ) as install_ui:
00034         process = subprocess.Popen(
00035             [
00036                 unreal.get_interpreter_executable_path(),
00037                 "-m",
00038                 "pip",
00039                 "install",
00040             ]
00041             + deps_to_install,
00042             shell=True,
00043             stdin=subprocess.PIPE,
00044             stdout=subprocess.PIPE,
00045             stderr=None,
00046         )
00047
00048         while process.poll() is None:
00049             install_ui.enter_progress_frame(
00050                 0, process.stdout.readline().decode("utf-8")
00051             )
00052
00053         process.wait()
00054
00055
00056
00057
00058 unreal.log("|| OpenAccessibility Python || Initializing ||")
```

```

00059
00060
00061
00062 # Verify Required Dependencies
00063
00064 missing_deps = [
00065     dep
00066     for dep in get_requirements(path.dirname(path.realpath(__file__)))
00067     if not is_dependency_satisfied(dep)
00068 ]
00069
00070 if missing_deps:
00071     install_dependencies(missing_deps)
00072
00073
00074
00075 # Initialize the Python Runtime
00076
00077 unreal.log("|| OpenAccessibility Python || Starting Python Runtime ||")
00078
00079 import OpenAccessibilityPy as OAPy
00080
00081 # Run Utilities for Better Project Runtime.
00082
00083 # Helps Circumvent CUDA and CUDNN Issues
00084 # during the inference process with the Whisper Model.
00085 # OAPy.forward_CUDA_CUDNN_to_path()
00086
00087 # Initialize the Runtime
00088 OpenAccessibilityPy = OAPy.OpenAccessibilityPy()
00089
00090

```

## 5.2 \_\_init\_\_.py

```

00001 import unreal as ue
00002 import zmq
00003 import numpy as np
00004 from gc import collect as gc_collect
00005
00006 from concurrent.futures import ThreadPoolExecutor as ThreadPool
00007
00008 from .CommunicationServer import CommunicationServer
00009 from .WhisperInterface import WhisperInterface
00010 from .Audio import AudioResampler
00011 from .Logging import Log, LogLevel
00012
00013 from .LibUtils import (
00014     get_filtered_path_list,
00015     get_child_directories,
00016     append_paths_to_library_path,
00017 )
00018
00019
00020 def forward_CUDA_CUDNN_to_path():
00021     """
00022     Forces any found CUDA and CUDNN Paths to the System Path.
00023
00024     This is useful for circumventing issues with CUDA and CUDNN not being found on the embedded
    interpreter.
00025     Not always needed, but useful for some systems.
00026     """
00027
00028     filtered_path_list = get_filtered_path_list(["CUDA", "CUDNN"])
00029
00030     for path in filtered_path_list:
00031         append_paths_to_library_path(get_child_directories(path, depth=1))
00032
00033
00034 class OpenAccessibilityPy:
00035     """Python Runtime Class for Open Accessibility Plugin"""
00036
00037     def __init__(
00038         self,
00039         # General Runtime Specifics
00040         worker_count: int = 2,
00041         # Whisper Specifics
00042         whisper_model: str = "distil-small.en",
00043         device: str = "auto",
00044         compute_type: str = "default",
00045         # Communication Specifics
00046         poll_timeout: int = 10,
00047     ):

```

```

00048         """Constructor of Python Runtime Class for Open Accessibility Plugin
00049
00050         Args:
00051             worker_count (int, optional): Amount of Thread Workers for Audio Transcription. Defaults
to 2.
00052             whisper_model (str, optional): Hugging-Face Model Specifier for CTranslate Compatible
Models. Defaults to "distil-small.en".
00053             device (str, optional): Device host for the Whisper Model (Can be "auto", "cpu", "cuda").
Defaults to "auto".
00054             compute_type (str, optional): Data Structure for Whisper Compute. Defaults to "default".
00055             poll_timeout (int, optional): Amount of time (ms) for event polling on the Transcription
Socket. Defaults to 10.
00056         """
00057
00058         self.worker_pool = ThreadPool(
00059             max_workers=worker_count, thread_name_prefix="TranscriptionWorker"
00060         )
00061
00062         self.whisper_interface = WhisperInterface(
00063             model_name=whisper_model,
00064             device=device,
00065             compute_type=compute_type,
00066             transcribe_workers=worker_count,
00067         )
00068         self.com_server = CommunicationServer(
00069             send_socket_type=zmq.PUSH,
00070             recv_socket_type=zmq.PULL,
00071             poll_timeout=poll_timeout,
00072         )
00073         self.audio_resampler = AudioResampler(target_sample_rate=16000)
00074
00075         self.tick_handle = ue.register_slate_post_tick_callback(self.Tick)
00076
00077         self.pyshutdown_handle = ue.register_python_shutdown_callback(self.Shutdown)
00078
00079     def __del__(self):
00080         """Destructor of Python Runtime Class for Open Accessibility Plugin"""
00081
00082         self.Shutdown()
00083
00084     def Tick(self, delta_time: float):
00085         """Tick Callback for Unreal Engine Slate Post Tick.
00086
00087         Detecting Incoming Transcription Requests and Handling them, through the Worker Pool.
00088
00089         Args:
00090             delta_time (float): Time since last tick
00091         """
00092
00093         if self.com_server.EventOccured():
00094             Log("Event Occured")
00095
00096             message, metadata = self.com_server.ReceiveNDArrayWithMeta(dtype=np.float32)
00097
00098             self.worker_pool.submit(self.HandleTranscriptionRequest, message, metadata)
00099
00100     def HandleTranscriptionRequest(
00101         self, recv_message: np.ndarray, metadata: dict = None
00102     ):
00103         """Handles Incoming Transcription Requests
00104
00105         Takes the Incoming AudioBuffer, Resamples it to 16kHz and Transcribes it using Whisper.
00106
00107         Args:
00108             recv_message (np.ndarray): ndarray of the incoming audio buffer.
00109             metadata (dict, optional): metadata of the incoming audio buffer, if any is recieved.
Defaults to None.
00110         """
00111
00112         Log(
00113             f"Handling Transcription Request | Message: {recv_message} | Size: {recv_message.size} |
Shape: {recv_message.shape}"
00114         )
00115
00116         sample_rate = metadata.get("sample_rate", 48000)
00117         num_channels = metadata.get("num_channels", 2)
00118
00119         message_ndarray = self.audio_resampler.resample(
00120             recv_message, sample_rate, num_channels
00121         )
00122
00123         trans_segments, trans_metadata = self.whisper_interface.process_audio_buffer(
00124             message_ndarray
00125         )
00126
00127         encoded_segments = [
00128             transcription.text.encode() for transcription in trans_segments

```

```

00129         ]
00130
00131         Log(f"Encoded Segments: {encoded_segments}")
00132
00133         if len(encoded_segments) > 0:
00134             try:
00135                 self.com_server.SendMultipartWithMeta(
00136                     message=encoded_segments, meta=trans_metadata
00137                 )
00138
00139             except:
00140                 Log("Error Sending Encoded Transcription Segments", LogLevel.ERROR)
00141
00142         else:
00143             Log("No Transcription Segments Returned", LogLevel.WARNING)
00144
00145     def Shutdown(self):
00146         """Shutsdown the Python Runtime Components, and Forces a Garbage Collection."""
00147
00148         if self.tick_handle:
00149             ue.unregister_slate_post_tick_callback(self.tick_handle)
00150             del self.tick_handle
00151
00152         if self.worker_pool:
00153             self.worker_pool.shutdown(wait=False, cancel_futures=True)
00154             del self.worker_pool
00155
00156         if self.audio_resampler:
00157             del self.audio_resampler
00158
00159         if self.com_server:
00160             del self.com_server
00161
00162         if self.whisper_interface:
00163             del self.whisper_interface
00164
00165         # Force a Garbage Collection
00166         gc.collect()

```

## 5.3 \_\_main\_\_.py

```

00001 import numpy as np
00002 from zmq import PUSH as zmq_PUSH, PULL as zmq_PULL
00003
00004 from faster_whisper.transcribe import decode_audio
00005
00006 from CommunicationServer import CommunicationServer
00007 from WhisperInterface import WhisperInterface
00008 from Audio import AudioResampler
00009 import LibUtils
00010
00011 from Logging import Log, LogLevel
00012
00013
00014 PERFORM_COMPARE = False
00015
00016
00017 def PlotAudioBuffers(
00018     recv_audio_buffer: np.ndarray,
00019     decoded_audio_buffer: np.ndarray,
00020     name: str = "BufferComparison",
00021 ):
00022     """
00023     Plots the received audio buffer and the decoded audio buffer to compare the two.
00024     """
00025
00026     try:
00027         import matplotlib as mpl
00028         from matplotlib import pyplot as plt
00029
00030         mpl.interactive(False)
00031
00032         fig, axs = plt.subplots(3)
00033
00034         axs[0].plot(recv_audio_buffer)
00035
00036         axs[1].plot(decoded_audio_buffer)
00037
00038         axs[2].plot(recv_audio_buffer)
00039         axs[2].plot(decoded_audio_buffer)
00040         axs[2].set_title("Buffer Comparison")
00041
00042         fig.savefig(

```

```

00043         f"D:/dev/Unreal Engine/AccessibilityProject/Saved/Debug/OpenAccessibility/{name}.png",
00044         dpi=300,
00045     )
00046     fig.clear()
00047
00048     except Exception as e:
00049         Log(f"Error Plotting Audio Buffers: {e}", LogLevel.ERROR)
00050
00051
00052
00053 def main():
00054
00055     whisper_interface = WhisperInterface("distil-small.en")
00056     com_server = CommunicationServer(
00057         send_socket_type=zmq_PUSH, recv_socket_type=zmq_PULL, poll_timeout=10
00058     )
00059     audio_resampler = AudioResampler(target_sample_rate=16000)
00060
00061     should_run = True
00062
00063     print("Starting Run Loop")
00064
00065     while should_run:
00066
00067         if com_server.EventOccured():
00068             Log("Event Occured")
00069
00070             recv_message, metadata = com_server.ReceiveNDArrayWithMeta()
00071
00072             message_ndarray: np.ndarray = np.frombuffer(recv_message, dtype=np.float32)
00073
00074             sample_rate = metadata.get("sample_rate", 48000)
00075             num_channels = metadata.get("num_channels", 1)
00076
00077             if PERFORM_COMPARE:
00078                 decoded_ndarray = decode_audio(
00079                     "D:/dev/Unreal
Engine/AccessibilityProject/Saved/BouncedWavFiles/OpenAccessibility/Audioclips/Captured_User_Audio.wav",
00080                     sampling_rate=16000,
00081                 )
00082
00083                 PlotAudioBuffers(message_ndarray, decoded_ndarray)
00084
00085                 isSame = np.array_equal(message_ndarray, decoded_ndarray)
00086                 # isClose = np.allclose(message_ndarray, decoded_ndarray)
00087
00088                 # difference = np.subtract(message_ndarray, decoded_ndarray)
00089
00090                 Log(
00091                     f"Recieved Buffer | {message_ndarray} | Shape: {message_ndarray.shape}"
00092                 )
00093
00094                 Log(
00095                     f"Decoded Buffer | {decoded_ndarray} | Shape: {decoded_ndarray.shape}"
00096                 )
00097                 Log(f"Comparisons | Is Same: {isSame}")
00098
00099                 # Apply Resampling to the Audio Buffer, to match samplerate of 16000Hz
00100                 message_ndarray = audio_resampler.resample(message_ndarray, sample_rate)
00101
00102             if PERFORM_COMPARE:
00103                 PlotAudioBuffers(
00104                     message_ndarray, decoded_ndarray, name="ResampledBufferComparison"
00105                 )
00106
00107             transcription_segments, metadata = whisper_interface.process_audio_buffer(
00108                 message_ndarray
00109             )
00110
00111             encoded_segments = [
00112                 transcription.text.encode() for transcription in transcription_segments
00113             ]
00114
00115             mock_encoded_segments = [
00116                 "VIEW NODE 0".encode(),
00117                 "NODE 0 MOVE UP 50".encode(),
00118             ]
00119
00120             Log(f"Encoded Segments: {encoded_segments}")
00121             Log(f"Encoded Mock Segments: {mock_encoded_segments}")
00122
00123             if len(encoded_segments) > 0:
00124                 try:
00125                     com_server.SendMultipartWithMeta(encoded_segments, metadata)
00126                 except:
00127                     Log("Error Sending Encoded Transcription Segments", LogLevel.ERROR)
00128             else:

```

```

00129             Log("No Transcription Segments Returned", LogLevel.WARNING)
00130
00131
00132 if __name__ == "__main__":
00133     main()

```

## 5.4 Audio.py

```

00001 import gc
00002 from itertools import chain as iter_chain
00003 from multiprocessing import Lock
00004
00005 import numpy as np
00006 import av
00007
00008
00009 try:
00010     from .Logging import Log
00011 except ImportError:
00012     from Logging import Log
00013
00014
00015 class AudioResampler:
00016     """Audio Resampler for Resampling Incoming Audio to the Target Sample Rate. Using FFmpeg for
    Resampling."""
00017
00018     def __init__(self, target_sample_rate: int = 16000):
00019         """Constructor of Audio Resampler Class
00020
00021         Args:
00022             target_sample_rate (int, optional): The Target for all incoming resampling requests.
    Defaults to 16000 (Required by Whisper).
00023         """
00024
00025         self._audio_resampler = av.AudioResampler(
00026             format="sl6", layout="mono", rate=target_sample_rate
00027         )
00028         self._resample_mutex = Lock()
00029
00030     def __del__(self):
00031         """Destructor of Audio Resampler Class.
00032
00033         Ensures PyAV Resampler Object is Properly Deleted, calling Garbage Collection in the process.
00034         """
00035
00036         # Try Deleting the resampler object to cleanly free up memory
00037         try:
00038             del self._audio_resampler
00039         except:
00040             pass
00041
00042         try: # Delete the mutex
00043             del self._resample_mutex
00044         except:
00045             pass
00046
00047         # Force Garbage Collection, due to resampler not being properly deleted otherwise.
00048         gc.collect()
00049
00050     def resample(
00051         self,
00052         audio_data: np.ndarray,
00053         buffer_sample_rate: int = 48000,
00054         buffer_num_channels: int = 2,
00055     ) -> np.ndarray:
00056         """Resamples the Incoming Audio Data to the Classes Assigned Target Sample Rate.
00057
00058         Args:
00059             audio_data (np.ndarray): Audio Data to Resample.
00060             buffer_sample_rate (int, optional): Sample Rate of the Incoming Audio Data. Defaults to
    48000.
00061             buffer_num_channels (int, optional): Number of Channels in the Incoming Audio Data.
    Defaults to 2 (Stereo).
00062
00063         Returns:
00064             np.ndarray: Resampled Version of the Incoming Audio Data.
00065         """
00066
00067         audio_data = self._convert_to_sl6(audio_data).reshape(-1, 1)
00068
00069         frame: av.AudioFrame = av.AudioFrame.from_ndarray(
00070             audio_data.T,
00071             format="sl6",

```

```

00072         layout="stereo" if buffer_num_channels == 2 else "mono",
00073     )
00074
00075     frame.sample_rate = buffer_sample_rate
00076
00077     resampled_frames: list[av.AudioFrame] = []
00078     with self._resample_mutex:
00079         resampled_frames = self._audio_resampler.resample(frame)
00080
00081     return self._convert_to_float32(resampled_frames[0].to_ndarray()).reshape(
00082         -1,
00083     )
00084
00085 def _resample_frame(self, frame: av.AudioFrame) -> list[av.AudioFrame]:
00086     """Resamples an AudioFrame to the target sample rate.
00087
00088     Args:
00089         frame (av.AudioFrame): An AudioFrame to resample.
00090
00091     Returns:
00092         list[av.AudioFrame]: A List of Resampled AudioFrames generated from the input frame,
00093     """
00094     with self._resample_mutex:
00095         return self._audio_resampler.resample(frame)
00096
00097 def _resample_frames(self, frames: list[av.AudioFrame]):
00098     """Resamples an array of AudioFrames to the target sample rate.
00099
00100     Args:
00101         frames (list[av.AudioFrame]): An array of AudioFrames to resample.
00102
00103     Yields:
00104         An Array of Generators for the Resampled AudioFrames from the frame inputs.
00105     """
00106
00107     for frame in iter_chain(frames, [None]):
00108         yield from self._audio_resampler.resample(frame)
00109
00110 def _convert_to_float32(self, audio_data: np.ndarray) -> np.ndarray:
00111     """Converts the provided audio data to float32 format.
00112
00113     Args:
00114         audio_data (np.ndarray): The audio data to convert.
00115
00116     Raises:
00117         ValueError: If the data type is not supported.
00118
00119     Returns:
00120         np.ndarray: The Input Audio Data in float32 format.
00121     """
00122
00123     if audio_data.dtype == np.float32:
00124         return audio_data
00125
00126     elif audio_data.dtype == np.int16:
00127         return audio_data.astype(np.float32) / 32768.0
00128
00129     else:
00130         raise ValueError("Unsupported data type")
00131
00132 def _convert_to_s16(self, audio_data: np.ndarray) -> np.ndarray:
00133     """Converts the provided audio data to int16 format.
00134
00135     Args:
00136         audio_data (np.ndarray): The audio data to convert.
00137
00138     Raises:
00139         ValueError: If the data type is not supported.
00140
00141     Returns:
00142         np.ndarray: The Input Audio Data in int16 format.
00143     """
00144
00145     if audio_data.dtype == np.int16:
00146         return audio_data
00147
00148     elif audio_data.dtype == np.float32:
00149         return (audio_data * 32768.0).astype(np.int16)
00150
00151     else:
00152         raise ValueError("Unsupported data type")

```

## 5.5 CommunicationServer.py

```

00001 import numpy as np
00002 import json
00003 import zmq
00004
00005 try:
00006     from .Logging import Log, LogLevel
00007 except ImportError:
00008     from Logging import Log, LogLevel
00009
00010
00011 class CommunicationServer:
00012     """Communication Server Class for Handling Communication Between Python and C++.
00013
00014     Using ZeroMQ for Socket Communication. (Push / PULL Architecture)
00015     """
00016
00017     def __init__(
00018         self,
00019         send_socket_type: int,
00020         recv_socket_type: int,
00021         send_socket_addr: str = "tcp://127.0.0.1:5556",
00022         recv_socket_addr: str = "tcp://127.0.0.1:5555",
00023         poll_timeout: int = 10,
00024     ):
00025         """Constructor of Communication Server Class
00026
00027         Args:
00028             send_socket_type (int): ZeroMQ Socket Type for Sending Messages.
00029             recv_socket_type (int): ZeroMQ Socket Type for Receiving Messages.
00030             send_socket_addr (str, optional): Local Address / Port for Sending Communication Data.
00031             Defaults to "tcp://127.0.0.1:5556".
00032             recv_socket_addr (str, optional): Local Address / Port for Receiving Communication Data.
00033             Defaults to "tcp://127.0.0.1:5555".
00034             poll_timeout (int, optional): Amount of time (ms) for event polling on the Receive Socket.
00035             Defaults to 10.
00036         """
00037
00038         # Create the Context
00039         self.context = zmq.Context()
00040
00041         # Create a Socket
00042         self.send_socket: zmq.Socket = self.context.socket(send_socket_type)
00043         self.send_socket_context = self.send_socket.connect(send_socket_addr)
00044
00045         self.recv_socket = self.context.socket(recv_socket_type)
00046         self.recv_socket_context = self.recv_socket.bind(recv_socket_addr)
00047
00048         self.poller = zmq.Poller()
00049         self.poller.register(self.recv_socket, zmq.POLLIN)
00050         self.poller_timeout_time = poll_timeout
00051
00052     def __del__(self):
00053         """Destructor of Communication Server Class.
00054
00055         Closes the Sockets and Terminates the ZeroMQ Context.
00056         """
00057
00058         self.send_socket.close()
00059         self.recv_socket.close()
00060
00061         self.context.term()
00062
00063     def EventOccured(self) -> bool:
00064         """Checks if a Receive Event has Occured on the Receive Socket.
00065
00066         Returns:
00067             bool: True if an Event has Occured, False Otherwise.
00068         """
00069
00070         polled_events = dict(self.poller.poll(self.poller_timeout_time))
00071         if len(polled_events) > 0 and polled_events.get(self.recv_socket) == zmq.POLLIN:
00072             return True
00073         else:
00074             return False
00075
00076     def SendString(self, message: str) -> bool:
00077         """Sends a String Message on the Send Socket.
00078
00079         Args:
00080             message (str): String Message to Send.
00081
00082         Returns:
00083             bool: True if the Message was Sent Successfully, False Otherwise.
00084         """

```



```
00083         try:
00084             self.send_socket.send_string(message)
00085             return True
00086         except:
00087             Log("CommunicationServer | Error Sending String Message", LogLevel.WARNING)
00088             return False
00089
00090     def SendJSON(self, message: dict) -> bool:
00091         """Sends a JSON Message on the Send Socket.
00092
00093         Args:
00094             message (dict): Stringified JSON Message to Send.
00095
00096         Returns:
00097             bool: True if the Message was Sent Successfully, False Otherwise.
00098         """
00099
00100         try:
00101             self.send_socket.send_json(message)
00102             return True
00103         except:
00104             Log(
00105                 "CommunicationServer | Error Sending JSON Message",
00106                 LogLevel.WARNING,
00107             )
00108             return False
00109
00110     def SendNDArray(self, message: np.ndarray) -> bool:
00111         """Sends a Numpy NDArray Message on the Send Socket.
00112
00113         Args:
00114             message (np.ndarray): NDArray of Data to Send.
00115
00116         Returns:
00117             bool: True if the Data was Sent Successfully, False Otherwise.
00118         """
00119
00120         try:
00121             self.send_socket.send(message)
00122             return True
00123         except:
00124             Log(
00125                 "CommunicationServer | Error Sending NDArray Message",
00126                 LogLevel.WARNING,
00127             )
00128             return False
00129
00130     def SendNDArrayWithMeta(self, message: np.ndarray, meta: dict) -> bool:
00131         """Sends a Numpy NDArray Message with Metadata on the Send Socket.
00132
00133         Args:
00134             message (np.ndarray): NDArray of Data to Send.
00135             meta (dict): A Dictionary of Metadata to Send.
00136
00137         Returns:
00138             bool: True if the Data was Sent Successfully, False Otherwise.
00139         """
00140
00141         try:
00142             self.send_socket.send_multipart([json.dumps(meta).encode(), message.data])
00143
00144             return True
00145         except:
00146             Log(
00147                 "CommunicationServer | Error Sending NDArray With Meta Message",
00148                 LogLevel.WARNING,
00149             )
00150             return False
00151
00152     def SendMultipart(self, message: list) -> bool:
00153         """Sends a Multipart Message on the Send Socket.
00154
00155         Args:
00156             message (list): List of Messages to Send.
00157
00158         Returns:
00159             bool: True if the MultiPart Message was Sent Successfully, False Otherwise.
00160         """
00161
00162         try:
00163             self.send_socket.send_multipart(message)
00164             return True
00165         except:
00166             Log(
00167                 "CommunicationServer | Error Sending Multipart Message",
00168                 LogLevel.WARNING,
00169             )
```

```

00170         return False
00171
00172     def SendMultipartWithMeta(self, message: list, meta: dict) -> bool:
00173         """Sends a Multipart Message with Metadata on the Send Socket.
00174
00175         Args:
00176             message (list): List of Messages to Send.
00177             meta (dict): Metadata to Send.
00178
00179         Returns:
00180             bool: True if the MultiPart Message with Metadata was Sent Successfully, False Otherwise.
00181         """
00182
00183         try:
00184             self.send_socket.send_multipart([json.dumps(meta).encode(), *message])
00185             return True
00186         except:
00187             Log(
00188                 "CommunicationServer | Error Sending Multipart With Meta Message",
00189                 LogLevel.WARNING,
00190             )
00191             return False
00192
00193     def RecieveRaw(self):
00194         """Receives a Raw Message of Bytes from the Receive Socket.
00195
00196         Returns:
00197             bytes: Raw Received Bytes from the Receive Socket.
00198         """
00199
00200         return self.recv_socket.recv(zmq.DONTWAIT)
00201
00202     def ReceiveString(self) -> str:
00203         """Receives a String Message from the Receive Socket.
00204
00205         Returns:
00206             str: Received String Message.
00207         """
00208
00209         return self.recv_socket.recv_string(zmq.DONTWAIT)
00210
00211     def ReceiveJSON(self):
00212         """Receive a JSON Message from the Receive Socket.
00213
00214         Returns:
00215             dict: Dictionary of the Received JSON Message.
00216         """
00217
00218         return json.loads(self.recv_socket.recv_json(zmq.DONTWAIT))
00219
00220     def ReceiveNDArray(self, dtype=np.float32) -> np.ndarray:
00221         """Receives a Numpy NDArray from the Receive Socket.
00222
00223         Args:
00224             dtype (optional): Type of NDArray of Received Data. Defaults to np.float32.
00225
00226         Returns:
00227             np.ndarray: Receieved NDArray Message.
00228         """
00229
00230         return np.frombuffer(
00231             self.recv_socket.recv(zmq.DONTWAIT),
00232             dtype=dtype,
00233         )
00234
00235     def ReceiveNDArrayWithMeta(self, dtype=np.float32) -> tuple[np.ndarray, dict]:
00236         """Receives a Numpy NDArray with Metadata from the Receive Socket.
00237
00238         Args:
00239             dtype (optional): Type of NDArray of Received Data. Defaults to np.float32.
00240
00241         Returns:
00242             tuple[np.ndarray, dict]: Tuple of Received NDArray and Dict Metadata Object.
00243         """
00244
00245         recv_message = self.recv_socket.recv_multipart(zmq.DONTWAIT)
00246
00247         if len(recv_message) > 1:
00248             return (
00249                 np.frombuffer(recv_message[1], dtype=dtype),
00250                 json.loads(recv_message[0]),
00251             )
00252
00253         elif len(recv_message) == 1:
00254             Log(
00255                 "CommunicationServer | Error Receiving NDArray With Meta. Only Contains One Message,
Assumed Data.",

```

```

00256         LogLevel.WARNING,
00257     )
00258     return (np.frombuffer(recv_message[0], dtype=dtype), {})
00259
00260     Log("CommunicationServer | Error Receiving NDArray With Meta", LogLevel.WARNING)
00261
00262     def ReceiveMultipart(self) -> list[bytes]:
00263         """Receieved a Raw Multipart Message from the Receive Socket.
00264
00265         Returns:
00266             list[bytes]: Raw List of Bytes from the Received Multipart Message.
00267         """
00268
00269         return self.recv_socket.recv_multipart(zmq.DONTWAIT)

```

## 5.6 LibUtils.py

```

00001 import sys
00002 import os
00003
00004
00005 def append_paths_to_library_path(paths: list[str]):
00006     """Appends the given paths to the systems active library path.
00007
00008     Args:
00009         paths (list[str]): List of Paths to Append.
00010     """
00011
00012     sys.path.extend(paths)
00013
00014
00015 def get_path_list() -> list[str]:
00016     """Gets a list of paths in the PATH environment variable.
00017
00018     Returns:
00019         list[str]: _description_
00020     """
00021
00022     return os.getenv("PATH").split(";")
00023
00024
00025 def get_filtered_path_list(filter_list: list[str]) -> list[str]:
00026     """Gets a list of paths in the PATH environment variable that contain any of the given filters.
00027
00028     Args:
00029         filter_list (list[str]): List of Filter Strings to Search for in the PATH env.
00030
00031     Returns:
00032         list[str]: List of Found Paths.
00033     """
00034
00035     return [
00036         path for path in get_path_list() for filter in filter_list if filter in path
00037     ]
00038
00039
00040 def get_child_directories(path: str, depth: int = 0) -> list[str]:
00041     """
00042     Recursively searches the given directory, for any further child directories.
00043
00044     Args:
00045         path (str): The path to the directory to search.
00046         depth (int): The depth to search for child directories. Defaults to 0.
00047     """
00048
00049     assert os.path.isdir(path), f"Path: {path} is not a directory."
00050
00051     return [
00052         root
00053         for root, _, _ in os.walk(path, topdown=True)
00054         if root[len(path) :].count(os.sep) < depth
00055     ]

```

## 5.7 Logging.py

```

00001 from enum import Enum
00002
00003
00004 class LogLevel(Enum):

```

```

00005     INFO = 0
00006     WARNING = 1
00007     ERROR = 2
00008
00009
00010 def Log(message: str, log_level: LogLevel = LogLevel.INFO):
00011     """Logs a Message to the Unreal Engine Console.
00012
00013     Displays the given message in the Unreal Engine Console, with the given log level.
00014     If the Unreal Engine Python API is not available, the message is printed to the python terminal.
00015
00016     Args:
00017         message (str): Message to Log.
00018         log_level (LogLevel, optional): Log Level of the Message. Defaults to LogLevel.INFO.
00019     """
00020
00021     message = f"|| LogOpenAccessibilityPy || {message} ||"
00022
00023     try:
00024         from unreal import log, log_warning, log_error
00025
00026         if log_level == LogLevel.INFO:
00027             log(message)
00028         elif log_level == LogLevel.WARNING:
00029             log_warning(message)
00030         elif log_level == LogLevel.ERROR:
00031             log_error(message)
00032         else:
00033             log(message)
00034
00035     except ImportError:
00036         print(message)
00037         pass

```

## 5.8 WhisperInterface.py

```

00001 from ctypes import Union
00002 import numpy as np
00003
00004 from faster_whisper import WhisperModel
00005 from faster_whisper.transcribe import Segment
00006
00007 try:
00008     from .Logging import Log, LogLevel
00009 except ImportError:
00010     from Logging import Log, LogLevel
00011
00012
00013 class WhisperInterface:
00014     """Interface Class for Interacting with the CTranslate2 Faster Whisper Model."""
00015
00016     def __init__(
00017         self,
00018         model_name: str = "distil-small.en",
00019         device: str = "auto",
00020         cpu_threads: int = 4,
00021         transcribe_workers: int = 2,
00022         compute_type: str = "default",
00023     ):
00024         """Constructor of Whisper Interface Class
00025
00026         Args:
00027             model_name (str, optional): Hugging-Face Model Specifier for CTranslate Compatible Models.
00028             Defaults to "distil-small.en".
00029             device (str, optional): Device host for the Whisper Model (Can be "auto", "cpu", "cuda").
00030             Defaults to "auto".
00031             cpu_threads (int, optional): Amount of CPU Threads to use, if Hosting the Model on a CPU.
00032             Defaults to 4.
00033             transcribe_workers (int, optional): Amount of Thread Workers for Audio Transcription.
00034             Defaults to 2.
00035             compute_type (str, optional): Data Structure for Whisper Compute. Defaults to "default".
00036
00037         """
00038
00039         # Whisper Focused Variables
00040         self.whisper_model = WhisperModel(
00041             model_name,
00042             device=device,
00043             compute_type=compute_type,
00044             num_workers=transcribe_workers,
00045             cpu_threads=cpu_threads,
00046             local_files_only=True,
00047         )
00048         self.beam_size = 5

```

```

00044
00045     def __del__(self):
00046         """Destructor of Whisper Interface Class."""
00047
00048         del self.whisper_model
00049
00050     def process_file_from_dir(self, filepath: str):
00051         """Transcribes an Audio Files from a Given WAV File Path.
00052
00053         Args:
00054             filepath (str): Path to the Audio Files to Transcribe.
00055
00056         Returns:
00057             A List of Segments containing the Transcribed Text and their Time Stamps.
00058         """
00059
00060         segments, info = self.whisper_model.transcribe(
00061             filepath,
00062             beam_size=self.beam_size,
00063             language="en",
00064             prepend_punctuations="",
00065             append_punctuations="",
00066             vad_filter=True,
00067         )
00068
00069         Log(
00070             f"WhisperInterface | Detected Language: {info.language} | Probability:
00071             {info.language_probability} | Duration: {info.duration}"
00072         )
00073
00074         for segment in segments:
00075             Log(
00076                 f"WhisperInterface | Segment : {segment.text} | Start: {segment.start} | End:
00077                 {segment.end}"
00078             )
00079
00080             return list(segments)
00081
00082     def process_audio_buffer(
00083         self, audio_buffer: np.ndarray
00084     ) -> tuple[list[Segment], dict]:
00085         """Transcribes an NDArrary AudioBuffer.
00086
00087         Args:
00088             audio_buffer (np.ndarray): AudioBuffer to Transcribe.
00089
00090         Returns:
00091             tuple[list[Segment], dict]: Tuple Containing a List of Transcription Segments and a
00092             Dictionary of Collected Metadata.
00093         """
00094
00095         segments, info = self.whisper_model.transcribe(
00096             audio_buffer,
00097             beam_size=self.beam_size,
00098             language="en",
00099         )
00100
00101         Log(
00102             f"WhisperInterface || Detected Language: {info.language} | Probability:
00103             {info.language_probability} | Duration: {info.duration}"
00104         )
00105
00106         collected_metadata = {
00107             "duration": info.duration,
00108             "language": info.language,
00109             "language_probability": info.language_probability,
00110         }
00111
00112         return list(segments), collected_metadata

```

## 5.9 OpenAccessibility.Build.cs

```

00001 // Copyright Epic Games, Inc. All Rights Reserved.
00002
00003 using System.IO;
00004 using UnrealBuildTool;
00005
00006 public class OpenAccessibility : ModuleRules
00007 {
00008     public OpenAccessibility(ReadOnlyTargetRules Target) : base(Target)
00009     {
00010         PCHUsage = ModuleRules.PCHUsageMode.UseExplicitOrSharedPCHs;
00011

```

```

00012     PublicIncludePaths.AddRange(
00013         new string[] {
00014             // ... add public include paths required here ...
00015         }
00016     );
00017
00018     PrivateIncludePaths.AddRange(
00019         new string[] {
00020             // ... add other private include paths required here ...
00021         }
00022     );
00023
00024
00025     PublicDependencyModuleNames.AddRange(
00026         new string[]
00027         {
00028             "Core",
00029             // ... add other public dependencies that you statically link with here ...
00030         }
00031     );
00032
00033
00034     PrivateDependencyModuleNames.AddRange(
00035         new string[]
00036         {
00037             // Internal Plugin Modules
00038             "OpenAccessibilityCommunication",
00039
00040             // Core Modules
00041             "CoreUObject",
00042             "Engine",
00043             "Json",
00044
00045             // Editor Modules
00046             "UnrealEd",
00047             "GraphEditor",
00048             "Kismet",
00049             "AIModule",
00050
00051             // Slate UI
00052             "Slate",
00053             "SlateCore",
00054             "EditorStyle",
00055         }
00056     );
00057
00058
00059     DynamicallyLoadedModuleNames.AddRange(
00060         new string[]
00061         {
00062             // ... add any modules that your module loads dynamically here ...
00063         }
00064     );
00065
00066     CircularlyReferencedDependentModules.AddRange(
00067         new string[]
00068         {
00069
00070         }
00071     );
00072 }
00073 }

```

## 5.10 SAccessibilityTranscriptionVis.cpp

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #include "AccessibilityWidgets/SAccessibilityTranscriptionVis.h"
00004
00005 SAccessibilityTranscriptionVis::~SAccessibilityTranscriptionVis()
00006 {
00007 }
00008
00009 void SAccessibilityTranscriptionVis::Construct(const FArguments& InArgs)
00010 {
00011     // Transcription Holder
00012     TSharedPtr<SVerticalBox> TranscriptionHolder = SNew(SVerticalBox)
00013         + SVerticalBox::Slot()
00014         .Padding(4.0f)
00015         .AutoHeight();
00016
00017     // Verify a least one slot will be constructed
00018     int TranscriptionSlotAmount = FMath::Max(1, InArgs._VisAmount);

```

```

00019
00020     FSlateFontInfo FontInfo = FAppStyle::GetFontStyle("NormalText");
00021     FontInfo.Size = 12;
00022
00023     TSharedPtr<STextBlock> CurrentTranscriptionSlot;
00024     for (int i = 0; i < TranscriptionSlotAmount; i++)
00025     {
00026         TranscriptionHolder->AddSlot()
00027             .HAlign(HAlign_Center)
00028             .Padding(4.0f)
00029             .AutoHeight()
00030             [
00031                 SAssignNew(CurrentTranscriptionSlot, STextBlock)
00032                     .Text(FText::GetEmpty())
00033                     .Font(FontInfo)
00034                     .SimpleTextMode(true)
00035                     .ColorAndOpacity(i == 0 ? FSlateColor(FLinearColor(1.0f, 1.0f, 0, 1.0f)) :
FSlateColor(FLinearColor(0.5f, 0.5f, 0.5f, 1.0f)))
00036             ];
00037
00038         TranscriptionSlots.Add(CurrentTranscriptionSlot);
00039     }
00040
00041     // Construct the Main Component
00042
00043     ChildSlot
00044     .Padding(FMargin(5.0f))
00045     [
00046         SNew(SOverlay)
00047         + SOverlay::Slot()
00048         .ZOrder(1)
00049         [
00050             SNew(SBorder)
00051             .BorderBackgroundColor(FSlateColor(FLinearColor::Gray))
00052             [
00053                 SNew(SBox)
00054                 .MinDesiredWidth(250.0f)
00055                 .MinDesiredHeight(60.0f)
00056                 [
00057                     TranscriptionHolder.ToSharedRef()
00058                 ]
00059             ]
00060         ]
00061     ];
00062
00063     this->TranscriptionContainer = TranscriptionHolder;
00064 }
00065
00066 void SAccessibilityTranscriptionVis::Tick(const FGeometry& AllottedGeometry, const double
InCurrentTime, const float InDeltaTime)
00067 {
00068     SBox::Tick(AllottedGeometry, InCurrentTime, InDeltaTime);
00069 }
00070
00071 void SAccessibilityTranscriptionVis::UpdateTopTranscription(const FString& InTopTranscription)
00072 {
00073     FString LastTopText = InTopTranscription;
00074     FString TempText;
00075
00076     TSharedPtr<STextBlock> CurrentTranscriptionSlot;
00077     for (TWeakPtr<STextBlock>& TranscriptionSlot : TranscriptionSlots)
00078     {
00079         CurrentTranscriptionSlot = TranscriptionSlot.Pin();
00080
00081         TempText = FString(CurrentTranscriptionSlot->GetText().ToString());
00082         CurrentTranscriptionSlot->SetText(FText::FromString(LastTopText));
00083
00084         CurrentTranscriptionSlot->Invalidate(EInvalidateWidgetReason::PaintAndVolatility);
00085
00086         LastTopText = TempText;
00087     }
00088
00089     TranscriptionContainer.Pin()->Invalidate(EInvalidateWidget::Layout);
00090 }

```

## 5.11 SContentIndexer.cpp

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #include "AccessibilityWidgets/SContentIndexer.h"
00004 #include "AccessibilityWidgets/SIndexer.h"
00005
00006 SContentIndexer::~SContentIndexer()

```

```

00007 {
00008 }
00009
00010 void SContentIndexer::Construct(const FArguments& InArgs)
00011 {
00012     TSharedPtr<SWidget> Content;
00013
00014     switch (InArgs._IndexPositionToContent)
00015     {
00016     case EIndexerPosition::Top:
00017         Content = ConstructTopIndexer(InArgs);
00018         break;
00019
00020     case EIndexerPosition::Bottom:
00021         Content = ConstructBottomIndexer(InArgs);
00022         break;
00023
00024     default:
00025     case EIndexerPosition::Left:
00026         Content = ConstructLeftIndexer(InArgs);
00027         break;
00028
00029     case EIndexerPosition::Right:
00030         Content = ConstructRightIndexer(InArgs);
00031         break;
00032     }
00033
00034     ChildSlot
00035     [
00036         Content.ToSharedRef()
00037     ];
00038 }
00039
00040 void SContentIndexer::Tick(const FGeometry& AllottedGeometry, const double InCurrentTime, const float
    InDeltaTime)
00041 {
00042     SBox::Tick(AllottedGeometry, InCurrentTime, InDeltaTime);
00043 }
00044
00045 void SContentIndexer::UpdateIndex(const int32 IndexValue)
00046 {
00047     if (IndexerWidget.IsValid())
00048         IndexerWidget.Pin()->UpdateIndex(IndexValue);
00049 }
00050
00051 TSharedPtr<SWidget> SContentIndexer::ConstructTopIndexer(const FArguments& InArgs)
00052 {
00053     return SNew(SVerticalBox)
00054         .Visibility(AccessWidgetVisibilityAttribute(InArgs._ContentToIndex.ToSharedRef()))
00055         + SVerticalBox::Slot()
00056         .HAlign(HAlign_Center)
00057         .VAlign(VAlign_Center)
00058         .AutoHeight()
00059         .Padding(.1f, .25f)
00060         [
00061             ConstructIndexContainer(InArgs).ToSharedRef()
00062         ]
00063         + SVerticalBox::Slot()
00064         .HAlign(HAlign_Center)
00065         .VAlign(VAlign_Center)
00066         .AutoHeight()
00067         [
00068             ConstructContentContainer(InArgs._ContentToIndex.ToSharedRef()).ToSharedRef()
00069         ];
00070 }
00071
00072 }
00073
00074 TSharedPtr<SWidget> SContentIndexer::ConstructBottomIndexer(const FArguments& InArgs)
00075 {
00076     return SNew(SVerticalBox)
00077         .Visibility(AccessWidgetVisibilityAttribute(InArgs._ContentToIndex.ToSharedRef()))
00078         + SVerticalBox::Slot()
00079         .HAlign(HAlign_Center)
00080         .VAlign(VAlign_Center)
00081         .AutoHeight()
00082         [
00083             ConstructContentContainer(InArgs._ContentToIndex.ToSharedRef()).ToSharedRef()
00084         ]
00085         + SVerticalBox::Slot()
00086         .HAlign(HAlign_Center)
00087         .VAlign(VAlign_Center)
00088         .AutoHeight()
00089         .Padding(.1f, .25f)
00090         [
00091

```



```

00093         ConstructIndexContainer(InArgs).ToSharedRef()
00094     };
00095 }
00096
00097 TSharedPtr<SWidget> SContentIndexer::ConstructLeftIndexer(const FArguments& InArgs)
00098 {
00099     return SNew(SHorizontalBox)
00100         .Visibility(AccessWidgetVisibilityAttribute(InArgs._ContentToIndex.ToSharedRef()))
00101         + SHorizontalBox::Slot()
00102         .VAlign(VAlign_Center)
00103         .HAlign(HAlign_Center)
00104         .AutoWidth()
00105         .Padding(.25f, .1f)
00106         [
00107             ConstructIndexContainer(InArgs).ToSharedRef()
00108         ]
00109         + SHorizontalBox::Slot()
00110         .VAlign(VAlign_Center)
00111         .HAlign(HAlign_Center)
00112         .AutoWidth()
00113         [
00114             ConstructContentContainer(InArgs._ContentToIndex.ToSharedRef()).ToSharedRef()
00115         ];
00116 }
00117
00118 TSharedPtr<SWidget> SContentIndexer::ConstructRightIndexer(const FArguments& InArgs)
00119 {
00120     return SNew(SHorizontalBox)
00121         .Visibility(AccessWidgetVisibilityAttribute(InArgs._ContentToIndex.ToSharedRef()))
00122         + SHorizontalBox::Slot()
00123         .VAlign(VAlign_Center)
00124         .HAlign(HAlign_Center)
00125         .AutoWidth()
00126         [
00127             ConstructContentContainer(InArgs._ContentToIndex.ToSharedRef()).ToSharedRef()
00128         ]
00129         + SHorizontalBox::Slot()
00130         .VAlign(VAlign_Center)
00131         .HAlign(HAlign_Center)
00132         .AutoWidth()
00133         .Padding(.25f, .1f)
00134         [
00135             ConstructIndexContainer(InArgs).ToSharedRef()
00136         ];
00137 }
00138
00139 TSharedPtr<SWidget> SContentIndexer::ConstructContentContainer(TSharedRef<SWidget> ContentToIndex)
00140 {
00141     IndexedContent = ContentToIndex;
00142     return ContentToIndex;
00143 }
00144
00145 TSharedPtr<SWidget> SContentIndexer::ConstructIndexContainer(const FArguments& InArgs, FLinearColor
    TextColor)
00146 {
00147     return SAssignNew(IndexerWidget, SIndexer)
00148         .TextColor(TextColor)
00149         .BorderColor(FLinearColor::Gray)
00150         .IndexValue(InArgs._IndexValue)
00151         .IndexVisibility(InArgs._IndexVisibility);
00152 }
00153
00154 FText SContentIndexer::ConstructIndexText(int32 Index)
00155 {
00156     return FText::FromString(FString::FromInt(Index));
00157 }
00158
00159
00160
00161

```

## 5.12 SIndexer.cpp

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #include "AccessibilityWidgets/SIndexer.h"
00004
00005 SIndexer::~SIndexer()
00006 {
00007 }
00008
00009
00010 void SIndexer::Tick(const FGeometry& AllottedGeometry, const double InCurrentTime, const float
    InDeltaTime)

```

```

00011 {
00012     SBox::Tick(AllotedGeometry, InCurrentTime, InDeltaTime);
00013 }
00014
00015 void SIndexer::Construct(const FArguments& InArgs)
00016 {
00017     ChildSlot
00018     [
00019         SNew(SBorder)
00020         .HAlign(HAlign_Center)
00021         .VAlign(VAlign_Center)
00022         .Visibility(InArgs._IndexVisibility)
00023         .Padding(FMargin(4.f, 2.f))
00024         .BorderBackgroundColor( FSlateColor(InArgs._BorderColor) )
00025     [
00026         SAssignNew(IndexTextBlock, STextBlock)
00027         .Text( FText::FromString(FString::FromInt(InArgs._IndexValue)) )
00028         .TextShapingMethod( ETextShapingMethod::KerningOnly )
00029         .ColorAndOpacity( FSlateColor(InArgs._TextColor) )
00030     ]
00031     ];
00032 }
00033
00034 void SIndexer::UpdateIndex(const int32 NewIndex)
00035 {
00036     if (!IndexTextBlock.IsValid())
00037         return;
00038
00039     IndexTextBlock.Pin()->SetText(
00040         FText::FromString( FString::FromInt(NewIndex) )
00041     );
00042 }
00043
00044 void SIndexer::UpdateIndex(const FString& NewIndex)
00045 {
00046     if (!IndexTextBlock.IsValid())
00047         return;
00048
00049     IndexTextBlock.Pin()->SetText(
00050         FText::FromString(NewIndex)
00051     );
00052 }
00053
00054 void SIndexer::UpdateIndex(const FText& NewIndex)
00055 {
00056     if (!IndexTextBlock.IsValid())
00057         return;
00058
00059     IndexTextBlock.Pin()->SetText(NewIndex);
00060 }

```

## 5.13 AccessibilityAddNodeContextMenu.cpp

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #include "AccessibilityWrappers/AccessibilityAddNodeContextMenu.h"
00004 #include "OpenAccessibilityLogging.h"
00005
00006 #include "Widgets/Input/SSearchBar.h"
00007
00008 #include "AccessibilityWidgets/SContentIndexer.h"
00009
00010
00011 #include "Styling/AppStyle.h"
00012
00013 UAccessibilityAddNodeContextMenu::UAccessibilityAddNodeContextMenu()
00014     : UPhraseTreeContextMenuObject()
00015 {
00016 }
00017
00018
00019 UAccessibilityAddNodeContextMenu::UAccessibilityAddNodeContextMenu(TSharedRef<IMenu> Menu)
00020     : UPhraseTreeContextMenuObject(Menu)
00021 {
00022 }
00023
00024
00025 UAccessibilityAddNodeContextMenu::UAccessibilityAddNodeContextMenu(TSharedRef<IMenu> Menu,
00026     TSharedRef<SGraphActionMenu> GraphMenu)
00027     : UPhraseTreeContextMenuObject(Menu)
00028 {
00029     this->GraphMenu = GraphMenu;
00029     this->FilterTextBox = GraphMenu->GetFilterTextBox();

```

```

00030 }
00031
00032 UAccessibilityAddNodeContextMenu::UAccessibilityAddNodeContextMenu(TSharedRef<IMenu> Menu,
    TSharedRef<SGraphActionMenu> GraphMenu, TSharedRef<STreeView<TSharedPtr<FGraphNode>> TreeView)
00033 : UPhraseTreeContextMenuObject(Menu)
00034 {
00035     this->GraphMenu = GraphMenu;
00036     this->TreeView = TreeView;
00037     this->FilterTextBox = GraphMenu->GetFilterTextBox();
00038 }
00039
00040 UAccessibilityAddNodeContextMenu::~UAccessibilityAddNodeContextMenu()
00041 {
00042 }
00043 }
00044
00045 void UAccessibilityAddNodeContextMenu::Init(TSharedRef<IMenu> InMenu, TSharedRef<FPhraseNode>
    InContextRoot)
00046 {
00047     Init(InMenu);
00048
00049     this->ContextRoot = InContextRoot;
00050 }
00051
00052 void UAccessibilityAddNodeContextMenu::Init(TSharedRef<IMenu> InMenu)
00053 {
00054     UPhraseTreeContextMenuObject::Init(InMenu);
00055
00056     // This is a Mess but half the Menu Containers are private, so have to move myself to key aspects
    of the Menu.
00057
00058     TSharedPtr<SWidget> KeyboardFocusedWidget = StaticCastSharedPtr<SEditableText>(
00059         FSlateApplication::Get().GetKeyboardFocusedWidget());
00060 };
00061 if (!KeyboardFocusedWidget.IsValid())
00062 {
00063     UE_LOG(LogOpenAccessibility, Warning, TEXT("AddNodeContextWrapper::Init: KeyboardFocusedWidget
    is Invalid."));
00064     return;
00065 }
00066
00067 this->GraphMenu = StaticCastSharedPtr<SGraphActionMenu>(
00068     KeyboardFocusedWidget
00069     ->GetParentWidget()
00070     ->GetParentWidget()
00071     ->GetParentWidget()
00072     ->GetParentWidget()
00073     ->GetParentWidget());
00074 };
00075
00076 {
00077     TSharedPtr<SSearchBar> SearchBox = StaticCastSharedPtr<SSearchBar>(
00078         KeyboardFocusedWidget
00079         ->GetParentWidget()
00080         ->GetParentWidget()
00081         ->GetParentWidget());
00082 };
00083
00084 TSharedRef<SWidget> SearchBoxSibling =
00085     SearchBox->GetParentWidget()->GetChildren()->GetChildAt(1);
00086 this->TreeView = StaticCastSharedRef<STreeView<TSharedPtr<FGraphNode>>>(
00087     SearchBoxSibling->GetChildren()->GetChildAt(0)->GetChildren()->GetChildAt(0)
00088 );
00089
00090 {
00091     TSharedRef<SCheckBox> CheckBox = StaticCastSharedRef<SCheckBox>(
00092         this->GraphMenu.Pin()->GetParentWidget()->GetChildren()->GetChildAt(0)->GetChildren()->GetChildAt(2)
00093     );
00094
00095     this->ContextAwarenessCheckBox = CheckBox;
00096 }
00097
00098 this->FilterTextBox = this->GraphMenu.Pin()->GetFilterTextBox();
00099
00100 FSlateApplication::Get().SetKeyboardFocus(this->TreeView.Pin());
00101 }
00102
00103 void UAccessibilityAddNodeContextMenu::Init(TSharedRef<IMenu> InMenu, TSharedRef<SGraphActionMenu>
    InGraphMenu, TSharedRef<STreeView<TSharedPtr<FGraphNode>> InTreeView)
00104 {
00105     UPhraseTreeContextMenuObject::Init(InMenu);
00106
00107     this->GraphMenu = InGraphMenu;
00108     this->TreeView = InTreeView;
00109     this->FilterTextBox = InGraphMenu->GetFilterTextBox();

```

```

00110 }
00111
00112 bool UAccessibilityAddNodeContextMenu::Tick(float DeltaTime)
00113 {
00114     if (!GraphMenu.IsValid() || !Menu.IsValid())
00115         return false;
00116
00117     if (DoesItemsRequireRefresh())
00118         RefreshAccessibilityWidgets();
00119
00120     TSharedPtr<STreeView<TSharedPtr<FGraphNode>>> TreeViewPtr = TreeView.Pin();
00121
00122     // Set Previous Vars For Next Tick
00123     PrevFilterString = FilterTextBox.Pin()->GetText().ToString();
00124     PrevNumItemsBeingObserved = TreeViewPtr->GetNumItemsBeingObserved();
00125     PrevNumGeneratedChildren = TreeViewPtr->GetNumGeneratedChildren();
00126     PrevScrollDistance = TreeViewPtr->GetScrollDistance().Y;
00127
00128     return true;
00129 }
00130
00131 bool UAccessibilityAddNodeContextMenu::Close()
00132 {
00133     RemoveTickDelegate();
00134     Menu.Pin()->Dismiss();
00135
00136     return true;
00137 }
00138
00139 void UAccessibilityAddNodeContextMenu::ScaleMenu(const float ScaleFactor)
00140 {
00141     // Scale TreeView Element
00142     {
00143         TSharedPtr<STreeView<TSharedPtr<FGraphNode>>> TreeViewPtr = TreeView.Pin();
00144
00145         TreeViewPtr->SetItemHeight(16 * ScaleFactor);
00146     }
00147
00148     // Scale Window Element
00149     {
00150         TSharedPtr<SWindow> WindowPtr = Window.Pin();
00151
00152         WindowPtr->SetSizingRule(ESizingRule::UserSized);
00153         WindowPtr->Resize(WindowPtr->GetSizeInScreen() * ScaleFactor);
00154     }
00155 }
00156
00157 bool UAccessibilityAddNodeContextMenu::DoesItemsRequireRefresh()
00158 {
00159     TSharedPtr<STreeView<TSharedPtr<FGraphNode>>> TreeViewPtr = TreeView.Pin();
00160
00161     return (
00162         FilterTextBox.Pin()->GetText().ToString() != PrevFilterString ||
00163         TreeViewPtr->GetNumItemsBeingObserved() != PrevNumItemsBeingObserved ||
00164         TreeViewPtr->GetNumGeneratedChildren() != PrevNumGeneratedChildren ||
00165         TreeViewPtr->GetScrollDistance().Y != PrevScrollDistance
00166     );
00167 }
00168
00169 void UAccessibilityAddNodeContextMenu::RefreshAccessibilityWidgets()
00170 {
00171     TSharedPtr<STreeView<TSharedPtr<FGraphNode>>> TreeViewPtr = TreeView.Pin();
00172
00173     TArray<TSharedPtr<FGraphNode>> Items =
00174         TArray<TSharedPtr<FGraphNode>>(TreeViewPtr->GetRootItems());
00175
00176     {
00177         TSharedPtr<STableRow<TSharedPtr<FGraphNode>>> ItemWidget = nullptr;
00178
00179         while (Items.Num() > 0)
00180         {
00181             const TSharedPtr<FGraphNode> Item = Items[0];
00182             Items.RemoveAt(0);
00183
00184             if (TreeViewPtr->IsItemExpanded(Item))
00185                 Items.Append(Item->Children);
00186
00187             ItemWidget = StaticCastSharedPtr<STableRow<TSharedPtr<FGraphNode>>>(
00188                 TreeViewPtr->WidgetFromItem(Item)
00189             );
00190
00191             if (!ItemWidget.IsValid())
00192                 continue;
00193
00194             // TO-DO: Change To Non-HardCoded Type Comparison.
00195             if (ItemWidget->GetContent()->GetType() == "SContentIndexer")

```

```

00196         {
00197             UpdateAccessibilityWidget (ItemWidget.ToSharedRef());
00198         }
00199         else
00200         {
00201             ApplyAccessibilityWidget (ItemWidget.ToSharedRef());
00202         }
00203     }
00204 }
00205 }
00206
00207 FGraphActionNode* UAccessibilityAddNodeContextMenu::GetGraphActionFromIndex(const int32 InIndex)
00208 {
00209     TArrayView<const TSharedPtr<FGraphActionNode>> Items = TreeView.Pin()->GetItems();
00210
00211     if (Items.Num() > InIndex)
00212         return Items[InIndex].Get();
00213
00214     else return nullptr;
00215 }
00216
00217 void UAccessibilityAddNodeContextMenu::GetGraphActionFromIndex(const int32 InIndex, FGraphActionNode*
    OutGraphAction)
00218 {
00219     TArrayView<const TSharedPtr<FGraphActionNode>> Items = TreeView.Pin()->GetItems();
00220
00221     if (Items.Num() > InIndex)
00222         OutGraphAction = Items[InIndex].Get();
00223
00224     else OutGraphAction = nullptr;
00225 }
00226
00227 TSharedPtr<FGraphActionNode> UAccessibilityAddNodeContextMenu::GetGraphActionFromIndexSP(const int32
    InIndex)
00228 {
00229     if (TreeView.Pin()->GetItems().Num() <= InIndex)
00230     {
00231         UE_LOG(LogOpenAccessibility, Warning, TEXT("GetGraphActionFromIndexSP: Provided Index is Out
    of Range."));
00232         return nullptr;
00233     }
00234     return TreeView.Pin()->GetItems()[InIndex];
00235 }
00236
00237 void UAccessibilityAddNodeContextMenu::SelectGraphAction(const int32 InIndex)
00238 {
00239     TSharedPtr<FGraphActionNode> GraphAction = GetGraphActionFromIndexSP(InIndex);
00240
00241     if (GraphAction.IsValid())
00242     {
00243         TreeView.Pin()->Private_OnItemClicked(GraphAction);
00244     }
00245     else
00246     {
00247         UE_LOG(LogOpenAccessibility, Warning, TEXT("SelectGraphAction: Provided GraphAction is
    Invalid."));
00248     }
00249 }
00250
00251 void UAccessibilityAddNodeContextMenu::PerformGraphAction(const int32 InIndex)
00252 {
00253     TSharedPtr<FGraphActionNode> GraphAction = GetGraphActionFromIndexSP(InIndex);
00254
00255     if (!GraphAction.IsValid())
00256     {
00257         UE_LOG(LogOpenAccessibility, Warning, TEXT("PerformGraphAction: Provided GraphAction is
    Invalid."));
00258     }
00259
00260     if (GraphAction->IsActionNode())
00261     {
00262         TreeView.Pin()->Private_ClearSelection();
00263         TreeView.Pin()->Private_SetItemSelection(GraphAction, true, true);
00264         TreeView.Pin()->Private_SignalSelectionChanged(ESelectInfo::OnMouseClicked);
00265     }
00266     else
00267     {
00268         TreeView.Pin()->Private_OnItemDoubleClicked(GraphAction);
00269     }
00270 }
00271
00272 FString UAccessibilityAddNodeContextMenu::GetFilterText()
00273 {
00274     return FilterTextBox.Pin()->GetText().ToString();
00275 }
00276
00277 void UAccessibilityAddNodeContextMenu::SetFilterText(const FString& InFilterText)

```

```

00278 {
00279     FilterTextBox.Pin()->SetText(FText::FromString(InFilterText));
00280 }
00281
00282 void UAccessibilityAddNodeContextMenu::AppendFilterText(const FString& InFilterText)
00283 {
00284     FilterTextBox.Pin()->SetText (
00285         FText::FromString (
00286             FilterTextBox.Pin()->GetText().ToString() + TEXT(" ") + InFilterText
00287         )
00288     );
00289 }
00290
00291 void UAccessibilityAddNodeContextMenu::ResetFilterText ()
00292 {
00293     FilterTextBox.Pin()->SetText (FText::FromString(TEXT("")));
00294 }
00295
00296 void UAccessibilityAddNodeContextMenu::SetScrollDistance(const float InScrollDistance)
00297 {
00298     TreeView.Pin()->SetScrollOffset(InScrollDistance);
00299 }
00300
00301 void UAccessibilityAddNodeContextMenu::AppendScrollDistance(const float InScrollDistance)
00302 {
00303     if (TreeView.Pin()->GetScrollOffset() + InScrollDistance < 0.0f)
00304     {
00305         TreeView.Pin()->SetScrollOffset(0.0f);
00306         return;
00307     }
00308     TreeView.Pin()->AddScrollOffset(InScrollDistance, true);
00309 }
00310
00311 void UAccessibilityAddNodeContextMenu::SetScrollDistanceTop ()
00312 {
00313     TreeView.Pin()->ScrollToTop();
00314 }
00315
00316 void UAccessibilityAddNodeContextMenu::SetScrollDistanceBottom ()
00317 {
00318     TreeView.Pin()->ScrollToBottom();
00319 }
00320
00321 void UAccessibilityAddNodeContextMenu::ToggleContextAwareness ()
00322 {
00323     ContextAwarenessCheckBox.Pin()->ToggleCheckedState();
00324 }
00325
00326 void
00327 UAccessibilityAddNodeContextMenu::ApplyAccessibilityWidget (TSharedRef<STableRow<TSharedPtr<FGraphNode>>
    ItemWidget)
00328 {
00329     TSharedPtr<SWidget> ItemContent = ItemWidget->GetContent();
00330
00331     ItemWidget->SetContent (
00332         SNew(SContentIndexer)
00333             .IndexValue (ItemWidget->GetIndexInList())
00334             .IndexPositionToContent (EIndexerPosition::Left)
00335             .ContentToIndex (ItemContent)
00336     );
00337 }
00338
00339 void
00340 UAccessibilityAddNodeContextMenu::UpdateAccessibilityWidget (TSharedRef<STableRow<TSharedPtr<FGraphNode>>
    ItemWidget)
00341 {
00342     TSharedPtr<SContentIndexer> ItemContent =
00343         StaticCastSharedPtr<SContentIndexer> (ItemWidget->GetContent());
00344
00345     ItemContent->UpdateIndex (ItemWidget->GetIndexInList());
00346 }

```

## 5.14 AccessibilityGraphEditorContext.cpp

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #include "AccessibilityWrappers/AccessibilityGraphEditorContext.h"
00004
00005 #include "OpenAccessibilityLogging.h"
00006 #include "AccessibilityWidgets/SIndexer.h"
00007 #include "AccessibilityWidgets/SContentIndexer.h"
00008 #include "Utils/WidgetUtils.h"

```

```

00009
00010 #include "Widgets/SWindow.h"
00011 #include "Widgets/Input/SEditableTextBox.h"
00012
00013 UAccessibilityGraphEditorContext::UAccessibilityGraphEditorContext()
00014     : Super()
00015 {
00016 }
00017
00018
00019 void UAccessibilityGraphEditorContext::Init(TSharedRef<IMenu> InMenu, TSharedRef<FPhraseNode>
    InContextRoot)
00020 {
00021     Super::Init(InMenu, InContextRoot);
00022
00023     TSharedRef<SWindow> WindowRef = Window.Pin().ToSharedRef();
00024
00025     if (!FindGraphActionMenu(WindowRef))
00026     {
00027         UE_LOG(LogOpenAccessibility, Warning, TEXT("GraphEditorContext: Cannot Find a SGraphActionMenu
    Widget"));
00028     }
00029
00030     if (!FindStaticComponents(WindowRef))
00031     {
00032         UE_LOG(LogOpenAccessibility, Warning, TEXT("GraphEditorContext: Cannot Find Any Static
    Components"));
00033     }
00034
00035     if (!FindTreeView(WindowRef))
00036     {
00037         UE_LOG(LogOpenAccessibility, Warning, TEXT("GraphEditorContext: Cannot Find a STreeView
    Widget"));
00038     }
00039     else
00040     {
00041         TreeViewTickRequirements = FTreeViewTickRequirements();
00042     }
00043 }
00044
00045 bool UAccessibilityGraphEditorContext::Tick(float DeltaTime)
00046 {
00047     Super::Tick(DeltaTime);
00048
00049     if (TreeViewCanTick())
00050     {
00051         TickTreeViewAccessibility();
00052
00053         TSharedPtr<STreeView<TSharedPtr<FGraphNode>>> TreeViewPtr = TreeView.Pin();
00054
00055         TreeViewTickRequirements.PrevSearchText = FilterTextBox.Pin()->GetText().ToString();
00056         TreeViewTickRequirements.PrevNumGeneratedChildren = TreeViewPtr->GetNumGeneratedChildren();
00057         TreeViewTickRequirements.PrevNumItemsBeingObserved = TreeViewPtr->GetNumItemsBeingObserved();
00058         TreeViewTickRequirements.PrevScrollDistance = TreeViewPtr->GetScrollDistance().Y;
00059     }
00060
00061     return true;
00062 }
00063
00064 bool UAccessibilityGraphEditorContext::Close()
00065 {
00066     Super::Close();
00067
00068     return true;
00069 }
00070
00071 void UAccessibilityGraphEditorContext::ScaleMenu(const float ScaleFactor)
00072 {
00073     Super::ScaleMenu(ScaleFactor);
00074
00075     // Scale TreeView
00076     if (TreeView.IsValid())
00077     {
00078         TSharedPtr<STreeView<TSharedPtr<FGraphNode>>> TreeViewPtr = TreeView.Pin();
00079
00080         TreeViewPtr->SetItemHeight(16 * ScaleFactor);
00081     }
00082
00083     // Scale Window Element
00084     if (Window.IsValid())
00085     {
00086         TSharedPtr<SWindow> WindowPtr = Window.Pin();
00087
00088         WindowPtr->SetSizingRule(ESizingRule::UserSized);
00089         WindowPtr->Resize(WindowPtr->GetSizeInScreen() * ScaleFactor);
00090     }
00091 }

```

```

00092
00093 TSharedPtr<FGraphNode> UAccessibilityGraphEditorContext::GetTreeViewAction(const int32& InIndex)
00094 {
00095     TArrayView<const TSharedPtr<FGraphNode> Items = TreeView.Pin()->GetItems();
00096
00097     if (TreeView.IsValid() && Items.Num() > InIndex && InIndex >= 0)
00098         return TreeView.Pin()->GetItems()[InIndex];
00099
00100     return TSharedPtr<FGraphNode>();
00101 }
00102
00103 void UAccessibilityGraphEditorContext::SelectAction(const int32& InIndex)
00104 {
00105     if (InIndex < 0)
00106         return;
00107
00108     if (!CheckBoxes.IsEmpty() && InIndex < CheckBoxes.Num())
00109     {
00110         if (CheckBoxes[InIndex].IsValid())
00111         {
00112             CheckBoxes[InIndex].Pin()->ToggleCheckedState();
00113             return;
00114         }
00115     }
00116
00117     TSharedPtr<FGraphNode> ChosenTreeViewAction = GetTreeViewAction(InIndex -
GetStaticIndexOffset());
00118     if (!ChosenTreeViewAction.IsValid())
00119     {
00120         UE_LOG(LogOpenAccessibility, Warning, TEXT("SelectGraphAction: Provided TreeView Action is
Invalid"))
00121         return;
00122     }
00123
00124     auto TreeViewPtr = TreeView.Pin();
00125     if (ChosenTreeViewAction->IsActionNode())
00126     {
00127         TreeViewPtr->Private_ClearSelection();
00128         TreeViewPtr->Private_SetItemSelection(ChosenTreeViewAction, true, true);
00129         TreeViewPtr->Private_SignalSelectionChanged(ESelectInfo::Type::OnClick);
00130     }
00131     else
00132     {
00133         TreeViewPtr->Private_OnItemDoubleClicked(ChosenTreeViewAction);
00134     }
00135 }
00136
00137 FString UAccessibilityGraphEditorContext::GetFilterText()
00138 {
00139     return FilterTextBox.IsValid() ? FilterTextBox.Pin()->GetText().ToString() : FString();
00140 }
00141
00142 void UAccessibilityGraphEditorContext::SetFilterText(const FString& NewString)
00143 {
00144     if (!FilterTextBox.IsValid())
00145         return;
00146
00147     FilterTextBox.Pin()->SetText(
00148         FText::FromString(NewString)
00149     );
00150 }
00151
00152 void UAccessibilityGraphEditorContext::AppendFilterText(const FString& StringToAdd)
00153 {
00154     if (!FilterTextBox.IsValid())
00155         return;
00156
00157     TSharedPtr<SEditableTextBox> FilterTextBoxPtr = FilterTextBox.Pin();
00158
00159     FilterTextBoxPtr->SetText(
00160         FText::FromString( FilterTextBoxPtr->GetText().ToString() + TEXT(" ") + StringToAdd )
00161     );
00162 }
00163
00164 void UAccessibilityGraphEditorContext::SetScrollDistance(const float NewDistance)
00165 {
00166     if (TreeView.IsValid())
00167         return;
00168
00169     TreeView.Pin()->SetScrollOffset(NewDistance);
00170 }
00171
00172 void UAccessibilityGraphEditorContext::AppendScrollDistance(const float DistanceToAdd)
00173 {
00174     auto TreeViewPtr = TreeView.Pin();
00175
00176     if (TreeViewPtr->GetScrollOffset() + DistanceToAdd < 0.0f)

```



```

00177     {
00178         TreeViewPtr->SetScrollOffset(0.0f);
00179         return;
00180     }
00181
00182     TreeViewPtr->AddScrollOffset(DistanceToAdd);
00183 }
00184
00185 void UAccessibilityGraphEditorContext::SetScrollDistanceTop()
00186 {
00187     TreeView.Pin()->ScrollToTop();
00188 }
00189
00190 void UAccessibilityGraphEditorContext::SetScrollDistanceBottom()
00191 {
00192     TreeView.Pin()->ScrollToBottom();
00193 }
00194
00195 const int32 UAccessibilityGraphEditorContext::GetStaticIndexOffset()
00196 {
00197     return CheckBoxes.Num();
00198 }
00199
00200 bool UAccessibilityGraphEditorContext::FindGraphActionMenu(const TSharedRef<SWidget>& SearchRoot)
00201 {
00202     TSharedPtr<SGraphActionMenu> GraphActionMenu = GetWidgetDescendant<SGraphActionMenu>(SearchRoot,
00203     TEXT("SGraphActionMenu"));
00204     if (GraphActionMenu.IsValid())
00205     {
00206         GraphMenu = GraphActionMenu;
00207         FilterTextBox = GraphActionMenu->GetFilterTextBox();
00208         return true;
00209     }
00210     return false;
00211 }
00212
00213
00214 bool UAccessibilityGraphEditorContext::FindTreeView(const TSharedRef<SWidget>& SearchRoot)
00215 {
00216     TSharedPtr<STreeView<TSharedPtr<FGraphActionNode>> ContextTreeView =
00217     GetWidgetDescendant<STreeView<TSharedPtr<FGraphActionNode>>(
00218     SearchRoot,
00219     TEXT("STreeView<TSharedPtr<FGraphActionNode>"));
00220     if (ContextTreeView.IsValid())
00221     {
00222         TreeView = ContextTreeView;
00223         return true;
00224     }
00225     return false;
00226 }
00227
00228
00229
00230 bool UAccessibilityGraphEditorContext::FindStaticComponents(const TSharedRef<SWidget>& SearchRoot)
00231 {
00232     TArray<FSlotBase*> FoundComponentSlots = GetWidgetSlotsByType(
00233     SearchRoot,
00234     TSet<FString> {
00235         TEXT("SCheckBox")
00236     });
00237
00238     if (!FoundComponentSlots.IsEmpty())
00239     {
00240         // Sort and Index the Static Components.
00241         for (int i = 0; i < FoundComponentSlots.Num(); i++)
00242         {
00243             FSlotBase* FoundComponentSlot = FoundComponentSlots[i];
00244
00245             TSharedPtr<SWidget> DetachedWidget = FoundComponentSlot->DetachWidget();
00246             if (!DetachedWidget.IsValid())
00247                 continue;
00248
00249             int32 ComponentIndex = -1;
00250             FString ComponentType = DetachedWidget->GetTypeAsString();
00251
00252             if (ComponentType == "SCheckBox")
00253             {
00254                 ComponentIndex = CheckBoxes.Num();
00255                 CheckBoxes.Add(StaticCastSharedPtr<SCheckBox>(DetachedWidget));
00256             }
00257
00258             FoundComponentSlot->AttachWidget(
00259                 SNew(SContentIndexer)
00260                 .IndexValue(ComponentIndex)

```

```

00262         .IndexPositionToContent (EIndexerPosition::Left)
00263         .ContentToIndex (DetachedWidget)
00264     );
00265 }
00266
00267     return true;
00268 }
00269
00270     return false;
00271 }
00272
00273 bool UAccessibilityGraphEditorContext::TreeViewCanTick()
00274 {
00275     return TreeView.IsValid() && GraphMenu.IsValid();
00276 }
00277
00278 bool UAccessibilityGraphEditorContext::TreeViewRequiresTick()
00279 {
00280     if (!TreeView.IsValid() || !GraphMenu.IsValid())
00281         return false;
00282
00283     bool bFilterTextChange = FilterTextBox.IsValid()
00284         ? FilterTextBox.Pin()->GetText().ToString() != TreeViewTickRequirements.PrevSearchText
00285         : false;
00286
00287     TSharedPtr<STreeView<TSharedPtr<FGraphNode>>> TreeViewPtr = TreeView.Pin();
00288
00289     return (
00290         bFilterTextChange ||
00291         TreeViewPtr->GetNumItemsBeingObserved() != TreeViewTickRequirements.PrevNumItemsBeingObserved
00292     ||
00293         TreeViewPtr->GetNumGeneratedChildren() != TreeViewTickRequirements.PrevNumGeneratedChildren ||
00294         TreeViewPtr->GetScrollDistance().Y != TreeViewTickRequirements.PrevScrollDistance
00295     );
00296 }
00297 void UAccessibilityGraphEditorContext::TickTreeViewAccessibility()
00298 {
00299     if (!TreeViewRequiresTick())
00300         return;
00301
00302     TSharedPtr<STreeView<TSharedPtr<FGraphNode>>> TreeViewPtr = TreeView.Pin();
00303
00304     TArray<TSharedPtr<FGraphNode>> Items = TArray<TSharedPtr<FGraphNode>>(
00305         TreeViewPtr->GetRootItems()
00306     );
00307
00308
00309     TSharedPtr<STableRow<TSharedPtr<FGraphNode>>> ItemWidget = nullptr;
00310     const int32 IndexOffset = GetStaticIndexOffset();
00311
00312     while (Items.Num() > 0)
00313     {
00314         const TSharedPtr<FGraphNode> Item = Items[0];
00315         Items.RemoveAt(0);
00316
00317         if (TreeViewPtr->IsItemExpanded(Item))
00318             Items.Append(Item->Children);
00319
00320         ItemWidget = StaticCastSharedPtr<STableRow<TSharedPtr<FGraphNode>>>(
00321             TreeViewPtr->WidgetFromItem(Item)
00322         );
00323         if (!ItemWidget.IsValid())
00324             continue;
00325
00326         TSharedPtr<SWidget> ItemContent = ItemWidget->GetContent();
00327
00328         if (ItemContent->GetType() == "SContentIndexer")
00329         {
00330             UpdateAccessibilityWidget (
00331                 StaticCastSharedPtr<SContentIndexer>(ItemContent.ToSharedRef()),
00332                 IndexOffset + ItemWidget->GetIndexInList()
00333             );
00334         }
00335         else
00336         {
00337             ItemWidget->SetContent (
00338                 CreateAccessibilityWrapper (ItemContent.ToSharedRef(), IndexOffset +
00339                 ItemWidget->GetIndexInList())
00340             );
00341         }
00342     }
00343
00344 void UAccessibilityGraphEditorContext::UpdateAccessibilityWidget (const TSharedRef<SContentIndexer>&
00345     ContentIndexer, const int32& NewIndex)
00346 {

```

```

00346     ContentIndexer->UpdateIndex(NewIndex);
00347 }
00348
00349 const TSharedRef<SContentIndexer> UAccessibilityGraphEditorContext::CreateAccessibilityWrapper(const
    TSharedRef<SWidget>& ContentToWrap, const int32& Index)
00350 {
00351     return SNew(SContentIndexer)
00352         .IndexValue(Index)
00353         .IndexPositionToContent(EIndexerPosition::Left)
00354         .ContentToIndex(ContentToWrap);
00355 }
00356

```

## 5.15 AccessibilityGraphLocomotionContext.cpp

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #include "AccessibilityWrappers/AccessibilityGraphLocomotionContext.h"
00004 #include "AccessibilityWidgets/SIndexer.h"
00005 #include "OpenAccessibilityLogging.h"
00006
00007 #include "SGraphPanel.h"
00008
00009 UAccessibilityGraphLocomotionContext::UAccessibilityGraphLocomotionContext(const FObjectInitializer&
    ObjectInitializer)
00010     : UPhraseTreeContextObject()
00011 {
00012     LinkedEditor = TWeakPtr<SGraphEditor>();
00013 }
00014
00015 UAccessibilityGraphLocomotionContext::~UAccessibilityGraphLocomotionContext()
00016 {
00017     Close();
00018
00019     UE_LOG(LogOpenAccessibility, Warning, TEXT("GraphLocomotion: CONTEXT DESTROYED."));
00020 }
00021
00022 void UAccessibilityGraphLocomotionContext::Init()
00023 {
00024     {
00025         TSharedPtr<SDockTab> ActiveTab = FGlobalTabmanager::Get()->GetActiveTab();
00026         if (!ActiveTab.IsValid())
00027         {
00028             UE_LOG(LogOpenAccessibility, Warning, TEXT("GraphLocomotion: NO ACTIVE TAB FOUND."));
00029             return;
00030         }
00031
00032         LinkedEditor = StaticCastSharedRef<SGraphEditor>(ActiveTab->GetContent());
00033         if (!LinkedEditor.IsValid())
00034         {
00035             UE_LOG(LogOpenAccessibility, Warning, TEXT("GraphLocomotion: CURRENT ACTIVE TAB IS NOT OF
                TYPE - SGraphEditor"));
00036             return;
00037         }
00038     }
00039
00040     TSharedPtr<SGraphEditor> LinkedEditorPtr = LinkedEditor.Pin();
00041
00042     Init(LinkedEditorPtr.ToSharedRef());
00043 }
00044
00045 void UAccessibilityGraphLocomotionContext::Init(const TSharedRef<SGraphEditor> InGraphEditor)
00046 {
00047     LinkedEditor = InGraphEditor;
00048
00049     InGraphEditor->GetViewLocation(StartViewPosition, StartViewZoom);
00050     InGraphEditor->ZoomToFit(false);
00051
00052     CreateVisualGrid(InGraphEditor);
00053     GenerateVisualChunks(InGraphEditor, FIntVector2(6, 4));
00054
00055     HideNativeVisuals();
00056
00057     BindFocusChangedEvent();
00058 }
00059
00060 bool UAccessibilityGraphLocomotionContext::SelectChunk(const int32& Index)
00061 {
00062     if (Index > ChunkArray.Num() || Index < 0)
00063         return false;
00064
00065     const FGraphLocomotionChunk SelectedChunk = ChunkArray[Index];
00066

```

```

00067     const SGraphPanel* LinkedPanel = LinkedEditor.Pin()->GetGraphPanel();
00068
00069     const FVector2D GraphTopLeftCoord =
LinkedPanel->PanelCoordToGraphCoord(SelectedChunk.GetChunkTopLeft());
00070     const FVector2D GraphBottomRightCoord =
LinkedPanel->PanelCoordToGraphCoord(SelectedChunk.GetChunkBottomRight());
00071
00072     ChangeChunkVis(Index, FLinearColor::Red);
00073
00074     GEditor->GetTimerManager()->SetTimer(
00075         SelectionTimerHandle,
00076         [this, Index, GraphTopLeftCoord, GraphBottomRightCoord]()
00077         {
00078             ChangeChunkVis(Index);
00079
00080             if (MoveViewport(GraphTopLeftCoord, GraphBottomRightCoord))
00081             {
00082                 if (CurrentViewPosition != FVector2D::ZeroVector)
00083                     PreviousPositions.Push(CurrentViewPosition);
00084
00085                 CurrentViewPosition = FPanelViewPosition(GraphTopLeftCoord, GraphBottomRightCoord);
00086             }
00087             else
00088             {
00089                 UE_LOG(LogOpenAccessibility, Log, TEXT("Failed To Jump To Viewport Coords (TopLeft: %s
| BottomRight: %s)"),
00090                     *GraphTopLeftCoord.ToString(), *GraphBottomRightCoord.ToString());
00091             }
00092         },
00093         0.5f,
00094         false
00095     );
00096
00097     return true;
00098 }
00099
00100 bool UAccessibilityGraphLocomotionContext::RevertToPreviousView()
00101 {
00102     if (PreviousPositions.IsEmpty())
00103     {
00104         LinkedEditor.Pin()->ZoomToFit(false);
00105         return true;
00106     }
00107
00108     if (!MoveViewport(PreviousPositions.Pop()))
00109     {
00110         return false;
00111     }
00112
00113     return true;
00114 }
00115
00116 void UAccessibilityGraphLocomotionContext::ConfirmSelection()
00117 {
00118     Close();
00119 }
00120
00121 void UAccessibilityGraphLocomotionContext::CancelLocomotion()
00122 {
00123     if (LinkedEditor.IsValid())
00124     {
00125         LinkedEditor.Pin()->SetViewLocation(StartViewPosition, StartViewZoom);
00126
00127         Close();
00128     }
00129 }
00130
00131 bool UAccessibilityGraphLocomotionContext::Close()
00132 {
00133     UnbindFocusChangedEvent();
00134
00135     if (SelectionTimerHandle.IsValid())
00136         GEditor->GetTimerManager()->ClearTimer(SelectionTimerHandle);
00137
00138     RemoveVisualGrid();
00139     UnHideNativeVisuals();
00140
00141     bIsActive = false;
00142
00143     RemoveFromRoot();
00144     MarkAsGarbage();
00145
00146     UE_LOG(LogOpenAccessibility, Warning, TEXT("GraphLocomotion: CONTEXT CLOSED."));
00147
00148     return true;
00149 }
00150

```

```

00151 bool UAccessibilityGraphLocomotionContext::MoveViewport(const FVector2D& InTopLeft, const FVector2D&
    InBottomRight) const
00152 {
00153     if (!LinkedEditor.IsValid())
00154         return false;
00155
00156     TSharedPtr<SGraphEditor> LinkedEditorPtr = LinkedEditor.Pin();
00157     SGraphPanel* LinkedPanel = LinkedEditorPtr->GetGraphPanel();
00158
00159     return LinkedPanel->JumpToRect(InTopLeft, InBottomRight);
00160 }
00161
00162 bool UAccessibilityGraphLocomotionContext::MoveViewport(const FPanelViewPosition& NewViewPosition)
    const
00163 {
00164     if (!LinkedEditor.IsValid())
00165         return false;
00166
00167     SGraphPanel* LinkedPanel = LinkedEditor.Pin()->GetGraphPanel();
00168
00169     return LinkedPanel->JumpToRect(NewViewPosition.TopLeft, NewViewPosition.BotRight);
00170 }
00171
00172 void UAccessibilityGraphLocomotionContext::ChangeChunkVis(const int32& Index, const FLinearColor&
    NewColor)
00173 {
00174     check(Index < ChunkArray.Num() && Index >= 0)
00175
00176     ChunkArray[Index].SetVisColor(NewColor);
00177 }
00178
00179 void UAccessibilityGraphLocomotionContext::CreateVisualGrid(const TSharedRef<SGraphEditor>
    InGraphEditor)
00180 {
00181     TSharedPtr<SOverlay> GraphViewport =
    StaticCastSharedPtr<SOverlay>(InGraphEditor->GetGraphPanel()->GetParentWidget());
00182     if (!GraphViewport.IsValid())
00183     {
00184         UE_LOG(LogOpenAccessibility, Warning, TEXT("GraphLocomotion: NO GRAPH VIEWPORT FOUND.));
00185         return;
00186     }
00187
00188     GridParent = GraphViewport;
00189
00190     GraphViewport->AddSlot()
00191         .ZOrder(1)
00192         .Valign(VAlign_Fill)
00193         .HAlign(HAlign_Fill)
00194         [
00195             SAssignNew(GridContainer, SUniformGridPanel)
00196         ];
00197 }
00198
00199 void UAccessibilityGraphLocomotionContext::GenerateVisualChunks(const TSharedRef<SGraphEditor>
    InGraphEditor, FIntVector2 InVisualChunkSize)
00200 {
00201     ChunkArray.Reset(InVisualChunkSize.X * InVisualChunkSize.Y);
00202     ChunkSize = InVisualChunkSize;
00203
00204     TSharedPtr<SUniformGridPanel> GridContainerPtr = GridContainer.Pin();
00205
00206     int32 ChunkIndex = -1;
00207     TSharedPtr<SBox> ChunkWidget;
00208     TSharedPtr<SBorder> ChunkVisWidget;
00209     TSharedPtr<SIndexer> ChunkIndexer;
00210
00211     for (int32 Y = 0; Y < InVisualChunkSize.Y; Y++)
00212     {
00213         for (int32 X = 0; X < InVisualChunkSize.X; X++)
00214         {
00215             ChunkIndex = X + (Y * InVisualChunkSize.X);
00216             FGraphLocomotionChunk& GraphChunk = ChunkArray.EmplaceAt_GetRef(ChunkIndex);
00217
00218             GridContainerPtr->AddSlot(X, Y)
00219             [
00220                 SAssignNew(ChunkWidget, SBox)
00221                 [
00222                     SAssignNew(ChunkVisWidget, SBorder)
00223                     .Padding(0.5f)
00224                     .BorderBackgroundColor(FLinearColor::Yellow)
00225                     [
00226                         SNew(SBorder)
00227                         .HAlign(HAlign_Center)
00228                         .Valign(VAlign_Center)
00229                         .BorderBackgroundColor(FLinearColor::Yellow)
00230                     [
00231                         SAssignNew(ChunkIndexer, SIndexer)

```

```

00232             .TextColor(FLinearColor::Yellow)
00233             .IndexValue(ChunkIndex)
00234         ]
00235     ]
00236 ]
00237 ];
00238
00239     GraphChunk.ChunkWidget = ChunkWidget;
00240     GraphChunk.ChunkVisWidget = ChunkVisWidget;
00241     GraphChunk.ChunkIndexer = ChunkIndexer;
00242 }
00243 }
00244
00245     CalculateVisualChunksBounds();
00246 }
00247
00248 void UAccessibilityGraphLocomotionContext::CalculateVisualChunksBounds()
00249 {
00250     if (!LinkedEditor.IsValid())
00251         return;
00252
00253     SGraphPanel* LinkedPanel = LinkedEditor.Pin()->GetGraphPanel();
00254     FVector2D PanelGeoSize = LinkedPanel->GetTickSpaceGeometry().GetLocalSize();
00255
00256     double ChunkWidgetSizeX = PanelGeoSize.X / ChunkSize.X;
00257     double ChunkWidgetSizeY = PanelGeoSize.Y / ChunkSize.Y;
00258
00259     FGraphLocomotionChunk Chunk;
00260     double ChunkX, ChunkY;
00261
00262     int32 ArrIndex;
00263     for (int Y = 0; Y < ChunkSize.Y; Y++)
00264     {
00265         for (int X = 0; X < ChunkSize.X; X++)
00266         {
00267             ArrIndex = (Y * ChunkSize.X) + X;
00268
00269             Chunk = ChunkArray[ArrIndex];
00270
00271             ChunkX = X * ChunkWidgetSizeX;
00272             ChunkY = Y * ChunkWidgetSizeY;
00273
00274             Chunk.SetChunkBounds(
00275                 FVector2D(ChunkX, ChunkY),
00276                 FVector2D(ChunkWidgetSizeX + ChunkX, ChunkWidgetSizeY + ChunkY)
00277             );
00278
00279             ChunkArray[ArrIndex] = Chunk;
00280         }
00281     }
00282 }
00283
00284 void UAccessibilityGraphLocomotionContext::RemoveVisualGrid()
00285 {
00286     TSharedPtr<SUniformGridPanel> GridContainerPtr = GridContainer.Pin();
00287     if (GridContainerPtr.IsValid())
00288     {
00289         TSharedPtr<SOverlay> ParentWidget = StaticCastSharedPtr<SOverlay>(
00290             GridContainerPtr->GetParentWidget()
00291         );
00292
00293         if (ParentWidget.IsValid()) {
00294             ParentWidget->RemoveSlot(GridContainerPtr.ToSharedRef());
00295
00296             GridParent = ParentWidget;
00297         }
00298         else UE_LOG(LogOpenAccessibility, Warning, TEXT("GraphLocomotion: PARENT WIDGET NOT FOUND,
00299             CANNOT REMOVE LOCOMOTION WIDGETS."))
00300     }
00301 }
00302
00303 void UAccessibilityGraphLocomotionContext::HideNativeVisuals()
00304 {
00305     NativeWidgetVisibility.Empty();
00306
00307     TSharedPtr<SOverlay> GraphViewport = GridParent.Pin();
00308     TSharedPtr<SUniformGridPanel> VisualGrid = GridContainer.Pin();
00309     SGraphPanel* GraphPanel = LinkedEditor.Pin()->GetGraphPanel();
00310
00311     FChildren* ViewportChildren = GraphViewport->GetChildren();
00312
00313     TSharedPtr<SWidget> ChildWidget;
00314     for (int32 i = 0; i < ViewportChildren->Num(); i++)
00315     {
00316         ChildWidget = ViewportChildren->GetChildAt(i);
00317
00318         if (ChildWidget != VisualGrid && ChildWidget.Get() != GraphPanel)

```

```

00318     {
00319         NativeWidgetVisibility.Add(ChildWidget.Get(), ChildWidget->GetVisibility());
00320     }
00321     ChildWidget->SetVisibility(EVisibility::Hidden);
00322 }
00323 }
00324 }
00325
00326 void UAccessibilityGraphLocomotionContext::UnHideNativeVisuals()
00327 {
00328     FChildren* ViewportChildren = GridParent.Pin()->GetChildren();
00329
00330     TSharedPtr<SWidget> ChildWidget;
00331     for (int32 i = 0; i < ViewportChildren->Num(); i++)
00332     {
00333         ChildWidget = ViewportChildren->GetChildAt(i);
00334
00335         if (NativeWidgetVisibility.Contains(ChildWidget.Get()))
00336         {
00337             ChildWidget->SetVisibility(NativeWidgetVisibility[ChildWidget.Get()]);
00338         }
00339     }
00340
00341     NativeWidgetVisibility.Empty();
00342 }
00343
00344 void UAccessibilityGraphLocomotionContext::OnFocusChanged(
00345     const FFocusEvent& FocusEvent,
00346     const FWeakWidgetPath& OldFocusedWidgetPath, const TSharedPtr<SWidget>& OldFocusedWidget,
00347     const FWidgetPath& NewFocusedWidgetPath, const TSharedPtr<SWidget>& NewFocusedWidget
00348 )
00349 {
00350     if (!bIsActive)
00351         return;
00352
00353     UE_LOG(LogOpenAccessibility, Warning, TEXT("GraphLocomotion: FOCUS CHANGED.));
00354
00355     TSharedPtr<SGraphEditor> LinkedEditorPtr = LinkedEditor.Pin();
00356
00357     if (!NewFocusedWidgetPath.ContainsWidget(LinkedEditorPtr.ToSharedRef()))
00358     {
00359         bIsActive = false;
00360         Close();
00361     }
00362 }
00363
00364 void UAccessibilityGraphLocomotionContext::BindFocusChangedEvent()
00365 {
00366     FocusChangedHandle = FSlateApplication::Get().OnFocusChanging()
00367         .AddUObject(this, &UAccessibilityGraphLocomotionContext::OnFocusChanged);
00368 }
00369
00370 void UAccessibilityGraphLocomotionContext::UnbindFocusChangedEvent()
00371 {
00372     if (FocusChangedHandle.IsValid())
00373     {
00374         FSlateApplication::Get().OnFocusChanging().Remove(FocusChangedHandle);
00375     }
00376 }

```

## 5.16 AccessibilityWindowToolbar.cpp

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #include "AccessibilityWrappers/AccessibilityWindowToolbar.h"
00004 #include "AccessibilityWidgets/SContentIndexer.h"
00005
00006 #include "PhraseTree/Containers/ParseRecord.h"
00007 #include "PhraseTree/Containers/Input/UParseIntInput.h"
00008
00009 UAccessibilityWindowToolbar::UAccessibilityWindowToolbar() : UObject()
00010 {
00011     LastToolkit = TWeakPtr<SWidget>();
00012     LastTopWindow = TWeakPtr<SWindow>();
00013     LastToolkitParent = TWeakPtr<SBorder>();
00014
00015     ConsoleCommands.Add(IConsoleManager::Get().RegisterConsoleCommand(
00016         TEXT("OpenAccessibiliy.ToolBar.ShowIndexerStats"),
00017         TEXT("Displays the Indexer Stats for the Toolbar."),
00018
00019         FConsoleCommandDelegate::CreateLambda([this]() {
00020             UE_LOG(LogOpenAccessibility, Display, TEXT("| ToolBar Indexer Stats | Indexed Amount: %d |
00021                 "), ToolbarIndex.Num())

```

```

00021     })
00022     });
00023
00024     BindTicker();
00025 }
00026
00027 UAccessibilityWindowToolBar::~UAccessibilityWindowToolBar()
00028 {
00029     UE_LOG(LogOpenAccessibility, Log, TEXT("AccessibilityToolBar: Destroyed.));
00030
00031     UnbindTicker();
00032 }
00033
00034 bool UAccessibilityWindowToolBar::Tick(float DeltaTime)
00035 {
00036     TSharedPtr<SWindow> TopWindow = FSlateApplication::Get().GetActiveTopLevelRegularWindow();
00037     if (!TopWindow.IsValid())
00038     {
00039         return true;
00040     }
00041
00042     TSharedPtr<SBorder> ContentContainer;
00043     if (TopWindow != LastTopWindow)
00044         ContentContainer = GetWindowContentContainer(TopWindow.ToSharedRef());
00045     else ContentContainer = LastToolkitParent.Pin();
00046
00047     if (!ContentContainer.IsValid())
00048     {
00049         return true;
00050     }
00051
00052     TSharedPtr<SWidget> Toolkit = ContentContainer->GetContent();
00053     if (!Toolkit.IsValid())
00054     {
00055         return true;
00056     }
00057
00058     if (ApplyToolBarIndexing(Toolkit.ToSharedRef(), TopWindow.ToSharedRef()))
00059     {
00060         LastToolkit = Toolkit;
00061         //UE_LOG(LogOpenAccessibility, Log, TEXT("AccessibilityToolBar: Toolkit Indexing Applied To
00062 %s"), *Toolkit->GetTypeAsString());
00063     }
00064
00065     LastTopWindow = TopWindow;
00066     LastToolkitParent = ContentContainer;
00067
00068     return true;
00069 }
00070
00071 bool UAccessibilityWindowToolBar::ApplyToolBarIndexing(TSharedRef<SWidget> ToolkitWidget,
TSharedRef<SWindow> ToolkitWindow)
00072 {
00073     TSharedPtr<SWidget> ToolBarContainer;
00074     if (!GetToolKitToolBar(ToolkitWidget, ToolBarContainer))
00075     {
00076         UE_LOG(LogOpenAccessibility, Log, TEXT("Failed to get Toolbar.));
00077         return false;
00078     }
00079
00080     if (!ToolBarContainer.IsValid())
00081     {
00082         UE_LOG(LogOpenAccessibility, Log, TEXT("Toolbar Container Is Not Valid.));
00083         return false;
00084     }
00085
00086     TArray<FChildren*> ChildrenToFilter = TArray<FChildren*> {
00087         ToolBarContainer->GetChildren()
00088     };
00089
00090     FString WidgetType;
00091     TSet<FString> AllowedWidgetTypes = TSet<FString>{
00092         TEXT("SToolBarButtonBlock"),
00093         TEXT("SToolBarComboButtonBlock"),
00094         TEXT("SToolBarStackButtonBlock"),
00095         TEXT("SUniformToolBarButtonBlock")
00096     };
00097
00098     ToolBarIndex.Reset();
00099
00100     int32 Index = -1;
00101     while (ChildrenToFilter.Num() > 0)
00102     {
00103         FChildren* Children = ChildrenToFilter[0];
00104         ChildrenToFilter.RemoveAt(0);
00105     }

```



```

00106         // To-Do: Learn How to Write Readable Code.
00107         for (int i = 0; i < Children->NumSlot(); i++)
00108         {
00109             FSlotBase& ChildSlot = const_cast<FSlotBase&>(Children->GetSlotAt(i));
00110
00111             TSharedPtr<SWidget> ChildWidget = Children->GetChildAt(i);
00112             if (!ChildWidget.IsValid() || ChildWidget->GetDesiredSize() == FVector2D::ZeroVector)
00113                 continue;
00114
00115             WidgetType = ChildWidget->GetTypeAsString();
00116
00117             if (ChildWidget.IsValid() && AllowedWidgetTypes.Contains(WidgetType))
00118             {
00119                 TSharedPtr<SMultiBlockBaseWidget> ToolBarButtonWidget =
00120                     StaticCastSharedPtr<SMultiBlockBaseWidget>(ChildWidget);
00121
00122                 ChildSlot.DetachWidget();
00123
00124                 ToolbarIndex.GetKeyOrAddValue(
00125                     ToolBarButtonWidget.Get(),
00126                     Index
00127                 );
00128
00129                 ChildSlot.AttachWidget(
00130                     SNew(SContentIndexer)
00131                     .IndexValue(Index)
00132                     .IndexPositionToContent(EIndexerPosition::Bottom)
00133                     .ContentToIndex(ToolBarButtonWidget)
00134                     .IndexVisibility_Lambda([this, ToolkitWidget]() -> EVisibility {
00135                         return (this->IsActiveToolBar(ToolkitWidget))
00136                             ? EVisibility::Visible
00137                             : EVisibility::Hidden;
00138                     })
00139                 );
00140             }
00141             else if (ChildWidget.IsValid() && WidgetType == "SContentIndexer")
00142             {
00143                 TSharedPtr<SContentIndexer> IndexerWidget =
00144                     StaticCastSharedPtr<SContentIndexer>(ChildWidget);
00145
00146                 TSharedPtr<SMultiBlockBaseWidget> IndexedContent =
00147                     StaticCastSharedPtr<SMultiBlockBaseWidget>(IndexerWidget->GetContent());
00148                 if (!IndexedContent.IsValid())
00149                     continue;
00150
00151                 ToolbarIndex.GetKeyOrAddValue(
00152                     IndexedContent.Get(),
00153                     Index
00154                 );
00155
00156                 IndexerWidget->UpdateIndex(Index);
00157             }
00158             else ChildrenToFilter.Add(ChildWidget->GetChildren());
00159         }
00160     }
00161     return true;
00162 }
00163
00164 // -- Util Widget Function --
00165
00166 template<typename T = SWidget>
00167 FORCEINLINE TSharedPtr<T> GetWidgetDescendantOfType(TSharedPtr<SWidget> Widget, FName TypeName)
00168 {
00169     if (Widget->GetType() == TypeName)
00170     {
00171         return Widget;
00172     }
00173
00174     TArray<FChildren*> ChildrenToFilter;
00175     ChildrenToFilter.Add(Widget->GetChildren());
00176
00177     while (ChildrenToFilter.Num() > 0)
00178     {
00179         FChildren* Children = ChildrenToFilter.Pop();
00180
00181         for (int i = 0; i < Children->Num(); i++)
00182         {
00183             TSharedPtr<SWidget> Child = Children->GetChildAt(i);
00184
00185             ChildrenToFilter.Add(Child->GetChildren());
00186
00187             if (Child->GetType() == TypeName)
00188             {
00189                 return StaticCastSharedPtr<T>(Child.ToSharedPtr());
00190             }
00191         }
00192     }

```

```

00190     }
00191
00192     return nullptr;
00193 }
00194
00195 // -- --
00196
00197 void UAccessibilityWindowToolBar::SelectToolBarItem(int32 Index)
00198 {
00199     if (ToolBarIndex.IsEmpty())
00200     {
00201         UE_LOG(LogOpenAccessibility, Warning, TEXT("ToolBar Index is Empty.))
00202         return;
00203     }
00204
00205     SMultiBlockBaseWidget* LinkedButton;
00206     if (!ToolBarIndex.GetValue(Index, LinkedButton))
00207     {
00208         UE_LOG(LogOpenAccessibility, Warning, TEXT("Provided Index is Not Linked to a ToolBar
00209         Button.))
00210         return;
00211     }
00212
00213     TSharedPtr<const FMultiBlock> MultiBlock = LinkedButton->GetBlock();
00214     if (!MultiBlock.IsValid())
00215     {
00216         UE_LOG(LogOpenAccessibility, Warning, TEXT("Provided ToolBar MultiBlock is Not Valid.))
00217     }
00218
00219     TSharedPtr<const FUICommandList> ActionList = MultiBlock->GetActionList();
00220     TSharedPtr<const FUICommandInfo> Action = MultiBlock->GetAction();
00221
00222     if (ActionList.IsValid() && Action.IsValid())
00223     {
00224         ActionList->ExecuteAction( Action.ToSharedRef() );
00225     }
00226     else
00227     {
00228         const FUIAction& DirectAction = MultiBlock->GetDirectActions();
00229         DirectAction.Execute();
00230     }
00231 }
00232
00233 bool UAccessibilityWindowToolBar::IsActiveToolBar(const TSharedRef<SWidget>& ToolkitWidget)
00234 {
00235     return LastToolkit.IsValid()
00236         ? LastToolkit.Pin() == ToolkitWidget
00237         : false;
00238 }
00239
00240 TSharedPtr<SWidget> UAccessibilityWindowToolBar::GetActiveToolkitWidget() const
00241 {
00242     if (LastToolkit.IsValid())
00243         return LastToolkit.Pin();
00244
00245     return TSharedPtr<SWidget>();
00246 }
00247
00248 TSharedPtr<SBorder> UAccessibilityWindowToolBar::GetWindowContentContainer(TSharedRef<SWindow>
00249     WindowToFindContainer)
00250 {
00251     // Find SDockingTabStack
00252     TSharedPtr<SWidget> DockingTabStack = GetWidgetDescendantOfType(WindowToFindContainer,
00253         "SDockingTabStack");
00254     if (!DockingTabStack.IsValid())
00255     {
00256         UE_LOG(LogOpenAccessibility, Log, TEXT("DockingTabStack is not Valid"));
00257         return nullptr;
00258     }
00259
00260     return StaticCastSharedRef<SBorder>(
00261         DockingTabStack
00262         ->GetChildren()->GetChildAt(0) // SVerticalBox
00263         ->GetChildren()->GetChildAt(1) // SOverlay
00264         ->GetChildren()->GetChildAt(0) // SBorder
00265     );
00266 }
00267
00268 bool UAccessibilityWindowToolBar::GetToolKitToolBar(TSharedRef<SWidget> ToolKitWidget,
00269     TSharedPtr<SWidget>& OutToolBar)
00270 {
00271     TSharedPtr<SWidget> CurrChild;
00272     FChildren* CurrChildren = ToolKitWidget->GetChildren();
00273     if (CurrChildren->Num() == 0)
00274         return false;
00275 }

```

```

00273     CurrChild = CurrChildren->GetChildAt(0); // Get SVerticalBox
00274     CurrChildren = CurrChild->GetChildren();
00275     if (CurrChildren->Num() == 0)
00276         return false;
00277
00278     OutToolBar = CurrChildren->GetChildAt(0); // Get SHorizontalBox
00279     if (!OutToolBar.IsValid())
00280         return false;
00281
00282     return true;
00283 }
00284
00285 void UAccessibilityWindowToolBar::BindTicker()
00286 {
00287     FTickerDelegate TickDelegate = FTickerDelegate::CreateUObject(this,
        &UAccessibilityWindowToolBar::Tick);
00288
00289     TickDelegateHandle = FTSTicker::GetCoreTicker()
        .AddTicker(TickDelegate);
00290 }
00291
00292 void UAccessibilityWindowToolBar::UnbindTicker()
00293 {
00294     FTSTicker::GetCoreTicker()
        .RemoveTicker(TickDelegateHandle);
00295 }
00296
00297
00298 }

```

## 5.17 AssetAccessibilityRegistry.cpp

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #include "AssetAccessibilityRegistry.h"
00004 #include "OpenAccessibilityLogging.h"
00005 #include "BehaviorTree/BehaviorTree.h"
00006
00007 #include "Subsystems/AssetEditorSubsystem.h"
00008 #include "EdGraph/EdGraph.h"
00009 #include "EdGraph/EdGraphNode.h"
00010 #include "MaterialGraph/MaterialGraph.h"
00011
00012 #include "UObject/Class.h"
00013 #include "Misc/Guid.h"
00014
00015 FAssetAccessibilityRegistry::FAssetAccessibilityRegistry()
00016 {
00017     GraphAssetIndex = TMap<FGuid, TSharedPtr<FGraphIndexer>>();
00018     //GameWorldAssetIndex = TMap<FGuid, FGameWorldIndexer*>();
00019
00020     AssetOpenedInEditorHandle =
        GEditor->GetEditorSubsystem<UAssetEditorSubsystem>()->OnAssetOpenedInEditor()
        .AddRaw(this, &FAssetAccessibilityRegistry::OnAssetOpenedInEditor);
00021
00022     AssetEditorRequestCloseHandle =
        GEditor->GetEditorSubsystem<UAssetEditorSubsystem>()->OnAssetEditorRequestClose()
        .AddRaw(this, &FAssetAccessibilityRegistry::OnAssetEditorRequestClose);
00023 }
00024
00025 ~FAssetAccessibilityRegistry()
00026 {
00027     GEditor->GetEditorSubsystem<UAssetEditorSubsystem>()->OnAssetOpenedInEditor()
        .Remove(AssetOpenedInEditorHandle);
00028
00029     GEditor->GetEditorSubsystem<UAssetEditorSubsystem>()->OnAssetEditorRequestClose()
        .Remove(AssetEditorRequestCloseHandle);
00030
00031     EmptyGraphAssetIndex();
00032 }
00033
00034 void FAssetAccessibilityRegistry::OnAssetOpenedInEditor(UObject* OpenedAsset, IAssetEditorInstance*
    EditorInstance)
00035 {
00036     UE_LOG(LogOpenAccessibility, Log, TEXT("|| AssetRegistry || Asset { %s } Opened In Editor: { %s }
    ||"), *OpenedAsset->GetName(), *EditorInstance->GetEditorName().ToString());
00037
00038     // Find Asset Type for correct Parsing.
00039     if (UBlueprint* OpenedBlueprint = Cast<UBlueprint>(OpenedAsset))
00040     {
00041         UE_LOG(LogOpenAccessibility, Log, TEXT("|| AssetRegistry || Asset { %s } Is A Blueprint ||"),
            *OpenedBlueprint->GetName());
00042
00043         RegisterBlueprintAsset(OpenedBlueprint);
00044     }
00045 }

```

```

00049     else if (UMaterial* OpenedMaterial = Cast<UMaterial>(OpenedAsset))
00050     {
00051         UE_LOG(LogOpenAccessibility, Log, TEXT("|| AssetRegistry || Asset { %s } Is A Material ||"),
        *OpenedMaterial->GetName());
00052
00053         RegisterMaterialAsset(OpenedMaterial);
00054     }
00055     else if (UBehaviorTree* OpenedBehaviorTree = Cast<UBehaviorTree>(OpenedAsset))
00056     {
00057         UE_LOG(LogOpenAccessibility, Log, TEXT("|| AssetRegistry || Asset { %s } Is A Behavior Tree
        ||"), *OpenedBehaviorTree->GetName());
00058
00059         RegisterBehaviorTreeAsset(OpenedBehaviorTree);
00060     }
00061 }
00062
00063 void FAssetAccessibilityRegistry::OnAssetEditorRequestClose(UObject* ClosingAsset,
        EAssetEditorCloseReason CloseReason)
00064 {
00065     if (ClosingAsset == nullptr)
00066         return;
00067
00068     UE_LOG(LogOpenAccessibility, Log, TEXT("|| AssetRegistry || Asset { %s } Closed | Reason: { %d }
        ||"), *ClosingAsset->GetFName().ToString(), int64(CloseReason));
00069 }
00070
00071 bool FAssetAccessibilityRegistry::IsGraphAssetRegistered(const UEdGraph* InUEdGraph) const
00072 {
00073     return GraphAssetIndex.Contains(InUEdGraph->GraphGuid);
00074 }
00075
00076 bool FAssetAccessibilityRegistry::RegisterGraphAsset(const UEdGraph* InGraph)
00077 {
00078     if (!InGraph->IsValidLowLevel())
00079         return false;
00080
00081     GraphAssetIndex.Add(InGraph->GraphGuid, MakeShared<FGraphIndexer>(InGraph));
00082
00083     for (auto& ChildGraph : InGraph->SubGraphs)
00084     {
00085         if (!RegisterGraphAsset(ChildGraph))
00086         {
00087             UE_LOG(LogOpenAccessibility, Error, TEXT("|| AssetRegistry || Error When Logging Child
            Graph: { %s } From Parent: { %s }||"), *ChildGraph->GetName(), *InGraph->GetName())
00088
00089             return false;
00090         }
00091     }
00092
00093     return true;
00094 }
00095
00096 bool FAssetAccessibilityRegistry::RegisterGraphAsset(const UEdGraph* InGraph, const
        TSharedRef<FGraphIndexer> InGraphIndexer)
00097 {
00098     if (!InGraph->IsValidLowLevel())
00099         return false;
00100
00101     GraphAssetIndex.Add(InGraph->GraphGuid, InGraphIndexer.ToSharedPtr());
00102
00103     for (auto& ChildGraph : InGraph->SubGraphs)
00104     {
00105         if (!RegisterGraphAsset(ChildGraph))
00106         {
00107             UE_LOG(LogOpenAccessibility, Error, TEXT("|| AssetRegistry || Error When Logging Child
            Graph: { %s } From Parent: { %s }||"), *ChildGraph->GetName(), *InGraph->GetName());
00108             return false;
00109         }
00110     }
00111
00112     return true;
00113 }
00114
00115 bool FAssetAccessibilityRegistry::UnregisterGraphAsset(const UEdGraph* UEdGraph)
00116 {
00117     GraphAssetIndex.Remove(UEdGraph->GraphGuid);
00118
00119     for (auto& ChildGraph : UEdGraph->SubGraphs)
00120     {
00121         if (!UnregisterGraphAsset(ChildGraph))
00122         {
00123             UE_LOG(LogOpenAccessibility, Error, TEXT("|| AssetRegistry || Error When Unregistering
            Child Graph: { %s } From Parent: { %s }||"), *ChildGraph->GetName(), *UEdGraph->GetName())
00124
00125             return false;
00126         }
00127     }

```

```

00128
00129     return true;
00130 }
00131
00132 void FAssetAccessibilityRegistry::GetAllGraphKeyIndexes(TArray<FGuid>& OutGraphKeys) const
00133 {
00134     GraphAssetIndex.GetKeys(OutGraphKeys);
00135 }
00136
00137 TArray<FGuid> FAssetAccessibilityRegistry::GetAllGraphKeyIndexes() const
00138 {
00139     TArray<FGuid> GraphKeys;
00140     GraphAssetIndex.GetKeys(GraphKeys);
00141
00142     return GraphKeys;
00143 }
00144
00145 void FAssetAccessibilityRegistry::GetAllGraphIndexes(TArray<TSharedPtr<FGraphIndexer>&
    OutGraphIndexes) const
00146 {
00147     return GraphAssetIndex.GenerateValueArray(OutGraphIndexes);
00148 }
00149
00150 TArray<TSharedPtr<FGraphIndexer> FAssetAccessibilityRegistry::GetAllGraphIndexes()
00151 {
00152     TArray<TSharedPtr<FGraphIndexer> GraphIndexArray;
00153
00154     GraphAssetIndex.GenerateValueArray(GraphIndexArray);
00155
00156     return GraphIndexArray;
00157 }
00158
00159 bool FAssetAccessibilityRegistry::IsGameWorldAssetRegistered(const UWorld* UWorld) const
00160 {
00161     throw std::exception("The method or operation is not implemented.");
00162 }
00163
00164 bool FAssetAccessibilityRegistry::RegisterGameWorldAsset(const UWorld* UWorld)
00165 {
00166     throw std::exception("The method or operation is not implemented.");
00167 }
00168
00169 bool FAssetAccessibilityRegistry::UnregisterGameWorldAsset(const UWorld* UWorld)
00170 {
00171     throw std::exception("The method or operation is not implemented.");
00172 }
00173
00174 void FAssetAccessibilityRegistry::EmptyGraphAssetIndex()
00175 {
00176     for (auto& GraphIndexer : GraphAssetIndex)
00177     {
00178         GraphIndexer.Value.Reset();
00179     }
00180
00181     GraphAssetIndex.Empty();
00182 }
00183
00184 void FAssetAccessibilityRegistry::EmptyGameWorldAssetIndex()
00185 {
00186     throw std::exception("The method or operation is not implemented.");
00187 }
00188
00189 void FAssetAccessibilityRegistry::RegisterBlueprintAsset(const UBlueprint* InBlueprint)
00190 {
00191     // Register the Blueprint's Graphs
00192     TArray<UEdGraph*> Graphs;
00193
00194     InBlueprint->GetAllGraphs(Graphs);
00195     for (auto& Graph : Graphs)
00196     {
00197         RegisterGraphAsset(Graph);
00198     }
00199
00200     // Register the Blueprint's World
00201     // Some Blueprints have no connected World / GameObjects,
00202     // so we need to check if the World is valid
00203
00204     UWorld* BlueprintDebugWorld = InBlueprint->GetWorldBeingDebugged();
00205     if (BlueprintDebugWorld != nullptr)
00206     {
00207         RegisterUWorldAsset(BlueprintDebugWorld);
00208     }
00209 }
00210
00211 void FAssetAccessibilityRegistry::RegisterMaterialAsset(const UMaterial* InMaterial)
00212 {
00213     if (InMaterial->MaterialGraph.IsNull())

```

```

00214         return;
00215
00216         TSharedPtr<FGraphIndexer> GraphIndexer =
            MakeShared<FGraphIndexer>(InMaterial->MaterialGraph.Get());
00217
00218         RegisterGraphAsset(InMaterial->MaterialGraph.Get(), GraphIndexer.ToSharedRef());
00219     }
00220
00221 void FAssetAccessibilityRegistry::RegisterBehaviorTreeAsset(const UBehaviorTree* InBehaviorTree)
00222 {
00223     if (InBehaviorTree->BTGraph->IsValidLowLevel())
00224     {
00225         RegisterGraphAsset(InBehaviorTree->BTGraph);
00226     }
00227 }
00228
00229 void FAssetAccessibilityRegistry::RegisterUWorldAsset(const UWorld* InWorld)
00230 {
00231     throw std::exception("The method or operation is not implemented.");
00232 }

```

## 5.18 GraphIndexer.cpp

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003
00004 #include "GraphIndexer.h"
00005
00006 #include "EdGraph/EdGraph.h"
00007 #include "EdGraph/EdGraphNode.h"
00008 #include "EdGraph/EdGraphPin.h"
00009 #include "GraphEditAction.h"
00010 #include "OpenAccessibilityLogging.h"
00011
00012 FGraphIndexer::FGraphIndexer()
00013 {
00014
00015 }
00016
00017 FGraphIndexer::FGraphIndexer(const UEdGraph* GraphToIndex)
00018     : LinkedGraph(const_cast<UEdGraph*>(GraphToIndex))
00019 {
00020     BuildGraphIndex();
00021
00022     OnGraphChangedHandle = LinkedGraph->AddOnGraphChangedHandler(
00023         FOnGraphChanged::FDelegate::CreateRaw(this, &FGraphIndexer::OnGraphEvent)
00024     );
00025 }
00026
00027 FGraphIndexer::~FGraphIndexer()
00028 {
00029     IndexMap.Empty();
00030     NodeSet.Empty();
00031     AvailableIndices.Empty();
00032
00033     LinkedGraph->RemoveOnGraphChangedHandler(OnGraphChangedHandle);
00034
00035     LinkedGraph = nullptr;
00036 }
00037
00038 bool FGraphIndexer::ContainsKey(const int& InKey)
00039 {
00040     return IndexMap.Contains(InKey);
00041 }
00042
00043 int FGraphIndexer::ContainsNode(UEdGraphNode* InNode)
00044 {
00045     check(InNode != nullptr);
00046
00047     if (!InNode->IsValidLowLevelFast() || !NodeSet.Contains(InNode->GetUniqueID()))
00048         return -1;
00049
00050     const int* ReturnedIndex = IndexMap.FindKey(InNode);
00051
00052     if (ReturnedIndex != nullptr)
00053     {
00054         return *ReturnedIndex;
00055     }
00056     else return -1;
00057 }
00058
00059 void FGraphIndexer::ContainsNode(UEdGraphNode* InNode, int& OutIndex)
00060 {

```

```

00061     OutIndex = ContainsNode(InNode);
00062 }
00063
00064 int FGraphIndexer::GetKey(const UEdGraphNode* InNode)
00065 {
00066     check(InNode != nullptr);
00067
00068     if (!InNode->IsValidLowLevelFast())
00069         return -1;
00070
00071     const int* FoundKey = IndexMap.FindKey(const_cast<UEdGraphNode*>(InNode));
00072
00073     if (FoundKey != nullptr) return *FoundKey;
00074     else return -1;
00075 }
00076
00077 bool FGraphIndexer::GetKey(const UEdGraphNode* InNode, int& OutKey)
00078 {
00079     check(InNode != nullptr);
00080
00081     if (!InNode->IsValidLowLevelFast())
00082         return false;
00083
00084     const int* FoundKey = IndexMap.FindKey(const_cast<UEdGraphNode*>(InNode));
00085     if (FoundKey != nullptr)
00086     {
00087         OutKey = *FoundKey;
00088         return true;
00089     }
00090     else return false;
00091 }
00092
00093 UEdGraphNode* FGraphIndexer::GetNode(const int& InIndex)
00094 {
00095     if (!IndexMap.Contains(InIndex))
00096     {
00097         UE_LOG(LogOpenAccessibility, Warning, TEXT("Provided Index is not recognised"))
00098
00099         return nullptr;
00100     }
00101
00102     return IndexMap[InIndex];
00103 }
00104
00105 void FGraphIndexer::GetPin(const int& InNodeIndex, const int& InPinIndex, UEdGraphPin* OutPin)
00106 {
00107     UEdGraphNode* Node = GetNode(InNodeIndex);
00108     if (Node == nullptr)
00109     {
00110         UE_LOG(LogOpenAccessibility, Warning, TEXT("Requested Node at index %d is not valid."),
00111             InNodeIndex);
00112         return;
00113     }
00114
00115     OutPin = Node->GetPinAt(InPinIndex); // Returns nullptr if invalid
00116 }
00117
00118 UEdGraphPin* FGraphIndexer::GetPin(const int& InNodeIndex, const int& InPinIndex)
00119 {
00120     UEdGraphNode* Node = GetNode(InNodeIndex);
00121     if (Node == nullptr)
00122     {
00123         UE_LOG(LogOpenAccessibility, Warning, TEXT("Requested Node at index %d is not valid."),
00124             InNodeIndex);
00125         return nullptr;
00126     }
00127
00128     return Node->GetPinAt(InPinIndex); // Returns nullptr if invalid
00129 }
00130
00131 void FGraphIndexer::GetNode(const int& InIndex, UEdGraphNode* OutNode)
00132 {
00133     OutNode = GetNode(InIndex);
00134 }
00135
00136 int FGraphIndexer::AddNode(const UEdGraphNode* InNode)
00137 {
00138     check(InNode != nullptr);
00139
00140     if (!InNode->IsValidLowLevelFast())
00141     {
00142         UE_LOG(LogOpenAccessibility, Warning, TEXT("Provided Node is not valid.))
00143     }
00144
00145     int Index = ContainsNode(const_cast<UEdGraphNode*>(InNode));
00146     if (Index != -1)
00147     {

```

```

00146         return Index;
00147     }
00148
00149     GetAvailableIndex(Index);
00150
00151     NodeSet.Add(InNode->GetUniqueID());
00152     IndexMap.Add(Index, const_cast<UEdGraphNode*>(InNode));
00153
00154     return Index;
00155 }
00156
00157 void FGraphIndexer::AddNode(int& OutIndex, const UEdGraphNode& InNode)
00158 {
00159     OutIndex = AddNode(&InNode);
00160 }
00161
00162 int FGraphIndexer::GetOrAddNode(const UEdGraphNode* InNode)
00163 {
00164     int Key = GetKey(InNode);
00165     if (Key != -1)
00166     {
00167         return Key;
00168     }
00169
00170     return AddNode(InNode);
00171 }
00172
00173 void FGraphIndexer::GetOrAddNode(const UEdGraphNode* InNode, int& OutIndex)
00174 {
00175     OutIndex = GetKey(InNode);
00176     if (OutIndex != -1)
00177     {
00178         return;
00179     }
00180
00181     OutIndex = AddNode(InNode);
00182 }
00183
00184 void FGraphIndexer::RemoveNode(const int& InIndex)
00185 {
00186     if (!IndexMap.Contains(InIndex))
00187     {
00188         UE_LOG(LogOpenAccessibility, Warning, TEXT("Provided Index is not recognised"))
00189     }
00190
00191     const UEdGraphNode* Node = IndexMap[InIndex];
00192
00193     if (Node->IsValidLowLevelFast())
00194     {
00195         NodeSet.Remove(Node->GetUniqueID());
00196         IndexMap.Remove(InIndex);
00197         AvailableIndices.Enqueue(InIndex);
00198     }
00199     else
00200     {
00201         UE_LOG(LogOpenAccessibility, Warning, TEXT("Stored Node in IndexMap is not valid."))
00202     }
00203 }
00204
00205 void FGraphIndexer::RemoveNode(const UEdGraphNode* InNode)
00206 {
00207     check(InNode != nullptr);
00208
00209     int Key = GetKey(InNode);
00210     if (Key == -1)
00211     {
00212         UE_LOG(LogOpenAccessibility, Warning, TEXT("Node does not exist in IndexMap."))
00213         return;
00214     }
00215
00216     RemoveNode(Key);
00217 }
00218
00219
00220
00221 // -----
00222 // Graph Events
00223 // -----
00224
00225 void FGraphIndexer::OnGraphEvent(const FEdGraphEditAction& InAction)
00226 {
00227     if (InAction.Graph != LinkedGraph)
00228     {
00229         return;
00230     }
00231
00232     switch (InAction.Action)

```



```

00233     {
00234         case EEdGraphActionType::GRAPHACTION_AddNode:
00235         {
00236             for (const UEdGraphNode* Node : InAction.Nodes)
00237             {
00238                 AddNode(Node);
00239             }
00240
00241             break;
00242         }
00243
00244         case EEdGraphActionType::GRAPHACTION_RemoveNode:
00245         {
00246             for (const UEdGraphNode* Node : InAction.Nodes)
00247             {
00248                 RemoveNode(Node);
00249             }
00250
00251             break;
00252         }
00253     }
00254 }
00255
00256 void FGraphIndexer::OnGraphRebuild()
00257 {
00258     IndexMap.Reset();
00259     NodeSet.Reset();
00260     AvailableIndices.Empty();
00261
00262     BuildGraphIndex();
00263 }
00264
00265 int FGraphIndexer::GetAvailableIndex()
00266 {
00267     if (!AvailableIndices.IsEmpty())
00268     {
00269         int Index;
00270         if (AvailableIndices.Dequeue(Index))
00271             return Index;
00272     }
00273
00274     return IndexMap.Num();
00275 }
00276
00277 void FGraphIndexer::GetAvailableIndex(int& OutIndex)
00278 {
00279     if (!AvailableIndices.IsEmpty() && AvailableIndices.Dequeue(OutIndex))
00280     {
00281         return;
00282     }
00283     else OutIndex = IndexMap.Num();
00284 }
00285
00286 void FGraphIndexer::BuildGraphIndex()
00287 {
00288     if (LinkedGraph == nullptr)
00289         return;
00290
00291     for (TObjectPtr<UEdGraphNode> Node : LinkedGraph->Nodes)
00292     {
00293         AddNode(Node);
00294     }
00295 }

```

## 5.19 OAccessibilityNodeFactory.cpp

```

00001 // Fill out your copyright notice in the Description page of Project Settings.
00002
00003
00004 #include "OAccessibilityNodeFactory.h"
00005 #include "OpenAccessibilityLogging.h"
00006
00007 #include "Logging/StructuredLog.h"
00008
00009 #include "NodeFactory.h"
00010 #include "EdGraphUtilities.h"
00011
00012 #include "Styling/AppStyle.h"
00013 #include "SGraphPanel.h"
00014 #include "SNodePanel.h"
00015 #include "SGraphNode.h"
00016 #include "SGraphPin.h"
00017 #include "Widgets/SBoxPanel.h"

```

```

00018 #include "Widgets/Text/STextBlock.h"
00019
00020 #include "OpenAccessibility.h"
00021 #include "AccessibilityWidgets/SIndexer.h"
00022
00023 FAccessibilityNodeFactory::FAccessibilityNodeFactory() : FGraphPanelNodeFactory()
00024 {
00025     UE_LOG_FMT(LogOpenAccessibility, Display, "Accessibility Node Factory Constructed");
00026 }
00027
00028 FAccessibilityNodeFactory::~FAccessibilityNodeFactory()
00029 {
00030 }
00031 }
00032
00033 TSharedPtr<class SGraphNode> FAccessibilityNodeFactory::CreateNode(UEdGraphNode* InNode) const
00034 {
00035     UE_LOG(LogOpenAccessibility, Display, TEXT("Accessibility Node Factory Used to construct %s"),
00036         *InNode->GetName());
00037
00038     check(InNode);
00039
00040     // Hack to get around the possible infinite loop of using
00041     // this factory to create the node from the factory itself.
00042
00043     FEdGraphUtilities::UnregisterVisualNodeFactory(FOpenAccessibilityModule::Get().AccessibilityNodeFactory);
00044     TSharedPtr<SGraphNode> OutNode = FNodeFactory::CreateNodeWidget(InNode);
00045
00046     FEdGraphUtilities::RegisterVisualNodeFactory(FOpenAccessibilityModule::Get().AccessibilityNodeFactory);
00047
00048     // Get Node Accessibility Index, from registry
00049     TSharedPtr<FGraphIndexer> GraphIndexer = FOpenAccessibilityModule::Get()
00050         .AssetAccessibilityRegistry->GetGraphIndexer(InNode->GetGraph());
00051
00052     int NodeIndex = -1;
00053     GraphIndexer->GetOrAddNode(InNode, NodeIndex);
00054
00055     {
00056         // Create Accessibility Widgets For Pins
00057         TArray<UEdGraphPin*> Pins = InNode->GetAllPins();
00058         TSharedPtr<SGraphPin> PinWidget;
00059
00060         for (int i = 0; i < Pins.Num(); i++)
00061         {
00062             UEdGraphPin* Pin = Pins[i];
00063
00064             PinWidget = OutNode->FindWidgetForPin(Pin);
00065             if (!PinWidget.IsValid())
00066             {
00067                 continue;
00068             }
00069
00070             WrapPinWidget(Pin, PinWidget.ToSharedRef(), i, OutNode.Get());
00071         }
00072
00073         PinWidget.Reset();
00074
00075         // Wrap The Node Widget
00076         WrapNodeWidget(InNode, OutNode.ToSharedRef(), NodeIndex);
00077
00078         return OutNode;
00079     }
00080
00081 void FAccessibilityNodeFactory::WrapNodeWidget(UEdGraphNode* Node, TSharedPtr<SGraphNode> NodeWidget,
00082     int NodeIndex) const
00083 {
00084     TSharedPtr<SWidget> WidgetToWrap = NodeWidget->GetSlot(ENodeZone::Center)->GetWidget();
00085     check(WidgetToWrap != SNullWidget::NullWidget);
00086
00087     NodeWidget->GetOrAddSlot(ENodeZone::Center)
00088         .Halign(HAlign_Fill)
00089         [
00090             SNew(SVerticalBox)
00091
00092             + SVerticalBox::Slot()
00093             .Halign(HAlign_Fill)
00094             .AutoHeight()
00095             .Padding(FMargin(1.5f, 0.25f))
00096             [
00097                 SNew(SOverlay)
00098
00099                 + SOverlay::Slot()
00100                 [
00101                     SNew(SImage)
00102                     .Image(FAppStyle::Get().GetBrush("Graph.Node.Body"))
00103                 ]
00104             ]
00105         ]

```

```

00101
00102         + SOverlay::Slot()
00103         .Padding(FMargin(4.0f, 0.0f))
00104     [
00105         SNew(SHorizontalBox)
00106         + SHorizontalBox::Slot()
00107         .HAlign(HAlign_Right)
00108         .VAlign(VAlign_Center)
00109         .Padding(1.f)
00110     [
00111         SNew(SOverlay)
00112         + SOverlay::Slot()
00113     [
00114         SNew(SIndexer)
00115         .IndexValue(NodeIndex)
00116         .TextColor(FLinearColor::White)
00117         .BorderColor(FLinearColor::Gray)
00118     ]
00119     ]
00120 ]
00121 ]
00122
00123     + SVerticalBox::Slot()
00124     .HAlign(HAlign_Fill)
00125     .AutoHeight()
00126     [
00127         WidgetToWrap
00128     ]
00129 ];
00130 }
00131
00132 void FAccessibilityNodeFactory::WrapPinWidget(UEdGraphPin* Pin, TSharedRef<SGraphPin> PinWidget, int
PinIndex, SGraphNode* OwnerNode) const
00133 {
00134     TSharedRef<SWidget> PinWidgetContent = PinWidget->GetContent();
00135     check(PinWidgetContent != SNullWidget::NullWidget);
00136
00137     TSharedRef<SWidget> AccessibilityWidget = SNew(SOverlay)
00138         .Visibility_Lambda([OwnerNode]() -> EVisibility {
00139
00140             if (OwnerNode->HasAnyUserFocusOrFocusedDescendants() || OwnerNode->IsHovered() ||
OwnerNode->GetOwnerPanel()->SelectionManager.IsNodeSelected(OwnerNode->GetNodeObj()))
00141                 return EVisibility::Visible;
00142
00143             return EVisibility::Hidden;
00144         })
00145     + SOverlay::Slot()
00146     [
00147         SNew(SIndexer)
00148         .IndexValue(PinIndex)
00149         .TextColor(FLinearColor::White)
00150         .BorderColor(FLinearColor::Gray)
00151     ];
00152
00153     switch (Pin->Direction)
00154     {
00155     case EEdGraphPinDirection::EGPD_Input:
00156     {
00157         PinWidget->SetContent(
00158             SNew(SHorizontalBox)
00159             + SHorizontalBox::Slot()
00160             .AutoWidth()
00161             [
00162                 PinWidgetContent
00163             ]
00164             + SHorizontalBox::Slot()
00165             .AutoWidth()
00166             [
00167                 AccessibilityWidget
00168             ]
00169         );
00170
00171         break;
00172     }
00173
00174     case EEdGraphPinDirection::EGPD_Output:
00175     {
00176         PinWidget->SetContent(
00177             SNew(SHorizontalBox)
00178             + SHorizontalBox::Slot()
00179             .AutoWidth()
00180             [
00181                 AccessibilityWidget
00182             ]
00183             + SHorizontalBox::Slot()
00184             .AutoWidth()
00185             [

```

```

00186             PinWidgetContent
00187         }
00188     };
00189     break;
00190 }
00191
00192     default:
00193     {
00194         UE_LOG(LogOpenAccessibility, Error, TEXT("Pin Direction Not Recognized"));
00195         break;
00196     }
00197 }
00198 }

```

## 5.20 OAEEditorAccessibilityManager.cpp

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003
00004 #include "OAEEditorAccessibilityManager.h"
00005
00006 OAEEditorAccessibilityManager::OAEEditorAccessibilityManager()
00007 {
00008 }
00009
00010 OAEEditorAccessibilityManager::~OAEEditorAccessibilityManager()
00011 {
00012 }

```

## 5.21 OpenAccessibility.cpp

```

00001 // Copyright Epic Games, Inc. All Rights Reserved.
00002
00003 #include "OpenAccessibility.h"
00004 #include "OpenAccessibilityCommunication.h"
00005 #include "OpenAccessibilityLogging.h"
00006
00007 #include "PhraseTree/PhraseNode.h"
00008 #include "PhraseTree/PhraseInputNode.h"
00009 #include "PhraseTree/PhraseStringInputNode.h"
00010 #include "PhraseTree/PhraseDirectionalInputNode.h"
00011 #include "PhraseTree/PhraseContextNode.h"
00012 #include "PhraseTree/PhraseContextMenuNode.h"
00013 #include "PhraseTree/PhraseEventNode.h"
00014
00015 #include "PhraseEvents/LocalizedInputLibrary.h"
00016 #include "PhraseEvents/WindowInteractionLibrary.h"
00017 #include "PhraseEvents/ViewInteractionLibrary.h"
00018 #include "PhraseEvents/NodeInteractionLibrary.h"
00019
00020 #include "TranscriptionVisualizer.h"
00021 #include "AccessibilityWrappers/AccessibilityAddNodeContextMenu.h"
00022 #include "AccessibilityWrappers/AccessibilityGraphLocomotionContext.h"
00023
00024 #include "GraphActionNode.h"
00025 #include "SGraphPanel.h"
00026 #include "AccessibilityWrappers/AccessibilityGraphEditorContext.h"
00027 #include "Widgets/Text/SMultiLineEditableText.h"
00028 #include "Widgets/Input/SSearchBar.h"
00029
00030 #include "Framework/Docking/TabManager.h"
00031 #include "HAL/PlatformFileManager.h"
00032 #include "Misc/FileHelper.h"
00033 #include "Logging/StructuredLog.h"
00034
00035 #define LOCTEXT_NAMESPACE "FOpenAccessibilityModule"
00036
00037 void FOpenAccessibilityModule::StartupModule()
00038 {
00039     UE_LOG(LogOpenAccessibility, Display, TEXT("OpenAccessibilityModule::StartupModule()"));
00040
00041     // Create the Asset Registry
00042     AssetAccessibilityRegistry = MakeShared<FAssetAccessibilityRegistry, ESPMode::ThreadSafe>();
00043
00044     // Register the Accessibility Node Factory
00045     AccessibilityNodeFactory = MakeShared<FAccessibilityNodeFactory, ESPMode::ThreadSafe>();
00046     FEdGraphUtilities::RegisterVisualNodeFactory(AccessibilityNodeFactory);
00047
00048     // Construct Base Phrase Tree Libraries

```

```

00049     FOpenAccessibilityCommunicationModule::Get ()
00050     .PhraseTreeUtils->RegisterFunctionLibrary (
00051         NewObject<ULocalizedInputLibrary>()
00052     );
00053
00054     FOpenAccessibilityCommunicationModule::Get ()
00055     .PhraseTreeUtils->RegisterFunctionLibrary (
00056         NewObject<UWindowInteractionLibrary>()
00057     );
00058
00059     FOpenAccessibilityCommunicationModule::Get ()
00060     .PhraseTreeUtils->RegisterFunctionLibrary (
00061         NewObject<UViewInteractionLibrary>()
00062     );
00063
00064     FOpenAccessibilityCommunicationModule::Get ()
00065     .PhraseTreeUtils->RegisterFunctionLibrary (
00066         NewObject<UNodeInteractionLibrary>()
00067     );
00068
00069     CreateTranscriptionVisualization();
00070
00071     // Register Console Commands
00072     RegisterConsoleCommands();
00073 }
00074
00075 void FOpenAccessibilityModule::ShutdownModule()
00076 {
00077     UE_LOG(LogOpenAccessibility, Display, TEXT("OpenAccessibilityModule::ShutdownModule()"));
00078
00079     UnregisterConsoleCommands();
00080 }
00081
00082 void FOpenAccessibilityModule::CreateTranscriptionVisualization()
00083 {
00084     TranscriptionVisualizer = MakeShared<FTranscriptionVisualizer, ESPMode::ThreadSafe>();
00085
00086     FOpenAccessibilityCommunicationModule::Get ().OnTranscriptionRecieved
00087         .AddSP(TranscriptionVisualizer.ToSharedRef(),
00088             &FTranscriptionVisualizer::OnTranscriptionRecieved);
00089 }
00090
00091 void FOpenAccessibilityModule::RegisterConsoleCommands()
00092 {
00093     ConsoleCommands.Add(IConsoleManager::Get ().RegisterConsoleCommand(
00094         TEXT("OpenAccessibility.Debug.SendPhraseEvent"),
00095         TEXT("Sends the provided Phrase to the Phrase Tree, replicating the STT Communication Module's
00096         Transcription Recieving."),
00097         FConsoleCommandWithArgsDelegate::CreateLambda([this](const TArray<FString> &Args) {
00098             if (Args.Num() == 0)
00099                 return;
00100
00101             FString ProvidedPhrase;
00102             for (const FString& Arg : Args)
00103             {
00104                 ProvidedPhrase += Arg + TEXT(" ");
00105             }
00106
00107             ProvidedPhrase.TrimStartAndEndInline();
00108             ProvidedPhrase.ToUpperInline();
00109
00110             FOpenAccessibilityCommunicationModule::Get ()
00111                 .OnTranscriptionRecieved.Broadcast(TArray<FString>{ ProvidedPhrase });
00112         })),
00113     ECVF_Default
00114 );
00115
00116     ConsoleCommands.Add(IConsoleManager::Get ().RegisterConsoleCommand(
00117         TEXT("OpenAccessibility.Debug.LogActiveIndexes"),
00118         TEXT("Logs the Active Indexes of the Active Tab"),
00119         FConsoleCommandDelegate::CreateLambda([this]() {
00120             TSharedPtr<SDockTab> ActiveTab = FGlobalTabmanager::Get()->GetActiveTab();
00121             SGraphEditor* ActiveGraphEditor =
00122                 (SGraphEditor*)ActiveTab->GetContent().ToSharedPtr().Get();
00123             if (ActiveGraphEditor == nullptr)
00124             {
00125                 UE_LOG(LogOpenAccessibility, Display, TEXT("Active Tab Not SGraphEditor"));
00126                 return;
00127             }
00128
00129             TSharedRef<FGraphIndexer> GraphIndexer =
00130                 AssetAccessibilityRegistry->GetGraphIndexer(ActiveGraphEditor->GetCurrentGraph());
00131         })),

```



```

00212 *TreeView->GetTypeAsString());
00213     MenuWindow =
    FSlateApplication::Get().FindWidgetWindow(KeyboardFocusedWidget.ToSharedRef());
00214 }
00215
00216     UAccessibilityAddNodeContextMenu* AddNodeContextMenu =
    NewObject<UAccessibilityAddNodeContextMenu>();
00217     AddNodeContextMenu->AddToRoot();
00218     AddNodeContextMenu->Init(
00219         Menu.ToSharedRef(),
00220         FOpenAccessibilityCommunicationModule::Get().PhraseTree->AsShared()
00221     );
00222
00223     AddNodeContextMenu->ScaleMenu(1.5f);
00224
00225     FSlateApplication::Get().SetKeyboardFocus(TreeView);
00226
00227     FPhraseTreeContextManager& ContextManager
    = FOpenAccessibilityCommunicationModule::Get()
00228       .PhraseTree->GetContextManager();
00229
00230     ContextManager.PushContextObject(AddNodeContextMenu);
00231 },
00232
00233     ECVF_Default
00234 );
00235
00236     ConsoleCommands.Add(IConsoleManager::Get().RegisterConsoleCommand(
00237         TEXT("OpenAccessibility.Debug.OpenAccessibilityGraph_GenericContextMenu"),
00238         TEXT("Opens the Context Menu for the Active Graph Editor, and Uses Generic Bindings For
    Commands"),
00239
00240         FConsoleCommandDelegate::CreateLambda(
00241             [this]()
00242             {
00243                 TSharedPtr<SGraphEditor> ActiveGraphEditor = nullptr;
00244                 {
00245                     // Getting Graph Editor Section
00246
00247                     TSharedPtr<SDockTab> ActiveTab = FGlobalTabmanager::Get()->GetActiveTab();
00248                     if (!ActiveTab.IsValid())
00249                         return;
00250
00251                     ActiveGraphEditor =
    StaticCastSharedPtr<SGraphEditor>(ActiveTab->GetContent().ToSharedPtr());
00252                     if (!ActiveGraphEditor.IsValid() && ActiveGraphEditor->GetType() ==
    "SGraphEditor")
00253                     {
00254                         UE_LOG(LogOpenAccessibility, Display, TEXT("Active Tab Not SGraphEditor"));
00255                         return;
00256                     }
00257                 }
00258
00259                 SGraphPanel* ActiveGraphPanel = ActiveGraphEditor->GetGraphPanel();
00260
00261                 FVector2D SpawnLocation;
00262                 {
00263                     TSharedPtr<SWindow> TopLevelWindow =
    FSlateApplication::Get().GetActiveTopLevelRegularWindow();
00264                     if (TopLevelWindow.IsValid())
00265                     {
00266                         SpawnLocation = TopLevelWindow->GetPositionInScreen();
00267                         FVector2D WindowSize = TopLevelWindow->GetSizeInScreen();
00268
00269                         SpawnLocation.X += WindowSize.X / 5;
00270                         SpawnLocation.Y += WindowSize.Y / 5;
00271                     }
00272                     else
00273                     {
00274                         FDisplayMetrics DisplayMetrics;
00275                         FSlateApplication::Get().GetDisplayMetrics(DisplayMetrics);
00276
00277                         SpawnLocation = FVector2D(
00278                             DisplayMetrics.PrimaryDisplayWidth / 5,
00279                             DisplayMetrics.PrimaryDisplayHeight / 5
00280                         );
00281                     }
00282                 }
00283
00284                 TSharedPtr<SWidget> ContextWidgetToFocus = ActiveGraphPanel->SummonContextMenu(
00285                     SpawnLocation,
00286                     ActiveGraphPanel->GetPastePosition(),
00287                     nullptr,
00288                     nullptr,
00289                     TArray<UEdGraphPin*>()
00290                 );

```

```

00291         FWidgetPath ContextWidgetToFocusPath;
00292         if (FSlateApplication::Get().FindPathToWidget(ContextWidgetToFocus.ToSharedRef(),
00293 ContextWidgetToFocusPath))
00294         {
00295             UAccessibilityGraphEditorContext* GraphContext =
00296             NewObject<UAccessibilityGraphEditorContext>();
00297             GraphContext->AddToRoot();
00298             GraphContext->Init(
00299             FSlateApplication::Get().FindMenuInWidgetPath(ContextWidgetToFocusPath).ToSharedRef(),
00300             FOpenAccessibilityCommunicationModule::Get().PhraseTree->AsShared()
00301             );
00302             GraphContext->ScaleMenu(1.5f);
00303         }
00304     }
00305 }
00306
00307 )
00308 );
00309
00310 ConsoleCommands.Add(IConsoleManager::Get().RegisterConsoleCommand(
00311     TEXT("OpenAccessibility.Debug.DumpActiveTabManagerObject"),
00312     TEXT("Dumps the Active FTabManager to a JSON Object File"),
00313     FConsoleCommandDelegate::CreateLambda(
00314         [this]() {
00315             TSharedPtr<SDockTab> ActiveTab = FGlobalTabmanager::Get()->GetActiveTab();
00316             if (!ActiveTab.IsValid())
00317             {
00318                 UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("DumpActiveTabManagerObject:
00319 No Active Tab Was Found"));
00320                 return;
00321             }
00322             TSharedPtr<FTabManager> ActiveTabManager = ActiveTab->GetTabManagerPtr();
00323             if (!ActiveTabManager.IsValid())
00324             {
00325                 UE_LOG(LogOpenAccessibility, Display, TEXT("DumpActiveTabManagerObject: No Parent
00326 Tab Manager Found For Active Tab."))
00327                 return;
00328             }
00329             TSharedRef<FTabManager::FLayout> ManagerLayout = ActiveTabManager->PersistLayout();
00330             FString JsonString;
00331             if (!FJsonSerializer::Serialize(ManagerLayout->ToJson(),
00332 TJsonWriterFactory<>::Create(&JsonString, 0)))
00333             {
00334                 UE_LOG(LogOpenAccessibility, Display, TEXT("DumpActiveTabManagerObject: Failed to
00335 Serialize Json Object to String."))
00336                 return;
00337             }
00338             if (!FFileHelper::SaveStringToFile(
00339                 JsonString,
00340                 *FPaths::ConvertRelativePathToFull(FPaths::ProjectSavedDir() +
00341 TEXT("OpenAccessibility/Debug/ActiveTabManager_Dump.json"))
00342             ))
00343             {
00344                 UE_LOG(LogOpenAccessibility, Display, TEXT("DumpActiveTabManagerObject: Failed to
00345 Save Serialized JSON Object to File."))
00346             }
00347         }
00348     )
00349 );
00350
00351 ConsoleCommands.Add(IConsoleManager::Get().RegisterConsoleCommand(
00352     TEXT("OpenAccessibility.Debug.OpenAccessibilityGraph_SummonImprovedLocomotion"),
00353     TEXT("Summons the Improved Locomotion Menu for the Active Graph Editor."),
00354     FConsoleCommandDelegate::CreateLambda(
00355         [this]() {
00356             TSharedPtr<SGraphEditor> ActiveGraphEditor = nullptr;
00357             // Getting Graph Editor Section
00358             TSharedPtr<SDockTab> ActiveTab = FGlobalTabmanager::Get()->GetActiveTab();
00359             if (!ActiveTab.IsValid())
00360                 return;
00361             ActiveGraphEditor =
00362             StaticCastSharedPtr<SGraphEditor>(ActiveTab->GetContent().ToSharedPtr());
00363             if (!ActiveGraphEditor.IsValid() || ActiveGraphEditor->GetTypeAsString() !=

```



```

    "SGraphEditor")
00368         {
00369             UE_LOG(LogOpenAccessibility, Display, TEXT("Active Tab Not SGraphEditor"));
00370             return;
00371         }
00372     }
00373
00374     UAccessibilityGraphLocomotionContext* LocomotionContext =
    NewObject<UAccessibilityGraphLocomotionContext>();
00375     LocomotionContext->AddToRoot();
00376     LocomotionContext->Init(ActiveGraphEditor.ToSharedRef());
00377
00378     FPhraseTreeContextManager& ContextManager =
    FOpenAccessibilityCommunicationModule::Get()
00379         .PhraseTree->GetContextManager();
00380
00381     ContextManager.PushContextObject(LocomotionContext);
00382 },
00383     ECVF_Default
00384 );
00385 };
00386 }
00387
00388 void FOpenAccessibilityModule::UnregisterConsoleCommands()
00389 {
00390     IConsoleCommand* ConsoleCommand = nullptr;
00391     while (ConsoleCommands.Num() > 0)
00392     {
00393         ConsoleCommand = ConsoleCommands.Pop();
00394
00395         IConsoleManager::Get().UnregisterConsoleObject(ConsoleCommand);
00396
00397         delete ConsoleCommand;
00398         ConsoleCommand = nullptr;
00399     }
00400 }
00401
00402 #undef LOCTEXT_NAMESPACE
00403
00404 IMPLEMENT_MODULE(FOpenAccessibilityModule, OpenAccessibility)

```

## 5.22 LocalizedInputLibrary.cpp

```

00001 #include "PhraseEvents/LocalizedInputLibrary.h"
00002
00003 #include "ToolContextInterfaces.h"
00004 #include "PhraseEvents/Utils.h"
00005
00006 #include "PhraseTree/PhraseStringInputNode.h"
00007 #include "PhraseTree/PhraseEventNode.h"
00008
00009 #include "PhraseTree/Containers/Input/UParseStringInput.h"
00010 #include "PhraseTree/Containers/Input/UParseIntInput.h"
00011 #include "Widgets/Text/SMultiLineEditableText.h"
00012
00013 ULocalizedInputLibrary::ULocalizedInputLibrary(const FObjectInitializer &ObjectInitializer)
00014 {
00015 }
00016
00017
00018 ULocalizedInputLibrary::~ULocalizedInputLibrary()
00019 {
00020 }
00021
00022
00023 void ULocalizedInputLibrary::BindBranches(TSharedRef<FPhraseTree> PhraseTree)
00024 {
00025     PhraseTree->BindBranch(
00026         MakeShared<FPhraseNode>(TEXT("INPUT"),
00027             TPhraseNodeArray {
00028
00029                 MakeShared<FPhraseNode>(TEXT("ADD"),
00030                     TPhraseNodeArray {
00031
00032                         MakeShared<FPhraseStringInputNode>(TEXT("PHRASE_TO_ADD"),
00033                             TPhraseNodeArray {
00034
00035                                 MakeShared<FPhraseEventNode>(CreateParseDelegate(this,
00036                                     &ULocalizedInputLibrary::KeyboardInputAdd))
00037                             }
00038                         )
00039                     }
00040                 )
00041             }
00042         )

```

```

00040
00041         MakeShared<FPhraseNode>(TEXT("REMOVE"),
00042         TPhraseNodeArray {
00043
00044             MakeShared<FPhraseInputNode<int32>(TEXT("AMOUNT"),
00045             TPhraseNodeArray {
00046
00047         MakeShared<FPhraseEventNode>(CreateParseDelegate(this,&ULocalizedInputLibrary::KeyboardInputRemove))
00048
00049             })
00050
00051         }),
00052
00053         MakeShared<FPhraseNode>(TEXT("RESET"),
00054         TPhraseNodeArray {
00055
00056             MakeShared<FPhraseEventNode>(CreateParseDelegate(this,
00057             &ULocalizedInputLibrary::KeyboardInputReset))
00058
00059             }),
00060
00061         /*
00062         MakeShared<FPhraseNode>(TEXT("CONFIRM"),
00063         TPhraseNodeArray {
00064
00065             MakeShared<FPhraseEventNode>(CreateParseDelegate(this,
00066             &ULocalizedInputLibrary::KeyboardInputConfirm))
00067
00068             }),
00069         */
00070
00071         MakeShared<FPhraseNode>(TEXT("EXIT"),
00072         TPhraseNodeArray {
00073
00074             MakeShared<FPhraseEventNode>(CreateParseDelegate(this,
00075             &ULocalizedInputLibrary::KeyboardInputExit))
00076
00077             })
00078         });
00079
00080 void ULocalizedInputLibrary::KeyboardInputAdd(FParseRecord &Record) {
00081     GET_ACTIVE_KEYBOARD_WIDGET(KeyboardFocusedWidget);
00082
00083     FString WidgetType = KeyboardFocusedWidget->GetTypeAsString();
00084
00085     UParseStringInput *PhraseInput = Record.GetPhraseInput<UParseStringInput>(TEXT("PHRASE_TO_ADD"));
00086     if (PhraseInput == nullptr)
00087         return;
00088
00089     if (WidgetType == "SEditableText")
00090     {
00091         TSharedPtr<SEditableText> EditableText =
00092         StaticCastSharedPtr<SEditableText>(KeyboardFocusedWidget);
00093         if (!EditableText.IsValid()) {
00094             UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("KeyboardInputAdd: CURRENT ACTIVE
00095             WIDGET IS NOT OF TYPE - SEditableText"));
00096             return;
00097         }
00098         FString CurrText = EditableText->GetText().ToString();
00099         EditableText->SetText(
00100             FText::FromString(CurrText.TrimStartAndEnd() + TEXT(" ") + PhraseInput->GetValue()));
00101     }
00102     else if (WidgetType == "SMultiLineEditableText")
00103     {
00104         TSharedPtr<SMultiLineEditableText> MultiLineEditableText =
00105         StaticCastSharedPtr<SMultiLineEditableText>(KeyboardFocusedWidget);
00106         if (!MultiLineEditableText.IsValid()) {
00107             UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("KeyboardInputAdd: CURRENT ACTIVE
00108             WIDGET IS NOT OF TYPE - SMultiLineEditableText"));
00109             return;
00110         }
00111         FString CurrText = MultiLineEditableText->GetText().ToString();
00112         MultiLineEditableText->SetText(
00113             FText::FromString(CurrText.TrimStartAndEnd() + TEXT(" ") + PhraseInput->GetValue()));
00114     }
00115     else UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("KeyboardInputAdd: CURRENT ACTIVE
00116     WIDGET IS NOT AN INTERFACEABLE TYPE"));
00117 }

```

```

00118 void ULocalizedInputLibrary::KeyboardInputRemove(FParseRecord& Record)
00119 {
00120     GET_ACTIVE_KEYBOARD_WIDGET(KeyboardFocusedWidget);
00121
00122     FString WidgetType = KeyboardFocusedWidget->GetTypeAsString();
00123
00124     UParseIntInput* RemoveInput = Record.GetPhraseInput<UParseIntInput>(TEXT("AMOUNT"));
00125     if (RemoveInput == nullptr)
00126         return;
00127
00128     if (WidgetType == "SEditableText")
00129     {
00130         TSharedPtr<SEditableText> EditableText =
00131             StaticCastSharedPtr<SEditableText>(KeyboardFocusedWidget);
00132         if (!EditableText.IsValid()) {
00133             UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("KeyboardInputRemove: CURRENT ACTIVE
00134 WIDGET IS NOT OF TYPE - SEditableText"));
00135             return;
00136         }
00137         EditableText->SetText(
00138             FText::FromString(
00139                 EventUtils::RemoveWordsFromEnd(EditableText->GetText().ToString(),
00140 RemoveInput->GetValue())
00141             ));
00142     }
00143     else if (WidgetType == "SMultiLineEditableText")
00144     {
00145         TSharedPtr<SMultiLineEditableText> MultiLineEditableText =
00146             StaticCastSharedPtr<SMultiLineEditableText>(KeyboardFocusedWidget);
00147         if (!MultiLineEditableText.IsValid()) {
00148             UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("KeyboardInputRemove: CURRENT ACTIVE
00149 WIDGET IS NOT OF TYPE - SMultiLineEditableText"));
00150             return;
00151         }
00152         MultiLineEditableText->SetText(
00153             FText::FromString(
00154                 EventUtils::RemoveWordsFromEnd(MultiLineEditableText->GetText().ToString(),
00155 RemoveInput->GetValue())
00156             ));
00157     }
00158     else UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("KeyboardInputRemove: CURRENT ACTIVE
00159 WIDGET IS NOT AN INTERFACEABLE TYPE"));
00160 }
00161 void ULocalizedInputLibrary::KeyboardInputReset(FParseRecord &Record)
00162 {
00163     GET_ACTIVE_KEYBOARD_WIDGET(KeyboardFocusedWidget);
00164
00165     FString WidgetType = KeyboardFocusedWidget->GetTypeAsString();
00166
00167     if (WidgetType == "SEditableText")
00168     {
00169         TSharedPtr<SEditableText> EditableText =
00170             StaticCastSharedPtr<SEditableText>(KeyboardFocusedWidget);
00171         if (!EditableText.IsValid()) {
00172             UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("KeyboardInputReset: CURRENT ACTIVE
00173 WIDGET IS NOT OF TYPE - SEditableText"));
00174             return;
00175         }
00176         EditableText->SetText(
00177             FText::FromString(TEXT(""))
00178             );
00179     }
00180     else if (WidgetType == "SMultiLineEditableText")
00181     {
00182         TSharedPtr<SMultiLineEditableText> MultiLineEditableText =
00183             StaticCastSharedPtr<SMultiLineEditableText>(KeyboardFocusedWidget);
00184         if (!MultiLineEditableText.IsValid()) {
00185             UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("KeyboardInputReset: CURRENT ACTIVE
00186 WIDGET IS NOT OF TYPE - SMultiLineEditableText"));
00187             return;
00188         }
00189         MultiLineEditableText->SetText(
00190             FText::FromString(TEXT(""))
00191             );
00192     }
00193     else UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("KeyboardInputReset: CURRENT ACTIVE
00194 WIDGET IS NOT AN INTERFACEABLE TYPE"));
00195 }
00196 void ULocalizedInputLibrary::KeyboardInputConfirm(FParseRecord& Record)

```

```

00193 {
00194     GET_ACTIVE_KEYBOARD_WIDGET(KeyboardFocusedWidget);
00195
00196     FName WidgetType = KeyboardFocusedWidget->GetType();
00197
00198     if (WidgetType == SEditableText::StaticWidgetClass().GetWidgetType())
00199     {
00200         TSharedPtr<SEditableText> EditableText =
00201             StaticCastSharedPtr<SEditableText>(KeyboardFocusedWidget);
00202         if (!EditableText.IsValid())
00203         {
00204             UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("KeyboardInputConfirm: CURRENT
ACTIVE WIDGET IS NOT OF TYPE - SEditableText"))
00205             return;
00206         }
00207     }
00208     else if (WidgetType == SMultiLineEditableText::StaticWidgetClass().GetWidgetType())
00209     {
00210         TSharedPtr<SMultiLineEditableText> MultiLineEditableText =
00211             StaticCastSharedPtr<SMultiLineEditableText>(KeyboardFocusedWidget);
00212         if (!MultiLineEditableText.IsValid())
00213         {
00214             UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("KeyboardInputConfirm: CURRENT
ACTIVE WIDGET IS NOT OF TYPE - SMultiLineEditableText"))
00215             return;
00216         }
00217     }
00218     else UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("KeyboardInputConfirm: CURRENT ACTIVE
WIDGET IS NOT AN INTERFACEABLE TYPE"))
00219 }
00220
00221 void ULocalizedInputLibrary::KeyboardInputExit(FParseRecord &Record)
00222 {
00223     FSlateApplication& SlateApp = FSlateApplication::Get();
00224     if (!SlateApp.IsInitialized())
00225         return;
00226
00227     SlateApp.ClearKeyboardFocus();
00228 }

```

## 5.23 NodeInteractionLibrary.cpp

```

00001 #include "PhraseEvents/NodeInteractionLibrary.h"
00002 #include "PhraseEvents/Utils.h"
00003
00004 #include "BlueprintEditor.h"
00005 #include "SNodePanel.h"
00006 #include "SGraphPanel.h"
00007 #include "Kismet2/KismetEditorUtilities.h"
00008 #include "Kismet2/BlueprintEditorUtils.h"
00009
00010 #include "PhraseTree/Containers/Input/InputContainers.h"
00011 #include "AccessibilityWrappers/AccessibilityGraphEditorContext.h"
00012 #include "AccessibilityWrappers/AccessibilityGraphLocomotionContext.h"
00013
00014 #include "PhraseTree/PhraseInputNode.h"
00015 #include "PhraseTree/PhraseStringInputNode.h"
00016 #include "PhraseTree/PhraseDirectionalInputNode.h"
00017 #include "PhraseTree/PhraseContextNode.h"
00018 #include "PhraseTree/PhraseContextMenuNode.h"
00019 #include "PhraseTree/PhraseEventNode.h"
00020
00021 UNodeInteractionLibrary::UNodeInteractionLibrary(const FObjectInitializer& ObjectInitializer)
00022     : Super(ObjectInitializer)
00023 {
00024
00025 }
00026
00027 UNodeInteractionLibrary::~UNodeInteractionLibrary()
00028 {
00029
00030 }
00031
00032 void UNodeInteractionLibrary::BindBranches(TSharedPtr<FPhraseTree> PhraseTree)
00033 {
00034     // Events
00035     TDelegate<void(int32)> NodeIndexFocusDelegate = CreateInputDelegate(this,
&UNodeInteractionLibrary::NodeIndexFocus);
00036
00037
00038     // Add Node Children Branch

```

```

00039     TPhraseNodeArray AddNodeContextChildren = TPhraseNodeArray {
00040
00041         MakeShared<FPhraseNode>(TEXT("SELECT"),
00042             TPhraseNodeArray {
00043
00044                 MakeShared<FPhraseInputNode<int32>>(TEXT("SELECTION_INDEX"),
00045                     TPhraseNodeArray {
00046
00047                         MakeShared<FPhraseEventNode>(CreateParseDelegate(this,
00048                             &UNodeInteractionLibrary::NodeAddSelect))
00049
00050                     })
00051             }},
00052
00053         MakeShared<FPhraseNode>(TEXT("SEARCH"),
00054             TPhraseNodeArray{
00055
00056                 MakeShared<FPhraseNode>(TEXT("ADD"),
00057                     TPhraseNodeArray {
00058
00059                         MakeShared<FPhraseStringInputNode>(TEXT("SEARCH_PHRASE"),
00060                             TPhraseNodeArray{
00061
00062                                 MakeShared<FPhraseEventNode>(CreateParseDelegate(this,
00063                                     &UNodeInteractionLibrary::NodeAddSearchAdd))
00064
00065                             })
00066                         }},
00067
00068                     MakeShared<FPhraseNode>(TEXT("REMOVE"),
00069                         TPhraseNodeArray {
00070
00071                             MakeShared<FPhraseInputNode<int32>>(TEXT("AMOUNT"),
00072                                 TPhraseNodeArray {
00073
00074                                     MakeShared<FPhraseEventNode>(CreateParseDelegate(this,
00075                                         &UNodeInteractionLibrary::NodeAddSearchRemove))
00076
00077                                 })
00078                             }},
00079
00080                     MakeShared<FPhraseNode>(TEXT("RESET"),
00081                         TPhraseNodeArray {
00082
00083                             MakeShared<FPhraseEventNode>(CreateParseDelegate(this,
00084                                 &UNodeInteractionLibrary::NodeAddSearchReset))
00085
00086                             })
00087                         }},
00088
00089                     MakeShared<FPhraseNode>(TEXT("SCROLL"),
00090                         TPhraseNodeArray {
00091
00092                             MakeShared<FPhraseScrollInputNode>(TEXT("DIRECTION"),
00093                                 TPhraseNodeArray {
00094
00095                                     MakeShared<FPhraseInputNode<int32>>(TEXT("AMOUNT"),
00096                                         TPhraseNodeArray {
00097
00098                                             MakeShared<FPhraseEventNode>(CreateParseDelegate(this,
00099                                                 &UNodeInteractionLibrary::NodeAddScroll))
00100
00101                                             })
00102                                         }},
00103                                     }},
00104                                 }},
00105                         });
00106
00107         PhraseTree->BindBranches(
00108             TPhraseNodeArray
00109             {
00110
00111                 MakeShared<FPhraseNode>(TEXT("NODE"),
00112                     TPhraseNodeArray {
00113
00114                         MakeShared<FPhraseInputNode<int32>>(TEXT("NODE_INDEX"),
00115                             TPhraseNodeArray {
00116
00117                                 MakeShared<FPhraseNode>(TEXT("MOVE"),
00118                                     TPhraseNodeArray {
00119
00120                                         MakeShared<FPhrase2DDirectionalInputNode>(TEXT("DIRECTION"),

```

```

00121         TPhraseNodeArray {
00122             MakeShared<FPhraseInputNode<int32>>(TEXT("AMOUNT"),
00123             TPhraseNodeArray {
00124                 MakeShared<FPhraseEventNode>(CreateParseDelegate(this,
00125 &UNodeInteractionLibrary::MoveNode))
00126             })
00127         })
00128     })
00129     })
00130     })
00131     })
00132     },
00133     MakeShared<FPhraseNode>(TEXT("REMOVE"),
00134     TPhraseNodeArray {
00135         MakeShared<FPhraseEventNode>(CreateParseDelegate(this,
00136 &UNodeInteractionLibrary::DeleteNode))
00137     })
00138     },
00139     MakeShared<FPhraseInputNode<int32>>(TEXT("PIN_INDEX"),
00140     TPhraseNodeArray {
00141         MakeShared<FPhraseNode>(TEXT("CONNECT"),
00142         TPhraseNodeArray {
00143             MakeShared<FPhraseContextMenuNode<UAccessibilityGraphEditorContext>>(
00144             TEXT("ADD"),
00145             1.5f,
00146             CreateMenuDelegate(this, &UNodeInteractionLibrary::NodeAddPinMenu),
00147             AddNodeContextChildren
00148             ),
00149             MakeShared<FPhraseInputNode<int32>>(TEXT("NODE_INDEX"),
00150             TPhraseNodeArray {
00151                 MakeShared<FPhraseInputNode<int32>>(TEXT("PIN_INDEX"),
00152                 TPhraseNodeArray {
00153                     MakeShared<FPhraseEventNode>(CreateParseDelegate(this,
00154 &UNodeInteractionLibrary::PinConnect))
00155                 })
00156             })
00157             }, NodeIndexFocusDelegate)
00158         })
00159         MakeShared<FPhraseNode>(TEXT("DISCONNECT"),
00160         TPhraseNodeArray {
00161             MakeShared<FPhraseInputNode<int32>>(TEXT("NODE_INDEX"),
00162             TPhraseNodeArray {
00163                 MakeShared<FPhraseInputNode<int32>>(TEXT("PIN_INDEX"),
00164                 TPhraseNodeArray {
00165                     MakeShared<FPhraseEventNode>(CreateParseDelegate(this,
00166 &UNodeInteractionLibrary::PinDisconnect))
00167                 })
00168             })
00169             })
00170             }, NodeIndexFocusDelegate),
00171         MakeShared<FPhraseNode>(TEXT("SELECT"),
00172         TPhraseNodeArray {
00173             MakeShared<FPhraseInputNode<int32>>(TEXT("NODE_INDEX"),
00174             TPhraseNodeArray {
00175                 MakeShared<FPhraseEventNode>(CreateParseDelegate(this,
00176 &UNodeInteractionLibrary::SelectionNodeToggle))
00177             })
00178             })
00179             })
00180             })
00181             })
00182             })
00183             })
00184             })
00185             })
00186             }, NodeIndexFocusDelegate),
00187         MakeShared<FPhraseNode>(TEXT("RESET"),
00188         TPhraseNodeArray {
00189             MakeShared<FPhraseInputNode<int32>>(TEXT("NODE_INDEX"),
00190             TPhraseNodeArray {
00191                 MakeShared<FPhraseEventNode>(CreateParseDelegate(this,
00192 &UNodeInteractionLibrary::SelectionNodeToggle))
00193             })
00194             })
00195             })
00196             })
00197             })
00198             })
00199             })
00200             })
00201             })
00202             MakeShared<FPhraseEventNode>(CreateParseDelegate(this,

```

Generated by Doxygen

```

00283         }},
00284
00285         MakeShared<FPhraseNode>(TEXT("REVERT"),
00286         TPhraseNodeArray {
00287
00288             MakeShared<FPhraseEventNode>(CreateParseDelegate(this,
00289 &UNodeInteractionLibrary::LocomotionRevert))
00290
00291         }},
00292
00293         MakeShared<FPhraseNode>(TEXT("CONFIRM"),
00294         TPhraseNodeArray {
00295
00296             MakeShared<FPhraseEventNode>(CreateParseDelegate(this,
00297 &UNodeInteractionLibrary::LocomotionConfirm))
00298
00299         }},
00300
00301         MakeShared<FPhraseNode>(TEXT("CANCEL"),
00302         TPhraseNodeArray {
00303
00304             MakeShared<FPhraseEventNode>(CreateParseDelegate(this,
00305 &UNodeInteractionLibrary::LocomotionCancel))
00306
00307         }},
00308     }
00309 };
00310
00311 };
00312
00313
00314 void UNodeInteractionLibrary::MoveNode(FParseRecord &Record) {
00315     GET_CAST_ACTIVE_TAB_CONTENT(ActiveGraphEditor, SGraphEditor)
00316
00317     UParseIntInput* IndexInput = Record.GetPhraseInput<UParseIntInput>(TEXT("NODE_INDEX"));
00318     UParseEnumInput* DirectionInput = Record.GetPhraseInput<UParseEnumInput>(TEXT("DIRECTION"));
00319     UParseIntInput* AmountInput = Record.GetPhraseInput<UParseIntInput>(TEXT("AMOUNT"));
00320     if (IndexInput == nullptr || DirectionInput == nullptr || AmountInput == nullptr)
00321         return;
00322
00323     TSharedRef<FAssetAccessibilityRegistry> AssetRegistry = GetAssetRegistry();
00324     TSharedRef<FGraphIndexer> Indexer =
00325         AssetRegistry->GetGraphIndexer(ActiveGraphEditor->GetCurrentGraph());
00326
00327     UEdGraphNode* Node = Indexer->GetNode(IndexInput->GetValue());
00328     if (Node == nullptr)
00329     {
00330         UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("MoveNode: Node Not Found"));
00331         return;
00332     }
00333
00334     FVector2D PositionDelta = FVector2D::ZeroVector;
00335     switch (EPhrase2DDirectionalInput(DirectionInput->GetValue()))
00336     {
00337     case EPhrase2DDirectionalInput::UP:
00338         PositionDelta.Y -= AmountInput->GetValue();
00339         break;
00340
00341     case EPhrase2DDirectionalInput::DOWN:
00342         PositionDelta.Y += AmountInput->GetValue();
00343         break;
00344
00345     case EPhrase2DDirectionalInput::LEFT:
00346         PositionDelta.X -= AmountInput->GetValue();
00347         break;
00348
00349     case EPhrase2DDirectionalInput::RIGHT:
00350         PositionDelta.X += AmountInput->GetValue();
00351         break;
00352
00353     default:
00354         UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("MoveNode: Invalid Direction"));
00355         return;
00356     }
00357
00358     SGraphPanel* GraphPanel = ActiveGraphEditor->GetGraphPanel();
00359     if (GraphPanel == nullptr)
00360     {
00361         UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("MoveNode: Linked Graph Panel Not Found"));
00362     }
00363
00364     TSharedPtr<SGraphNode> NodeWidget = GraphPanel ? GraphPanel->GetNodeWidgetFromGuid(Node->NodeGuid)
00365 : TSharedPtr<SGraphNode>();

```



```

00364     if (NodeWidget.IsValid())
00365     {
00366         SNodePanel::SNode::FNodeSet NodeFilter;
00367         NodeWidget->MoveTo(FVector2D(Node->NodePosX, Node->NodePosY) + PositionDelta, NodeFilter);
00368     }
00369     else
00370     {
00371         Node->Modify();
00372         Node->NodePosX += PositionDelta.X;
00373         Node->NodePosY += PositionDelta.Y;
00374     }
00375
00376     // Move Comment Node Children
00377     // Note: This is a workaround for the MoveTo Function not calling the override in
    UEdGraphNode_Comment
00378     if (UEdGraphNode_Comment* CommentNode = Cast<UEdGraphNode_Comment>(Node))
00379     {
00380         for (UObject* _CommentChildNode : CommentNode->GetNodesUnderComment())
00381         {
00382             if (UEdGraphNode* CommentChildNode = Cast<UEdGraphNode>(_CommentChildNode))
00383             {
00384                 if (!GraphPanel->SelectionManager.IsNodeSelected(CommentChildNode))
00385                 {
00386                     CommentChildNode->Modify();
00387                     CommentChildNode->NodePosX += PositionDelta.X;
00388                     CommentChildNode->NodePosY += PositionDelta.Y;
00389                 }
00390             }
00391         }
00392     }
00393 }
00394
00395 void UNodeInteractionLibrary::DeleteNode(FParseRecord& Record)
00396 {
00397     GET_CAST_ACTIVE_TAB_CONTENT(ActiveGraphEditor, SGraphEditor)
00398
00399     UParseIntInput* IndexInput = Record.GetPhraseInput<UParseIntInput>(TEXT("NODE_INDEX"));
00400     if (IndexInput == nullptr)
00401         return;
00402
00403     TSharedRef<FAssetAccessibilityRegistry> AssetRegistry = GetAssetRegistry();
00404     TSharedRef<FGraphIndexer> Indexer =
    AssetRegistry->GetGraphIndexer(ActiveGraphEditor->GetCurrentGraph());
00405
00406     UEdGraphNode* Node = Indexer->GetNode(IndexInput->GetValue());
00407     if (Node == nullptr)
00408     {
00409         UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("DeleteNode: Node Not Found"));
00410         return;
00411     }
00412
00413     Node->Modify();
00414     Node->DestroyNode();
00415 }
00416
00417 void UNodeInteractionLibrary::NodeIndexFocus(int32 Index)
00418 {
00419     GET_CAST_ACTIVE_TAB_CONTENT(ActiveGraphEditor, SGraphEditor)
00420
00421     TSharedRef<FGraphIndexer> Indexer = GetAssetRegistry()->GetGraphIndexer(
    ActiveGraphEditor->GetCurrentGraph()
00422 );
00423
00424     UEdGraphNode* Node = Indexer->GetNode(Index);
00425     if (Node == nullptr)
00426     {
00427         UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("NodeSelectionFocus: Node Not Found"));
00428         return;
00429     }
00430
00431     ActiveGraphEditor->SetNodeSelection(Node, true);
00432 }
00433
00434
00435 void UNodeInteractionLibrary::PinConnect(FParseRecord& Record)
00436 {
00437     GET_CAST_ACTIVE_TAB_CONTENT(ActiveGraphEditor, SGraphEditor)
00438
00439     UEdGraph* Graph = ActiveGraphEditor->GetCurrentGraph();
00440
00441     TArray<UParseInput*> NodeInputs = Record.GetPhraseInputs(TEXT("NODE_INDEX"));
00442     TArray<UParseInput*> PinInputs = Record.GetPhraseInputs(TEXT("PIN_INDEX"));
00443
00444     if (NodeInputs.Num() != 2 || PinInputs.Num() != 2)
00445     {
00446         UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("PinConnect: Invalid Inputs Amount"));
00447         return;
00448     }

```

```

00449
00450     TSharedRef<FGraphIndexer> Indexer = GetAssetRegistry()->GetGraphIndexer(Graph);
00451
00452     UEdGraphPin* SourcePin = Indexer->GetPin(
00453         Cast<UParseIntInput>(NodeInputs[0])->GetValue(),
00454         Cast<UParseIntInput>(PinInputs[0])->GetValue()
00455     );
00456
00457     UEdGraphPin* TargetPin = Indexer->GetPin(
00458         Cast<UParseIntInput>(NodeInputs[1])->GetValue(),
00459         Cast<UParseIntInput>(PinInputs[1])->GetValue()
00460     );
00461
00462     if (SourcePin == nullptr || TargetPin == nullptr)
00463     {
00464         UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("PinConnect: Pins Not Found"));
00465         return;
00466     }
00467
00468     if (!Graph->GetSchema()->TryCreateConnection(SourcePin, TargetPin))
00469     {
00470         UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("PinConnect: Pin Connection Failed"));
00471     }
00472 }
00473
00474 void UNodeInteractionLibrary::PinDisconnect(FParseRecord& Record)
00475 {
00476     GET_CAST_ACTIVE_TAB_CONTENT(ActiveGraphEditor, SGraphEditor)
00477
00478     UEdGraph* Graph = ActiveGraphEditor->GetCurrentGraph();
00479
00480     TArray<UParseIntInput*> NodeInputs = Record.GetPhraseInputs(TEXT("NODE_INDEX"));
00481     TArray<UParseIntInput*> PinInputs = Record.GetPhraseInputs(TEXT("PIN_INDEX"));
00482
00483     if (NodeInputs.Num() != 2 || PinInputs.Num() != 2)
00484     {
00485         UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("PinDisconnect: Invalid Inputs
00486 Amount"));
00487         return;
00488     }
00489
00490     TSharedRef<FGraphIndexer> Indexer = GetAssetRegistry()->GetGraphIndexer(Graph);
00491
00492     UEdGraphPin* SourcePin = Indexer->GetPin(
00493         Cast<UParseIntInput>(NodeInputs[0])->GetValue(),
00494         Cast<UParseIntInput>(PinInputs[0])->GetValue()
00495     );
00496
00497     UEdGraphPin* TargetPin = Indexer->GetPin(
00498         Cast<UParseIntInput>(NodeInputs[1])->GetValue(),
00499         Cast<UParseIntInput>(PinInputs[1])->GetValue()
00500     );
00501
00502     if (SourcePin == nullptr || TargetPin == nullptr)
00503     {
00504         UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("PinDisconnect: Pins Not Found"));
00505         return;
00506     }
00507
00508     Graph->GetSchema()->BreakSinglePinLink(SourcePin, TargetPin);
00509 }
00510 TSharedPtr<IMenu> UNodeInteractionLibrary::NodeAddMenu(FParseRecord& Record)
00511 {
00512     GET_CAST_ACTIVE_TAB_CONTENT_RETURN(ActiveGraphEditor, SGraphEditor, TSharedPtr<IMenu>())
00513
00514     SGraphPanel* GraphPanel = ActiveGraphEditor->GetGraphPanel();
00515
00516     FVector2D SpawnLocation;
00517     {
00518         TSharedPtr<SWindow> TopLevelWindow =
00519             FSlateApplication::Get().GetActiveTopLevelRegularWindow();
00520
00521         if (TopLevelWindow.IsValid())
00522         {
00523             SpawnLocation = TopLevelWindow->GetPositionInScreen();
00524             FVector2D WindowSize = TopLevelWindow->GetSizeInScreen();
00525
00526             SpawnLocation.X += WindowSize.X / 5;
00527             SpawnLocation.Y += WindowSize.Y / 5;
00528         }
00529         else
00530         {
00531             FDisplayMetrics DisplayMetrics;
00532             FSlateApplication::Get().GetDisplayMetrics(DisplayMetrics);
00533
00534             SpawnLocation = FVector2D(

```

```

00534         DisplayMetrics.PrimaryDisplayWidth / 5,
00535         DisplayMetrics.PrimaryDisplayHeight / 5
00536     );
00537 }
00538
00539 TSharedPtr<SWidget> ContextWidgetToFocus = GraphPanel->SummonContextMenu(
00540     SpawnLocation,
00541     GraphPanel->GetPastePosition(),
00542     nullptr,
00543     nullptr,
00544     TArray<UEdGraphPin*>()
00545 );
00546
00547 if (!ContextWidgetToFocus.IsValid())
00548 {
00549     UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("NodeAddMenu: Context Keyboard Focus
Widget Not Found"));
00550     return TSharedPtr<IMenu>();
00551 }
00552
00553 FWidgetPath KeyboardFocusPath;
00554 if (FSlateApplication::Get().FindPathToWidget(ContextWidgetToFocus.ToSharedRef(),
KeyboardFocusPath))
00555 {
00556     return FSlateApplication::Get().FindMenuInWidgetPath(KeyboardFocusPath);
00557 }
00558 else
00559 {
00560     UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("NodeAddMenu: IMenu Could Not Be
Found In Widget Path"));
00561     return TSharedPtr<IMenu>();
00562 }
00563 }
00564 }
00565
00566 TSharedPtr<IMenu> UNodeInteractionLibrary::NodeAddPinMenu(FParseRecord &Record)
00567 {
00568     GET_CAST_ACTIVE_TAB_CONTENT_RETURN(ActiveGraphEditor, SGraphEditor, TSharedPtr<IMenu>())
00569
00570     SGraphPanel* GraphPanel = ActiveGraphEditor->GetGraphPanel();
00571
00572     FVector2D SpawnLocation;
00573     {
00574         TSharedPtr<SWindow> TopLevelWindow =
FSlateApplication::Get().GetActiveTopLevelRegularWindow();
00575
00576         if (TopLevelWindow.IsValid())
00577         {
00578             SpawnLocation = TopLevelWindow->GetPositionInScreen();
00579             FVector2D WindowSize = TopLevelWindow->GetSizeInScreen();
00580
00581             SpawnLocation.X += WindowSize.X / 5;
00582             SpawnLocation.Y += WindowSize.Y / 5;
00583         }
00584         else
00585         {
00586             FDisplayMetrics DisplayMetrics;
00587             FSlateApplication::Get().GetDisplayMetrics(DisplayMetrics);
00588
00589             SpawnLocation = FVector2D(
00590                 DisplayMetrics.PrimaryDisplayWidth / 5,
00591                 DisplayMetrics.PrimaryDisplayHeight / 5
00592             );
00593         }
00594     }
00595
00596     TSharedRef<FGraphIndexer> Indexer =
GetAssetRegistry()->GetGraphIndexer(ActiveGraphEditor->GetCurrentGraph());
00597
00598     UParseIntInput* NodeIndexInput = Record.GetPhraseInput<UParseIntInput>(TEXT("NODE_INDEX"));
00599     UParseIntInput* PinIndexInput = Record.GetPhraseInput<UParseIntInput>(TEXT("PIN_INDEX"));
00600
00601     if (NodeIndexInput == nullptr || PinIndexInput == nullptr)
00602     {
00603         UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("NodeAddMenu: Invalid Inputs"));
00604
00605         return TSharedPtr<IMenu>();
00606     }
00607
00608     TSharedPtr<SWidget> ContextWidgetToFocus = GraphPanel->SummonContextMenu(
00609         SpawnLocation,
00610         GraphPanel->GetPastePosition(),
00611         nullptr,
00612         nullptr,
00613         TArray<UEdGraphPin*> {
00614             Indexer->GetPin(
00615                 NodeIndexInput->GetValue(),
00616                 PinIndexInput->GetValue()

```

```

00616         )
00617     }
00618 };
00619
00620     if (!ContextWidgetToFocus.IsValid())
00621     {
00622         UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("NodeAddMenu: Context Keyboard Focus
Widget Not Found"));
00623         return TSharedPtr<IMenu>();
00624     }
00625
00626     FWidgetPath KeyboardFocusPath;
00627     if (FSlateApplication::Get().FindPathToWidget(ContextWidgetToFocus.ToSharedRef(),
KeyboardFocusPath))
00628     {
00629         return FSlateApplication::Get().FindMenuInWidgetPath(KeyboardFocusPath);
00630     }
00631     else
00632     {
00633         UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("NodeAddMenu: IMenu Could Not Be
Found In Widget Path"));
00634         return TSharedPtr<IMenu>();
00635     }
00636 }
00637 }
00638
00639 void UNodeInteractionLibrary::NodeAddSelect (FParseRecord& Record)
00640 {
00641     GET_TOP_CONTEXT(Record, ContextMenu, UAccessibilityGraphEditorContext)
00642
00643     UParseIntInput* IndexInput = Record.GetPhraseInput<UParseIntInput>(TEXT("SELECTION_INDEX"));
00644     if (IndexInput == nullptr)
00645         return;
00646
00647     ContextMenu->SelectAction(IndexInput->GetValue());
00648 }
00649
00650 void UNodeInteractionLibrary::NodeAddSearchAdd (FParseRecord& Record)
00651 {
00652     GET_TOP_CONTEXT(Record, ContextMenu, UAccessibilityGraphEditorContext)
00653
00654     UParseStringInput *SearchInput = Record.GetPhraseInput<UParseStringInput>(TEXT("SEARCH_PHRASE"));
00655     if (SearchInput == nullptr)
00656         return;
00657
00658     ContextMenu->AppendFilterText(SearchInput->GetValue());
00659 }
00660
00661 void UNodeInteractionLibrary::NodeAddSearchRemove (FParseRecord& Record)
00662 {
00663     GET_TOP_CONTEXT(Record, ContextMenu, UAccessibilityGraphEditorContext);
00664
00665     UParseIntInput* RemoveAmountInput = Record.GetPhraseInput<UParseIntInput>(TEXT("AMOUNT"));
00666     if (RemoveAmountInput == nullptr)
00667         return;
00668
00669     ContextMenu->SetFilterText (
00670         EventUtils::RemoveWordsFromEnd(ContextMenu->GetFilterText(), RemoveAmountInput->GetValue())
00671     );
00672 }
00673
00674 void UNodeInteractionLibrary::NodeAddSearchReset (FParseRecord& Record)
00675 {
00676     GET_TOP_CONTEXT(Record, ContextMenu, UAccessibilityGraphEditorContext)
00677
00678     ContextMenu->SetFilterText(TEXT(""));
00679 }
00680
00681 void UNodeInteractionLibrary::NodeAddScroll (FParseRecord& Record)
00682 {
00683     GET_TOP_CONTEXT(Record, ContextMenu, UAccessibilityGraphEditorContext)
00684
00685     UParseEnumInput* DirectionInput = Record.GetPhraseInput<UParseEnumInput>(TEXT("DIRECTION"));
00686     UParseIntInput* AmountInput = Record.GetPhraseInput<UParseIntInput>(TEXT("AMOUNT"));
00687     if (DirectionInput == nullptr || AmountInput == nullptr)
00688         return;
00689
00690     switch (EPhraseScrollInput(DirectionInput->GetValue()))
00691     {
00692     case EPhraseScrollInput::UP:
00693         ContextMenu->AppendScrollDistance(-AmountInput->GetValue());
00694         break;
00695
00696     case EPhraseScrollInput::DOWN:
00697         ContextMenu->AppendScrollDistance(AmountInput->GetValue());
00698         break;
00699     }

```

```

00700         case EPhraseScrollInput::TOP:
00701             ContextMenu->SetScrollDistanceTop();
00702             break;
00703
00704         case EPhraseScrollInput::BOTTOM:
00705             ContextMenu->SetScrollDistanceBottom();
00706             break;
00707
00708         default:
00709             UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("NodeAddScroll: Invalid Scroll
Position / Direction"));
00710             return;
00711     }
00712 }
00713
00714 void UNodeInteractionLibrary::SelectionNodeToggle(FParseRecord& Record)
00715 {
00716     GET_CAST_ACTIVE_TAB_CONTENT(ActiveGraphEditor, SGraphEditor);
00717
00718     UParseIntInput* IndexInput = Record.GetPhraseInput<UParseIntInput>(TEXT("NODE_INDEX"));
00719     if (IndexInput == nullptr)
00720         return;
00721
00722     TSharedRef<FGraphIndexer> Indexer = GetAssetRegistry()->GetGraphIndexer(
ActiveGraphEditor->GetCurrentGraph()
00723     );
00724
00725     UEdGraphNode* Node = Indexer->GetNode(IndexInput->GetValue());
00726     if (Node == nullptr)
00727     {
00728         UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("SelectionToggle: Node Not Found"));
00729         return;
00730     }
00731
00732     ActiveGraphEditor->SetNodeSelection(
Node,
00733     !ActiveGraphEditor->GetSelectedNodes().Contains(Node)
00734     );
00735 }
00736
00737 void UNodeInteractionLibrary::SelectionReset(FParseRecord &Record) {
00738     GET_CAST_ACTIVE_TAB_CONTENT(ActiveGraphEditor, SGraphEditor)
00739
00740     ActiveGraphEditor->ClearSelectionSet();
00741 }
00742
00743 void UNodeInteractionLibrary::SelectionMove(FParseRecord& Record)
00744 {
00745     GET_CAST_ACTIVE_TAB_CONTENT(ActiveGraphEditor, SGraphEditor)
00746
00747     UParseEnumInput* Direction = Record.GetPhraseInput<UParseEnumInput>(TEXT("DIRECTION"));
00748     UParseIntInput* Amount = Record.GetPhraseInput<UParseIntInput>(TEXT("AMOUNT"));
00749     if (Direction == nullptr || Amount == nullptr)
00750         return;
00751
00752     for (UObject* NodeObj : ActiveGraphEditor->GetSelectedNodes())
00753     {
00754         UEdGraphNode* Node = Cast<UEdGraphNode>(NodeObj);
00755         if (Node == nullptr)
00756             continue;
00757
00758         switch (EPhrase2DDirectionalInput(Direction->GetValue()))
00759         {
00760             case EPhrase2DDirectionalInput::UP:
00761                 Node->NodePosY -= Amount->GetValue();
00762                 break;
00763
00764             case EPhrase2DDirectionalInput::DOWN:
00765                 Node->NodePosY += Amount->GetValue();
00766                 break;
00767
00768             case EPhrase2DDirectionalInput::LEFT:
00769                 Node->NodePosX -= Amount->GetValue();
00770                 break;
00771
00772             case EPhrase2DDirectionalInput::RIGHT:
00773                 Node->NodePosX += Amount->GetValue();
00774                 break;
00775
00776             default:
00777                 UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("SelectionMove: Invalid
Direction"));
00778                 return;
00779         }
00780     }
00781 }
00782 }
00783 }
00784

```

```

00785 void UNodeInteractionLibrary::SelectionAlignment(FParseRecord& Record)
00786 {
00787     GET_CAST_ACTIVE_TAB_CONTENT(ActiveGraphEditor, SGraphEditor)
00788
00789     UParseEnumInput* PositionInput = Record.GetPhraseInput<UParseEnumInput>(TEXT("POSITION"));
00790     if (PositionInput == nullptr)
00791         return;
00792
00793     switch (EPhrasePositionalInput(PositionInput->GetValue()))
00794     {
00795         case EPhrasePositionalInput::TOP:
00796             ActiveGraphEditor->OnAlignTop();
00797             break;
00798
00799         case EPhrasePositionalInput::MIDDLE:
00800             ActiveGraphEditor->OnAlignMiddle();
00801             break;
00802
00803         case EPhrasePositionalInput::BOTTOM:
00804             ActiveGraphEditor->OnAlignBottom();
00805             break;
00806
00807         case EPhrasePositionalInput::LEFT:
00808             ActiveGraphEditor->OnAlignLeft();
00809             break;
00810
00811         case EPhrasePositionalInput::RIGHT:
00812             ActiveGraphEditor->OnAlignRight();
00813             break;
00814
00815         case EPhrasePositionalInput::CENTER:
00816             ActiveGraphEditor->OnAlignCenter();
00817             break;
00818     }
00819 }
00820
00821 void UNodeInteractionLibrary::SelectionStraighten(FParseRecord& Record)
00822 {
00823     GET_CAST_ACTIVE_TAB_CONTENT(ActiveGraphEditor, SGraphEditor)
00824
00825     ActiveGraphEditor->OnStraightenConnections();
00826 }
00827
00828 void UNodeInteractionLibrary::SelectionComment(FParseRecord& Record)
00829 {
00830     GET_CAST_ACTIVE_TAB_CONTENT(ActiveGraphEditor, SGraphEditor)
00831
00832     UEdGraph* Graph = ActiveGraphEditor->GetCurrentGraph();
00833
00834     TSharedPtr<FEdGraphSchemaAction> CommentCreateAction =
00835         Graph->GetSchema()->GetCreateCommentAction();
00836     if (CommentCreateAction.IsValid())
00837     {
00838         CommentCreateAction->PerformAction(Graph, nullptr, FVector2D(0, 0), true);
00839     }
00840     else UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("SelectionComment: Comment Creation
00841         Failed"));
00842 }
00843
00844 void UNodeInteractionLibrary::LocomotionSelect(FParseRecord& Record)
00845 {
00846     GET_TOP_CONTEXT(Record, LocomotionContext, UAccessibilityGraphLocomotionContext);
00847
00848     UParseIntInput* ViewSelectionInput = Record.GetPhraseInput<UParseIntInput>(TEXT("INDEX"));
00849     if (ViewSelectionInput == nullptr)
00850         return;
00851
00852     if (!LocomotionContext->SelectChunk(ViewSelectionInput->GetValue()))
00853     {
00854         UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("Locomotion Select: Failed to Choose New
00855             View."));
00856     }
00857 }
00858
00859 void UNodeInteractionLibrary::LocomotionRevert(FParseRecord& Record)
00860 {
00861     GET_TOP_CONTEXT(Record, LocomotionContext, UAccessibilityGraphLocomotionContext);
00862
00863     if (!LocomotionContext->RevertToPreviousView())
00864     {
00865         UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("Locomotion Revert: Failed to Revert to
00866             Previous View."));
00867     }
00868 }
00869
00870 void UNodeInteractionLibrary::LocomotionConfirm(FParseRecord& Record)
00871 {

```

```

00868     GET_TOP_CONTEXT(Record, LocomotionContext, UAccessibilityGraphLocomotionContext);
00869
00870     LocomotionContext->ConfirmSelection();
00871 }
00872
00873 void UNodeInteractionLibrary::LocomotionCancel(FParseRecord& Record)
00874 {
00875     GET_TOP_CONTEXT(Record, LocomotionContext, UAccessibilityGraphLocomotionContext);
00876
00877     LocomotionContext->CancelLocomotion();
00878 }
00879
00880 void UNodeInteractionLibrary::BlueprintCompile(FParseRecord& Record)
00881 {
00882     GET_CAST_ACTIVE_TAB_CONTENT(ActiveGraphEditor, SGraphEditor)
00883
00884     UEdGraph* ActiveGraph = ActiveGraphEditor->GetCurrentGraph();
00885     if (ActiveGraph == nullptr)
00886     {
00887         UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("BlueprintCompile: Active Graph Not
Found"));
00888         return;
00889     }
00890
00891     UBlueprint* FoundBlueprint = FBlueprintEditorUtils::FindBlueprintForGraph(ActiveGraph);
00892     if (FoundBlueprint == nullptr)
00893     {
00894         UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("BlueprintCompile: Blueprint Not
Found"));
00895         return;
00896     }
00897
00898     TSharedPtr<FBlueprintEditor> BlueprintEditor =
StaticCastSharedPtr<FBlueprintEditor>(FKismetEditorUtilities::GetIBlueprintEditorForObject(FoundBlueprint,
false));
00899     if (!BlueprintEditor.IsValid())
00900     {
00901         UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("BlueprintCompile: BlueprintEditor Not
Found"));
00902         return;
00903     }
00904
00905     BlueprintEditor->Compile();
00906 }

```

## 5.24 ViewInteractionLibrary.cpp

```

00001 #include "PhraseEvents/ViewInteractionLibrary.h"
00002 #include "PhraseEvents/Utils.h"
00003
00004 #include "PhraseTree/Containers/Input/InputContainers.h"
00005
00006 #include "AssetAccessibilityRegistry.h"
00007
00008 #include "PhraseTree/PhraseInputNode.h"
00009 #include "PhraseTree/PhraseDirectionalInputNode.h"
00010 #include "PhraseTree/PhraseEventNode.h"
00011
00012 UViewInteractionLibrary::UViewInteractionLibrary(const FObjectInitializer &ObjectInitializer)
00013     : Super(ObjectInitializer)
00014 {
00015 }
00016
00017
00018 UViewInteractionLibrary::~UViewInteractionLibrary()
00019 {
00020 }
00021
00022
00023 void UViewInteractionLibrary::BindBranches(TSharedPtr<FPhraseTree> PhraseTree)
00024 {
00025     PhraseTree->BindBranch(
00026         MakeShared<FPhraseNode>(TEXT("VIEW"),
00027             TPhraseNodeArray {
00028
00029                 MakeShared<FPhraseNode>(TEXT("MOVE"),
00030                     TPhraseNodeArray {
00031
00032                         MakeShared<FPhrase2DDirectionalInputNode>(TEXT("DIRECTION"),
00033                             TPhraseNodeArray {
00034
00035                                 MakeShared<FPhraseInputNode<int32>>(TEXT("AMOUNT"),
00036                                     TPhraseNodeArray {

```

```

00037
00038             MakeShared<FPhraseEventNode>(CreateParseDelegate(this,
&UIViewInteractionLibrary::MoveViewport))
00039
00040             })
00041
00042             })
00043
00044             }},
00045
00046             MakeShared<FPhraseNode>(TEXT("ZOOM"),
00047             TPhraseNodeArray {
00048
00049                 MakeShared<FPhrase2DDirectionalInputNode>(TEXT("DIRECTION"),
00050                 TPhraseNodeArray {
00051
00052                     MakeShared<FPhraseInputNode<int32>>(TEXT("AMOUNT"),
00053                     TPhraseNodeArray {
00054
00055                         MakeShared<FPhraseEventNode>(CreateParseDelegate(this,
&UIViewInteractionLibrary::ZoomViewport))
00056
00057                         })
00058
00059                         })
00060
00061                     }},
00062
00063                     MakeShared<FPhraseNode>(TEXT("FOCUS"),
00064                     TPhraseNodeArray {
00065
00066                         MakeShared<FPhraseInputNode<int32>>(TEXT("INDEX"),
00067                         TPhraseNodeArray {
00068
00069                             MakeShared<FPhraseEventNode>(CreateParseDelegate(this,
&UIViewInteractionLibrary::IndexFocus))
00070
00071                             })
00072
00073                             })
00074
00075                         })
00076                     });
00077 }
00078
00079 void UIViewInteractionLibrary::MoveViewport(FParseRecord &Record) {
00080     GET_ACTIVE_TAB_CONTENT(ActiveTab)
00081
00082     FString TabType = ActiveTab->GetTypeAsString();
00083
00084     UParseEnumInput* DirectionInput = Record.GetPhraseInput<UParseEnumInput>(TEXT("DIRECTION"));
00085     UParseIntInput* AmountInput = Record.GetPhraseInput<UParseIntInput>(TEXT("AMOUNT"));
00086     if (DirectionInput == nullptr || AmountInput == nullptr)
00087         return;
00088
00089     if (TabType == "SGraphEditor")
00090     {
00091         TSharedPtr<SGraphEditor> GraphEditor = StaticCastSharedPtr<SGraphEditor>(ActiveTab);
00092
00093         FVector2D ViewLocation;
00094         float ZoomAmount;
00095         GraphEditor->GetViewLocation(ViewLocation, ZoomAmount);
00096
00097         switch (EPhrase2DDirectionalInput(DirectionInput->GetValue()))
00098         {
00099             case EPhrase2DDirectionalInput::UP:
00100                 ViewLocation.Y -= AmountInput->GetValue();
00101                 break;
00102
00103             case EPhrase2DDirectionalInput::DOWN:
00104                 ViewLocation.Y += AmountInput->GetValue();
00105                 break;
00106
00107             case EPhrase2DDirectionalInput::LEFT:
00108                 ViewLocation.X -= AmountInput->GetValue();
00109                 break;
00110
00111             case EPhrase2DDirectionalInput::RIGHT:
00112                 ViewLocation.X += AmountInput->GetValue();
00113                 break;
00114
00115             default:
00116                 UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("MoveViewport: INVALID DIRECTION
INPUT"));
00117                 return;
00118         }
00119

```



```

00120         GraphEditor->SetViewLocation(ViewLocation, ZoomAmount);
00121     }
00122
00123     // Further Viewport Implementation Here
00124 }
00125
00126 void UViewInteractionLibrary::ZoomViewport(FParseRecord &Record)
00127 {
00128     GET_ACTIVE_TAB_CONTENT(ActiveTab)
00129
00130     FString TabType = ActiveTab->GetTypeAsString();
00131
00132     UParseEnumInput* DirectionInput = Record.GetPhraseInput<UParseEnumInput>(TEXT("DIRECTION"));
00133     UParseIntInput* AmountInput = Record.GetPhraseInput<UParseIntInput>(TEXT("AMOUNT"));
00134     if (DirectionInput == nullptr || AmountInput == nullptr)
00135         return;
00136
00137     if (TabType == "SGraphEditor")
00138     {
00139         TSharedPtr<SGraphEditor> GraphEditor = StaticCastSharedPtr<SGraphEditor>(ActiveTab);
00140
00141         FVector2D ViewLocation;
00142         float ZoomAmount;
00143         GraphEditor->GetViewLocation(ViewLocation, ZoomAmount);
00144
00145         switch (EPhrase2DDirectionalInput(DirectionInput->GetValue())) {
00146             case EPhrase2DDirectionalInput::UP:
00147                 ZoomAmount += AmountInput->GetValue();
00148                 break;
00149
00150             case EPhrase2DDirectionalInput::DOWN:
00151                 ZoomAmount -= AmountInput->GetValue();
00152                 break;
00153
00154             default:
00155                 UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("ZoomViewport: INVALID DIRECTION
INPUT"));
00156                 return;
00157         }
00158
00159         GraphEditor->SetViewLocation(ViewLocation, ZoomAmount);
00160     }
00161
00162     // Further Viewport Specific Implementation Here
00163 }
00164
00165 void UViewInteractionLibrary::IndexFocus(FParseRecord& Record)
00166 {
00167     GET_ACTIVE_TAB_CONTENT(ActiveTab)
00168
00169     FString TabType = ActiveTab->GetTypeAsString();
00170
00171     UParseIntInput* IndexInput = Record.GetPhraseInput<UParseIntInput>(TEXT("INDEX"));
00172     if (IndexInput == nullptr)
00173         return;
00174
00175     if (TabType == "SGraphEditor")
00176     {
00177         TSharedPtr<SGraphEditor> GraphEditor = StaticCastSharedPtr<SGraphEditor>(ActiveTab);
00178         if (!GraphEditor.IsValid())
00179             return;
00180
00181         TSharedPtr<FAssetAccessibilityRegistry> AssetRegistry = GetAssetRegistry();
00182
00183         TSharedPtr<FGraphIndexer> GraphIndexer =
AssetRegistry->GetGraphIndexer(GraphEditor->GetCurrentGraph());
00184
00185         UEdGraphNode* Node = GraphIndexer->GetNode(IndexInput->GetValue());
00186         if (Node == nullptr)
00187         {
00188             UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("IndexFocus: INVALID INDEX INPUT"))
00189             return;
00190         }
00191
00192         GraphEditor->JumpToNode(Node);
00193     }
00194
00195     // Further ViewportS Specific Implementation Here
00196 }

```

## 5.25 WindowInteractionLibrary.cpp

```
00001 #include "PhraseEvents/WindowInteractionLibrary.h"
```

```

00002 #include "PhraseEvents/Utils.h"
00003
00004 #include "PhraseTree/PhraseInputNode.h"
00005 #include "PhraseTree/PhraseEventNode.h"
00006 #include "PhraseTree/Containers/Input/UParseIntInput.h"
00007
00008 #include "AccessibilityWrappers/AccessibilityWindowToolBar.h"
00009 #include "Framework/Docking/TabManager.h"
00010
00011 UWindowInteractionLibrary::UWindowInteractionLibrary(const FObjectInitializer& ObjectInitializer)
00012 : Super(ObjectInitializer)
00013 {
00014     WindowToolBar = NewObject<UAccessibilityWindowToolBar>();
00015 }
00016
00017 UWindowInteractionLibrary::~UWindowInteractionLibrary()
00018 {
00019
00020 }
00021
00022 void UWindowInteractionLibrary::BindBranches(TSharedRef<FPhraseTree> PhraseTree)
00023 {
00024     PhraseTree->BindBranches(
00025         TPhraseNodeArray{
00026
00027             MakeShared<FPhraseNode>(TEXT("WINDOW"),
00028                 TPhraseNodeArray{
00029
00030                     MakeShared<FPhraseNode>(TEXT("NEXT"),
00031                         TPhraseNodeArray {
00032
00033                             MakeShared<FPhraseEventNode>(CreateParseDelegate(this,
00034                                     &UWindowInteractionLibrary::SwitchNextActiveWindow))
00035                         }},
00036
00037                     MakeShared<FPhraseNode>(TEXT("PREVIOUS"),
00038                         TPhraseNodeArray {
00039
00040                             MakeShared<FPhraseEventNode>(CreateParseDelegate(this,
00041                                     &UWindowInteractionLibrary::SwitchPrevActiveWindow))
00042                         }},
00043
00044                     MakeShared<FPhraseNode>(TEXT("CLOSE"),
00045                         TPhraseNodeArray {
00046
00047                             MakeShared<FPhraseEventNode>(CreateParseDelegate(this,
00048                                     &UWindowInteractionLibrary::CloseActiveWindow))
00049                         }},
00050
00051                     }},
00052
00053                     MakeShared<FPhraseNode>(TEXT("TAB"),
00054                         TPhraseNodeArray {
00055
00056                             MakeShared<FPhraseNode>(TEXT("NEXT"),
00057                                 TPhraseNodeArray {
00058
00059                                     MakeShared<FPhraseEventNode>(CreateParseDelegate(this,
00060                                             &UWindowInteractionLibrary::SwitchNextTabInStack))
00061                                 }},
00062
00063                             MakeShared<FPhraseNode>(TEXT("PREVIOUS"),
00064                                 TPhraseNodeArray {
00065
00066                                     MakeShared<FPhraseEventNode>(CreateParseDelegate(this,
00067                                             &UWindowInteractionLibrary::SwitchPrevTabInStack))
00068                                 }},
00069
00070                             }},
00071
00072                     MakeShared<FPhraseNode>(TEXT("TOOLBAR"),
00073                         TPhraseNodeArray {
00074
00075                             MakeShared<FPhraseInputNode<int32>>(TEXT("ITEM_INDEX"),
00076                                 TPhraseNodeArray {
00077
00078                                     MakeShared<FPhraseEventNode>(CreateParseDelegate(this,
00079                                             &UWindowInteractionLibrary::SelectToolBarItem))
00080                                 }},
00081
00082                     }},

```

```

00083
00084     }
00085 );
00086 }
00087
00088 void UWindowInteractionLibrary::SwitchNextActiveWindow(FParseRecord& Record)
00089 {
00090     GET_ACTIVE_REGULAR_WINDOW(ActiveWindow)
00091
00092     TArray<TSharedRef<SWindow>> AllWindows;
00093     FSlateApplication::Get().GetAllVisibleWindowsOrdered(AllWindows);
00094
00095     int32 FoundIndex;
00096     if (!AllWindows.Find(ActiveWindow.ToSharedRef(), FoundIndex))
00097     {
00098         UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("SwitchNextActiveWindow: Cannot Find the
Current Active Window."))
00099         return;
00100     }
00101
00102     TSharedRef<SWindow> NextWindow = AllWindows[FoundIndex + 1 % AllWindows.Num()];
00103
00104     NextWindow->BringToFront(true);
00105
00106     // Set Window Major Tab Focus?
00107 }
00108
00109 void UWindowInteractionLibrary::SwitchPrevActiveWindow(FParseRecord& Record)
00110 {
00111     GET_ACTIVE_REGULAR_WINDOW(ActiveWindow)
00112
00113     TArray<TSharedRef<SWindow>> AllWindows;
00114     FSlateApplication::Get().GetAllVisibleWindowsOrdered(AllWindows);
00115
00116     int32 FoundIndex;
00117     if (!AllWindows.Find(ActiveWindow.ToSharedRef(), FoundIndex))
00118     {
00119         UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("SwitchPrevActiveWindow: Cannot Find the
Current Active Window."))
00120         return;
00121     }
00122
00123     TSharedRef<SWindow> PrevWindow = AllWindows[
00124         FoundIndex - 1 < 0
00125         ? AllWindows.Num() - 1
00126         : FoundIndex - 1
00127     ];
00128
00129     PrevWindow->BringToFront(true);
00130
00131     // Set Window Major Tab Focus?
00132 }
00133
00134 void UWindowInteractionLibrary::CloseActiveWindow(FParseRecord &Record) {
00135     FSlateApplication& SlateApp = FSlateApplication::Get();
00136     if (!SlateApp.CanDisplayWindows())
00137     {
00138         UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("CloseActiveWindow: Slate Application
cannot display windows."));
00139         return;
00140     }
00141
00142     TSharedPtr<SWindow> ActiveWindow = SlateApp.GetActiveTopLevelWindow();
00143     if (!ActiveWindow.IsValid())
00144     {
00145         UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("CloseActiveWindow: No Active Window
Found."));
00146         return;
00147     }
00148
00149     TSharedPtr<SWindow> RootWindow = FGlobalTabmanager::Get()->GetRootWindow();
00150     if (ActiveWindow->IsVisible() && ActiveWindow != RootWindow)
00151     {
00152         ActiveWindow->RequestDestroyWindow();
00153     }
00154 }
00155
00156 void UWindowInteractionLibrary::SelectToolBarItem(FParseRecord& Record)
00157 {
00158     UParseIntInput* ItemIndexInput = Record.GetPhraseInput<UParseIntInput>(TEXT("ITEM_INDEX"));
00159     if (ItemIndexInput == nullptr)
00160     {
00161         UE_LOG(LogOpenAccessibilityPhraseEvent, Display, TEXT("SelectToolBarItem: No Item Index
Found."));
00162         return;
00163     }
00164

```

```

00165     WindowToolBar->SelectToolBarItem(ItemIndexInput->GetValue());
00166 }
00167
00168 namespace TabUtils
00169 {
00170     class FOpenArea : public FTabManager::FArea
00171     {
00172     public:
00173         const TArray<TSharedRef<FLayoutNode>& GetChildNodes()
00174         {
00175             return ChildNodes;
00176         }
00177     };
00178
00179     class FOpenStack : public FTabManager::FStack
00180     {
00181     public:
00182         const TArray<FTabManager::FTab> GetTabs()
00183         {
00184             return Tabs;
00185         }
00186     };
00187
00188     class FOpenSplitter : public FTabManager::FSplitter
00189     {
00190     public:
00191         const TArray<TSharedRef<FLayoutNode>& GetChildNodes()
00192         {
00193             return ChildNodes;
00194         }
00195     };
00196
00197     [[nodiscard]] TArray<FTabManager::FTab> CollectManagedTabs(const TSharedRef<FTabManager>&
    TabManager)
00198     {
00199         TArray<FTabManager::FTab> ManagedTabs = TArray<FTabManager::FTab>();
00200
00201         const TSharedRef<FTabManager::FLayout> Layout = TabManager->PersistLayout();
00202         const TSharedPtr<FOpenArea> MainArea =
    StaticCastSharedPtr<FOpenArea>(Layout->GetPrimaryArea().Pin());
00203         if (!MainArea.IsValid())
00204         {
00205             UE_LOG(LogOpenAccessibility, Warning, TEXT("CollectManagedTabs: Primary Area is Not
    Valid"))
00206             return ManagedTabs;
00207         }
00208
00209         TSharedPtr<FTabManager::FLayoutNode> CurrentNode = nullptr;
00210         TArray<TSharedRef<FTabManager::FLayoutNode>& NodesToCheck = TArray{
00211             MainArea->GetChildNodes()
00212         };
00213
00214         while (!NodesToCheck.IsEmpty())
00215         {
00216             CurrentNode = NodesToCheck[0];
00217             NodesToCheck.RemoveAt(0);
00218
00219             if (TSharedPtr<FTabManager::FStack> AsStack = CurrentNode->AsStack())
00220             {
00221                 ManagedTabs.Append(StaticCastSharedPtr<FOpenStack>(AsStack->GetTabs()));
00222             }
00223             else if (TSharedPtr<FTabManager::FArea> AsArea = CurrentNode->AsArea())
00224             {
00225                 NodesToCheck.Append(StaticCastSharedPtr<FOpenArea>(AsArea->GetChildNodes()));
00226             }
00227             else if (TSharedPtr<FTabManager::FSplitter> AsSplitter = CurrentNode->AsSplitter())
00228             {
00229                 NodesToCheck.Append(StaticCastSharedPtr<FOpenArea>(AsSplitter->GetChildNodes()));
00230             }
00231             else
00232             {
00233                 UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("CollectManagedTabs: Unknown
    Node Type."))
00234             }
00235         }
00236
00237         return ManagedTabs;
00238     }
00239 }
00240
00241 void UWindowInteractionLibrary::SwitchNextTabInStack(FParseRecord& Record)
00242 {
00243     GET_ACTIVE_TAB(ActiveTab);
00244
00245     TSharedPtr<FTabManager> ActiveTabManager = ActiveTab->GetTabManagerPtr();
00246     if (!ActiveTabManager.IsValid())
00247     {

```

```

00248         UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("SwitchNextTabInStack: Cannot Find
Active Tab Manager"))
00249         return;
00250     }
00251
00252     TArray<FTabManager::FTab> FoundTabs =
TabUtils::CollectManagedTabs(ActiveTabManager.ToSharedRef());
00253
00254     const FTabId ActiveTabId = ActiveTab->GetLayoutIdentifier();
00255     for (int32 i = 0; i < FoundTabs.Num(); i++)
00256     {
00257         if (FoundTabs[i].TabId == ActiveTabId)
00258         {
00259             TSharedPtr<SDockTab> NextTabWidget = TSharedPtr<SDockTab>();
00260
00261             int CurrentTabIndex = i;
00262             while (!NextTabWidget.IsValid())
00263             {
00264                 CurrentTabIndex += 1;
00265                 CurrentTabIndex %= FoundTabs.Num();
00266
00267                 const FTabManager::FTab NextTab = FoundTabs[
CurrentTabIndex
00268             ];
00269
00270             if (NextTab.TabId == ActiveTabId)
00271             {
00272                 UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("SwitchNextTabInStack: No
Next Tab Found.))
00273                 return;
00274             }
00275
00276             NextTabWidget = ActiveTabManager->FindExistingLiveTab(
NextTab.TabId
00277         );
00278     }
00279
00280     FGlobalTabmanager::Get()->SetActiveTab(NextTabWidget);
00281     NextTabWidget->ActivateInParent(SetDirectly);
00282
00283     break;
00284 }
00285
00286 /*
00287 // Most Straightforward Implementation, But Requires Private Core Tab Classes
00288
00289 TSharedPtr<SDockingTabStack> ActiveTabStack = ActiveTab->GetParentDockTabStack();
00290 if (!ActiveTabStack.IsValid())
00291 {
00292     UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("SwitchNextTabInStack: No Parent Tab
Stack Found.))
00293     return;
00294 }
00295
00296 TArray<TSharedPtr<SDockTab>> AllTabs = ActiveTabStack->GetAllChildTabs();
00297
00298 int32 FoundIndex;
00299 if (!AllTabs.Find(ActiveTab.ToSharedRef(), FoundIndex))
00300 {
00301     UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("SwitchNextTabInStack: Active Tab Not
Found In Tab Stack.))
00302     return;
00303 }
00304
00305 TSharedPtr<SDockTab> NextTab = AllTabs[FoundIndex + 1 % AllTabs.Num()];
00306
00307 FGlobalTabmanager::Get()->SetActiveTab(NextTab);
00308 NextTab->ActivateInParent(SetDirectly);
00309 */
00310 }
00311
00312 void UWindowInteractionLibrary::SwitchPrevTabInStack(FParseRecord& Record)
00313 {
00314     GET_ACTIVE_TAB(ActiveTab);
00315
00316     TSharedPtr<FTabManager> ActiveTabManager = ActiveTab->GetTabManagerPtr();
00317     if (!ActiveTabManager.IsValid())
00318     {
00319         UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("SwitchPrevTabInStack: Cannot Find
Active Tab Manager"))
00320         return;
00321     }
00322
00323     TArray<FTabManager::FTab> FoundTabs =
TabUtils::CollectManagedTabs(ActiveTabManager.ToSharedRef());
00324
00325     break;
00326 }
00327

```

```

00328     const FTabId ActiveTabId = ActiveTab->GetLayoutIdentifier();
00329     for (int32 i = 0; i < FoundTabs.Num(); i++)
00330     {
00331         if (FoundTabs[i].TabId == ActiveTabId)
00332         {
00333             TSharedPtr<SDockTab> PrevTabWidget = TSharedPtr<SDockTab>();
00334
00335             int CurrentTabIndex = i;
00336             while (!PrevTabWidget.IsValid())
00337             {
00338                 CurrentTabIndex -= 1;
00339                 if (CurrentTabIndex < 0)
00340                 {
00341                     CurrentTabIndex = FoundTabs.Num() - 1;
00342                 }
00343
00344                 const FTabManager::FTab NextTab = FoundTabs[
00345                     CurrentTabIndex
00346                 ];
00347
00348                 if (NextTab.TabId == ActiveTabId)
00349                 {
00350                     UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("SwitchNextTabInStack: No
Next Tab Found.))
00351                         return;
00352                 }
00353
00354                 PrevTabWidget = ActiveTabManager->FindExistingLiveTab(
00355                     NextTab.TabId
00356                 );
00357             }
00358
00359             FGlobalTabmanager::Get()->SetActiveTab(PrevTabWidget);
00360             PrevTabWidget->ActivateInParent(SetDirectly);
00361
00362             break;
00363         }
00364     }
00365
00366     /*
00367     // Most Straightforward Implementation, But Requires Private Core Tab Classes
00368
00369     TSharedPtr<SDockingTabStack> ActiveTabStack = ActiveTab->GetParentDockTabStack();
00370     if (!ActiveTabStack.IsValid())
00371     {
00372         UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("SwitchPrevTabInStack: No Parent Tab
Stack Found.))
00373             return;
00374     }
00375
00376     TArray<TSharedPtr<SDockTab>> AllTabs = ActiveTabStack->GetAllChildTabs();
00377
00378     int32 FoundIndex;
00379     if (!AllTabs.Find(ActiveTab.ToSharedRef(), FoundIndex))
00380     {
00381         UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, TEXT("SwitchPrevTabInStack: Active Tab Not
Found In Tab Stack.))
00382             return;
00383     }
00384
00385     TSharedPtr<SDockTab> PrevTab = AllTabs[
00386         FoundIndex - 1 < 0
00387         ? AllTabs.Num() - 1
00388         : FoundIndex - 1
00389     ];
00390
00391     FGlobalTabmanager::Get()->SetActiveTab(PrevTab);
00392     PrevTab->ActivateInParent(SetDirectly);
00393     */
00394 }

```

## 5.26 TranscriptionVisualizer.cpp

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #include "TranscriptionVisualizer.h"
00004
00005 #include "AccessibilityWidgets/SAccessibilityTranscriptionVis.h"
00006
00007 FTranscriptionVisualizer::FTranscriptionVisualizer()
00008 {
00009     RegisterTicker();
00010 }

```

```

00011
00012 FTranscriptionVisualizer::~FTranscriptionVisualizer()
00013 {
00014     UnregisterTicker();
00015 }
00016
00017 bool FTranscriptionVisualizer::Tick(float DeltaTime)
00018 {
00019     if (VisWindow.IsValid())
00020     {
00021         UpdateVisualizer();
00022     }
00023     else if (FSlateApplication::Get().GetActiveTopLevelRegularWindow().IsValid() &&
FSlateApplication::Get().IsActive())
00024     {
00025         ConstructVisualizer();
00026     }
00027
00028     return true;
00029 }
00030
00031 void FTranscriptionVisualizer::ConstructVisualizer()
00032 {
00033     TSharedPtr<SAccessibilityTranscriptionVis> MenuContent = SNew(SAccessibilityTranscriptionVis)
00034         .VisAmount(2);
00035
00036     MenuContent->ForceVolatile(true);
00037
00038     FDisplayMetrics DisplayMetrics;
00039     FSlateApplication::Get().GetDisplayMetrics(DisplayMetrics);
00040
00041     FVector2D VisPosition = FVector2D();
00042
00043     if (FSlateApplication::Get().GetActiveTopLevelRegularWindow().IsValid())
00044     {
00045         VisPosition =
FSlateApplication::Get().GetActiveTopLevelRegularWindow()->GetPositionInScreen();
00046     }
00047     VisPosition.X = DisplayMetrics.PrimaryDisplayWidth;
00048     VisPosition.Y = DisplayMetrics.PrimaryDisplayHeight;
00049
00050     TSharedRef<SWindow> MenuWindow = SNew(SWindow)
00051         .Type(EWindowType::Normal)
00052         .SizingRule(ESizingRule::Autosized)
00053         .ScreenPosition(VisPosition)
00054         .ClientSize(FVector2D(10, 10))
00055         .IsPopupWindow(true)
00056         //.InitialOpacity(0.5f)
00057         .SupportsTransparency(EWindowTransparency::PerWindow)
00058         .ActivationPolicy(EWindowActivationPolicy::Always)
00059         .AdjustInitialSizeAndPositionForDPIScale(true)
00060     [
00061         MenuContent.ToSharedRef()
00062     ];
00063
00064     TSharedPtr<SWindow> TopLevelWindow = FSlateApplication::Get().GetActiveTopLevelRegularWindow();
00065
00066     MenuWindow->AssignParentWidget(TopLevelWindow);
00067     FSlateApplication::Get().AddWindowAsNativeChild(MenuWindow, TopLevelWindow.ToSharedRef(), true);
00068
00069     VisWindow = MenuWindow.ToWeakPtr();
00070     VisContent = MenuContent.ToWeakPtr();
00071 }
00072
00073 void FTranscriptionVisualizer::UpdateVisualizer()
00074 {
00075     if (FSlateApplication::Get().IsActive())
00076     {
00077         VisWindow.Pin()->ShowWindow();
00078
00079         // ReparentWindow();
00080
00081         MoveVisualizer();
00082     }
00083     else VisWindow.Pin()->HideWindow();
00084 }
00085
00086 void FTranscriptionVisualizer::ReparentWindow()
00087 {
00088     TSharedPtr<SWindow> TopLevelActiveWindow =
FSlateApplication::Get().GetActiveTopLevelRegularWindow();
00089     if (!TopLevelActiveWindow.IsValid())
00090         return;
00091
00092     TSharedPtr<SWindow> VisWindowPtr = VisWindow.Pin();
00093
00094     if (TopLevelActiveWindow == VisWindow.Pin() ||

```

```

00095         TopLevelActiveWindow->GetContent() == VisWindowPtr->GetParentWidget())
00096         return;
00097
00098         TSharedPtr<SWindow> PrevParentWindow = VisWindowPtr->GetParentWindow();
00099         if (PrevParentWindow.IsValid())
00100         {
00101             PrevParentWindow->RemoveDescendantWindow(VisWindowPtr.ToSharedRef());
00102         }
00103
00104         VisWindowPtr->AssignParentWidget(TopLevelActiveWindow);
00105         TopLevelActiveWindow->AddChildWindow(VisWindowPtr.ToSharedRef());
00106     }
00107
00108     void FTranscriptionVisualizer::MoveVisualizer()
00109     {
00110         FVector2D NewPosition = FVector2D();
00111
00112         if (!GetTopScreenVisualizerPosition(NewPosition))
00113         {
00114             GetDisplayVisualizerPosition(NewPosition);
00115         }
00116
00117         VisWindow.Pin()->MoveWindowTo(NewPosition);
00118     }
00119
00120     void FTranscriptionVisualizer::OnTranscriptionRecieved(TArray<FString> InTranscription)
00121     {
00122         for (int i = 0; i < InTranscription.Num(); i++)
00123         {
00124             VisContent.Pin()->UpdateTopTranscription(InTranscription[i]);
00125         }
00126     }
00127
00128     bool FTranscriptionVisualizer::GetTopScreenVisualizerPosition(FVector2D& OutPosition)
00129     {
00130         TSharedPtr<SWindow> TopLevelWindow = FSlateApplication::Get().GetActiveTopLevelRegularWindow();
00131         if (!TopLevelWindow.IsValid())
00132             return false;
00133
00134         FVector2D ActiveWindowPosition = TopLevelWindow->GetPositionInScreen();
00135         FVector2D ActiveWindowBounds = TopLevelWindow->GetClientSizeInScreen();
00136
00137         TSharedPtr<SWindow> VisWindowPtr = VisWindow.Pin();
00138
00139         OutPosition.X = (ActiveWindowPosition.X + ActiveWindowBounds.X / 2) -
00140             (VisWindowPtr->GetClientSizeInScreen().X / 2);
00141         OutPosition.Y = (ActiveWindowPosition.Y + ActiveWindowBounds.Y - 50) -
00142             VisWindowPtr->GetClientSizeInScreen().Y;
00143
00144         return true;
00145     }
00146
00147     bool FTranscriptionVisualizer::GetDisplayVisualizerPosition(FVector2D& OutPosition)
00148     {
00149         FDisplayMetrics DisplayMetrics;
00150         FSlateApplication::Get().GetDisplayMetrics(DisplayMetrics);
00151
00152         OutPosition.X = DisplayMetrics.PrimaryDisplayWidth;
00153         OutPosition.Y = DisplayMetrics.PrimaryDisplayHeight;
00154
00155         return true;
00156     }
00157
00158     void FTranscriptionVisualizer::RegisterTicker()
00159     {
00160         FTickerDelegate TickDelegate = FTickerDelegate::CreateRaw(this, &FTranscriptionVisualizer::Tick);
00161         TickDelegateHandle = FTSTicker::GetCoreTicker().AddTicker(TickDelegate);
00162     }
00163
00164     void FTranscriptionVisualizer::UnregisterTicker()
00165     {
00166         FTSTicker::GetCoreTicker().RemoveTicker(TickDelegateHandle);
00167     }

```

## 5.27 WidgetUtils.h

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #pragma once
00004
00005 #include "CoreMinimal.h"
00006

```



```

00014 template<class T>
00015 [[nodiscard]] FORCEINLINE TSharedPtr<T> GetWidgetDescendant(const TSharedRef<SWidget>& SearchRoot,
00016     FString TargetWidgetType)
00017 {
00018     static_assert(TIsDerivedFrom<T, SWidget>::IsDerived, "Provided Type Is Not a Valid Widget Type.");
00019     TargetWidgetType.RemoveSpacesInline();
00020     if (SearchRoot->GetType() == TargetWidgetType)
00021         return StaticCastSharedRef<T>(SearchRoot);
00022     {
00023         TArray<FChildren*> ChildrenToSearch = TArray{
00024             SearchRoot->GetChildren();
00025         };
00026         FChildren* CurrentChildren;
00027         TSharedPtr<SWidget> CurrentChild;
00028         FString CurrentChildString;
00029         while (ChildrenToSearch.Num() > 0)
00030         {
00031             CurrentChildren = ChildrenToSearch.Pop();
00032             for (int i = 0; i < CurrentChildren->Num(); i++)
00033             {
00034                 CurrentChild = CurrentChildren->GetChildAt(i);
00035                 CurrentChildString = CurrentChild->GetTypeAsString();
00036                 CurrentChildString.RemoveSpacesInline();
00037                 if (CurrentChildString == TargetWidgetType)
00038                     return StaticCastSharedRef<T>(CurrentChild);
00039                 ChildrenToSearch.Add(CurrentChild->GetChildren());
00040             }
00041         }
00042     }
00043     return TSharedPtr<T>();
00044 }
00045
00046 template <class T>
00047 [[nodiscard]] FORCEINLINE TArray<TSharedPtr<T>> GetWidgetDescendants(const TSharedRef<SWidget>&
00048     SearchRoot, FString TargetWidgetType)
00049 {
00050     static_assert(TIsDerivedFrom<T, SWidget>::IsDerived, "Provided Type Is Not a Valid Widget Type.");
00051     TargetWidgetType.RemoveSpacesInline();
00052     TArray<TSharedPtr<T>> FoundDescendants = TArray<TSharedPtr<T>>();
00053     if (SearchRoot->GetTypeAsString() == TargetWidgetType)
00054         FoundDescendants.Add(StaticCastSharedRef<T>(SearchRoot));
00055     {
00056         TArray<FChildren*> ChildrenToSearch = TArray {
00057             SearchRoot->GetChildren();
00058         };
00059         while (ChildrenToSearch.Num() > 0)
00060         {
00061             FChildren* CurrentChildren = ChildrenToSearch.Pop();
00062             for (int i = 0; i < CurrentChildren->Num(); i++)
00063             {
00064                 TSharedPtr<SWidget> CurrentChild = CurrentChildren->GetChildAt(i);
00065                 FString CurrentChildString = CurrentChild->GetTypeAsString();
00066                 CurrentChildString.RemoveSpacesInline();
00067                 if (CurrentChildString == TargetWidgetType)
00068                     FoundDescendants.Add(StaticCastSharedRef<T>(CurrentChild));
00069                 ChildrenToSearch.Add(CurrentChild->GetChildren());
00070             }
00071         }
00072     }
00073     return FoundDescendants;
00074 }
00075
00076 [[nodiscard]] FORCEINLINE TArray<FSlotBase*> GetWidgetSlotsByType(const TSharedRef<SWidget>&
00077     SearchRoot, const TSet<FString>& TargetTypes)
00078 {
00079     TArray<FSlotBase*> FoundDescendants = TArray<FSlotBase*>();

```

```

00111
00112     {
00113         TArray<FChildren*> ChildrenToSearch = TArray{
00114             SearchRoot->GetChildren();
00115         };
00116
00117         FChildren* CurrentChildren;
00118         FString CurrentWidgetString;
00119
00120         while (ChildrenToSearch.Num() > 0)
00121         {
00122             CurrentChildren = ChildrenToSearch.Pop();
00123
00124             for (int i = 0; i < CurrentChildren->NumSlot(); i++)
00125             {
00126                 FSlotBase& CurrentSlot = const_cast<FSlotBase&>(CurrentChildren->GetSlotAt(i));
00127
00128                 const TSharedRef<SWidget> CurrentWidget = CurrentSlot.GetWidget();
00129
00130                 CurrentWidgetString = CurrentWidget->GetTypeAsString();
00131                 CurrentWidgetString.RemoveSpacesInline();
00132
00133                 if (TargetTypes.Contains(CurrentWidgetString))
00134                     FoundDescendants.Add(&CurrentSlot);
00135
00136                 ChildrenToSearch.Add(CurrentWidget->GetChildren());
00137             }
00138         }
00139     }
00140
00141     return FoundDescendants;
00142 }

```

## 5.28 AccessibilityNodeFactory.h

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #pragma once
00004
00005 #include "CoreMinimal.h"
00006 #include "NodeFactory.h"
00007 #include "OpenAccessibility.h"
00008 #include "AccessibilityWidgets/SIndexer.h"
00009
00010 #include "SGraphNode.h"
00011 #include "SGraphPin.h"
00012
00016 template<class T>
00017 class OPENACCESSIBILITY_API TGraphAccessibilityNodeFactory : public FGraphNodeFactory
00018 {
00019 public:
00020
00021     static_assert(TIsDerivedFrom<T, FGraphNodeFactory>::IsDerived, "Provided Template Type Must Derive From FGraphNodeFactory");
00022
00023     TGraphAccessibilityNodeFactory()
00024     {
00025         Implementation = TSharedPtr<T>();
00026
00027         AccessibilityRegistry =
00028             FOpenAccessibilityModule::Get().AssetAccessibilityRegistry.ToSharedRef();
00029
00030     }
00031
00032     TGraphAccessibilityNodeFactory(TSharedRef<FAssetAccessibilityRegistry> InAccessibilityRegistry)
00033     : AccessibilityRegistry(InAccessibilityRegistry)
00034     {
00035         Implementation = TSharedPtr<T>();
00036     }
00037
00038     virtual ~TGraphAccessibilityNodeFactory()
00039     {
00040     }
00041
00042     /* FGraphNodeFactory Implementation */
00043
00048     virtual TSharedPtr<class SGraphNode> CreateNodeWidget(UEdGraphNode* InNode) override;
00049
00055     virtual TSharedPtr<class SGraphPin> CreatePinWidget(UEdGraphPin* InPin) override;
00056
00057     /* End Of FGraphNodeFactory Implementation*/
00058
00059 protected:

```

```

00060
00064     TSharedRef<FAssetAccessibilityRegistry> AccessibilityRegistry;
00065
00066     TSharedPtr<T> Implementation;
00067 };
00068
00069 template<class T>
00070 TSharedPtr<class SGraphNode> TGraphAccessibilityNodeFactory<T>::CreateNodeWidget (UEdGraphNode* InNode)
00071 {
00072     check(InNode != nullptr);
00073
00074     TSharedPtr<SGraphNode> OutNode = Implementation->CreateNodeWidget(InNode);
00075
00076     // Apply Accessibility Visuals to the Node.
00077
00078     TSharedRef<FGraphIndexer> GraphIndexer =
        AccessibilityRegistry->GetGraphIndexer(InNode->GetGraph());
00079
00080     int NodeIndex = -1;
00081     GraphIndexer->GetOrAddNode(InNode);
00082
00083     TSharedRef<SWidget> WidgetToWrap = OutNode->GetSlot(ENodeZone::Center)->GetWidget();
00084
00085     check(WidgetToWrap != SNullWidget::NullWidget);
00086
00087     OutNode->GetOrAddSlot(ENodeZone::Center)
00088         .HAlign(HAlign_Fill)
00089         [
00090             SNew(SVerticalBox)
00091
00092             + SVerticalBox::Slot()
00093             .HAlign(HAlign_Fill)
00094             .AutoHeight()
00095             .Padding(FMargin(1.5f, 0.25f))
00096             [
00097                 SNew(SOverlay)
00098
00099                 + SOverlay::Slot()
00100                 [
00101                     SNew(SImage)
00102                     .Image(FAppStyle::Get().GetBrush("Graph.Node.Body"))
00103                 ]
00104
00105                 + SOverlay::Slot()
00106                 .Padding(FMargin(4.0f, 0.0f))
00107                 [
00108                     SNew(SHorizontalBox)
00109                     + SHorizontalBox::Slot()
00110                     .HAlign(HAlign_Right)
00111                     .VAlign(VAlign_Center)
00112                     .Padding(1.f)
00113                     [
00114                         SNew(SOverlay)
00115                         + SOverlay::Slot()
00116                         [
00117                             SNew(SIndexer)
00118                             .IndexValue(NodeIndex)
00119                             .TextColor(FLinearColor::White)
00120                             .BorderColor(FLinearColor::Gray)
00121                         ]
00122                     ]
00123                 ]
00124             ]
00125
00126             + SVerticalBox::Slot()
00127             .HAlign(HAlign_Fill)
00128             .AutoHeight()
00129             [
00130                 WidgetToWrap
00131             ]
00132         ];
00133
00134     return OutNode;
00135 }
00136
00137 template<class T>
00138 TSharedPtr<class SGraphPin> TGraphAccessibilityNodeFactory<T>::CreatePinWidget (UEdGraphPin* InPin)
00139 {
00140     check(InPin != nullptr);
00141
00142     TSharedPtr<SGraphPin> OutPin = Implementation->CreatePinWidget(InPin);
00143     SGraphPin* OutPinPtr = OutPin.Get();
00144
00145     TSharedRef<FGraphIndexer> GraphIndexer =
        AccessibilityRegistry->GetGraphIndexer(InPin->GetOwningNode()->GetGraph());
00146
00147     int PinIndex = -1;

```

```

00148     PinIndex = InPin->GetOwningNode()->GetPinIndex(InPin);
00149
00150     TSharedRef<SWidget> AccessibilityWidget = SNew(SOverlay)
00151         .Visibility_Lambda([OutPinPtr]() -> EVisibility {
00152             if (OutPinPtr->HasAnyUserFocusOrFocusedDescendants() || OutPinPtr->IsHovered())
00153                 return EVisibility::Visible;
00154             return EVisibility::Hidden;
00155         })
00156     + SOverlay::Slot()
00157     [
00158         SNew(STextBlock)
00159             .ColorAndOpacity(FLinearColor::White)
00160             .ShadowColorAndOpacity(FLinearColor::Black)
00161             .ShadowOffset(FIntPoint(-1, 1))
00162             .Font(FAppStyle::Get().GetFontStyle("Graph.Node.Pin.Font"))
00163             .Text(FText::FromString("[ " + FString::FromInt(PinIndex) + "]" ));
00164     ];
00165
00166     // Get Pin Widget Content, before modifying it.
00167     TSharedRef<SWidget> PinWidgetContent = OutPin->GetContent();
00168
00169     // Modify the Pin Widget Content, based on the Pin's Direction.
00170     switch (OutPin->GetDirection())
00171     {
00172     case EEdGraphPinDirection::EGPD_Input:
00173     {
00174         OutPin->SetContent(
00175             SNew(SHorizontalBox)
00176                 + SHorizontalBox::Slot()
00177                 [
00178                     PinWidgetContent
00179                 ]
00180                 + SHorizontalBox::Slot()
00181                 [
00182                     AccessibilityWidget
00183                 ]
00184             );
00185         break;
00186     }
00187     case EEdGraphPinDirection::EGPD_Output:
00188     {
00189         OutPin->SetContent(
00190             SNew(SHorizontalBox)
00191                 + SHorizontalBox::Slot()
00192                 .AutoWidth()
00193                 [
00194                     AccessibilityWidget
00195                 ]
00196                 + SHorizontalBox::Slot()
00197                 .AutoWidth()
00198                 [
00199                     PinWidgetContent
00200                 ]
00201             );
00202         break;
00203     }
00204     }
00205     return OutPin;
00206 }
00207 }
00208 }
00209
00210 }
00211 }

```

## 5.29 SAccessibilityTranscriptionVis.h

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #pragma once
00004
00005 #include "CoreMinimal.h"
00006 #include "Styling/AppStyle.h"
00007 #include "Widgets/Layout/SBorder.h"
00008
00009 class OPENACCESSIBILITY_API SAccessibilityTranscriptionVis : public SBox
00010 {
00011 public:
00012     SLATE_BEGIN_ARGS(SAccessibilityTranscriptionVis)
00013     : _VisAmount(1)
00014     {}
00015     SLATE_ARGUMENT(int, VisAmount)
00016 }

```

```

00017     SLATE_END_ARGS()
00018
00019     ~SAccessibilityTranscriptionVis();
00020
00021     void Construct(const FArguments& InArgs);
00022
00023     // SWidget Interface
00024
00025     virtual void Tick(const FGeometry& AllottedGeometry, const double InCurrentTime, const float
InDeltaTime) override;
00026
00027     // End of SWidget Interface
00028
00029     void UpdateTopTranscription(const FString& InTopTranscription);
00030
00031 protected:
00032
00033     TWeakPtr<SVerticalBox> TranscriptionContainer;
00034
00035     TArray<TWeakPtr<STextBlock>> TranscriptionSlots;
00036 };

```

## 5.30 SContentIndexer.h

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #pragma once
00004
00005 #include "CoreMinimal.h"
00006 #include "Widgets/DeclarativeSyntaxSupport.h"
00007
00008 enum class EIndexerPosition : uint8
00009 {
00010     Top,
00011     Bottom,
00012     Left,
00013     Right
00014 };
00015
00016 class OPENACCESSIBILITY_API SContentIndexer : public SBox
00017 {
00018 public:
00019
00020     SLATE_BEGIN_ARGS( SContentIndexer )
00021     : _IndexValue(0)
00022     , _IndexPositionToContent(EIndexerPosition::Left)
00023     , _ContentToIndex(SNullWidget::NullWidget)
00024     {}
00025     SLATE_ARGUMENT(int32, IndexValue)
00026     SLATE_ARGUMENT(EIndexerPosition, IndexPositionToContent)
00027     SLATE_ARGUMENT(TSharedPtr<SWidget>, ContentToIndex)
00028
00029     SLATE_PRIVATE_ATTRIBUTE_VARIABLE(EVisibility, IndexVisibility) = EVisibility::Visible;
00030     SLATE_PRIVATE_ATTRIBUTE_FUNCTION(EVisibility, IndexVisibility)
00031     SLATE_END_ARGS()
00032
00033     ~SContentIndexer();
00034
00035
00036     void Construct(const FArguments& InArgs);
00037
00038     // SWidget Implementation
00039
00040     virtual void Tick(const FGeometry& AllottedGeometry, const double InCurrentTime, const float
InDeltaTime) override;
00041
00042     // End SWidget Implementation
00043
00044     void UpdateIndex(const int32 IndexValue);
00045
00046     TSharedRef<SWidget> GetContent() const
00047     {
00048         return IndexedContent.Pin().ToSharedRef();
00049     }
00050
00051     template<typename CastType>
00052     TSharedRef<CastType> GetContent() const
00053     {
00054         return CastStaticSharedPtr<CastType>(IndexedContent.Pin());
00055     }
00056
00057 protected:

```

```

00072
00079     TSharedPtr<SWidget> ConstructTopIndexer(const FArguments& InArgs);
00080
00087     TSharedPtr<SWidget> ConstructBottomIndexer(const FArguments& InArgs);
00088
00095     TSharedPtr<SWidget> ConstructLeftIndexer(const FArguments& InArgs);
00096
00103     TSharedPtr<SWidget> ConstructRightIndexer(const FArguments& InArgs);
00104
00110     TSharedPtr<SWidget> ConstructContentContainer(TSharedPtr<SWidget> ContentToIndex);
00111
00118     TSharedPtr<SWidget> ConstructIndexContainer(const FArguments& InArgs, FLinearColor TextColor =
FLinearColor::White);
00119
00125     FText ConstructIndexText(int32 Index);
00126
00127 protected:
00128
00132     TWeakPtr<SWidget> IndexedContent;
00133
00137     TWeakPtr<class SIndexer> IndexerWidget;
00138 };

```

## 5.31 SIndexer.h

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #pragma once
00004
00005 #include "CoreMinimal.h"
00006
00007 class OPENACCESSIBILITY_API SIndexer : public SBox {
00008 public:
00009
00010     SLATE_BEGIN_ARGS( SIndexer )
00011     : _TextColor(FLinearColor::White)
00012     , _BorderColor(FLinearColor::Black)
00013     {}
00014
00015     SLATE_ARGUMENT(FLinearColor, TextColor)
00016     SLATE_ARGUMENT(FLinearColor, BorderColor)
00017
00018     SLATE_PRIVATE_ARGUMENT_VARIABLE(int32, IndexValue) = -1;
00019     SLATE_PRIVATE_ARGUMENT_FUNCTION(int32, IndexValue)
00020     SLATE_PRIVATE_ATTRIBUTE_VARIABLE(EVisibility, IndexVisibility) = EVisibility::Visible;
00021     SLATE_PRIVATE_ATTRIBUTE_FUNCTION(EVisibility, IndexVisibility)
00022     SLATE_END_ARGS()
00023
00024     ~SIndexer();
00025
00026     // SWidget Implementation
00027
00028     virtual void Tick(const FGeometry& AllottedGeometry, const double InCurrentTime, const float
InDeltaTime) override;
00029
00030     void Construct(const FArguments& InArgs);
00031
00032     // End SWidget Implementation
00033
00034     void UpdateIndex(const int32 NewIndex);
00035
00036     void UpdateIndex(const FString& NewIndex);
00037
00038     void UpdateIndex(const FText& NewIndex);
00039
00040     TSharedPtr<STextBlock> GetIndexText() const
00041     {
00042         return IndexTextBlock.IsValid() ? IndexTextBlock.Pin() : TSharedPtr<STextBlock>();
00043     }
00044
00045 protected:
00046
00047     TWeakPtr<STextBlock> IndexTextBlock;
00048 };

```

## 5.32 AccessibilityAddNodeContextMenu.h

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002

```

```

00003 #pragma once
00004
00005 #include "CoreMinimal.h"
00006
00007 #include "PhraseTree/Containers/ContextMenuObject.h"
00008
00009 #include "SGraphActionMenu.h"
00010 #include "GraphActionNode.h"
00011
00012 #include "AccessibilityAddNodeContextMenu.generated.h"
00013
00014 struct FGraphActionNode;
00015
00016 UCLASS()
00017 class OPENACCESSIBILITY_API UAccessibilityAddNodeContextMenu : public UPhraseTreeContextMenuObject
00018 {
00019     GENERATED_BODY()
00020
00021 public:
00022
00023     UAccessibilityAddNodeContextMenu();
00024     UAccessibilityAddNodeContextMenu(TSharedRef<IMenu> Menu);
00025     UAccessibilityAddNodeContextMenu(TSharedRef<IMenu> Menu, TSharedRef<SGraphActionMenu> GraphMenu);
00026     UAccessibilityAddNodeContextMenu(TSharedRef<IMenu> Menu, TSharedRef<SGraphActionMenu> GraphMenu,
    TSharedRef<STreeView<TSharedPtr<FGraphActionNode>> TreeView);
00027
00028     ~UAccessibilityAddNodeContextMenu();
00029
00030     virtual void Init(TSharedRef<IMenu> InMenu, TSharedRef<FPhraseNode> InContextRoot) override;
00031
00032     void Init(TSharedRef<IMenu> InMenu, TSharedRef<SGraphActionMenu> InGraphMenu,
    TSharedRef<STreeView<TSharedPtr<FGraphActionNode>> InTreeView);
00033
00034     // -- UAccessibilityContextMenu Implementation
00035
00036     virtual void Init(TSharedRef<IMenu> InMenu) override;
00037
00038     virtual bool Tick(float DeltaTime) override;
00039
00040     virtual bool Close() override;
00041
00042     virtual void ScaleMenu(const float ScaleFactor = 1.5f) override;
00043
00044     // -- End UAccessibilityContextMenu Implementation
00045
00046     bool DoesItemsRequireRefresh();
00047
00048     void RefreshAccessibilityWidgets();
00049
00050     // Utility Interactions
00051     // Useful for simplifying common interactions.
00052
00053     void GetGraphActionFromIndex(const int32 InIndex, FGraphActionNode* OutGraphAction);
00054
00055     FGraphActionNode* GetGraphActionFromIndex(const int32 InIndex);
00056
00057     TSharedPtr<FGraphActionNode> GetGraphActionFromIndexSP(const int32 InIndex);
00058
00059     void SelectGraphAction(const int32 InIndex);
00060
00061     void PerformGraphAction(const int32 InIndex);
00062
00063     FString GetFilterText();
00064
00065     void SetFilterText(const FString& InFilterText);
00066
00067     void AppendFilterText(const FString& InFilterText);
00068
00069     void ResetFilterText();
00070
00071     void SetScrollDistance(const float InScrollDistance);
00072
00073     void AppendScrollDistance(const float InScrollDistance);
00074
00075     void SetScrollDistanceTop();
00076
00077     void SetScrollDistanceBottom();
00078
00079     void ToggleContextAwareness();
00080
00081 protected:
00082
00083     void ApplyAccessibilityWidget(TSharedRef<STableRow<TSharedPtr<FGraphActionNode>> ItemWidget);
00084
00085     void UpdateAccessibilityWidget(TSharedRef<STableRow<TSharedPtr<FGraphActionNode>> ItemWidget);
00086
00087 public:

```

```

00182
00183     // Menu Components
00184
00188     TWeakPtr<SGraphActionMenu> GraphMenu;
00189
00193     TWeakPtr<STreeView<TSharedPtr<FGraphActionNode>>> TreeView;
00194
00198     TWeakPtr<SEditableTextBox> FilterTextBox;
00199
00203     TWeakPtr<SCheckBox> ContextAwarenessCheckBox;
00204
00205 protected:
00206
00207     FString PrevFilterString;
00208     int32 PrevNumItemsBeingObserved;
00209     int32 PrevNumGeneratedChildren;
00210     double PrevScrollDistance;
00211 };

```

## 5.33 AccessibilityGraphEditorContext.h

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #pragma once
00004
00005 #include "CoreMinimal.h"
00006
00007 #include "PhraseTree/Containers/ContextMenuObject.h"
00008
00009 #include "SGraphActionMenu.h"
00010 #include "GraphActionNode.h"
00011
00012 #include "AccessibilityGraphEditorContext.generated.h"
00013
00014 class SContentIndexer;
00015
00019 UCLASS()
00020 class OPENACCESSIBILITY_API UAccessibilityGraphEditorContext : public UPhraseTreeContextMenuObject
00021 {
00022     GENERATED_BODY()
00023
00024 public:
00025
00026     UAccessibilityGraphEditorContext();
00027
00028     // -- UPhraseTreeContextMenuObject Implementation
00029
00030
00036     virtual void Init(TSharedPtr<IMenu> InMenu, TSharedPtr<FPhraseNode> InContextRoot) override;
00037
00038     virtual bool Tick(float DeltaTime) override;
00039
00044     virtual bool Close() override;
00045
00050     virtual void ScaleMenu(const float ScaleFactor = 1.5f) override;
00051
00052     // -- End of UPhraseTreeContextMenuObject Implementation
00053
00054
00055     // -- Event Actions
00056
00057
00063     TSharedPtr<FGraphActionNode> GetTreeViewAction(const int32& InIndex);
00064
00069     void SelectAction(const int32& InIndex);
00070
00075     FString GetFilterText();
00076
00081     void SetFilterText(const FString& NewString);
00082
00087     void AppendFilterText(const FString& StringToAdd);
00088
00093     void SetScrollDistance(const float NewDistance);
00094
00099     void AppendScrollDistance(const float DistanceToAdd);
00100
00104     void SetScrollDistanceTop();
00105
00109     void SetScrollDistanceBottom();
00110
00111 protected:
00112
00113     // Index Utils

```



```

00114
00119     const int32 GetStaticIndexOffset();
00120
00121     // Component Finders
00122
00128     bool FindGraphActionMenu(const TSharedRef<SWidget>& SearchRoot);
00129
00135     bool FindTreeView(const TSharedRef<SWidget>& SearchRoot);
00136
00142     bool FindStaticComponents(const TSharedRef<SWidget>& SearchRoot);
00143
00144     // Component Tickers
00145
00146     struct FTreeViewTickRequirements
00147     {
00148     public:
00149
00150         FTreeViewTickRequirements()
00151             : PrevSearchText(FString())
00152             , PrevNumItemsBeingObserved(-1)
00153             , PrevNumGeneratedChildren(-1)
00154             , PrevScrollDistance(-1.00)
00155         { }
00156
00157         FString PrevSearchText;
00158         int32 PrevNumItemsBeingObserved;
00159         int32 PrevNumGeneratedChildren;
00160         double PrevScrollDistance;
00161     };
00162
00167     bool TreeViewCanTick();
00168
00173     bool TreeViewRequiresTick();
00174
00178     void TickTreeViewAccessibility();
00179
00180     // Widget Utils
00181
00187     void UpdateAccessibilityWidget(const TSharedRef<SContentIndexer>& ContextIndexer, const int32&
00188     NewIndex);
00189
00195     const TSharedRef<SContentIndexer> CreateAccessibilityWrapper(const TSharedRef<SWidget>&
00196     ContentToWrap, const int32& Index);
00197 protected:
00198
00199     FTreeViewTickRequirements TreeViewTickRequirements;
00200
00201     TWeakPtr<SGraphActionMenu> GraphMenu = TWeakPtr<SGraphActionMenu>();
00202     TWeakPtr<SEditableTextBox> FilterTextBox = TWeakPtr<SEditableTextBox>();
00203
00204     TWeakPtr<STreeView<TSharedPtr<FGraphActionNode>>> TreeView =
00205     TWeakPtr<STreeView<TSharedPtr<FGraphActionNode>>>();
00206
00206     TArray<TWeakPtr<SCheckBox>> CheckBoxes = TArray<TWeakPtr<SCheckBox>>();
00207 };

```

## 5.34 AccessibilityGraphLocomotionContext.h

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #pragma once
00004
00005 #include "CoreMinimal.h"
00006 #include "PhraseTree/Containers/ContextObject.h"
00007
00008 #include "Widgets/Layout/SUniformGridPanel.h"
00009
00010 #include "AccessibilityGraphLocomotionContext.generated.h"
00011
00012 USTRUCT()
00013 struct FGraphLocomotionChunk
00014 {
00015     GENERATED_BODY()
00016
00017 public:
00018
00019     void SetChunkBounds(FVector2D InTopLeft, FVector2D InBottomRight)
00020     {
00021         TopLeft = InTopLeft;
00022         BottomRight = InBottomRight;
00023     }
00024

```

```

00025     void GetChunkBounds(FVector2D& OutTopLeft, FVector2D& OutBottomRight) const
00026     {
00027         OutTopLeft = TopLeft;
00028         OutBottomRight = BottomRight;
00029     }
00030
00031     FVector2D GetChunkTopLeft() const { return TopLeft; }
00032
00033     FVector2D GetChunkBottomRight() const { return BottomRight; }
00034
00035     void SetVisColor(const FLinearColor& NewColor) const
00036     {
00037         if (ChunkVisWidget.IsValid())
00038             ChunkVisWidget.Pin()->SetBorderBackgroundColor(NewColor);
00039     }
00040
00041 public:
00042
00046     FVector2D TopLeft;
00047
00051     FVector2D BottomRight;
00052
00056     TWeakPtr<SBox> ChunkWidget;
00057
00061     TWeakPtr<SBorder> ChunkVisWidget;
00062
00066     TWeakPtr<class SIndexer> ChunkIndexer;
00067
00068 };
00069
00070 struct FPanelViewPosition
00071 {
00072 public:
00073
00074     FPanelViewPosition()
00075         : TopLeft(FVector2D::ZeroVector)
00076         , BotRight(FVector2D::ZeroVector)
00077     { }
00078
00079     FPanelViewPosition(FVector2D InTopLeft, FVector2D InBotRight)
00080         : TopLeft(InTopLeft)
00081         , BotRight(InBotRight)
00082     { }
00083
00084     bool operator!=(const FVector2D& Other)
00085     {
00086         return TopLeft != Other || BotRight != Other;
00087     }
00088
00089     bool operator!=(const FPanelViewPosition& Other)
00090     {
00091         return TopLeft != Other.TopLeft || BotRight != Other.BotRight;
00092     }
00093
00094     FVector2D TopLeft;
00095     FVector2D BotRight;
00096 };
00097
00098 UCLASS()
00099 class OPENACCESSIBILITY_API UAccessibilityGraphLocomotionContext : public UPhraseTreeContextObject
00100 {
00101     GENERATED_BODY()
00102
00103 public:
00104
00105     UAccessibilityGraphLocomotionContext(const FObjectInitializer& ObjectInitializer);
00106
00107     virtual ~UAccessibilityGraphLocomotionContext();
00108
00109     void Init();
00110     void Init(TSharedRef<SGraphEditor> InGraphEditor);
00111
00112     bool SelectChunk(const int32& Index);
00113
00114     bool RevertToPreviousView();
00115
00116     void ConfirmSelection();
00117
00118     void CancelLocomotion();
00119
00120     virtual bool Close() override;
00121
00122 protected:
00123
00124     bool MoveViewport(const FVector2D& InTopLeft, const FVector2D& InBottomRight) const;
00125
00126     bool MoveViewport(const FPanelViewPosition& NewViewPosition) const;

```

```

00127
00128     // Visuals Methods
00129
00130     void ChangeChunkVis(const int32& Index, const FLinearColor& NewColor = FLinearColor::Yellow);
00131
00132     void CreateVisualGrid(const TSharedRef<SGraphEditor> InGraphEditor);
00133
00134     void GenerateVisualChunks(const TSharedRef<SGraphEditor> InGraphEditor, FIntVector2
InVisualChunkSize = FIntVector2(10));
00135
00136     void CalculateVisualChunksBounds();
00137
00138     void RemoveVisualGrid();
00139
00140     void HideNativeVisuals();
00141
00142     void UnHideNativeVisuals();
00143
00144     void OnFocusChanged(const FFocusEvent& FocusEvent, const FWeakWidgetPath& OldFocusedWidgetPath,
const TSharedPtr<SWidget>& OldFocusedWidget, const FWidgetPath& NewFocusedWidgetPath, const
TSharedPtr<SWidget>& NewFocusedWidget);
00145
00146
00147     void BindFocusChangedEvent();
00148
00149     void UnbindFocusChangedEvent();
00150
00151 protected:
00152
00153     FVector2D StartViewPosition; float StartViewZoom;
00154
00155     FPanelViewPosition CurrentViewPosition;
00156     TArray<FPanelViewPosition> PreviousPositions;
00157
00158     // Chunking References
00159
00160     TArray<FGraphLocomotionChunk> ChunkArray;
00161
00162     FIntVector2 ChunkSize;
00163
00164     // Container References
00165
00166     TWeakPtr<SUniformGridPanel> GridContainer;
00167
00168     TWeakPtr<SOverlay> GridParent;
00169
00170     TWeakPtr<SGraphEditor> LinkedEditor;
00171
00172 private:
00173
00174     FTimerHandle SelectionTimerHandle;
00175
00176     TMap<SWidget*, EVisibility> NativeWidgetVisibility;
00177
00178     FDelegateHandle FocusChangedHandle;
00179
00180 };

```

## 5.35 AccessibilityWindowToolbar.h

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #pragma once
00004
00005 #include "CoreMinimal.h"
00006
00007 #include "Indexers/Indexer.h"
00008
00009 #include "AccessibilityWindowToolbar.generated.h"
00010
00014 UCLASS()
00015 class OPENACCESSIBILITY_API UAccessibilityWindowToolbar : public UObject
00016 {
00017     GENERATED_BODY()
00018
00019 public:
00020
00021     UAccessibilityWindowToolbar();
00022
00023     virtual ~UAccessibilityWindowToolbar();
00024
00025     bool Tick(float DeltaTime);
00026

```

```

00027     // -- Parse Events --
00028
00033     void SelectToolBarItem(int32 Index);
00034
00035     // -- End of Parse Events --
00036
00042     bool IsActiveToolBar(const TSharedRef<SWidget>& ToolkitWidget);
00043
00048     TSharedPtr<SWidget> GetActiveToolkitWidget() const;
00049
00050 private:
00051
00058     bool ApplyToolBarIndexing(TSharedRef<SWidget> ToolkitWidget, TSharedRef<SWindow> ToolkitWindow);
00059
00060     // Widget Getters
00061
00067     TSharedPtr<SBorder> GetWindowContentContainer(TSharedRef<SWindow> WindowToFindContainer);
00068
00075     bool GetToolKitToolBar(TSharedRef<SWidget> ToolKitWidget, TSharedPtr<SWidget>& OutToolBar);
00076
00080     void BindTicker();
00081
00085     void UnbindTicker();
00086
00087 public:
00088
00089 private:
00090
00094     TWeakPtr<SWindow> LastTopWindow;
00095
00099     TWeakPtr<SBorder> LastToolkitParent;
00100
00104     TWeakPtr<SWidget> LastToolkit;
00105
00109     FIndexer<int32, SMultiBlockBaseWidget*> ToolbarIndex;
00110
00114     FTSTicker::FDelegateHandle TickDelegateHandle;
00115
00119     TArray<IConsoleCommand*> ConsoleCommands;
00120
00121 };

```

## 5.36 AssetAccessibilityRegistry.h

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #pragma once
00004
00005 #include "CoreMinimal.h"
00006
00007 #include "GraphIndexer.h"
00008
00009 class UBehaviorTree;
00010
00014 class OPENACCESSIBILITY_API FAssetAccessibilityRegistry
00015 {
00016 public:
00017     FAssetAccessibilityRegistry();
00018     ~FAssetAccessibilityRegistry();
00019
00020     // Graph Indexing
00021
00027     bool IsGraphAssetRegistered(const UEdGraph* InGraph) const;
00028
00034     bool RegisterGraphAsset(const UEdGraph* InGraph);
00035
00036     bool RegisterGraphAsset(const UEdGraph* InGraph, const TSharedRef<FGraphIndexer> InGraphIndexer);
00037
00043     bool UnregisterGraphAsset(const UEdGraph* InGraph);
00044
00050     TSharedRef<FGraphIndexer> GetGraphIndexer(const UEdGraph* InGraph) const {
00051         if (GraphAssetIndex.Contains(InGraph->GraphGuid))
00052             return GraphAssetIndex[InGraph->GraphGuid].ToSharedRef();
00053
00054         return TSharedRef<FGraphIndexer>();
00055     }
00056
00061     void GetAllGraphKeyIndexes(TArray<FGuid>& OutGraphKeys) const;
00062
00067     TArray<FGuid> GetAllGraphKeyIndexes() const;
00068
00073     void GetAllGraphIndexes(TArray<TSharedPtr<FGraphIndexer>& OutGraphIndexes) const;
00074

```

```

00079     TArray<TSharedPtr<FGraphIndexer>> GetAllGraphIndexes();
00080
00081     // Game World Indexing
00082
00088     bool IsGameWorldAssetRegistered(const UWorld* InWorld) const;
00089
00095     bool RegisterGameWorldAsset(const UWorld* InWorld);
00096
00102     bool UnregisterGameWorldAsset(const UWorld* InWorld);
00103
00104 private:
00105
00106     // Asset Register Events
00107
00113     void OnAssetOpenedInEditor(UObject* OpenedAsset, IAssetEditorInstance* EditorInstance);
00114
00120     void OnAssetEditorRequestClose(UObject* ClosingAsset, EAssetEditorCloseReason CloseReason);
00121
00125     void EmptyGraphAssetIndex();
00126
00130     void EmptyGameWorldAssetIndex();
00131
00132     // Asset Editor Registers
00133
00138     void RegisterBlueprintAsset(const UBlueprint* InBlueprint);
00139
00144     void RegisterMaterialAsset(const UMaterial* InMaterial);
00145
00149     void RegisterBehaviorTreeAsset(const UBehaviorTree* InBehaviorTree);
00150
00155     void RegisterUWorldAsset(const UWorld* InWorld);
00156
00157 public:
00158
00162     TMap<FGuid, TSharedPtr<FGraphIndexer>> GraphAssetIndex;
00163
00167     //TMap<UWorld, FWorldIndexer*> GameWorldAssetIndex;
00168
00169 private:
00170
00171     FDelegateHandle AssetOpenedInEditorHandle;
00172     FDelegateHandle AssetEditorRequestCloseHandle;
00173 };

```

## 5.37 GraphIndexer.h

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #pragma once
00004
00005 #include "CoreMinimal.h"
00006
00007 class UEdGraph;
00008 class UEdGraphNode;
00009 struct FEdGraphEditAction;
00010
00014 class OPENACCESSIBILITY_API FGraphIndexer
00015 {
00016 public:
00017
00018     FGraphIndexer();
00019     FGraphIndexer(const UEdGraph* GraphToIndex);
00020     ~FGraphIndexer();
00021
00027     bool ContainsKey(const int& InKey);
00028
00034     int ContainsNode(UEdGraphNode* InNode);
00035
00041     void ContainsNode(UEdGraphNode* InNode, int& OutIndex);
00042
00048     int GetKey(const UEdGraphNode* InNode);
00049
00056     bool GetKey(const UEdGraphNode* InNode, int& OutKey);
00057
00063     void GetNode(const int& InIndex, UEdGraphNode* OutNode);
00064
00070     UEdGraphNode* GetNode(const int& InIndex);
00071
00078     void GetPin(const int& InNodeIndex, const int& InPinIndex, UEdGraphPin* OutPin);
00079
00086     UEdGraphPin* GetPin(const int& InNodeIndex, const int& InPinIndex);
00087
00093     int AddNode(const UEdGraphNode* Node);

```

```

00094
00100     void AddNode(int& OutIndex, const UEdGraphNode& InNode);
00101
00107     int GetOrAddNode(const UEdGraphNode* InNode);
00108
00114     void GetOrAddNode(const UEdGraphNode* InNode, int& OutIndex);
00115
00120     void RemoveNode(const int& InIndex);
00121
00126     void RemoveNode(const UEdGraphNode* InNode);
00127
00132     void OnGraphEvent(const FEdGraphEditAction& InAction);
00133
00137     void OnGraphRebuild();
00138
00139 private:
00140
00145     int GetAvailableIndex();
00146
00151     void GetAvailableIndex(int& OutIndex);
00152
00156     void BuildGraphIndex();
00157
00158 protected:
00159
00163     TMap<int, UEdGraphNode*> IndexMap;
00164
00168     TSet<int32> NodeSet;
00169
00173     TQueue<int32> AvailableIndices;
00174
00178     UEdGraph* LinkedGraph;
00179
00180     FDelegateHandle OnGraphChangedHandle;
00181 };

```

## 5.38 Indexer.h

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #pragma once
00004
00005 #include "CoreMinimal.h"
00006
00007 #include "OpenAccessibilityLogging.h"
00008
00014 template <typename KeyType, typename ValueType>
00015 class FIndexer
00016 {
00017 public:
00018
00019     FIndexer()
00020     {
00021
00022     }
00023
00024     virtual ~FIndexer()
00025     {
00026
00027     }
00028
00029
00034     bool IsEmpty() const
00035     {
00036         return IndexMap.IsEmpty();
00037     }
00038
00042     void Reset()
00043     {
00044         IndexMap.Reset();
00045         AvailableIndexes.Empty();
00046     }
00047
00051     void Empty()
00052     {
00053         IndexMap.Empty();
00054         AvailableIndexes.Empty();
00055     }
00056
00061     int32 Num() const
00062     {
00063         return IndexMap.Num();
00064     }

```

```

00065
00070 void Num(int32& OutNum) const
00071 {
00072     OutNum = IndexMap.Num();
00073 }
00074
00080 bool ContainsKey(const KeyType& InKey)
00081 {
00082     return IndexMap.Contains(InKey);
00083 }
00084
00090 bool ContainsValue(const ValueType& InValue)
00091 {
00092     check(InValue != nullptr);
00093
00094     const KeyType* FoundKey = IndexMap.FindKey(InValue);
00095
00096     return FoundKey != nullptr;
00097 }
00098
00104 const KeyType GetKey(const ValueType& InValue)
00105 {
00106     check(InValue != nullptr);
00107
00108     return *IndexMap.FindKey(InValue);
00109 }
00110
00117 bool GetKey(const ValueType& InValue, KeyType& OutKey)
00118 {
00119     check(InValue != nullptr);
00120
00121     const KeyType* FoundKey = IndexMap.FindKey(InValue);
00122
00123     if (FoundKey != nullptr)
00124     {
00125         OutKey = *FoundKey;
00126
00127         return true;
00128     }
00129     else return false;
00130 }
00131
00137 ValueType GetValue(const KeyType& InKey)
00138 {
00139     return *IndexMap.Find(InKey);
00140 }
00141
00148 bool GetValue(const KeyType& InKey, ValueType& OutValue)
00149 {
00150     if (!IndexMap.Contains(InKey))
00151     {
00152         UE_LOG(LogOpenAccessibility, Warning, TEXT("Provided Key is not recognised.));
00153         return false;
00154     }
00155
00156     OutValue = *IndexMap.Find(InKey);
00157
00158     return true;
00159 }
00160
00166 KeyType AddValue(const ValueType& InValue)
00167 {
00168     check(InValue != nullptr);
00169
00170     if (ContainsValue(InValue))
00171     {
00172         return GetKey(InValue);
00173     }
00174
00175     KeyType NewKey;
00176     GetAvailableKey(NewKey);
00177
00178     IndexMap.Add(NewKey, InValue);
00179
00180     return NewKey;
00181 }
00182
00188 void AddValue(const ValueType& InValue, KeyType& OutKey)
00189 {
00190     check(InValue != nullptr);
00191
00192     if (ContainsValue(InValue))
00193     {
00194         OutKey = GetKey(InValue);
00195         return;
00196     }
00197

```

```

00198         OutKey = GetAvailableKey();
00199
00200         IndexMap.Add(OutKey, InValue);
00201     }
00202
00208     KeyType GetKeyOrAddValue(const ValueType& InValue)
00209     {
00210         check(InValue != nullptr);
00211
00212         KeyType FoundKey;
00213         if (GetKey(InValue, FoundKey))
00214             return FoundKey;
00215
00216         return AddValue(InValue);
00217     }
00218
00224     void GetKeyOrAddValue(const ValueType& InValue, KeyType& OutKey)
00225     {
00226         check(InValue != nullptr);
00227
00228         if (GetKey(InValue, OutKey))
00229             return;
00230
00231         OutKey = AddValue(InValue);
00232     }
00233
00238     void RemoveValue(const KeyType& InKey)
00239     {
00240         if (!IndexMap.Contains(InKey))
00241         {
00242             UE_LOG(LogOpenAccessibility, Warning, TEXT("Provided Key Has No Pair in Index.));
00243             return;
00244         }
00245
00246         IndexMap.Remove(InKey);
00247         AvailableIndexes.Enqueue(InKey);
00248     }
00249
00254     void RemoveValue(const ValueType& InValue)
00255     {
00256         check(InValue != nullptr);
00257
00258         KeyType FoundKey;
00259         if (GetKey(InValue, FoundKey))
00260         {
00261             IndexMap.Remove(FoundKey);
00262             AvailableIndexes.Enqueue(FoundKey);
00263         }
00264         else UE_LOG(LogOpenAccessibility, Log, TEXT("Provided Value Had No Associated Key.));
00265     }
00266
00267 protected:
00268
00273     void GetAvailableKey(KeyType& OutKey)
00274     {
00275         if (!AvailableIndexes.IsEmpty() && AvailableIndexes.Dequeue(OutKey))
00276             return;
00277
00278         OutKey = IndexMap.Num();
00279     }
00280
00285     KeyType GetAvailableKey()
00286     {
00287         if (!AvailableIndexes.IsEmpty())
00288         {
00289             KeyType OutKey;
00290             if (AvailableIndexes.Dequeue(OutKey))
00291                 return OutKey;
00292         }
00293
00294         return IndexMap.Num();
00295     }
00296
00297 public:
00298
00299
00300 protected:
00301
00305     TMap<KeyType, ValueType> IndexMap;
00306
00310     TQueue<KeyType> AvailableIndexes;
00311 };

```



## 5.39 OAccessibilityNodeFactory.h

```

00001 // Fill out your copyright notice in the Description page of Project Settings.
00002
00003 #pragma once
00004
00005 #include "CoreMinimal.h"
00006 #include "EdGraphUtilities.h"
00007
00011 class OPENACCESSIBILITY_API FAccessibilityNodeFactory : public FGraphPanelNodeFactory
00012 {
00013
00014 public:
00015     /* Begin FGraphPanelNodeFactory */
00016     virtual TSharedPtr<class SGraphNode> CreateNode(UEdGraphNode* Node) const override;
00017     /* End FGraphPanelNodeFactory */
00018
00019 public:
00020     FAccessibilityNodeFactory();
00021     virtual ~FAccessibilityNodeFactory();
00022
00029     inline void WrapNodeWidget(UEdGraphNode* Node, TSharedRef<SGraphNode> NodeWidget, int NodeIndex)
00030     const;
00038     inline void WrapPinWidget(UEdGraphPin* Pin, TSharedRef<SGraphPin> PinWidget, int PinIndex,
00039                               SGraphNode* OwnerNode) const;
00040
00040     void SetSharedPtr(TSharedPtr<FAccessibilityNodeFactory> InSharedPtr)
00041     {
00042         ThisPtr = InSharedPtr;
00043     }
00044
00045 private:
00046
00047     TSharedPtr<FAccessibilityNodeFactory> ThisPtr;
00048 };

```

## 5.40 OEditorAccessibilityManager.h

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #pragma once
00004
00005 #include "CoreMinimal.h"
00006
00010 class OPENACCESSIBILITY_API OEditorAccessibilityManager
00011 {
00012 public:
00013     OEditorAccessibilityManager();
00014     ~OEditorAccessibilityManager();
00015 };

```

## 5.41 OpenAccessibility.h

```

00001 // Copyright Epic Games, Inc. All Rights Reserved.
00002
00003 #pragma once
00004
00005 #include "CoreMinimal.h"
00006 #include "Modules/ModuleManager.h"
00007
00008 #include "AssetAccessibilityRegistry.h"
00009 #include "OAccessibilityNodeFactory.h"
00010
00011 class FOpenAccessibilityModule : public IModuleInterface
00012 {
00013
00014 public:
00015
00017     virtual void StartupModule() override;
00018     virtual void ShutdownModule() override;
00021     static FOpenAccessibilityModule& Get()
00022     {
00023         return FModuleManager::GetModuleChecked<FOpenAccessibilityModule>("OpenAccessibility");
00024     }
00025
00026     virtual bool SupportsDynamicReloading() override
00027     {
00028         return false;
00029     }

```

```

00029     }
00030
00031 private:
00032
00033     // Phrase Branch Bindings
00034
00038     void BindLocalizedInteractionBranch();
00039
00043     void BindGraphInteractionBranch();
00044
00048     void BindViewportInteractionBranch();
00049
00050     // Transcription Visualization
00051
00055     void CreateTranscriptionVisualization();
00056
00060     void DestroyTranscriptionVisualization();
00061
00062     // Console Commands
00063
00067     void RegisterConsoleCommands();
00068
00072     void UnregisterConsoleCommands();
00073
00074 public:
00075
00076     // Accessibility Components
00077
00081     TSharedPtr<class FAccessibilityNodeFactory> AccessibilityNodeFactory;
00082
00086     TSharedPtr<class FAssetAccessibilityRegistry> AssetAccessibilityRegistry;
00087
00088 private:
00089
00090     TSharedPtr<class FTranscriptionVisualizer> TranscriptionVisualizer;
00091
00092     TArray<IConsoleCommand*> ConsoleCommands;
00093 };

```

## 5.42 OpenAccessibilityLogging.h

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #pragma once
00004
00005 OPENACCESSIBILITY_API DECLARE_LOG_CATEGORY_EXTERN(LogOpenAccessibility, Log, All);
00006
00007 DEFINE_LOG_CATEGORY(LogOpenAccessibility);

```

## 5.43 LocalizedInputLibrary.h

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #pragma once
00004
00005 #include "CoreMinimal.h"
00006
00007 #include "PhraseTree/PhraseTreeFunctionLibrary.h"
00008
00009 #include "LocalizedInputLibrary.generated.h"
00010
00011 UCLASS()
00012 class ULocalizedInputLibrary : public UPhraseTreeFunctionLibrary
00013 {
00014     GENERATED_BODY()
00015
00016 public:
00017
00018     ULocalizedInputLibrary(const FObjectInitializer& ObjectInitializer);
00019
00020     virtual ~ULocalizedInputLibrary();
00021
00022     // UPhraseTreeFunctionLibrary Implementation
00023
00028     virtual void BindBranches(TSharedPtr<FPhraseTree> PhraseTree) override;
00029
00030     // End of UPhraseTreeFunctionLibrary Implementation
00031
00032

```

```

00033 // Keyboard Input Implementation
00034
00039 UFUNCTION()
00040 void KeyboardInputAdd(FParseRecord& Record);
00041
00046 UFUNCTION()
00047 void KeyboardInputRemove(FParseRecord& Record);
00048
00053 UFUNCTION()
00054 void KeyboardInputReset(FParseRecord& Record);
00055
00060 UFUNCTION()
00061 void KeyboardInputConfirm(FParseRecord& Record);
00062
00067 UFUNCTION()
00068 void KeyboardInputExit(FParseRecord& Record);
00069
00070 // End of Keyboard Input Implementation
00071
00072
00073 // Mouse Input Implementation
00074
00075
00076
00077 // End of Keyboard Input Implementation
00078 };

```

## 5.44 NodeInteractionLibrary.h

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #pragma once
00004
00005 #include "CoreMinimal.h"
00006
00007 #include "PhraseTree/PhraseTreeFunctionLibrary.h"
00008
00009 #include "NodeInteractionLibrary.generated.h"
00010
00011 UCLASS()
00012 class UNodeInteractionLibrary : public UPhraseTreeFunctionLibrary
00013 {
00014     GENERATED_BODY()
00015
00016 public:
00017
00018     UNodeInteractionLibrary(const FObjectInitializer& ObjectInitializer);
00019
00020     virtual ~UNodeInteractionLibrary();
00021
00022     // UPhraseTreeFunctionLibrary Implementation
00023
00028     virtual void BindBranches(TSharedRef<FPhraseTree> PhraseTree) override;
00029
00030     // End of UPhraseTreeFunctionLibrary Implementation
00031
00032
00033     // Node Implementation
00034
00039     UFUNCTION()
00040     void MoveNode(FParseRecord& Record);
00041
00046     UFUNCTION()
00047     void DeleteNode(FParseRecord& Record);
00048
00053     UFUNCTION()
00054     void NodeIndexFocus(int32 Index);
00055
00056     // End of Node Implementation
00057
00058
00059     // Pin Implementation
00060
00065     UFUNCTION()
00066     void PinConnect(FParseRecord& Record);
00067
00072     UFUNCTION()
00073     void PinDisconnect(FParseRecord& Record);
00074
00075     // End of Pin Implementation
00076
00077
00078     // Node Add Implementation

```

```

00079
00085     TSharedPtr<IMenu> NodeAddMenu (FParseRecord& Record);
00086
00092     TSharedPtr<IMenu> NodeAddPinMenu (FParseRecord& Record);
00093
00094
00099     void NodeAddSelect (FParseRecord& Record);
00100
00105     void NodeAddSearchAdd (FParseRecord& Record);
00106
00111     void NodeAddSearchRemove (FParseRecord& Record);
00112
00117     void NodeAddSearchReset (FParseRecord& Record);
00118
00123     void NodeAddScroll (FParseRecord& Record);
00124
00125     // End of Node Add Implementation
00126
00127
00128     // Selection Implementation
00129
00134     UFUNCTION()
00135     void SelectionNodeToggle (FParseRecord& Record);
00136
00141     UFUNCTION()
00142     void SelectionReset (FParseRecord &Record);
00143
00148     UFUNCTION()
00149     void SelectionMove (FParseRecord &Record);
00150
00155     UFUNCTION()
00156     void SelectionAlignment (FParseRecord &Record);
00157
00162     UFUNCTION()
00163     void SelectionStraighten (FParseRecord &Record);
00164
00169     UFUNCTION()
00170     void SelectionComment (FParseRecord &Record);
00171
00172     // End of Selection Implementation
00173
00174
00175     // Locomotion Implementation
00176
00181     UFUNCTION()
00182     void LocomotionSelect (FParseRecord& Record);
00183
00188     UFUNCTION()
00189     void LocomotionRevert (FParseRecord& Record);
00190
00195     UFUNCTION()
00196     void LocomotionConfirm (FParseRecord& Record);
00197
00202     UFUNCTION()
00203     void LocomotionCancel (FParseRecord& Record);
00204
00205     // End of Locomotion Implementations
00206
00207
00208     // Blueprint Specifics
00209
00214     UFUNCTION()
00215     void BlueprintCompile (FParseRecord& Record);
00216
00217     // End of Blueprint Specifics
00218 };

```

## 5.45 ViewInteractionLibrary.h

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #pragma once
00004
00005 #include "CoreMinimal.h"
00006
00007 #include "PhraseTree/PhraseTreeFunctionLibrary.h"
00008
00009 #include "ViewInteractionLibrary.generated.h"
00010
00011 UCLASS()
00012 class UViewInteractionLibrary : public UPhraseTreeFunctionLibrary
00013 {
00014     GENERATED_BODY()

```

```

00015
00016 public:
00017
00018     UViewInteractionLibrary(const FObjectInitializer& ObjectInitializer);
00019
00020     virtual ~UViewInteractionLibrary();
00021
00022     // UPhraseTreeFunctionLibrary Implementation
00023
00028     void BindBranches(TSharedRef<FPhraseTree> PhraseTree) override;
00029
00030     // End of UPhraseTreeFunctionLibrary Implementation
00031
00032
00037     void MoveViewport(FParseRecord& Record);
00038
00043     void ZoomViewport(FParseRecord& Record);
00044
00049     void IndexFocus(FParseRecord& Record);
00050 };

```

## 5.46 WindowInteractionLibrary.h

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #pragma once
00004
00005 #include "CoreMinimal.h"
00006
00007 #include "PhraseTree/PhraseTreeFunctionLibrary.h"
00008
00009 #include "WindowInteractionLibrary.generated.h"
00010
00011 UCLASS()
00012 class UWindowInteractionLibrary : public UPhraseTreeFunctionLibrary
00013 {
00014     GENERATED_BODY()
00015
00016 public:
00017
00018     UWindowInteractionLibrary(const FObjectInitializer& ObjectInitializer);
00019
00020     virtual ~UWindowInteractionLibrary();
00021
00022     // UPhraseTreeFunctionLibrary Implementation
00023
00028     void BindBranches(TSharedRef<FPhraseTree> PhraseTree) override;
00029
00030     // End of UPhraseTreeFunctionLibrary Implementation
00031
00032
00033     // Window Interaction
00034
00039     void SwitchNextActiveWindow(FParseRecord& Record);
00040
00045     void SwitchPrevActiveWindow(FParseRecord& Record);
00046
00051     void CloseActiveWindow(FParseRecord& Record);
00052
00053     // End Window Interaction
00054
00055
00056     // Window ToolBar Interaction
00057
00062     void SelectToolBarItem(FParseRecord& Record);
00063
00064     // End Window ToolBar Interaction
00065
00066
00067     // Window Tab Interaction
00068
00073     void SwitchNextTabInStack(FParseRecord& Record);
00074
00079     void SwitchPrevTabInStack(FParseRecord& Record);
00080
00081     // End of Window Tab Interaction
00082
00083 protected:
00084
00085     UPROPERTY(BlueprintReadOnly)
00086     class UAccessibilityWindowToolBar* WindowToolBar;
00087
00088 };

```

## 5.47 TranscriptionVisualizer.h

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #pragma once
00004
00005 #include "CoreMinimal.h"
00006
00007 class OPENACCESSIBILITY_API FTranscriptionVisualizer
00008 {
00009 public:
00010     FTranscriptionVisualizer();
00011     ~FTranscriptionVisualizer();
00012
00013     virtual bool Tick(float DeltaTime);
00014
00015     // Visualizer Methods
00016
00017     void ConstructVisualizer();
00018
00019     void UpdateVisualizer();
00020
00021     void ReparentWindow();
00022
00023     void MoveVisualizer();
00024
00025     void OnTranscriptionRecieved(TArray<FString> InTranscription);
00026
00027 protected:
00028
00029     bool GetTopScreenVisualizerPosition(FVector2D& OutPosition);
00030
00031     bool GetDisplayVisualizerPosition(FVector2D& OutPosition);
00032
00033     // Ticker Manager Methods
00034
00035     void RegisterTicker();
00036
00037     void UnregisterTicker();
00038
00039 protected:
00040
00041     // Ticker Vars
00042
00043     FTSTicker::FDelegateHandle TickDelegateHandle;
00044
00045     // Vis Components
00046
00047     TWeakPtr<SWindow> VisWindow;
00048
00049     TWeakPtr<class SAccessibilityTranscriptionVis> VisContent;
00050 };

```

## 5.48 OpenAccessibilityAnalytics.Build.cs

```

00001 // Copyright Epic Games, Inc. All Rights Reserved.
00002
00003 using System.IO;
00004 using UnrealBuildTool;
00005
00006 public class OpenAccessibilityAnalytics : ModuleRules
00007 {
00008     public OpenAccessibilityAnalytics(ReadOnlyTargetRules Target) : base(Target)
00009     {
00010         PCHUsage = ModuleRules.PCHUsageMode.UseExplicitOrSharedPCHs;
00011
00012         PublicIncludePaths.AddRange(
00013             new string[] {
00014                 // ... add public include paths required here ...
00015             }
00016         );
00017
00018         PrivateIncludePaths.AddRange(
00019             new string[] {
00020                 // ... add other private include paths required here ...
00021             }
00022         );
00023
00024         PublicDependencyModuleNames.AddRange(
00025             new string[]
00026             {

```

```

00027         {
00028             "Core",
00029             // ... add other public dependencies that you statically link with here ...
00030         }
00031     };
00032
00033     PrivateDependencyModuleNames.AddRange(
00034         new string[]
00035         {
00036             "Engine",
00037         }
00038     );
00039
00040
00041     DynamicallyLoadedModuleNames.AddRange(
00042         new string[]
00043         {
00044             // ... add any modules that your module loads dynamically here ...
00045         }
00046     );
00047
00048     CircularlyReferencedDependentModules.AddRange(
00049         new string[]
00050         {
00051             // ...
00052         }
00053     );
00054 }
00055 }
00056 }

```

## 5.49 OpenAccessibilityAnalytics.cpp

```

00001 #include "OpenAccessibilityAnalytics.h"
00002 #include "OpenAccessibilityAnalyticsLogging.h"
00003
00004 #include "HAL/PlatformFileManager.h"
00005 #include "Misc/DateTime.h"
00006
00007 #define LOCTEXT_NAMESPACE "FOpenAccessibilityAnalyticsModule"
00008
00009 void FOpenAccessibilityAnalyticsModule::StartupModule()
00010 {
00011     SessionBufferFile = GenerateFileForSessionLog();
00012     EnableDumpTick();
00013     AddConsoleCommands();
00014 }
00015
00016 void FOpenAccessibilityAnalyticsModule::ShutdownModule()
00017 {
00018     DisableDumpTick();
00019     RemoveConsoleCommands();
00020 }
00021
00022 bool FOpenAccessibilityAnalyticsModule::DumpTick(float DeltaTime)
00023 {
00024     if (EventBuffer.IsEmpty())
00025         return true;
00026
00027     if (SessionBufferFile.IsEmpty())
00028         SessionBufferFile = GenerateFileForSessionLog();
00029
00030     UE_LOG(LogOpenAccessibilityAnalytics, Log, TEXT("Dumping Event Log To File.));
00031
00032     if (!WriteBufferToFile())
00033     {
00034         UE_LOG(LogOpenAccessibilityAnalytics, Warning, TEXT("EventLog Dumping Failed.));
00035     }
00036
00037     return true;
00038 }
00039
00040 FString FOpenAccessibilityAnalyticsModule::GenerateFileForSessionLog()
00041 {
00042     FDateTime CurrentDateTime = FDateTime::Now();
00043
00044     FString CombinedFileName = TEXT("[") + CurrentDateTime.ToString() + TEXT("] OA Event Log.log");
00045     return FPaths::ConvertRelativePathToFull(FPaths::ProjectSavedDir() +
00046         TEXT("OpenAccessibility/Logs/") + CombinedFileName);
00047 }
00048
00049 bool FOpenAccessibilityAnalyticsModule::WriteBufferToFile()

```

```

00050 {
00051     if (EventBuffer.IsEmpty())
00052         return false;
00053
00054     FString CombinedString = FString("");
00055     LoggedEvent CurrEvent;
00056     while (!EventBuffer.IsEmpty())
00057     {
00058         CurrEvent = EventBuffer[0];
00059         EventBuffer.RemoveAt(0);
00060
00061         CombinedString += FString::Printf(TEXT("| %s | - %s\r\n"), *CurrEvent.Title,
*CurrEvent.Body);
00062     }
00063
00064     if (FFileHelper::SaveStringToFile(
00065         CombinedString,
00066         *SessionBufferFile,
00067         FFileHelper::EEncodingOptions::AutoDetect,
00068         &IFileManager::Get(),
00069         EFileWrite::FILEWRITE_Append
00070     ))
00071     {
00072
00073         return true;
00074     }
00075
00076     return false;
00077 }
00078
00079 void FOpenAccessibilityAnalyticsModule::EnableDumpTick()
00080 {
00081     const double DumpDelayCheck = 20.0f;
00082
00083     FTickerDelegate TickDelegate = FTickerDelegate::CreateRaw(this,
&FOpenAccessibilityAnalyticsModule::DumpTick);
00084     DumpTickHandle = FTSTicker::GetCoreTicker().AddTicker(TickDelegate, DumpDelayCheck);
00085 }
00086
00087 void FOpenAccessibilityAnalyticsModule::DisableDumpTick()
00088 {
00089     if (DumpTickHandle.IsValid())
00090         FTSTicker::GetCoreTicker().RemoveTicker(DumpTickHandle);
00091 }
00092
00093 void FOpenAccessibilityAnalyticsModule::AddConsoleCommands()
00094 {
00095     ConsoleCommands.Add(IConsoleManager::Get().RegisterConsoleCommand(
00096         TEXT("OpenAccessibilityAnalytics.Debug.Add_Mock_Event"),
00097         TEXT("Adds a MOCK Event to the Eventbuffer"),
00098
00099         FConsoleCommandWithArgsDelegate::CreateLambda(
00100             [this](const TArray<FString>& Args) {
00101
00102                 if (Args.Num() < 2)
00103                     return;
00104
00105                 FString EventTitle = Args[0];
00106                 FString EventBody;
00107
00108                 for (int i = 1; i < Args.Num(); i++)
00109                 {
00110                     EventBody += Args[i] + TEXT(" ");
00111                 }
00112
00113                 this->LogEvent(*EventTitle, *EventBody);
00114             }
00115         )
00116     ));
00117
00118     ConsoleCommands.Add(IConsoleManager::Get().RegisterConsoleCommand(
00119         TEXT("OpenAccessibilityAnalytics.Debug.ForceLogDump"),
00120         TEXT("Forces a Dump of the Active To Log File."),
00121
00122         FConsoleCommandDelegate::CreateLambda(
00123             [this]() {
00124                 this->DumpTick(0.0f);
00125             }
00126         )
00127     ));
00128 }
00129
00130 void FOpenAccessibilityAnalyticsModule::RemoveConsoleCommands()
00131 {
00132     IConsoleCommand* ConsoleCommand = nullptr;
00133     while (ConsoleCommands.Num() > 0)
00134     {

```



```

00135         ConsoleCommand = ConsoleCommands.Pop();
00136
00137         IConsoleManager::Get().UnregisterConsoleObject(ConsoleCommand);
00138
00139         delete ConsoleCommand;
00140         ConsoleCommand = nullptr;
00141     }
00142 }
00143
00144 #undef LOCTEXT_NAMESPACE
00145
00146 IMPLEMENT_MODULE(FOpenAccessibilityAnalyticsModule, OpenAccessibilityAnalytics)

```

## 5.50 OpenAccessibilityAnalyticsLogging.h

```

00001 // Copyright Epic Games, Inc. All Rights Reserved.
00002
00003 #pragma once
00004
00005 DECLARE_LOG_CATEGORY_EXTERN(LogOpenAccessibilityAnalytics, Log, All);
00006
00007 DEFINE_LOG_CATEGORY(LogOpenAccessibilityAnalytics);

```

## 5.51 OpenAccessibilityAnalytics.h

```

00001 // Copyright Epic Games, Inc. All Rights Reserved.
00002
00003 #pragma once
00004
00005 #include "CoreMinimal.h"
00006 #include "Modules/ModuleManager.h"
00007
00008 #define OA_LOG(CategoryName, Verbosity, EventTitle, Format, ...) \
00009 { \
00010     UE_VALIDATE_FORMAT_STRING(Format, ##__VA_ARGS__); \
00011     UE_LOG(CategoryName, Verbosity, Format, ##__VA_ARGS__) \
00012     FOpenAccessibilityAnalyticsModule::Get().LogEvent(EventTitle, Format, ##__VA_ARGS__); \
00013 }
00014
00015 class FOpenAccessibilityAnalyticsModule : public IModuleInterface {
00016 public:
00017     virtual void StartupModule() override;
00018     virtual void ShutdownModule() override;
00019
00020     virtual bool SupportsDynamicReloading() override { return false; }
00021
00022     static FOpenAccessibilityAnalyticsModule& Get()
00023     {
00024         return
00025         FModuleManager::GetModuleChecked<FOpenAccessibilityAnalyticsModule>("OpenAccessibilityAnalytics");
00026     }
00027
00028     bool DumpTick(float DeltaTime);
00029
00030     // Analytics Logging
00031
00032     void LogEvent(const TCHAR* EventTitle, const TCHAR* LogString, ...);
00033
00034 private:
00035     FString GenerateFileForSessionLog();
00036
00037     bool WriteBufferToFile();
00038
00039     void EnableDumpTick();
00040
00041     void DisableDumpTick();
00042
00043     void AddConsoleCommands();
00044
00045     void RemoveConsoleCommands();
00046
00047 private:
00048     // Analytics Dumping
00049
00050     FString SessionBufferFile;

```

```

00092
00093     struct LoggedEvent
00094     {
00095     public:
00096
00097         LoggedEvent()
00098         { };
00099
00100         LoggedEvent(const TCHAR* EventTitle, const TCHAR* EventString, FDateTime EventTimestamp =
FDateTime::Now())
00101             : Title(EventTitle)
00102             , Body(EventString)
00103             , Timestamp(EventTimestamp)
00104             { };
00105
00106         LoggedEvent(const FString& EventTitle, const FString& EventString, FDateTime EventTimestamp =
FDateTime::Now())
00107             : Title(EventTitle)
00108             , Body(EventString)
00109             , Timestamp(EventTimestamp)
00110             { };
00111
00112     public:
00113         FString Title;
00114         FString Body;
00115
00116         FDateTime Timestamp;
00117     };
00118
00122     TArray<LoggedEvent> EventBuffer;
00123
00124     FTSTicker::FDelegateHandle DumpTickHandle;
00125
00126     // Console Commands
00127
00131     TArray<IConsoleCommand*> ConsoleCommands;
00132 };
00133
00134
00135 FORCEINLINE void FOpenAccessibilityAnalyticsModule::LogEvent(const TCHAR* EventTitle, const TCHAR*
LogString, ...)
00136 {
00137     va_list Args;
00138
00139     va_start(Args, LogString);
00140     TStringBuilder<1024> Message;
00141     Message.AppendV(LogString, Args);
00142     va_end(Args);
00143
00144     EventBuffer.Add(
00145         LoggedEvent(EventTitle, *Message)
00146     );
00147 }

```

## 5.52 OpenAccessibilityCommunication.Build.cs

```

00001 // Copyright Epic Games, Inc. All Rights Reserved.
00002
00003 using System.IO;
00004 using UnrealBuildTool;
00005 using UnrealBuildTool.Rules;
00006
00007 public class OpenAccessibilityCommunication : ModuleRules
00008 {
00009     public OpenAccessibilityCommunication(ReadOnlyTargetRules Target) : base(Target)
00010     {
00011         PCHUsage = ModuleRules.PCHUsageMode.UseExplicitOrSharedPCHs;
00012
00013         PublicIncludePaths.AddRange(
00014             new string[] {
00015                 // ... add public include paths required here ...
00016             }
00017         );
00018
00019         PrivateIncludePaths.AddRange(
00020             new string[] {
00021                 // ... add other private include paths required here ...
00022             }
00023         );
00024
00025
00026         PublicDependencyModuleNames.AddRange(
00027             new string[]

```

```

00028         {
00029             "Core",
00030             // ... add other public dependencies that you statically link with here ...
00031         }
00032     };
00033
00034     PrivateDependencyModuleNames.AddRange(
00035         new string[]
00036         {
00037             // Internal Plugin Dependencies
00038             "OpenAccessibilityAnalytics",
00039
00040             // Internal ThirdParty Dependencies
00041             "ZeroMQ",
00042
00043             // Core Modules
00044             "CoreUObject",
00045             "Engine",
00046             "Json",
00047
00048             // Editor Modules
00049             "UnrealEd",
00050             "Projects",
00051
00052             // Slate UI Modules
00053             "Slate",
00054             "SlateCore",
00055
00056             // Audio Modules
00057             "AudioMixer",
00058             "AudioCaptureCore",
00059             "AudioCapture",
00060             "InputCore",
00061         }
00062     );
00063
00064     DynamicallyLoadedModuleNames.AddRange(
00065         new string[]
00066         {
00067             // ... add any modules that your module loads dynamically here ...
00068         }
00069     );
00070
00071     CircularlyReferencedDependentModules.AddRange(
00072         new string[]
00073         {
00074             // ... add any modules that your module depends on here ...
00075         }
00076     );
00077 }
00078 }
00079 }

```

## 5.53 AudioManager.cpp

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #include "AudioManager.h"
00004 #include "OpenAccessibilityCommunication.h"
00005 #include "OpenAccessibilityComLogging.h"
00006 #include "SocketCommunicationServer.h"
00007
00008 #include "AudioCaptureCore.h"
00009 #include "AudioDeviceNotificationSubsystem.h"
00010 #include "Templates/Function.h"
00011
00012 UAudioManager::UAudioManager()
00013 {
00014     Settings = FAudioManagerSettings();
00015
00016     // Create Audio Capture Object and Initialize Audio Stream
00017     bIsCapturingAudio = false;
00018     AudioCapture = NewObject<UAudioCapture>();
00019     AudioCapture->OpenDefaultAudioStream();
00020     AudioCapture->StartCapturingAudio();
00021
00022     RegisterAudioGenerator();
00023
00024     // Create FileIO Objects
00025     FileWriter = new Audio::FSoundWavePCMWriter();
00026 }
00027
00028 UAudioManager::~UAudioManager()

```

```

00029 {
00030     UnregisterAudioGenerator();
00031
00032     AudioCapture->StopCapturingAudio();
00033     AudioCapture->RemoveFromRoot();
00034
00035     delete AudioCapture; AudioCapture = nullptr;
00036     delete FileWriter; FileWriter = nullptr;
00037 }
00038
00039 void UAudioManager::StartCapturingAudio()
00040 {
00041     AudioBuffer.Empty();
00042
00043     bIsCapturingAudio = true;
00044 }
00045
00046 void UAudioManager::StopCapturingAudio()
00047 {
00048     bIsCapturingAudio = false;
00049
00050     if (AudioBuffer.Num() == 0)
00051         return;
00052
00053     SaveAudioBufferToWAV(Settings.SavePath);
00054
00055     if (OnAudioReadyForTranscription.ExecuteIfBound(AudioBuffer))
00056     {
00057         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Executing Audio Ready For Transcription
00058 Delegate. ||"));
00059     }
00060     else
00061     {
00062         UE_LOG(LogOpenAccessibilityCom, Warning, TEXT("|| No Delegates Bound to Audio Ready For
00063 Transcription Delegate. ||"));
00064     }
00065     AudioBuffer.Empty();
00066 }
00067 void UAudioManager::PRIVATE_OnAudioGenerate(const float* InAudio, int32 NumSamples)
00068 {
00069     if (bIsCapturingAudio == false)
00070         return;
00071
00072     // Need to Check Samples are above threshold and ignore if their run length is too long.
00073
00074     AudioBuffer.Append(InAudio, NumSamples);
00075 }
00076
00077 void UAudioManager::SaveAudioBufferToWAV(const FString& FilePath)
00078 {
00079     UE_LOG(LogOpenAccessibilityCom, Log, TEXT("Starting to Save Audio Buffer to WAV"));
00080
00081     Audio::FSampleBuffer SampleBuffer = Audio::FSampleBuffer(AudioBuffer.GetData(), AudioBuffer.Num(),
00082 AudioCapture->GetNumChannels(), AudioCapture->GetSampleRate());
00083
00084     FileWriter->BeginWriteToWavFile(SampleBuffer, Settings.SaveName, const_cast<FString*>(FilePath),
00085 []() {
00086         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("Audio Buffer Saved to WAV"));
00087     });
00088 }
00089 void UAudioManager::OnDefaultDeviceChanged(EAudioDeviceChangedRole ChangedRole, FString DeviceID)
00090 {
00091     UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Default Device Changed || Role: %d || DeviceID: %s
00092 ||"), ChangedRole, *DeviceID);
00093
00094     this->UnregisterAudioGenerator();
00095     this->RegisterAudioGenerator();
00096 }
00097 void UAudioManager::RegisterAudioGenerator()
00098 {
00099     // Add Audio Generator Delegate to get audio data from stream,
00100     // and apply wrapper function due to wanting to reference class function.
00101     OnAudioGenerateHandle = AudioCapture->AddGeneratorDelegate(FOnAudioGenerate([this](const float*
00102 InAudio, int32 NumSamples) {
00103         if (this->IsCapturingAudio()) this->PRIVATE_OnAudioGenerate(InAudio, NumSamples);
00104     }));
00105 }
00106 void UAudioManager::UnregisterAudioGenerator()
00107 {
00108     AudioCapture->RemoveGeneratorDelegate(OnAudioGenerateHandle);
00109 }

```

## 5.54 OpenAccessibilityComLogging.cpp

```
00001
00002 #include "OpenAccessibilityComLogging.h"
```

## 5.55 OpenAccessibilityCommunication.cpp

```
00001 // Copyright Epic Games, Inc. All Rights Reserved.
00002
00003 #include "OpenAccessibilityCommunication.h"
00004 #include "OpenAccessibilityComLogging.h"
00005
00006 #include "OpenAccessibilityAnalytics.h"
00007
00008 #include "AudioManager.h"
00009 #include "SocketCommunicationServer.h"
00010
00011 #include "PhraseTree/PhraseNode.h"
00012 #include "PhraseTree/PhraseInputNode.h"
00013 #include "PhraseTree/PhraseDirectionalInputNode.h"
00014 #include "PhraseTree/PhraseEventNode.h"
00015
00016 #include "Containers/Ticker.h"
00017 #include "Dom/JsonObject.h"
00018 #include "Interfaces/IPluginManager.h"
00019 #include "Sound/SampleBufferIO.h"
00020 #include "HAL/PlatformProcess.h"
00021
00022 #define LOCTEXT_NAMESPACE "UOpenAccessibilityCommunicationModule"
00023
00024 void FOpenAccessibilityCommunicationModule::StartupModule()
00025 {
00026     LoadZMQDLL();
00027
00028     // This code will execute after your module is loaded into memory; the exact timing is specified
    in the .uplugin file per-module
00029     UE_LOG(LogOpenAccessibilityCom, Display, TEXT("OpenAccessibilityComModule::StartupModule()"));
00030
00031     // Initialize AudioManager
00032     AudioManager = NewObject<UAudioManager>();
00033     AudioManager->AddToRoot();
00034
00035     AudioManager->OnAudioReadyForTranscription
00036         .BindRaw(this, &FOpenAccessibilityCommunicationModule::TranscribeWaveForm);
00037
00038     // Initialize Socket Server
00039     SocketServer = MakeShared<FSocketCommunicationServer>();
00040
00041     // Build The Phrase Tree
00042     BuildPhraseTree();
00043
00044     // Bind Tick Event
00045     TickDelegate = FTickerDelegate::CreateRaw(this, &FOpenAccessibilityCommunicationModule::Tick);
00046     TickDelegateHandle = FTSTicker::GetCoreTicker().AddTicker(TickDelegate);
00047
00048     // Bind Input Events
00049     KeyDownEventHandle = FSlateApplication::Get().OnApplicationPreInputKeyDownListener().AddRaw(this,
    &FOpenAccessibilityCommunicationModule::HandleKeyDownEvent);
00050
00051     // Register Console Commands
00052     RegisterConsoleCommands();
00053 }
00054
00055 void FOpenAccessibilityCommunicationModule::ShutdownModule()
00056 {
00057     // This function may be called during shutdown to clean up your module. For modules that support
    dynamic reloading,
00058     // we call this function before unloading the module.
00059     UE_LOG(LogOpenAccessibilityCom, Display, TEXT("OpenAccessibilityComModule::ShutdownModule()"));
00060
00061     AudioManager->RemoveFromRoot();
00062     PhraseTreeUtils->RemoveFromRoot();
00063
00064     FSlateApplication::Get().OnApplicationPreInputKeyDownListener().Remove(KeyDownEventHandle);
00065
00066     UnloadZMQDLL();
00067
00068     UnregisterConsoleCommands();
00069 }
00070
00071 bool FOpenAccessibilityCommunicationModule::Tick(const float DeltaTime)
00072 {
00073     // Detect if any events are ready to be received.
```

```

00074     if (SocketServer->EventOccured())
00075     {
00076         TArray<FString> RecvStrings;
00077         TSharedPtr<FJsonObject> RecvMetadata;
00078
00079         // Receive the Detected Event, with separate transcriptions and metadata.
00080         if (SocketServer->RecvStringMultipartWithMeta(RecvStrings, RecvMetadata))
00081         {
00082             OA_LOG(LogOpenAccessibilityCom, Log, TEXT("TRANSCRIPTION RECIEVED"), TEXT("Recieved
Multipart - Message Count: %d"), RecvStrings.Num());
00083
00084             // Send Received Transcriptions to any bound events.
00085             OnTranscriptionRecieved.Broadcast(RecvStrings);
00086         }
00087     }
00088
00089     return true;
00090 }
00091
00092 void FOpenAccessibilityCommunicationModule::HandleKeyDownEvent(const FKeyEvent& InKeyEvent)
00093 {
00094     // If the Space Key is pressed, we will send a request to the Accessibility Server
00095     if (InKeyEvent.GetKey() == EKeys::SpaceBar)
00096     {
00097         if (InKeyEvent.IsShiftDown())
00098         {
00099             OA_LOG(LogOpenAccessibilityCom, Log, TEXT("AudioCapture Change"), TEXT("Stopping Audio
Capture"));
00100             AudioManager->StopCapturingAudio();
00101         }
00102         else
00103         {
00104             OA_LOG(LogOpenAccessibilityCom, Log, TEXT("AudioCapture Change"), TEXT("Starting Audio
Capture"));
00105             AudioManager->StartCapturingAudio();
00106         }
00107     }
00108 }
00109
00110 void FOpenAccessibilityCommunicationModule::TranscribeWaveForm(const TArray<float>
AudioBufferToTranscribe)
00111 {
00112     if (AudioBufferToTranscribe.Num() == 0)
00113     {
00114         UE_LOG(LogOpenAccessibilityCom, Warning, TEXT("|| Transcription Ready || Audio Buffer is Empty
||"));
00115         return;
00116     }
00117
00118     PrevAudioBuffer = TArray(AudioBufferToTranscribe);
00119
00120     UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| WaveForm Transcription || Array Size: %d || Byte
Size: %s ||"), AudioBufferToTranscribe.Num(), *FString::FromInt(AudioBufferToTranscribe.Num() *
sizeof(float)));
00121
00122     // Create Metadata of Audio Source.
00123     TSharedPtr<FJsonObject> AudioBufferMetadata = MakeShared<FJsonObject>();
00124     AudioBufferMetadata->SetNumberField(TEXT("sample_rate"),
AudioManager->GetAudioCaptureSampleRate());
00125     AudioBufferMetadata->SetNumberField(TEXT("num_channels"),
AudioManager->GetAudioCaptureNumChannels());
00126
00127     bool bArrayMessageSent = SocketServer->SendArrayMessageWithMeta(AudioBufferToTranscribe,
AudioBufferMetadata.ToSharedRef(), ComSendFlags::none);
00128
00129     OA_LOG(LogOpenAccessibilityCom, Log, TEXT("TRANSCRIPTION SENT"), TEXT("{%s} Send Audiobuffer
(float x %d / %d Hz / %d channels)"),
00130         bArrayMessageSent ? TEXT("Success") : TEXT("Failed"),
00131         AudioBufferToTranscribe.Num(), AudioManager->GetAudioCaptureSampleRate(),
00132         AudioManager->GetAudioCaptureNumChannels());
00133 }
00134
00135 void FOpenAccessibilityCommunicationModule::BuildPhraseTree()
00136 {
00137     // Initialize the Phrase Tree
00138     PhraseTree = MakeShared<FPhraseTree>();
00139     PhraseTreePhraseRecievedHandle = OnTranscriptionRecieved
.AddRaw(PhraseTree.Get(), &FPhraseTree::ParseTranscription);
00140
00141     PhraseTreeUtils = NewObject<UPhraseTreeUtils>();
00142     PhraseTreeUtils->SetPhraseTree(PhraseTree.ToSharedRef());
00143     PhraseTreeUtils->AddToRoot();
00144 }
00145
00146 void FOpenAccessibilityCommunicationModule::RegisterConsoleCommands()
00147 {
00148     // Audio Commands

```

```

00149
00150     ConsoleCommands.Add(IConsoleManager::Get().RegisterConsoleCommand(
00151         TEXT("OpenAccessibilityCom.Debug.ShowAudioSampleRate"),
00152         TEXT("Logs the Number of Samples being captured, from user input."),
00153
00154         FConsoleCommandDelegate::CreateLambda([this]() {
00155             UE_LOG(LogOpenAccessibilityCom, Display,
00156                 TEXT("OpenAccessibilityCom.Debug.ShowAudioSampleRate | Sample Rate: %d"),
00157                 this->AudioManager->GetAudioCaptureSampleRate());
00158         }));
00159
00160     ConsoleCommands.Add(IConsoleManager::Get().RegisterConsoleCommand(
00161         TEXT("OpenAccessibilityCom.Debug.ShowAudioNumChannels"),
00162         TEXT("Logs the Number of Audio Channels being captured, from user input."),
00163
00164         FConsoleCommandDelegate::CreateLambda([this]() {
00165             UE_LOG(LogOpenAccessibilityCom, Display,
00166                 TEXT("OpenAccessibilityCom.Debug.ShowAudioNumChannels | Num Channels: %d"),
00167                 this->AudioManager->GetAudioCaptureNumChannels());
00168         }));
00169
00170     ConsoleCommands.Add(IConsoleManager::Get().RegisterConsoleCommand(
00171         TEXT("OpenAccessibilityCom.Debug.SendLastBuffer"),
00172         TEXT("Sends the last saved audio buffer to the transcription service."),
00173
00174         FConsoleCommandDelegate::CreateLambda([this]() {
00175             UE_LOG(LogOpenAccessibilityCom, Display,
00176                 TEXT("OpenAccessibilityCom.Debug.SendLastBuffer"));
00177             TranscribeWaveForm(PrevAudioBuffer);
00178         }));
00179
00180 }
00181
00182 void FOpenAccessibilityCommunicationModule::UnregisterConsoleCommands()
00183 {
00184     IConsoleCommand* ConsoleCommand = nullptr;
00185     while (ConsoleCommands.Num() > 0)
00186     {
00187         ConsoleCommand = ConsoleCommands.Pop();
00188         IConsoleManager::Get().UnregisterConsoleObject(ConsoleCommand);
00189     }
00190 }
00191
00192
00193 void FOpenAccessibilityCommunicationModule::LoadZMQDLL()
00194 {
00195     FString BaseDir = IPluginManager::Get().FindPlugin("OpenAccessibility")->GetBaseDir();
00196
00197     FString LibraryPath;
00198     #if PLATFORM_WINDOWS
00199     #if UE_BUILD_DEBUG
00200         LibraryPath = FPaths::Combine(*BaseDir,
00201             TEXT("Binaries/ThirdParty/ZeroMQ/Win64/libzmq-mt-gd-4_3_5.dll"));
00202     #else
00203         LibraryPath = FPaths::Combine(*BaseDir,
00204             TEXT("Binaries/ThirdParty/ZeroMQ/Win64/libzmq-mt-4_3_5.dll"));
00205     #endif
00206 #elif PLATFORM_LINUX
00207     LibraryPath = FPaths::Combine(*BaseDir,
00208         TEXT("Binaries/ThirdParty/ZeroMQ/Linux/libzmq-mt-4_3_5.so"));
00209 #elif PLATFORM_MAC
00210     LibraryPath = FPaths::Combine(*BaseDir,
00211         TEXT("Source/ThirdParty/ZeroMQ/Mac/libzmq-mt-4_3_5.dylib"));
00212 #endif
00213
00214     ZMQDllHandle = !LibraryPath.IsEmpty() ? FPlatformProcess::GetDllHandle(*LibraryPath) : nullptr;
00215
00216     if (ZMQDllHandle)
00217     {
00218         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| LoadZMQDLL || Successfully Loaded ZMQ DLL ||"));
00219     }
00220     else
00221     {
00222         UE_LOG(LogOpenAccessibilityCom, Error, TEXT("|| LoadZMQDLL || Failed to Load ZMQ DLL ||"));
00223     }
00224 }
00225
00226 void FOpenAccessibilityCommunicationModule::UnloadZMQDLL()
00227 {
00228     FPlatformProcess::FreeDllHandle(ZMQDllHandle);
00229     ZMQDllHandle = nullptr;
00230 }

```

```

00227
00228 #undef LOCTEXT_NAMESPACE
00229
00230 IMPLEMENT_MODULE(FOpenAccessibilityCommunicationModule, OpenAccessibility)

```

## 5.56 PhraseTree.cpp

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003
00004 #include "PhraseTree.h"
00005 #include "PhraseTree/PhraseNode.h"
00006 #include "Algo/Reverse.h"
00007
00008 #include "Logging/StructuredLog.h"
00009 #include "OpenAccessibilityComLogging.h"
00010 #include "OpenAccessibilityAnalytics.h"
00011
00012 FPhraseTree::FPhraseTree() : FPhraseNode(TEXT("ROOT_NODE"))
00013 {
00014     ContextManager = FPhraseTreeContextManager();
00015
00016     FTickerDelegate TickDelegate = FTickerDelegate::CreateRaw(this, &FPhraseTree::Tick);
00017     TickDelegateHandle = FTSTicker::GetCoreTicker().AddTicker(TickDelegate);
00018 }
00019
00020 FPhraseTree::~FPhraseTree()
00021 {
00022     FTSTicker::GetCoreTicker().RemoveTicker(TickDelegateHandle);
00023 }
00024
00025 bool FPhraseTree::Tick(float DeltaTime)
00026 {
00027     // Filter InActive Context Objects out of the stack.
00028     ContextManager.FilterContextStack();
00029
00030     return true;
00031 }
00032
00033 void FPhraseTree::ParseTranscription(TArray<FString> InTranscriptionSegments)
00034 {
00035     if (InTranscriptionSegments.IsEmpty())
00036     {
00037         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Phrase Tree || Provided Transcription is Empty
||"))
00038         return;
00039     }
00040
00041     TArray<FString> SegmentWordArray = TArray<FString>();
00042     int SegmentCount = 0;
00043
00044     // Loop over any Transcription Segments.
00045     for (FString& TranscriptionSegment : InTranscriptionSegments)
00046     {
00047         if (TranscriptionSegment.IsEmpty())
00048         {
00049             UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Phrase Tree || Transcription Segment is
Empty ||"))
00050             continue;
00051         }
00052
00053         // Filter the Transcription Segment, to remove any unwanted characters.
00054         TranscriptionSegment.TrimStartAndEndInline();
00055         TranscriptionSegment.ReplaceInline(TEXT("."), TEXT(""), ESearchCase::IgnoreCase);
00056         TranscriptionSegment.ReplaceInline(TEXT(","), TEXT(""), ESearchCase::IgnoreCase);
00057         TranscriptionSegment.ToUpperInline();
00058
00059         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Phrase Tree || Filtered Transcription Segment: {
%s } ||"), *TranscriptionSegment)
00060
00061         // Parse the Transcription Segment into an Array of Words, removing any white space.
00062         TranscriptionSegment.ParseIntoArrayWS(SegmentWordArray);
00063         if (SegmentWordArray.Num() == 0)
00064         {
00065             UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Phrase Tree || Transcription Segment has no
Word Content ||"))
00066             continue;
00067         }
00068
00069         Algo::Reverse(SegmentWordArray);
00070
00071         // Loop until the Segment is Empty
00072         while (!SegmentWordArray.IsEmpty())

```



```

00073     {
00074
00075         FParseRecord ParseRecord = FParseRecord(ContextManager.GetContextStack());
00076         FParseResult ParseResult = ParsePhrase(SegmentWordArray, ParseRecord);
00077
00078         ContextManager.UpdateContextStack(ParseRecord.ContextObjectStack);
00079
00080         UE_LOG_FMT(LogOpenAccessibilityCom, Log, "|| Phrase Tree || Segment: {0} | Result: {1} ||",
SegmentCount, ParseResult.Result);
00081
00082         switch (ParseResult.Result)
00083         {
00084             case PHRASE_PARSED:
00085             case PHRASE_PARSED_AND_EXECUTED:
00086             {
00087                 OA_LOG(LogOpenAccessibilityCom, Log, TEXT("PhraseTree Propagation"),
TEXT("{Success} Phrase Tree Parsed Correctly (%s)",
00088                     *ParseRecord.GetPhraseString())
00089
00090                 LastVistedNode.Reset();
00091                 LastVistedParseRecord = FParseRecord();
00092
00093                 break;
00094             }
00095
00096             case PHRASE_REQUIRES_MORE:
00097             {
00098                 OA_LOG(LogOpenAccessibilityCom, Log, TEXT("PhraseTree Propagation"),
TEXT("{Failed} Phrase Tree Propagation Requires More Segments. (%s)",
00099                     *ParseRecord.GetPhraseString())
00100
00101                 // Store Reach Nodes, and the ParseRecord for future propagation attempts.
00102                 LastVistedNode = ParseResult.ReachedNode;
00103                 LastVistedParseRecord = ParseRecord;
00104             }
00105
00106             case PHRASE_REQUIRES_MORE_CORRECT_PHRASES:
00107             {
00108                 OA_LOG(LogOpenAccessibilityCom, Log, TEXT("PhraseTree Propagation"),
TEXT("{Failed} Phrase Tree Propagation Requires More Correct Segments. (%s)",
00109                     *ParseRecord.GetPhraseString())
00110
00111                 LastVistedNode = ParseResult.ReachedNode;
00112                 LastVistedParseRecord = ParseRecord;
00113
00114                 // Dirty Way of Ensuring all Segments in Transcription are Attempted.
00115                 if (!SegmentWordArray.IsEmpty())
00116                     SegmentWordArray.Pop();
00117
00118                 break;
00119             }
00120
00121             default:
00122             case PHRASE_UNABLE_TO_PARSE:
00123             {
00124                 OA_LOG(LogOpenAccessibilityCom, Log, TEXT("PhraseTree Propagation"),
TEXT("{Failed} Phrase Tree Propagation Failed. (%s)",
00125                     *ParseRecord.GetPhraseString())
00126
00127                 // Dirty Way of Ensuring all Segments in Transcription are Attempted.
00128                 if (!SegmentWordArray.IsEmpty())
00129                     SegmentWordArray.Pop();
00130
00131                 break;
00132             }
00133         }
00134     }
00135
00136     SegmentCount++;
00137     SegmentWordArray.Reset();
00138 }
00139 }
00140
00141 FParseResult FPhraseTree::ParsePhrase(TArray<FString>& InPhraseWordArray, FParseRecord& InParseRecord)
00142 {
00143     if (InPhraseWordArray.IsEmpty())
00144     {
00145         UE_LOG(LogOpenAccessibilityCom, Warning, TEXT("|| Phrase Tree || Provided Transcription
Segment is Empty ||"));
00146
00147         return FParseResult(PHRASE_NOT_PARSED);
00148     }
00149
00150     // First give the last visited node a chance to parse the phrase.
00151     // due to the possibility of connecting phrases over different transcription segments.
00152     if (LastVistedNode != nullptr && LastVistedNode.IsValid())
00153     {

```

```

00154     TArray<FString> PhraseWordArrayCopy = TArray(InPhraseWordArray);
00155
00156     FParseResult ParseResult = LastVistedNode->ParseChildren(PhraseWordArrayCopy,
LastVistedParseRecord);
00157     if (ParseResult.Result == PHRASE_PARSED)
00158     {
00159         LastVistedNode.Reset();
00160         InParseRecord = LastVistedParseRecord;
00161         LastVistedParseRecord = FParseRecord();
00162
00163         return ParseResult;
00164     }
00165     else if (ParseResult.Result != PHRASE_UNABLE_TO_PARSE)
00166     {
00167         return ParseResult;
00168     }
00169 }
00170
00171 // Check if the Context Stack has Objects, if so propagation from the Context Root.
00172 if (ContextManager.HasContextObjects())
00173 {
00174     // Propagate from the Context Root, that is the Top of the Context Stack.
00175     return
ContextManager.PeekContextObject()->GetContextRoot()->ParsePhraseAsContext(InPhraseWordArray,
InParseRecord);
00176 }
00177
00178 // Otherwise, start a new propagation entirely from the Tree Root.
00179 return ParseChildren(InPhraseWordArray, InParseRecord);
00180 }
00181
00182 void FPhraseTree::BindBranch(const TPhraseNode& InNode)
00183 {
00184     TArray<FPhraseTreeBranchBind> ToBindArray = TArray<FPhraseTreeBranchBind>();
00185
00186     ToBindArray.Add(FPhraseTreeBranchBind(AsShared(), InNode));
00187
00188     while (!ToBindArray.IsEmpty())
00189     {
00190         FPhraseTreeBranchBind BranchToBind = ToBindArray.Pop();
00191
00192         // Check all ChildNodes to see if they are similar in purpose.
00193         for (auto& ChildNode : BranchToBind.StartNode->ChildNodes)
00194         {
00195             // If a ChildNode meets the same requirements as the BranchRoot,
00196             // then Split Bind Process to the ChildNodes.
00197             if (ChildNode->RequiresPhrase(BranchToBind.BranchRoot->BoundPhrase))
00198             {
00199                 for (auto& BranchChildNode : BranchToBind.BranchRoot->ChildNodes)
00200                 {
00201                     ToBindArray.Add(FPhraseTreeBranchBind(ChildNode, BranchChildNode));
00202                 }
00203
00204                 continue;
00205             }
00206         }
00207
00208         // If the Start Node has no similar children, then bind the branch to the start node.
00209         // Can force bind, as previous checks show no child is similar.
00210         BranchToBind.StartNode->BindChildNodeForce(BranchToBind.BranchRoot);
00211     }
00212 }
00213
00214 void FPhraseTree::BindBranches(const TPhraseNodeArray& InNodes)
00215 {
00216     for (const TSharedPtr<FPhraseNode>& Node : InNodes)
00217     {
00218         BindBranch(Node);
00219     }
00220 }

```

## 5.57 ContextMenuObject.cpp

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #include "PhraseTree/Containers/ContextMenuObject.h"
00004
00005 #include "OpenAccessibilityComLogging.h"
00006
00007 UPhraseTreeContextMenuObject::UPhraseTreeContextMenuObject()
00008 : UPhraseTreeContextObject()
00009 {
00010

```

```

00011 }
00012
00013 UPhraseTreeContextMenuObject::UPhraseTreeContextMenuObject(TSharedRef<IMenu> Menu)
00014 : UPhraseTreeContextObject()
00015 {
00016
00017 }
00018
00019 UPhraseTreeContextMenuObject::~UPhraseTreeContextMenuObject()
00020 {
00021     // Unbind Tick Delegate
00022     RemoveTickDelegate();
00023
00024     if (Menu.IsValid())
00025         RemoveMenuDismissed(Menu.Pin().ToSharedRef());
00026
00027     UE_LOG(LogOpenAccessibilityCom, Warning, TEXT("|| Context Menu || Destroyed ||"))
00028 }
00029
00030 void UPhraseTreeContextMenuObject::Init(TSharedRef<IMenu> InMenu)
00031 {
00032     this->Menu = InMenu;
00033     this->Window = FSlateApplication::Get().FindWidgetWindow(
00034         InMenu->GetContent().ToSharedRef()
00035     );
00036
00037     BindMenuDismissed(InMenu);
00038     BindTickDelegate();
00039 }
00040
00041 void UPhraseTreeContextMenuObject::Init(TSharedRef<IMenu> InMenu, TSharedRef<FPhraseNode>
    InContextRoot)
00042 {
00043     this->Menu = InMenu;
00044     this->Window = FSlateApplication::Get().FindWidgetWindow(
00045         InMenu->GetContent().ToSharedRef()
00046     );
00047
00048     this->ContextRoot = InContextRoot;
00049
00050     BindMenuDismissed(InMenu);
00051     BindTickDelegate();
00052 }
00053
00054 void UPhraseTreeContextMenuObject::BindTickDelegate()
00055 {
00056     TickDelegate = FTickerDelegate::CreateUObject(this, &UPhraseTreeContextMenuObject::Tick);
00057     TickDelegateHandle = FTicker::GetCoreTicker().AddTicker(TickDelegate);
00058 }
00059
00060 void UPhraseTreeContextMenuObject::RemoveTickDelegate()
00061 {
00062     if (TickDelegateHandle != NULL)
00063         FTicker::GetCoreTicker().RemoveTicker(TickDelegateHandle);
00064 }
00065
00066 void UPhraseTreeContextMenuObject::BindMenuDismissed(TSharedRef<IMenu> InMenu)
00067 {
00068     MenuDismissedHandle = InMenu->GetOnMenuDismissed()
00069         .AddUObject(this, &UPhraseTreeContextMenuObject::OnMenuDismissed);
00070 }
00071
00072 void UPhraseTreeContextMenuObject::RemoveMenuDismissed(TSharedRef<IMenu> InMenu)
00073 {
00074     Menu.Pin()->GetOnMenuDismissed().Remove(MenuDismissedHandle);
00075 }
00076
00077 void UPhraseTreeContextMenuObject::OnMenuDismissed(TSharedRef<IMenu> InMenu)
00078 {
00079     RemoveTickDelegate();
00080
00081     RemoveFromRoot();
00082     MarkAsGarbage();
00083
00084     bIsActive = false;
00085
00086     UE_LOG(LogOpenAccessibilityCom, Warning, TEXT("|| Context Menu || Dismissed ||"))
00087 }

```

## 5.58 PhraseEnumInputNode.cpp

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002

```

```

00003
00004 #include "PhraseTree/PhraseEnumInputNode.h"
00005
00006 #include "PhraseTree/Containers/Input/UParseEnumInput.h"
00007
00008 template<typename TEnum>
00009 FPhraseEnumInputNode<TEnum>::FPhraseEnumInputNode(const TCHAR* NodeName)
00010 : FPhraseInputNode(NodeName)
00011 {
00012     static_assert(TIsEnum<TEnum>::Value, "Passed EnumType Must be an Enum.");
00013 };
00014
00015 template<typename TEnum>
00016 FPhraseEnumInputNode<TEnum>::FPhraseEnumInputNode(const TCHAR* NodeName, TPhraseNodeArray
    InChildNodes)
00017 : FPhraseInputNode(NodeName, InChildNodes)
00018 {
00019     static_assert(TIsEnum<TEnum>::Value, "Passed EnumType Must be an Enum");
00020 }
00021
00022 template<typename TEnum>
00023 FPhraseEnumInputNode<TEnum>::FPhraseEnumInputNode(const TCHAR* InInputString,
    TDelegate<void(FParseRecord& Record)> InOnPhraseParsed, TPhraseNodeArray InChildNodes)
00024 : FPhraseInputNode(InInputString, InOnPhraseParsed, InChildNodes)
00025 {
00026     static_assert(TIsEnum<TEnum>::Value, "Passed EnumType Must be an Enum");
00027 }
00028
00029 template<typename TEnum>
00030 FPhraseEnumInputNode<TEnum>::FPhraseEnumInputNode(const TCHAR* InInputString, TPhraseNodeArray
    InChildNodes, TDelegate<void(int32 Input)> InOnInputRecieved)
00031 : FPhraseInputNode(InInputString, InChildNodes, InOnInputRecieved)
00032 {
00033     static_assert(TIsEnum<TEnum>::Value, "Passed EnumType Must be an Enum");
00034 }
00035
00036 template<typename TEnum>
00037 FPhraseEnumInputNode<TEnum>::FPhraseEnumInputNode(const TCHAR* InInputString,
    TDelegate<void(FParseRecord& Record)> InOnPhraseParsed, TPhraseNodeArray InChildNodes,
    TDelegate<void(int32 Input)> InOnInputRecieved)
00038 : FPhraseInputNode(InInputString, InOnPhraseParsed, InChildNodes, InOnInputRecieved)
00039 {
00040     static_assert(TIsEnum<TEnum>::Value, "Passed EnumType Must be an Enum");
00041 }
00042
00043 template<typename TEnum>
00044 FPhraseEnumInputNode<TEnum>::~~FPhraseEnumInputNode()
00045 {
00046 }
00047 }
00048
00049 template<typename TEnum>
00050 bool FPhraseEnumInputNode<TEnum>::MeetsInputRequirements(const FString& InPhrase)
00051 {
00052     UEnum* EnumPtr = StaticEnum<TEnum>();
00053     if (!EnumPtr)
00054     {
00055         UE_LOG(LogTemp, Error, TEXT("FPhraseEnumInputNode::MeetsInputRequirements: EnumPtr is NULL"));
00056         return false;
00057     }
00058     return EnumPtr->IsValidEnumName(*EnumPtr->GenerateFullEnumName(*InPhrase.ToUpper()));
00059 }
00060
00061
00062 template<typename TEnum>
00063 bool FPhraseEnumInputNode<TEnum>::RecordInput(const FString& InInput, FParseRecord& OutParseRecord)
00064 {
00065     UEnum* EnumPtr = StaticEnum<TEnum>();
00066     if (!EnumPtr)
00067     {
00068         UE_LOG(LogTemp, Error, TEXT("FPhraseEnumInputNode::RecordInput: EnumPtr is NULL"));
00069         return false;
00070     }
00071
00072     int32 Val = EnumPtr->GetValueByNameString(EnumPtr->GenerateFullEnumName(*InInput.ToUpper()));
00073     if (Val == INDEX_NONE)
00074     {
00075         return false;
00076     }
00077
00078     UParseEnumInput* ParseInput = MakeParseInput<UParseEnumInput>();
00079     ParseInput->SetValue(Val);
00080     ParseInput->SetEnumType(EnumPtr);
00081
00082     OutParseRecord.AddPhraseInput(BoundPhrase, ParseInput);
00083
00084     return true;

```

```
00085 }
```

## 5.59 PhraseEventNode.cpp

```
00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003
00004 #include "PhraseTree/PhraseEventNode.h"
00005 #include "OpenAccessibilityComLogging.h"
00006
00007 FPhraseEventNode::FPhraseEventNode()
00008     : FPhraseNode(TEXT("EVENT_NODE"))
00009 {
00010     OnPhraseParsed = TDelegate<void(FParseRecord*)>();
00011 }
00012
00013 FPhraseEventNode::FPhraseEventNode(TDelegate<void(FParseRecord*)> InEvent)
00014     : FPhraseNode(TEXT("EVENT_NODE"), InEvent)
00015 {
00016 }
00017 }
00018
00019 FPhraseEventNode::FPhraseEventNode(TFunction<void(FParseRecord*)> InEventFunction)
00020     : FPhraseNode(TEXT("EVENT_NODE"), TDelegate<void(FParseRecord*)>::CreateLambda(InEventFunction))
00021 {
00022 }
00023 }
00024
00025 FPhraseEventNode::~FPhraseEventNode()
00026 {
00027 }
00028 }
00029
00030 bool FPhraseEventNode::RequiresPhrase(const FString InPhrase)
00031 {
00032     return true;
00033 }
00034
00035 bool FPhraseEventNode::RequiresPhrase(const FString InPhrase, int32& OutDistance)
00036 {
00037     OutDistance = 0;
00038     return true;
00039 }
00040
00041 FParseResult FPhraseEventNode::ParsePhrase(TArray<FString>& InPhraseArray, FParseRecord&
    InParseRecord)
00042 {
00043     if (OnPhraseParsed.ExecuteIfBound(InParseRecord))
00044     {
00045         return FParseResult(PHRASE_PARSED_AND_EXECUTED);
00046     }
00047
00048     UE_LOG(LogOpenAccessibilityCom, Warning, TEXT("|| Unable to Execute Event ||"))
00049
00050     return FParseResult(PHRASE_UNABLE_TO_PARSE, AsShared());
00051 }
```

## 5.60 PhraseInputNode.cpp

```
00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #include "PhraseTree/PhraseInputNode.h"
00004 #include "PhraseTree/Utils.h"
00005 #include "OpenAccessibilityComLogging.h"
00006
00007 #include "PhraseTree/Containers/Input/UParseIntInput.h"
00008
00009 template<typename InputType>
00010 FPhraseInputNode<InputType>::FPhraseInputNode(const TCHAR* InInputString)
00011     : FPhraseNode(InInputString)
00012 {
00013 }
00014 }
00015
00016 template<typename InputType>
00017 FPhraseInputNode<InputType>::FPhraseInputNode(const TCHAR* InInputString, TPhraseNodeArray
    InChildNodes)
00018     : FPhraseNode(InInputString, InChildNodes)
00019 {
```

```

00020
00021 }
00022
00023 template<typename InputType>
00024 FPhraseInputNode<InputType>::FPhraseInputNode(const TCHAR* InInputString, TDelegate<void(FParseRecord&
00025     Record)> InOnPhraseParsed, TPhraseNodeArray InChildNodes)
00026     : FPhraseNode(InInputString, InOnPhraseParsed, InChildNodes)
00027 {
00028 }
00029
00030 template<typename InputType>
00031 FPhraseInputNode<InputType>::FPhraseInputNode(const TCHAR* InInputString, TPhraseNodeArray
00032     InChildNodes, TDelegate<void(InputType Input)> InOnInputRecieved)
00033     : FPhraseNode(InInputString, InChildNodes)
00034 {
00035     OnInputReceived = InOnInputRecieved;
00036 }
00037 template<typename InputType>
00038 FPhraseInputNode<InputType>::FPhraseInputNode(const TCHAR* InInputString, TDelegate<void(FParseRecord&
00039     Record)> InOnPhraseParsed, TPhraseNodeArray InChildNodes, TDelegate<void(InputType Input)>
00040     InOnInputRecieved)
00041     : FPhraseNode(InInputString, InOnPhraseParsed, InChildNodes)
00042 {
00043     OnInputReceived = InOnInputRecieved;
00044 }
00045 template<typename InputType>
00046 FPhraseInputNode<InputType>::~~FPhraseInputNode()
00047 {
00048 }
00049
00050 template<typename InputType>
00051 bool FPhraseInputNode<InputType>::RequiresPhrase(const FString InPhrase)
00052 {
00053     return MeetsInputRequirements(InPhrase);
00054 }
00055
00056 template<typename InputType>
00057 bool FPhraseInputNode<InputType>::RequiresPhrase(const FString InPhrase, int32& OutDistance)
00058 {
00059     bool bMeetsRequirements = MeetsInputRequirements(InPhrase);
00060     OutDistance = bMeetsRequirements ? 0 : INT32_MAX;
00061
00062     return bMeetsRequirements;
00063 }
00064
00065 template<typename InputType>
00066 FParseResult FPhraseInputNode<InputType>::ParsePhrase(TArray<FString>& InPhraseArray, FParseRecord&
00067     InParseRecord)
00068 {
00069     if (InPhraseArray.Num() == 0)
00070     {
00071         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Emptied Phrase Array ||"))
00072
00073         return FParseResult(PHRASE_REQUIRES_MORE, AsShared());
00074     }
00075
00076     if (MeetsInputRequirements(InPhraseArray.Last()))
00077     {
00078         // Get the Input String.
00079         FString InputToRecord = InPhraseArray.Pop();
00080
00081         // Append the Input String to the Record.
00082         InParseRecord.AddPhraseString(InputToRecord);
00083
00084         if (!InputToRecord.IsNumeric() && NumericParser::IsValidNumeric(InputToRecord, false))
00085         {
00086             NumericParser::StringToNumeric(InputToRecord, false);
00087         }
00088
00089         if (!RecordInput(InputToRecord, InParseRecord))
00090         {
00091             UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Unable to Record Input ||"))
00092
00093             return FParseResult(PHRASE_UNABLE_TO_PARSE, AsShared());
00094         }
00095
00096         OnPhraseParsed.ExecuteIfBound(InParseRecord);
00097
00098         return ParseChildren(InPhraseArray, InParseRecord);
00099     }
00100
00101     return FParseResult(PHRASE_UNABLE_TO_PARSE, AsShared());
00102 }

```

```

00102
00103 template<typename InputType>
00104 bool FPhraseInputNode<InputType>::MeetsInputRequirements(const FString& InPhrase)
00105 {
00106     return InPhrase.IsNumeric() || NumericParser::IsValidNumeric(InPhrase, false);
00107 }
00108
00109 template<typename InputType>
00110 bool FPhraseInputNode<InputType>::RecordInput(const FString& InInput, FParseRecord& OutParseRecord)
00111 {
00112     return false;
00113 }
00114
00115 bool FPhraseInputNode<int32>::RecordInput(const FString& InInput, FParseRecord& OutParseRecord)
00116 {
00117     int32 Input = FCString::Atoi(*InInput);
00118
00119     UParseIntInput* ParseInput = MakeParseInput<UParseIntInput>();
00120     ParseInput->SetValue(Input);
00121
00122     OutParseRecord.AddPhraseInput(BoundPhrase, ParseInput);
00123
00124     OnInputReceived.ExecuteIfBound(Input);
00125
00126     return true;
00127 }

```

## 5.61 PhraseNode.cpp

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #include "PhraseTree/PhraseNode.h"
00004 #include "PhraseTree.h"
00005 #include "OpenAccessibilityComLogging.h"
00006
00007 #include "Algo/LevenshteinDistance.h"
00008
00009 FPhraseNode::FPhraseNode(const TCHAR* InBoundPhrase)
00010 {
00011     BoundPhrase = InBoundPhrase;
00012     BoundPhrase.ToUpperInline();
00013
00014     ChildNodes = TArray<TSharedPtr<FPhraseNode>>();
00015 }
00016
00017 FPhraseNode::FPhraseNode(const TCHAR* InBoundPhrase, TDelegate<void(FParseRecord& Record)>
    InOnPhraseParsed)
00018 {
00019     BoundPhrase = InBoundPhrase;
00020     BoundPhrase.ToUpperInline();
00021
00022     OnPhraseParsed = InOnPhraseParsed;
00023     ChildNodes = TArray<TSharedPtr<FPhraseNode>>();
00024 }
00025
00026 FPhraseNode::FPhraseNode(const TCHAR* InBoundPhrase, TPhraseNodeArray InChildNodes)
00027 {
00028     BoundPhrase = InBoundPhrase;
00029     BoundPhrase.ToUpperInline();
00030
00031     ChildNodes = InChildNodes;
00032 }
00033
00034 FPhraseNode::FPhraseNode(const TCHAR* InBoundPhrase, TDelegate<void(FParseRecord& Record)>
    InOnPhraseParsed, TPhraseNodeArray InChildNodes)
00035 {
00036     BoundPhrase = InBoundPhrase;
00037     BoundPhrase.ToUpperInline();
00038
00039     OnPhraseParsed = InOnPhraseParsed;
00040     ChildNodes = InChildNodes;
00041 }
00042
00043 FPhraseNode::~FPhraseNode()
00044 {
00045 }
00046
00047
00048 bool FPhraseNode::HasLeafChild() const
00049 {
00050     return bHasLeafChild;
00051 }
00052

```

```

00053 bool FPhraseNode::RequiresPhrase(FString InPhrase)
00054 {
00055     return InPhrase.Equals(BoundPhrase, ESearchCase::IgnoreCase) ||
        Algo::LevenshteinDistance(BoundPhrase, InPhrase) < 3;
00056 }
00057
00058 bool FPhraseNode::RequiresPhrase(const FString InPhrase, int32& OutDistance)
00059 {
00060     OutDistance = Algo::LevenshteinDistance(BoundPhrase, InPhrase);
00061
00062     return InPhrase.Equals(BoundPhrase, ESearchCase::IgnoreCase) || OutDistance < 3;
00063 }
00064
00065 FParseResult FPhraseNode::ParsePhrase(TArray<FString>& InPhraseArray,
00066                                         FParseRecord& InParseRecord) {
00067     if (InPhraseArray.IsEmpty())
00068     {
00069         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Emptied Phrase Array ||"))
00070
00071         return FParseResult(PHRASE_REQUIRES_MORE, AsShared());
00072     }
00073
00074     // Pop the Phrase Linked to this Node.
00075     // Apply to the Record.
00076     FString LinkedPhrase = InPhraseArray.Pop();
00077
00078     // Append Removed Phrase To Record.
00079     InParseRecord.AddPhraseString(LinkedPhrase);
00080
00081     OnPhraseParsed.ExecuteIfBound(InParseRecord);
00082
00083     // Pass
00084     return ParseChildren(InPhraseArray, InParseRecord);
00085 }
00086
00087 FParseResult FPhraseNode::ParsePhraseAsContext(TArray<FString>& InPhraseWordArray, FParseRecord&
    InParseRecord)
00088 {
00089     if (InPhraseWordArray.IsEmpty())
00090     {
00091         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Emptied Phrase Array ||"))
00092
00093         return FParseResult(PHRASE_REQUIRES_MORE, AsShared());
00094     }
00095
00096     OnPhraseParsed.ExecuteIfBound(InParseRecord);
00097
00098     return ParseChildren(InPhraseWordArray, InParseRecord);
00099 }
00100
00101 FParseResult FPhraseNode::ParsePhraseIfRequired(TArray<FString>& InPhraseWordArray, FParseRecord&
    InParseRecord)
00102 {
00103     if (RequiresPhrase(InPhraseWordArray.Last()))
00104     {
00105         return ParsePhrase(InPhraseWordArray, InParseRecord);
00106     }
00107
00108     return FParseResult(PHRASE_UNABLE_TO_PARSE);
00109 }
00110
00111 bool FPhraseNode::CanBindChild(TPhraseNode& InNode)
00112 {
00113     for (auto& ChildNode : ChildNodes)
00114     {
00115         if (ChildNode->RequiresPhrase(InNode->BoundPhrase) || ChildNode->IsLeafNode())
00116         {
00117             return false;
00118         }
00119     }
00120
00121     return true;
00122 }
00123
00124 bool FPhraseNode::BindChildNode(TPhraseNode InNode)
00125 {
00126     if (!InNode.IsValid())
00127         return false;
00128
00129     for (auto& ChildNode : ChildNodes)
00130     {
00131         if (ChildNode->RequiresPhrase(InNode->BoundPhrase))
00132         {
00133             return ChildNode->BindChildrenNodes(InNode->ChildNodes);
00134         }
00135         else
00136         {

```



```

00137         ChildNodes.AddUnique(ChildNode);
00138         return true;
00139     }
00140 }
00141
00142     return false;
00143 }
00144
00145 bool FPhraseNode::BindChildNodeForce(TPhraseNode InNode)
00146 {
00147     ChildNodes.AddUnique(InNode);
00148
00149     return true;
00150 }
00151
00152 bool FPhraseNode::BindChildrenNodes(TPhraseNodeArray InNodes)
00153 {
00154     for (auto& InNode : InNodes)
00155     {
00156         for (auto& ChildNode : ChildNodes)
00157         {
00158             if (ChildNode->RequiresPhrase(InNode->BoundPhrase))
00159             {
00160                 return ChildNode->BindChildrenNodes(InNode->ChildNodes);
00161             }
00162             else
00163             {
00164                 ChildNodes.AddUnique(ChildNode);
00165                 return true;
00166             }
00167         }
00168     }
00169
00170     return false;
00171 }
00172
00173 bool FPhraseNode::BindChildrenNodesForce(TPhraseNodeArray InNodes)
00174 {
00175     for (auto& InNode : InNodes)
00176     {
00177         ChildNodes.AddUnique(InNode);
00178     }
00179
00180     return true;
00181 }
00182
00183 bool FPhraseNode::HasLeafChild()
00184 {
00185     return ChildNodes.Num() == 1 && ChildNodes[0]->IsLeafNode();
00186 }
00187
00188 FParseResult FPhraseNode::ParseChildren(TArray<FString>& InPhraseArray, FParseRecord& InParseRecord)
00189 {
00190     if (HasLeafChild())
00191         return ChildNodes[0]->ParsePhrase(InPhraseArray, InParseRecord);
00192     if (InPhraseArray.IsEmpty())
00193         return FParseResult(PHRASE_REQUIRES_MORE, AsShared());
00194
00195     // Below Can Be Optimized.
00196     // Maybe bypass the loop if Distance == 0 and Sort ChildNodes with Derived PhraseNodes Last?
00197
00198     int FoundChildIndex = -1;
00199     {
00200         int32 FoundChildDistance = INT32_MAX, CurrentDistance = INT32_MAX;
00201
00202         for (int i = 0; i < ChildNodes.Num(); i++)
00203         {
00204             // Child Nodes Require Unique Phrases to Siblings.
00205             if (ChildNodes[i]->RequiresPhrase(InPhraseArray.Last(), CurrentDistance))
00206             {
00207                 if (FoundChildDistance > CurrentDistance)
00208                 {
00209                     FoundChildIndex = i;
00210                     FoundChildDistance = CurrentDistance;
00211                 }
00212             }
00213         }
00214     }
00215
00216     if (FoundChildIndex != -1)
00217     {
00218         return ChildNodes[FoundChildIndex]->ParsePhrase(InPhraseArray, InParseRecord);
00219     }
00220
00221     /*else if (!InPhraseArray.IsEmpty())
00222     {
00223         return FParseResult(PHRASE_REQUIRES_MORE_CORRECT_PHRASES, AsShared());
00224     }

```

```

00224     */
00225
00226     return FParseResult (PHRASE_UNABLE_TO_PARSE, AsShared());
00227 }

```

## 5.62 PhraseStringInputNode.cpp

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #include "PhraseTree/PhraseStringInputNode.h"
00004
00005 #include "PhraseTree/Containers/Input/UParseStringInput.h"
00006
00007 FPhraseStringInputNode::FPhraseStringInputNode(const TCHAR* InInputString)
00008     : FPhraseInputNode(InInputString)
00009 {
00010
00011 };
00012
00013 FPhraseStringInputNode::FPhraseStringInputNode(const TCHAR* InInputString, TPhraseNodeArray
    InChildNodes)
00014     : FPhraseInputNode(InInputString, InChildNodes)
00015 {
00016
00017 }
00018
00019 FPhraseStringInputNode::FPhraseStringInputNode(const TCHAR* InInputString,
    TDelegate<void(FParseRecord& Record)> InOnPhraseParse, TPhraseNodeArray InChildNodes)
00020     : FPhraseInputNode(InInputString, InOnPhraseParse, InChildNodes)
00021 {
00022
00023 }
00024
00025 FPhraseStringInputNode::FPhraseStringInputNode(const TCHAR* InInputString, TPhraseNodeArray
    InChildNodes, TDelegate<void(FString Input)> InOnInputRecieved)
00026     : FPhraseInputNode(InInputString, InChildNodes, InOnInputRecieved)
00027 {
00028
00029 }
00030
00031 FPhraseStringInputNode::~FPhraseStringInputNode()
00032 {
00033
00034 }
00035
00036 bool FPhraseStringInputNode::MeetsInputRequirements(const FString& InPhrase)
00037 {
00038     if (InPhrase.IsEmpty())
00039         return false;
00040     else return true;
00041 }
00042
00043 bool FPhraseStringInputNode::RecordInput(const FString& InInput, FParseRecord& OutParseRecord)
00044 {
00045     if (InInput.IsEmpty())
00046         return false;
00047
00048     UParseStringInput* ParseInput = MakeParseInput<UParseStringInput>();
00049     ParseInput->SetValue(InInput);
00050
00051     OutParseRecord.AddPhraseInput(BoundPhrase, ParseInput);
00052
00053     OnInputReceived.ExecuteIfBound(InInput);
00054
00055     return true;
00056 }

```

## 5.63 Utils.cpp

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #include "PhraseTree/Utils.h"
00004 #include "OpenAccessibilityComLogging.h"
00005
00006
00007 bool NumericParser::IsValidNumeric(const FString& StringToCheck, bool ConvertToUpper)
00008 {
00009     return StringMappings.Contains(ConvertToUpper ? StringToCheck.ToUpper() : StringToCheck);
00010 }

```

```

00011
00012 void NumericParser::StringToNumeric(FString& NumericString, bool ConvertToUpper)
00013 {
00014     if (const FString* FoundMapping = StringMappings.Find(NumericString))
00015     {
00016         NumericString = ConvertToUpper ? *FoundMapping->ToUpper() : *FoundMapping;
00017     }
00018     else UE_LOG(LogOpenAccessibilityCom, Warning, TEXT("|| Numeric Parser || No Mapping Found for
String: %s ||"), *NumericString);
00019 }
00020
00021 const TMap<const FString, const FString> NumericParser::StringMappings = TMap<const FString, const
FString>
00022 {
00023     { TEXT("ZERO"), TEXT("0") },
00024     { TEXT("ONE"), TEXT("1") },
00025     { TEXT("TWO"), TEXT("2") },
00026     { TEXT("TOO"), TEXT("2") },
00027     { TEXT("TO"), TEXT("2") },
00028     { TEXT("THREE"), TEXT("3") },
00029     { TEXT("FOUR"), TEXT("4") },
00030     { TEXT("FOR"), TEXT("4") },
00031     { TEXT("FIVE"), TEXT("5") },
00032     { TEXT("SIX"), TEXT("6") },
00033     { TEXT("SEVEN"), TEXT("7") },
00034     { TEXT("EIGHT"), TEXT("8") },
00035     { TEXT("NINE"), TEXT("9") },
00036     { TEXT("TEN"), TEXT("10") },
00037     { TEXT("TIN"), TEXT("10") },
00038     { TEXT("ELEVEN"), TEXT("11") },
00039     { TEXT("TWELVE"), TEXT("12") },
00040     { TEXT("THIRTEEN"), TEXT("13") },
00041     { TEXT("FOURTEEN"), TEXT("14") },
00042     { TEXT("FIFTEEN"), TEXT("15") },
00043     { TEXT("SIXTEEN"), TEXT("16") },
00044     { TEXT("SEVENTEEN"), TEXT("17") },
00045     { TEXT("EIGHTEEN"), TEXT("18") },
00046     { TEXT("NINETEEN"), TEXT("19") },
00047     { TEXT("TWENTY"), TEXT("20") },
00048     { TEXT("THIRTY"), TEXT("30") },
00049     { TEXT("FORTY"), TEXT("40") },
00050     { TEXT("FIFTY"), TEXT("50") },
00051     { TEXT("SIXTY"), TEXT("60") },
00052     { TEXT("SEVENTY"), TEXT("70") },
00053     { TEXT("EIGHTY"), TEXT("80") },
00054     { TEXT("NINETY"), TEXT("90") },
00055     { TEXT("HUNDRED"), TEXT("100") },
00056 };

```

## 5.64 PhraseTreeUtils.cpp

```

00001 #include "PhraseTreeUtils.h"
00002
00003 #include "OpenAccessibilityComLogging.h"
00004
00005 UPhraseTreeUtils::UPhraseTreeUtils()
00006 {
00007
00008 }
00009
00010 UPhraseTreeUtils::~UPhraseTreeUtils()
00011 {
00012
00013 }
00014
00015 void UPhraseTreeUtils::RegisterFunctionLibrary(UPhraseTreeFunctionLibrary* LibraryToRegister)
00016 {
00017     TSharedPtr<FPhraseTree> PhraseTreeSP = PhraseTree.Pin();
00018     if (!PhraseTreeSP.IsValid())
00019     {
00020         UE_LOG(LogOpenAccessibilityCom, Warning, TEXT("Cannot Register Phrase Tree Function Library
Due To Invalid Phrase Tree Reference.));
00021         return;
00022     }
00023
00024     // For some reason this needs to be told directly to be kept alive,
00025     // even though it is a UPROPERTY TArray and should be kept alive by the UObject system.
00026     LibraryToRegister->AddToRoot();
00027     LibraryToRegister->BindBranches(PhraseTreeSP.ToSharedRef());
00028 }

```

## 5.65 SocketCommunicationServer.cpp

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #include "SocketCommunicationServer.h"
00004 #include "OpenAccessibilityComLogging.h"
00005
00006 #include "Serialization/JsonSerializer.h"
00007
00008 FSocketCommunicationServer::FSocketCommunicationServer(const std::string SendAddress, std::string
    RecvAddress, const int PollTimeout)
00009     : SendAddress(SendAddress), RecvAddress(RecvAddress), PollTimeout(PollTimeout)
00010 {
00011     Context = new zmq::context_t(1);
00012     if (Context == nullptr)
00013     {
00014         UE_LOG(LogOpenAccessibilityCom, Error, TEXT("Failed to create ZMQ context"));
00015         return;
00016     }
00017
00018     SendSocket = new zmq::socket_t(*Context, ZMQ_PUSH);
00019     if (SendSocket == nullptr)
00020     {
00021         UE_LOG(LogOpenAccessibilityCom, Error, TEXT("Failed to create ZMQ socket"));
00022         return;
00023     }
00024
00025     RecvSocket = new zmq::socket_t(*Context, ZMQ_PULL);
00026     if (RecvSocket == nullptr)
00027     {
00028         UE_LOG(LogOpenAccessibilityCom, Error, TEXT("Failed to create ZMQ socket"));
00029         return;
00030     }
00031
00032     Poller = new zmq::poller_t<int>();
00033     if (Poller == nullptr)
00034     {
00035         UE_LOG(LogOpenAccessibilityCom, Error, TEXT("Failed to create ZMQ poller"));
00036         return;
00037     }
00038
00039     SendSocket->connect(SendAddress);
00040     RecvSocket->bind(RecvAddress);
00041
00042     Poller->add(*RecvSocket, zmq::event_flags::pollin);
00043 }
00044
00045 FSocketCommunicationServer::~FSocketCommunicationServer()
00046 {
00047     Poller->remove(*RecvSocket);
00048     delete Poller; Poller = nullptr;
00049
00050     SendSocket->disconnect(SendAddress);
00051     SendSocket->close();
00052     delete SendSocket; SendSocket = nullptr;
00053
00054     RecvSocket->unbind(RecvAddress);
00055     RecvSocket->close();
00056     delete RecvSocket; RecvSocket = nullptr;
00057
00058     Context->shutdown();
00059     Context->close();
00060     delete Context; Context = nullptr;
00061 }
00062
00063 bool FSocketCommunicationServer::EventOccured()
00064 {
00065     std::vector<zmq::poller_event<int>> PollEvents(1);
00066     if (Poller->wait_all(PollEvents, std::chrono::milliseconds(PollTimeout)) > 0)
00067     {
00068         PollEvents.clear();
00069         return true;
00070     }
00071
00072     PollEvents.clear();
00073     return false;
00074 }
00075
00076 bool FSocketCommunicationServer::SendArrayBuffer(const float* MessageData, size_t Size, ComSendFlags
    SendFlags)
00077 {
00078     auto Result = SendSocket->send(zmq::const_buffer(MessageData, Size * sizeof(float)), SendFlags);
00079     if (Result.has_value())
00080     {
00081         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Com Server: Sent Array || Sent %d of %d bytes"),
            Result.value(), Size * sizeof(float));
00082         return true;

```

```

00083     }
00084     else if (zmq_errno() == EAGAIN)
00085     {
00086         UE_LOG(LogOpenAccessibilityCom, Warning, TEXT("|| Com Server: Sent Array || EAGAIN Error
Occured ||"));
00087         return true;
00088     }
00089
00090     return false;
00091 }
00092
00093 bool FSocketCommunicationServer::SendArrayBuffer(const float MessageData[], ComSendFlags SendFlags)
00094 {
00095     auto Result = SendSocket->send(zmq::const_buffer(MessageData, sizeof MessageData), SendFlags);
00096     if (Result.has_value())
00097     {
00098         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Com Server: Sent Array || Sent %d of %d bytes"),
Result.value(), int(sizeof MessageData));
00099         return true;
00100     }
00101     else if (zmq_errno() == EAGAIN)
00102     {
00103         UE_LOG(LogOpenAccessibilityCom, Warning, TEXT("|| Com Server: Sent Array || EAGAIN Error
Occured ||"));
00104         return true;
00105     }
00106
00107     return false;
00108 }
00109
00110 bool FSocketCommunicationServer::SendArrayBuffer(const TArray<float>& ArrayMessage, ComSendFlags
SendFlag)
00111 {
00112     auto Result = SendSocket->send(zmq::const_buffer(ArrayMessage.GetData(), ArrayMessage.Num() *
sizeof(float)), SendFlag);
00113     if (Result.has_value())
00114     {
00115         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Com Server: Sent Array || Sent %d of %d bytes"),
Result.value(), int(ArrayMessage.Num() * sizeof(float)));
00116         return true;
00117     }
00118     else if (zmq_errno() == EAGAIN)
00119     {
00120         UE_LOG(LogOpenAccessibilityCom, Warning, TEXT("|| Com Server: Sent Array || EAGAIN Error
Occured ||"));
00121         return true;
00122     }
00123
00124     return false;
00125 }
00126
00127 bool FSocketCommunicationServer::SendArrayMessage(const float* MessageData, size_t Size, ComSendFlags
SendFlags)
00128 {
00129     auto Result = SendSocket->send(zmq::message_t(MessageData, Size * sizeof(float)), SendFlags);
00130     if (Result.has_value())
00131     {
00132         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Com Server: Sent Array || Sent %d of %d bytes"),
Result.value(), Size * sizeof(float));
00133         return true;
00134     }
00135     else if (zmq_errno() == EAGAIN)
00136     {
00137         UE_LOG(LogOpenAccessibilityCom, Warning, TEXT("|| Com Server: Sent Array || EAGAIN Error
Occured ||"));
00138         return true;
00139     }
00140
00141     return false;
00142 }
00143
00144 bool FSocketCommunicationServer::SendArrayMessage(const float MessageData[], ComSendFlags SendFlags)
00145 {
00146     auto Result = SendSocket->send(zmq::message_t(MessageData, sizeof MessageData), SendFlags);
00147     if (Result.has_value())
00148     {
00149         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Com Server: Sent Array || Sent %d of %d bytes"),
Result.value(), int(sizeof MessageData));
00150         return true;
00151     }
00152     else if (zmq_errno() == EAGAIN)
00153     {
00154         UE_LOG(LogOpenAccessibilityCom, Warning, TEXT("|| Com Server: Sent Array || EAGAIN Error
Occured ||"));
00155         return true;
00156     }
00157

```

```

00158     return false;
00159 }
00160
00161 bool FSocketCommunicationServer::SendArrayMessage(const TArray<float>& ArrayMessage, ComSendFlags
    SendFlags)
00162 {
00163     auto Result = SendSocket->send(zmq::message_t(ArrayMessage.GetData(), ArrayMessage.Num() *
    sizeof(float)), SendFlags);
00164     if (Result.has_value())
00165     {
00166         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Com Server: Sent Array || Sent %d of %d bytes"),
    Result.value(), int(ArrayMessage.Num() * sizeof(float)));
00167         return true;
00168     }
00169     else if (zmq_errno() == EAGAIN)
00170     {
00171         UE_LOG(LogOpenAccessibilityCom, Warning, TEXT("|| Com Server: Sent Array || EAGAIN Error
    Occured ||"));
00172         return true;
00173     }
00174 }
00175 return false;
00176 }
00177
00178 bool FSocketCommunicationServer::SendArrayMessageWithMeta(const float* MessageData, size_t Size, const
    TSharedRef<FJsonObject>& Metadata, ComSendFlags SendFlags)
00179 {
00180     FString MetaDataString;
00181     if (!SerializeJSON(Metadata, MetaDataString))
00182     {
00183         UE_LOG(LogOpenAccessibilityCom, Error, TEXT("|| Com Server: Sent Array || Failed to serialize
    metadata ||"));
00184         return false;
00185     }
00186
00187     std::vector<zmq::message_t> Messages;
00188     Messages.push_back(zmq::message_t(*MetaDataString, MetaDataString.Len() * sizeof(TCHAR)));
00189     Messages.push_back(zmq::message_t(MessageData, Size * sizeof(float)));
00190
00191     auto Result = zmq::send_multipart(*SendSocket, Messages, SendFlags);
00192
00193     if (Result.has_value())
00194     {
00195         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Com Server: Sent Array || Sent %d of %d bytes"),
    Result.value(), Size * sizeof(float));
00196         return true;
00197     }
00198     else if (zmq_errno() == EAGAIN)
00199     {
00200         UE_LOG(LogOpenAccessibilityCom, Warning, TEXT("|| Com Server: Sent Array || EAGAIN Error
    Occured ||"));
00201         return true;
00202     }
00203 }
00204 return false;
00205 }
00206
00207 bool FSocketCommunicationServer::SendArrayMessageWithMeta(const float MessageData[], const
    TSharedRef<FJsonObject>& Metadata, ComSendFlags SendFlags)
00208 {
00209     FString MetaDataString;
00210     if (!SerializeJSON(Metadata, MetaDataString))
00211     {
00212         UE_LOG(LogOpenAccessibilityCom, Error, TEXT("|| Com Server: Sent Array || Failed to serialize
    metadata ||"));
00213         return false;
00214     }
00215
00216     std::vector<zmq::message_t> Messages;
00217     Messages.push_back(zmq::message_t(*MetaDataString, MetaDataString.Len() * sizeof(TCHAR)));
00218     Messages.push_back(zmq::message_t(MessageData, sizeof MessageData));
00219
00220     auto Result = zmq::send_multipart(*SendSocket, Messages, SendFlags);
00221     if (Result.has_value())
00222     {
00223         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Com Server: Sent Array || Sent %d of %d bytes"),
    Result.value(), int(sizeof MessageData));
00224         return true;
00225     }
00226     else if (zmq_errno() == EAGAIN)
00227     {
00228         UE_LOG(LogOpenAccessibilityCom, Warning, TEXT("|| Com Server: Sent Array || EAGAIN Error
    Occured ||"));
00229         return true;
00230     }
00231 }
00232

```

```

00233     return false;
00234 }
00235
00236 bool FSocketCommunicationServer::SendArrayMessageWithMeta(const TArray<float>& ArrayMessage, const
    TSharedRef<FJsonObject>& Metadata, ComSendFlags SendFlags)
00237 {
00238     FString MetaDataString;
00239     if (!SerializeJSON(Metadata, MetaDataString))
00240     {
00241         UE_LOG(LogOpenAccessibilityCom, Error, TEXT("|| Com Server: Sent Array || Failed to serialize
            metadata ||"));
00242         return false;
00243     }
00244
00245     std::vector<zmq::message_t> Messages;
00246     Messages.push_back(zmq::message_t(*MetaDataString, MetaDataString.Len() * sizeof(TCHAR)));
00247     Messages.push_back(zmq::message_t(ArrayMessage.GetData(), ArrayMessage.Num() * sizeof(float)));
00248
00249     auto Result = zmq::send_multipart(*SendSocket, Messages, SendFlags);
00250     if (Result.has_value())
00251     {
00252         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Com Server: Sent Array || Sent %d of %d
            Messages"), Result.value(), Messages.size());
00253
00254         return true;
00255     }
00256     else if (zmq_errno() == EAGAIN)
00257     {
00258         UE_LOG(LogOpenAccessibilityCom, Warning, TEXT("|| Com Server: Sent Array || EAGAIN Error
            Occured ||"));
00259
00260         return true;
00261     }
00262
00263     return false;
00264 }
00265
00266 bool FSocketCommunicationServer::SendStringBuffer(const std::string StringMessage, ComSendFlags
    SendFlags)
00267 {
00268     auto Result = SendSocket->send(zmq::const_buffer(StringMessage.c_str(), StringMessage.size()),
        SendFlags);
00269     if (Result.has_value())
00270     {
00271         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Com Server: Sent String || Sent %d of %d
            bytes"), Result.value(), StringMessage.size());
00272         return true;
00273     }
00274     else if (zmq_errno() == EAGAIN)
00275     {
00276         UE_LOG(LogOpenAccessibilityCom, Warning, TEXT("|| Com Server: Sent String || EAGAIN Error
            Occured ||"));
00277         return true;
00278     }
00279
00280     return false;
00281 }
00282
00283 bool FSocketCommunicationServer::SendJsonBuffer(const std::string JsonMessage, ComSendFlags SendFlags)
00284 {
00285     auto Result = SendSocket->send(zmq::const_buffer(JsonMessage.c_str(), JsonMessage.size()),
        SendFlags);
00286     if (Result.has_value())
00287     {
00288         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Com Server: Sent JSON || Sent %d of %d bytes"),
            Result.value(), JsonMessage.size());
00289         return true;
00290     }
00291     else if (zmq_errno() == EAGAIN)
00292     {
00293         UE_LOG(LogOpenAccessibilityCom, Warning, TEXT("|| Com Server: Sent JSON || EAGAIN Error
            Occured ||"));
00294         return true;
00295     }
00296
00297     return false;
00298 }
00299
00300
00301
00302 template <typename T>
00303 bool FSocketCommunicationServer::RecvArray(TArray<T>& OutArrayData, size_t Size, ComRecvFlags
    RecvFlags)
00304 {
00305     zmq::message_t RecvMessage;
00306
00307     auto Result = RecvSocket->recv(RecvMessage, RecvFlags);

```

```

00308         if (Result.has_value())
00309         {
00310             UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Com Server: Recv Array || Recv %d bytes"),
Result.value());
00311             OutArrayData.Append(RecvMessage.data<T>(), Result.value());
00312             return true;
00313         }
00314         else if (zmq_errno() == EAGAIN)
00315         {
00316             UE_LOG(LogOpenAccessibilityCom, Warning, TEXT("|| Com Server: Recv Array || EAGAIN Error
Occured ||"));
00317             return true;
00318         }
00319         return false;
00320     }
00321 }
00322 }
00323 }
00324 }
00325 bool FSocketCommunicationServer::RecvString(FString& OutStringMessage, ComRecvFlags RecvFlags)
00326 {
00327     zmq::message_t RecvMessage;
00328     auto Result = RecvSocket->recv(RecvMessage, RecvFlags);
00329     if (Result.has_value())
00330     {
00331         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Com Server: Recv String || Recv %d bytes"),
Result.value());
00332         OutStringMessage = FString(Result.value(), UTF8_TO_TCHAR(RecvMessage.data()));
00333         return true;
00334     }
00335     else if (zmq_errno() == EAGAIN)
00336     {
00337         UE_LOG(LogOpenAccessibilityCom, Warning, TEXT("|| Com Server: Recv String || EAGAIN Error
Occured ||"));
00338         return true;
00339     }
00340     return false;
00341 }
00342 }
00343 }
00344 }
00345 }
00346 }
00347 }
00348 bool FSocketCommunicationServer::RecvJson(FString& OutJsonMessage, ComRecvFlags RecvFlags)
00349 {
00350     zmq::message_t RecvMessage;
00351     auto Result = RecvSocket->recv(RecvMessage, RecvFlags);
00352     if (Result.has_value())
00353     {
00354         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Com Server: Recv JSON || Recv %d bytes"),
Result.value());
00355         OutJsonMessage = FString(Result.value(), UTF8_TO_TCHAR(RecvMessage.data()));
00356         return true;
00357     }
00358     else if (zmq_errno() == EAGAIN)
00359     {
00360         UE_LOG(LogOpenAccessibilityCom, Warning, TEXT("|| Com Server: Recv JSON || EAGAIN Error
Occured ||"));
00361         return true;
00362     }
00363     return false;
00364 }
00365 }
00366 }
00367 }
00368 }
00369 }
00370 bool FSocketCommunicationServer::RecvStringMultipart(TArray<FString>& OutMessages, ComRecvFlags
RecvFlags)
00371 {
00372     std::vector<zmq::message_t> RecvMessages;
00373     auto Result = zmq::recv_multipart(*RecvSocket, std::back_inserter(RecvMessages), RecvFlags);
00374     if (Result.has_value())
00375     {
00376         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Com Server: Recv Multipart || Recv %d
messages"), Result.value());
00377         for (auto& Message : RecvMessages)
00378         {
00379             OutMessages.Add(FString(Message.size(), UTF8_TO_TCHAR(Message.data())));
00380         }
00381         return true;
00382     }
00383     else if (zmq_errno() == EAGAIN)

```



```

00387     {
00388         UE_LOG(LogOpenAccessibilityCom, Warning, TEXT("|| Com Server: Recv Multipart || EAGAIN Error
Occured ||"));
00389         return true;
00390     }
00391
00392     return false;
00393 }
00394
00395 bool FSocketCommunicationServer::RecvStringMultipartWithMeta(TArray<FString>& OutMessages,
TSharedPtr<FJsonObject>& OutMetadata, ComRecvFlags RecvFlag)
00396 {
00397     std::vector<zmq::message_t> RecvMessages;
00398     if (!RecvMultipartWithMeta(RecvMessages, OutMetadata, RecvFlag))
00399         return false;
00400
00401     for (auto& Message : RecvMessages)
00402     {
00403         OutMessages.Add(FString(Message.size(), UTF8_TO_TCHAR(Message.data())));
00404     }
00405
00406     return true;
00407 }
00408
00409 bool FSocketCommunicationServer::RecvMultipartWithMeta(std::vector<zmq::message_t>&
OutMultipartMessages, TSharedPtr<FJsonObject>& OutMetadata, ComRecvFlags RecvFlags)
00410 {
00411     auto Result = zmq::recv_multipart(*RecvSocket, std::back_inserter(OutMultipartMessages),
RecvFlags);
00412     if (Result.has_value())
00413     {
00414         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Com Server: Recv Multipart || Recv %d
messages"), Result.value());
00415
00416         // Pop Metadata Messages from the Front of Array.
00417         zmq::message_t MetadataMessage = MoveTempIfPossible(OutMultipartMessages[0]);
00418         OutMultipartMessages.erase(OutMultipartMessages.begin());
00419
00420         if (DeserializeJSON(FString(UTF8_TO_TCHAR(MetadataMessage.data()), MetadataMessage.size()),
OutMetadata))
00421         {
00422             return true;
00423         }
00424         else
00425         {
00426             UE_LOG(LogOpenAccessibilityCom, Error, TEXT("|| Com Server: Recv Multipart || Failed to
deserialize metadata ||"));
00427             return false;
00428         }
00429     }
00430     else if (zmq_errno() == EAGAIN)
00431     {
00432         UE_LOG(LogOpenAccessibilityCom, Warning, TEXT("|| Com Server: Recv Multipart || EAGAIN Error
Occured ||"));
00433         return true;
00434     }
00435
00436     return false;
00437 }
00438
00439 bool FSocketCommunicationServer::SerializeJSON(const TSharedRef<FJsonObject>& InJsonObject, FString&
OutJsonString)
00440 {
00441     return FJsonSerializer::Serialize(InJsonObject,
TJsonWriterFactory<TCHAR>::Create(&OutJsonString));
00442 }
00443
00444 bool FSocketCommunicationServer::DeserializeJSON(const FString& InJsonString, TSharedPtr<FJsonObject>&
OutJsonObject)
00445 {
00446     return FJsonSerializer::Deserialize(TJsonReaderFactory<TCHAR>::Create(InJsonString),
OutJsonObject);
00447 }

```

## 5.66 UBAudioCapture.cpp

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003
00004 #include "UBAudioCapture.h"
00005
00006 UBAudioCapture::UBAudioCapture() : UAudioCapture()
00007 {

```

```

00008
00009 }
00010
00011 UBAudioCapture::~UBAudioCapture()
00012 {
00013 }
00014
00015 bool UBAudioCapture::OpenDefaultAudioStream(int32 OverrideSampleRate, int32 OverrideInputChannels)
00016 {
00017     if (!AudioCapture.IsStreamOpen())
00018     {
00019         if (!AudioCapture.IsStreamOpen())
00020         {
00021             Audio::FOnAudioCaptureFunction OnCapture = [this](const void* AudioData, int32 NumFrames,
00022 int32 InNumChannels, int32 InSampleRate, double StreamTime, bool bOverflow)
00023             {
00024                 OnGeneratedAudio((const float*)AudioData, NumFrames * InNumChannels);
00025             };
00026             // Start the stream here to avoid hitching the audio render thread.
00027             Audio::FAudioCaptureDeviceParams Params;
00028             if (OverrideSampleRate != NULL)
00029                 Params.SampleRate = OverrideSampleRate;
00030             if (OverrideInputChannels != NULL)
00031                 Params.NumInputChannels = OverrideInputChannels;
00032
00033             if (AudioCapture.OpenAudioCaptureStream(Params, MoveTemp(OnCapture), 1024))
00034             {
00035                 // If we opened the capture stream succesfully, get the capture device info and
00036                 initialize the UAudioGenerator
00037                 Audio::FCaptureDeviceInfo Info;
00038                 if (AudioCapture.GetCaptureDeviceInfo(Info))
00039                 {
00040                     Init(
00041                         OverrideSampleRate != NULL ? OverrideSampleRate : Info.PreferredSampleRate,
00042                         OverrideInputChannels != NULL ? OverrideInputChannels : Info.InputChannels
00043                     );
00044                     return true;
00045                 }
00046             }
00047         }
00048     }
00049     return false;
00050 }
00051
00052 return false;
00053 }
00054 }

```

## 5.67 AudioManager.h

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #pragma once
00004
00005 #include "CoreMinimal.h"
00006
00007 #include "AudioCapture.h"
00008 #include "Sound/SampleBufferIO.h"
00009 #include "Delegates/DelegateCombinations.h"
00010 #include "AudioDeviceNotificationSubsystem.h"
00011
00012 #include "AudioManager.generated.h"
00013
00014 USTRUCT()
00015 struct FAudioManagerSettings
00016 {
00017     GENERATED_BODY()
00018
00019 public:
00020     FAudioManagerSettings()
00021     {
00022         // Default Settings
00023         LevelThreshold = -2.5f;
00024         SaveName = FString("Captured_User_Audio");
00025         SavePath = FString("./OpenAccessibility/Audioclips/");
00026     }
00027
00028     // The Threshold for incoming audio to be considered as input.
00029     UPROPERTY(Config, EditAnywhere, Category = "OpenAccessibility/Audio Manager")
00030     float LevelThreshold;
00031 }

```

```

00035     UPROPERTY(Config, EditAnywhere, Category = "OpenAccessibility/Audio Manager")
00036     FString SaveName;
00037
00041     UPROPERTY(Config, EditAnywhere, Category = "OpenAccessibility/Audio Manager")
00042     FString SavePath;
00043 };
00044
00045
00049 UClass(BlueprintType, Blueprintable, Config = OpenAccessibility)
00050 class OPENACCESSIBILITYCOMMUNICATION_API UAudioManager : public UObject
00051 {
00052     GENERATED_BODY()
00053
00054 public:
00055     UAudioManager();
00056     virtual ~UAudioManager();
00057
00061     void StartCapturingAudio();
00062
00066     void StopCapturingAudio();
00067
00073     void PRIVATE_OnAudioGenerate(const float* InAudio, int32 NumSamples);
00074
00079     void SaveAudioBufferToWAV(const FString& FilePath);
00080
00085     bool IsCapturingAudio() const { return bIsCapturingAudio; }
00086
00091     int32 GetAudioCaptureSampleRate() const { return AudioCapture->GetSampleRate(); }
00092
00097     int32 GetAudioCaptureNumChannels() const { return AudioCapture->GetNumChannels(); }
00098
00105     void OnDefaultDeviceChanged(EAudioDeviceChangedRole ChangedRole, FString DeviceID);
00106
00107 private:
00108
00109     void RegisterAudioGenerator();
00110
00111     void UnregisterAudioGenerator();
00112
00113 public:
00114
00118     UPROPERTY(Config, EditAnywhere, Category = "OpenAccessibility/Audio Manager")
00119     FAudioManagerSettings Settings;
00120
00124     TDelegate<void(const TArray<float>>> OnAudioReadyForTranscription;
00125
00126 private:
00127
00128     // Audio Capture
00129     bool bIsCapturingAudio = false;
00130
00131     UPROPERTY(EditDefaultsOnly, Category = "OpenAccessibility/Audio Capture")
00132     class UAudioCapture* AudioCapture;
00133
00134     UPROPERTY(EditDefaultsOnly, Category = "OpenAccessibility/Audio Capture")
00135     TArray<float> AudioBuffer;
00136
00137     FAudioGeneratorHandle OnAudioGenerateHandle;
00138
00139     FDelegateHandle OnDefaultDeviceChangedHandle;
00140
00141     // Audio Saving
00142     Audio::FSoundWavePCMWriter* FileWriter;
00143 };

```

## 5.68 OpenAccessibilityComLogging.h

```

00001 // Copyright Epic Games, Inc. All Rights Reserved.
00002
00003 #pragma once
00004
00005 DECLARE_LOG_CATEGORY_EXTERN(LogOpenAccessibilityCom, Log, All);
00006
00007 DEFINE_LOG_CATEGORY(LogOpenAccessibilityCom);

```

## 5.69 OpenAccessibilityCommunication.h

```

00001 // Copyright Epic Games, Inc. All Rights Reserved.
00002

```

```

00003 #pragma once
00004
00005 #include "CoreMinimal.h"
00006 #include "Modules/ModuleManager.h"
00007 #include "Modules/ModuleInterface.h"
00008 #include "Delegates/DelegateCombinations.h"
00009
00010 #include "PhraseTree.h"
00011 #include "PhraseTreeUtils.h"
00012
00013 //UDELEGATE()
00014 //DECLARE_DYNAMIC_MULTICAST_DELEGATE_OneParam(FTranscriptionRecievedSignature, const TArray<FString>,
    InTranscription);
00015
00016 class FOpenAccessibilityCommunicationModule : public IModuleInterface
00017 {
00018
00019 public:
00020
00021     virtual void StartupModule() override;
00022     virtual void ShutdownModule() override;
00023
00024     virtual bool SupportsDynamicReloading() override
00025     {
00026         return false;
00027     }
00028
00031 static FOpenAccessibilityCommunicationModule& Get()
00032 {
00033     return
    FModuleManager::GetModuleChecked<FOpenAccessibilityCommunicationModule>("OpenAccessibilityCommunication");
00034 }
00035
00036 bool Tick(const float DeltaTime);
00037
00038 void HandleKeyDownEvent(const FKeyEvent& InKeyEvent);
00039
00044 void TranscribeWaveForm(TArray<float> AudioBufferToTranscribe);
00045
00046 private:
00047
00051 void BuildPhraseTree();
00052
00056 void RegisterConsoleCommands();
00057
00061 void UnregisterConsoleCommands();
00062
00066 void LoadZMQDLL();
00067
00071 void UnloadZMQDLL();
00072 public:
00073
00077 TMulticastDelegate<void(TArray<FString>)> OnTranscriptionRecieved;
00078
00082 class UAudioManager* AudioManager;
00083
00087 TSharedPtr<class FSocketCommunicationServer> SocketServer;
00088
00092 TSharedPtr<FPhraseTree> PhraseTree;
00093
00097 class UPhraseTreeUtils* PhraseTreeUtils;
00098
00099 private:
00100
00104 TArray<float> PrevAudioBuffer;
00105
00106 FTickerDelegate TickDelegate;
00107 FTSTicker::FDelegateHandle TickDelegateHandle;
00108
00109 FDelegateHandle PhraseTreePhraseRecievedHandle;
00110
00111 FDelegateHandle KeyDownEventHandle;
00112
00116 void* ZMQDllHandle;
00117
00118 TArray<IConsoleCommand*> ConsoleCommands;
00119 };

```

## 5.70 PhraseTree.h

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #pragma once
00004

```

```

00005 #include "CoreMinimal.h"
00006
00007 #include "PhraseTree/PhraseNode.h"
00008 #include "PhraseTree/Containers/ParseRecord.h"
00009 #include "PhraseTree/Containers/ContextObject.h"
00010
00011 enum EPhraseTreeBranchBindResult : uint8_t
00012 {
00016     BRANCH_NOT_BOUND,
00017
00021     BRANCH_BOUND,
00022     BRANCH_SPLIT
00023 };
00024
00025 struct OPENACCESSIBILITYCOMMUNICATION_API FPhraseTreeBranchBind
00026 {
00027     FPhraseTreeBranchBind()
00028     {
00029     }
00030 }
00031
00032 FPhraseTreeBranchBind(TPhraseNode InRootNode, TPhraseNode InBranchRoot)
00033 {
00034     StartNode = InRootNode;
00035     BranchRoot = InBranchRoot;
00036 }
00037
00038 ~FPhraseTreeBranchBind()
00039 {
00040     StartNode.Reset();
00041     BranchRoot.Reset();
00042 }
00043
00047 TPhraseNode StartNode;
00048
00052 TPhraseNode BranchRoot;
00053 };
00054
00055 struct OPENACCESSIBILITYCOMMUNICATION_API FPhraseTreeContextManager
00056 {
00057     friend class FPhraseTree;
00058
00059 public:
00060
00061     FPhraseTreeContextManager()
00062     {
00063     }
00064 }
00065
00066 ~FPhraseTreeContextManager()
00067 {
00068 }
00069
00070
00071 // Context Stack Management
00072
00076 void IsEmpty()
00077 {
00078     this->ContextObjectStack.IsEmpty();
00079 }
00080
00085 bool HasContextObjects()
00086 {
00087     return this->ContextObjectStack.Num() > 0;
00088 }
00089
00095 bool HasContextObject(UPhraseTreeContextObject* InContextObject)
00096 {
00097     return this->ContextObjectStack.Contains(InContextObject);
00098 }
00099
00104 TArray<UPhraseTreeContextObject*> GetContextStack()
00105 {
00106     return this->ContextObjectStack;
00107 }
00108
00109 // Context Stack Ammendments
00110
00115 void PeekContextObject(UPhraseTreeContextObject* OutContextObject)
00116 {
00117     OutContextObject = this->ContextObjectStack.Top();
00118 }
00119
00124 UPhraseTreeContextObject* PeekContextObject()
00125 {
00126     return this->ContextObjectStack.Top();
00127 }

```

```

00128
00133 void PushContextObject (UPhraseTreeContextObject* InContextObject)
00134 {
00135     this->ContextObjectStack.Push(InContextObject);
00136 }
00137
00141 void PopContextObject ()
00142 {
00143     this->ContextObjectStack.Pop();
00144 }
00145
00151 template<class CastToContextType>
00152 void PopContextObject (CastToContextType* OutContextObject)
00153 {
00154     OutContextObject = Cast<CastToContextType>(this->ContextObjectStack.Pop());
00155 }
00156
00161 void PopContextObject (UPhraseTreeContextObject* OutContextObject)
00162 {
00163     OutContextObject = this->ContextObjectStack.Pop();
00164 }
00165
00166 private:
00167
00172 void UpdateContextStack (TArray<UPhraseTreeContextObject*> InContextObjectStack)
00173 {
00174     this->ContextObjectStack = InContextObjectStack;
00175
00176     FilterContextStack();
00177 }
00178
00179 // Context Stack Filtering
00180
00184 void FilterContextStack ()
00185 {
00186     bool bRemoveDerivedContextObjects = false;
00187
00188     int i = this->ContextObjectStack.Num() - 1;
00189     if (i < 0)
00190         return;
00191
00192     UPhraseTreeContextObject* CurrObj = nullptr;
00193
00194     do
00195     {
00196         CurrObj = this->ContextObjectStack[i];
00197
00198         if (CurrObj != nullptr && CurrObj->GetIsActive())
00199         {
00200             i--;
00201             continue;
00202         }
00203
00204         if (CurrObj->IsValidLowLevel())
00205         {
00206             CurrObj->RemoveFromRoot();
00207             CurrObj->MarkAsGarbage();
00208         }
00209
00210         this->ContextObjectStack.RemoveAt(i);
00211         i--;
00212     } while (i > 0);
00213
00214     CurrObj = nullptr;
00215 }
00216
00217 private:
00218
00219     TArray<UPhraseTreeContextObject*> ContextObjectStack;
00220 };
00221
00223
00227 class OPENACCESSIBILITYCOMMUNICATION_API FPhraseTree : public FPhraseNode
00228 {
00229 public:
00230     FPhraseTree();
00231     virtual ~FPhraseTree();
00232
00233     FPhraseTreeContextManager& GetContextManager() {
00234         return ContextManager;
00235     }
00236
00237     bool Tick(float DeltaTime);
00238
00239     // FPhraseNode Implementation
00240     virtual FParseResult ParsePhrase(TArray<FString>& InPhraseWordArray, FParseRecord& InParseRecord)

```

```

00241     override;
00242     // End FPhaseNode Implementation
00243
00244     void BindBranch(const TPhraseNode& InNode);
00245
00246     void BindBranches(const TPhraseNodeArray& InNodes);
00247
00248     void ParseTranscription(TArray<FString> InTranscriptionSegments);
00249
00250 private:
00251
00252     TSharedPtr<FPhraseNode> LastVistedNode;
00253
00254     FParseRecord LastVistedParseRecord;
00255
00256     FPhraseTreeContextManager ContextManager;
00257
00258     FTSTicker::FDelegateHandle TickDelegateHandle;
00259 };

```

## 5.71 ContextMenuObject.h

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #pragma once
00004
00005 #include "CoreMinimal.h"
00006
00007 #include "Framework/Application/IMenu.h"
00008
00009 #include "PhraseTree/Containers/ContextObject.h"
00010
00011 #include "ContextMenuObject.generated.h"
00012
00013 UCLASS()
00014 class OPENACCESSIBILITYCOMMUNICATION_API UPhraseTreeContextMenuObject : public
    UPhraseTreeContextObject
00015 {
00016     GENERATED_BODY()
00017
00018 public:
00019
00020     UPhraseTreeContextMenuObject();
00021     UPhraseTreeContextMenuObject(TSharedPtr<IMenu> Menu);
00022
00023     virtual ~UPhraseTreeContextMenuObject();
00024
00025     virtual void Init(TSharedPtr<IMenu> InMenu);
00026
00027     virtual void Init(TSharedPtr<IMenu> InMenu, TSharedPtr<FPhraseNode> InContextRoot);
00028
00029     virtual bool Tick(float DeltaTime) { return true; };
00030
00031     virtual bool Close() override
00032     {
00033         RemoveTickDelegate();
00034         Menu.Pin()->Dismiss();
00035
00036         return true;
00037     };
00038
00039     void BindTickDelegate();
00040
00041     void RemoveTickDelegate();
00042
00043     void BindMenuDismissed(TSharedPtr<IMenu> InMenu);
00044
00045     void RemoveMenuDismissed(TSharedPtr<IMenu> InMenu);
00046
00047     virtual void SetMenu(TSharedPtr<IMenu> InMenu)
00048     {
00049         Menu = InMenu;
00050     }
00051
00052     virtual void ScaleMenu(const float ScaleFactor) {};
00053
00054 protected:
00055
00056     TSharedPtr<SWindow> GetWindow()
00057     {
00058         return Menu.Pin()->GetOwnedWindow();
00059     }
00060
00061

```

```

00104     void OnMenuDismissed(TSharedRef<IMenu> Menu);
00105
00106 public:
00107
00111     TWeakPtr<IMenu> Menu;
00112
00116     TWeakPtr<SWindow> Window;
00117
00118 private:
00119
00120     // Ticker Components
00121
00122     FTickerDelegate TickDelegate;
00123     FTSTicker::FDelegateHandle TickDelegateHandle;
00124
00125     FDelegateHandle MenuDismissedHandle;
00126 };

```

## 5.72 ContextObject.h

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #pragma once
00004
00005 #include "CoreMinimal.h"
00006
00007 #include "ContextObject.generated.h"
00008
00009 class FPhraseNode;
00010
00011 UCLASS(Abstract)
00012 class OPENACCESSIBILITYCOMMUNICATION_API UPhraseTreeContextObject : public UObject
00013 {
00014     GENERATED_BODY()
00015
00016 public:
00017
00018     UPhraseTreeContextObject()
00019         : UObject()
00020     {
00021
00022     }
00023
00024     virtual ~UPhraseTreeContextObject()
00025     {
00026
00027     }
00028
00029     virtual bool Close() { return true; }
00030
00035     void SetContextRootNode(TSharedRef<FPhraseNode> InRootNode)
00036     {
00037         ContextRoot = InRootNode;
00038     }
00039
00044     TSharedPtr<FPhraseNode> GetContextRoot()
00045     {
00046         return ContextRoot.Pin();
00047     }
00048
00053     const bool GetIsActive()
00054     {
00055         return bIsActive;
00056     }
00057
00058 protected:
00059
00063     bool bIsActive = true;
00064
00069     TWeakPtr<FPhraseNode> ContextRoot;
00070 };

```

## 5.73 InputContainers.h

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #pragma once
00004
00005 #include "CoreMinimal.h"

```



```

00006
00007 UENUM()
00008 enum class EPhrasePositionalInput : uint8
00009 {
00010     TOP,
00011     MIDDLE,
00012     BOTTOM,
00013     LEFT,
00014     RIGHT,
00015     CENTER
00016 };
00017
00018 UENUM()
00019 enum class EPhraseDirectionalInput : int8
00020 {
00021     UP,
00022     DOWN,
00023     LEFT,
00024     RIGHT,
00025     FORWARD,
00026     BACKWARD
00027 };
00028
00029 UENUM()
00030 enum class EPhrase2DDirectionalInput : int8
00031 {
00032     UP = EPhraseDirectionalInput::UP,
00033     DOWN = EPhraseDirectionalInput::DOWN,
00034     LEFT = EPhraseDirectionalInput::LEFT,
00035     RIGHT = EPhraseDirectionalInput::RIGHT,
00036 };
00037
00038 UENUM()
00039 enum class EPhraseScrollInput : uint8
00040 {
00041     UP, // 0
00042     DOWN, // 1
00043     TOP, // 2
00044     BOTTOM // 3
00045 };

```

## 5.74 UParseEnumInput.h

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #pragma once
00004
00005 #include "CoreMinimal.h"
00006 #include "UParseIntInput.h"
00007
00008 #include "UParseEnumInput.generated.h"
00009
00010 UCLASS()
00011 class OPENACCESSIBILITYCOMMUNICATION_API UParseEnumInput : public UParseIntInput
00012 {
00013     GENERATED_BODY()
00014
00015 public:
00016
00017     UParseEnumInput() = default;
00018     virtual ~UParseEnumInput()
00019     {
00020         delete EnumType;
00021     };
00022
00027     void SetEnumType(UEnum* InEnumType)
00028     {
00029         EnumType = InEnumType;
00030     }
00031
00036     void GetEnumType(UEnum*& OutEnumType)
00037     {
00038         OutEnumType = EnumType;
00039     }
00040
00045     UEnum* GetEnumType()
00046     {
00047         return EnumType;
00048     }
00049
00050 protected:
00051
00052     UPROPERTY()

```

```

00053     UEnum* EnumType;
00054
00055 };

```

## 5.75 UParseInput.h

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #pragma once
00004
00005 #include "CoreMinimal.h"
00006 #include "UObject/Object.h"
00007
00008 #include "UParseInput.generated.h"
00009
00010 UCLASS()
00011 class OPENACCESSIBILITYCOMMUNICATION_API UParseInput : public UObject
00012 {
00013     GENERATED_BODY()
00014
00015 public:
00016
00017     UParseInput() = default;
00018     virtual ~UParseInput()
00019     {
00020
00021     };
00022 };
00023
00024 // Input Constructor Functions
00025
00031 template<class ParseInputType>
00032 [[nodiscard]] FORCEINLINE ParseInputType* MakeParseInput()
00033 {
00034     ParseInputType* NewObj = NewObject<ParseInputType>();
00035     NewObj->AddToRoot();
00036
00037     return NewObj;
00038 }

```

## 5.76 UParseIntInput.h

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #pragma once
00004
00005 #include "CoreMinimal.h"
00006 #include "UParseInput.h"
00007
00008 #include "UParseIntInput.generated.h"
00009
00010 UCLASS()
00011 class OPENACCESSIBILITYCOMMUNICATION_API UParseIntInput : public UParseInput
00012 {
00013     GENERATED_BODY()
00014
00015 public:
00016
00017     UParseIntInput() = default;
00018     virtual ~UParseIntInput()
00019     {
00020
00021     };
00022
00027     void SetValue(int32 InValue)
00028     {
00029         Value = InValue;
00030     }
00031
00036     void GetValue(int32& OutValue)
00037     {
00038         OutValue = Value;
00039     }
00040
00045     int32 GetValue()
00046     {
00047         return Value;
00048     }
00049

```

```

00050 protected:
00051
00052     int32 Value;
00053
00054 };

```

## 5.77 UParseStringInput.h

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #pragma once
00004
00005 #include "CoreMinimal.h"
00006 #include "UParseInput.h"
00007
00008 #include "UParseStringInput.generated.h"
00009
00010 UCLASS()
00011 class OPENACCESSIBILITYCOMMUNICATION_API UParseStringInput : public UParseInput
00012 {
00013     GENERATED_BODY()
00014
00015 public:
00016
00017     UParseStringInput() = default;
00018     virtual ~UParseStringInput()
00019     {
00020
00021     };
00022
00027     void SetValue(FString InValue)
00028     {
00029         Value = InValue;
00030     }
00031
00036     void GetValue(FString& OutValue)
00037     {
00038         OutValue = Value;
00039     }
00040
00045     FString GetValue()
00046     {
00047         return Value;
00048     }
00049
00050 protected:
00051
00052     FString Value;
00053 };

```

## 5.78 ParseRecord.h

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #pragma once
00004
00005 #include "CoreMinimal.h"
00006
00007 #include "Input/UParseInput.h"
00008 #include "PhraseTree/Containers/ContextObject.h"
00009
00010 #include "ParseRecord.generated.h"
00011
00015 USTRUCT(BlueprintType)
00016 struct OPENACCESSIBILITYCOMMUNICATION_API FParseRecord
00017 {
00018     GENERATED_BODY()
00019
00020 public:
00021     friend class FPhraseTree;
00022
00023     FParseRecord()
00024     {
00025         PhraseInputs = TMultiMap<FString, UParseInput*>();
00026         ContextObjectStack = TArray<UPhraseTreeContextObject*>();
00027     }
00028
00029     FParseRecord(TArray<UPhraseTreeContextObject*> InContextObjects)
00030     {

```

```

00031     PhraseInputs = TMultiMap<FString, UParseInput*>();
00032     ContextObjectStack = InContextObjects;
00033 }
00034
00035 ~FParseRecord()
00036 {
00037     PhraseInputs.Empty();
00038 }
00039
00040 // -- Phrase String
00041
00042 FString GetPhraseString() const
00043 {
00044     return FString::Join(PhraseRecord, TEXT(" "));
00045 }
00046
00047 void AddPhraseString(FString StringToRecord)
00048 {
00049     PhraseRecord.Add(StringToRecord);
00050 }
00051
00052 // --
00053
00054 UParseInput* GetPhraseInput(const FString& InString)
00055 {
00056     // Check If The Phrase Exits
00057     // This Error Will Be Thrown If: InString Is In Correct (Requires UpperCase) or The Phrase
00058     Does Not Exist.
00059     check(PhraseInputs.Contains(InString))
00060
00061     return *PhraseInputs.Find(InString);
00062 }
00063
00064 template<class CastToType>
00065 CastToType* GetPhraseInput(const FString& InString)
00066 {
00067     // Check If The Phrase Exits
00068     // This Error Will Be Thrown If: InString Is In Correct (Requires UpperCase) or The Phrase
00069     Does Not Exist.
00070     check(PhraseInputs.Contains(InString))
00071
00072     return Cast<CastToType>(*PhraseInputs.Find(InString));
00073 }
00074
00075 void GetPhraseInput(const FString& InString, UParseInput* OutInput)
00076 {
00077     // Check If The Phrase Exits
00078     // This Error Will Be Thrown If: InString Is In Correct (Requires UpperCase) or The Phrase
00079     Does Not Exist.
00080     check(PhraseInputs.Contains(InString))
00081
00082     OutInput = *PhraseInputs.Find(InString);
00083 }
00084
00085 template<class CastToType>
00086 void GetPhraseInput(const FString& InString, CastToType* OutInput)
00087 {
00088     // Check If The Phrase Exits
00089     // This Error Will Be Thrown If: InString Is In Correct (Requires UpperCase) or The Phrase
00090     Does Not Exist.
00091     check(PhraseInputs.Contains(InString))
00092
00093     OutInput = Cast<CastToType>(*PhraseInputs.Find(InString));
00094 }
00095
00096 // -- GetPhraseInputs
00097
00098 void GetPhraseInputs(const FString& InString, TArray<UParseInput*>& OutInputs, const bool
00099 MaintainOrder = true)
00100 {
00101     // Check If The Phrase Exits
00102     // This Error Will Be Thrown If: InString Is In Correct (Requires UpperCase) or The Phrase
00103     Does Not Exist.
00104     check(PhraseInputs.Contains(InString))
00105
00106     PhraseInputs.MultiFind(InString, OutInputs, MaintainOrder);
00107 }
00108
00109 TArray<UParseInput*> GetPhraseInputs(const FString& InString, const bool MaintainOrder = true)
00110 {
00111     // Check If The Phrase Exits
00112     // This Error Will Be Thrown If: InString Is In Correct (Requires UpperCase) or The Phrase
00113     Does Not Exist.
00114     check(PhraseInputs.Contains(InString))
00115
00116     TArray<UParseInput*> OutInputs;

```

```

00149
00150     PhraseInputs.MultiFind(InString, OutInputs, MaintainOrder);
00151
00152     return OutInputs;
00153 }
00154
00155 // -- PhraseInput
00156
00162 void AddPhraseInput(const FString& InString, UParseInput* InInput)
00163 {
00164     PhraseInputs.Add(InString.ToUpper(), InInput);
00165 }
00166
00171 void RemovePhraseInput(const FString& InString)
00172 {
00173     PhraseInputs.Remove(InString);
00174 }
00175
00176 // -- ContextObject Stack Modification
00177
00182 void PushContextObj(UPhraseTreeContextObject* InObject)
00183 {
00184     this->ContextObjectStack.Push(InObject);
00185 }
00186
00190 void PopContextObj()
00191 {
00192     if (ContextObjectStack.IsEmpty())
00193         return;
00194
00195     this->ContextObjectStack.Pop();
00196 }
00197
00202 void PopContextObj(UPhraseTreeContextObject* OutObject)
00203 {
00204     if (ContextObjectStack.IsEmpty())
00205     {
00206         OutObject = nullptr;
00207         return;
00208     }
00209
00210     OutObject = this->ContextObjectStack.Pop();
00211 }
00212
00217 void RemoveContextObj(UPhraseTreeContextObject* InObject)
00218 {
00219     this->ContextObjectStack.Remove(InObject);
00220 }
00221
00222 // -- HasContextObj
00223
00228 bool HasContextObj()
00229 {
00230     return this->ContextObjectStack.Num() > 0;
00231 }
00232
00238 bool HasContextObj(UPhraseTreeContextObject* InObject)
00239 {
00240     return HasContextObj() && this->ContextObjectStack.Contains(InObject);
00241 }
00242
00243 // -- GetContextObj
00244
00249 UPhraseTreeContextObject* GetContextObj()
00250 {
00251     if (ContextObjectStack.IsEmpty())
00252         return nullptr;
00253
00254     return this->ContextObjectStack.Last();
00255 }
00256
00261 void GetContextObj(UPhraseTreeContextObject* OutObject)
00262 {
00263     if (ContextObjectStack.IsEmpty())
00264     {
00265         OutObject = nullptr;
00266         return;
00267     }
00268
00269     OutObject = this->ContextObjectStack.Last();
00270 }
00271
00277 template<class CastToType>
00278 CastToType* GetContextObj()
00279 {
00280     if (ContextObjectStack.IsEmpty())
00281         return nullptr;

```

```

00282
00283         return Cast<CastToType>(this->ContextObjectStack.Last());
00284     }
00285
00291     template<class CastToType>
00292     void GetContextObj(CastToType* OutObject)
00293     {
00294         if (ContextObjectStack.IsEmpty())
00295         {
00296             OutObject = nullptr;
00297             return;
00298         }
00299
00300         OutObject = Cast<CastToType>(this->ContextObjectStack.Last());
00301     }
00302
00307     void GetContextStack(TArray<UPhraseTreeContextObject*> OutContextStack)
00308     {
00309         OutContextStack = ContextObjectStack;
00310     }
00311
00316     TArray<UPhraseTreeContextObject*> GetContextStack()
00317     {
00318         return ContextObjectStack;
00319     }
00320
00321 protected:
00322
00326     TArray<UPhraseTreeContextObject*> ContextObjectStack = TArray<UPhraseTreeContextObject*>();
00327
00331     TArray<FString> PhraseRecord;
00332
00336     TMultiMap<FString, UParseInput*> PhraseInputs;
00337
00338 };

```

## 5.79 ParseResult.h

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #pragma once
00004
00005 #include "CoreMinimal.h"
00006
00007 class FPhraseNode;
00008
00009 typedef TSharedPtr<FPhraseNode> TPhraseNode;
00010
00011 typedef TArray<TPhraseNode> TPhraseNodeArray;
00012
00013
00014 enum OPENACCESSIBILITYCOMMUNICATION_API PhrasePropogationType : uint8_t
00015 {
00019     PHRASE_NOT_PARSED = 0,
00020
00024     PHRASE_UNABLE_TO_PARSE = 1,
00025
00029     PHRASE_REQUIRES_MORE = 2,
00030
00035     PHRASE_REQUIRES_MORE_CORRECT_PHRASES = 3,
00036
00040     PHRASE_PARSED = 4,
00041
00045     PHRASE_PARSED_AND_EXECUTED = 5,
00046 };
00047
00051 struct OPENACCESSIBILITYCOMMUNICATION_API FParseResult
00052 {
00053     FParseResult()
00054     {
00055         Result = PHRASE_NOT_PARSED;
00056     }
00057
00058     FParseResult(PhrasePropogationType InResult)
00059     {
00060         Result = InResult;
00061     }
00062
00063     FParseResult(PhrasePropogationType InResult, TSharedPtr<FPhraseNode> InReachedNode)
00064     {
00065         Result = InResult;
00066         ReachedNode = InReachedNode;
00067     }

```

```

00068
00069 public:
00070
00074     uint8_t Result;
00075
00079     TSharedPtr<FPPhraseNode> ReachedNode;
00080 };

```

## 5.80 IPhraseContextNode.h

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #pragma once
00004
00005 #include "CoreMinimal.h"
00006
00007 #include "PhraseTree/Containers/ContextObject.h"
00008
00012 class IPhraseContextNodeBase
00013 {
00014 protected:
00015
00021     virtual bool HasContextObject(TArray<UPhraseTreeContextObject*> InContextObjects) const = 0;
00022
00027     virtual UPhraseTreeContextObject* CreateContextObject(FParseRecord& Record) = 0;
00028
00034     virtual void ConstructContextChildren(TArray<TSharedPtr<class FPPhraseNode>& InChildNodes) = 0;
00035 };

```

## 5.81 PhraseContextMenuNode.h

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #pragma once
00004
00005 #include "CoreMinimal.h"
00006
00007 #include "PhraseTree/PhraseNode.h"
00008 #include "PhraseTree/IPhraseContextNode.h"
00009 #include "PhraseTree/PhraseEventNode.h"
00010 #include "PhraseTree/Containers/ContextMenuObject.h"
00011 #include "OpenAccessibilityComLogging.h"
00012
00013 template<typename ContextMenuType = UPhraseTreeContextMenuObject>
00014 class FPPhraseContextMenuNode : public FPPhraseNode, public IPhraseContextNodeBase
00015 {
00016 public:
00017
00018     static_assert(std::is_base_of_v<UPhraseTreeContextMenuObject, ContextMenuType>, "ContextType must
be a subclass of UPhraseTreeContextMenuObject");
00019
00020     FPPhraseContextMenuNode(const TCHAR* InInputString)
00021         : FPPhraseNode(InInputString)
00022         , ContextMenuScalar(1.0f)
00023     {
00024         this->ChildNodes = TPhraseNodeArray();
00025     };
00026
00027     FPPhraseContextMenuNode(const TCHAR* InInputString, TPhraseNodeArray InChildNodes)
00028         : FPPhraseNode(InInputString)
00029         , ContextMenuScalar(1.0f)
00030     {
00031         ConstructContextChildren(InChildNodes);
00032     };
00033
00034     FPPhraseContextMenuNode(const TCHAR* InInputString, TDelegate<TSharedPtr<IMenu>(FParseRecord&
Record)> InOnGetMenu, TPhraseNodeArray InChildNodes)
00035         : FPPhraseNode(InInputString)
00036         , ContextMenuScalar(1.0f)
00037         , OnGetMenu(InOnGetMenu)
00038     {
00039         ConstructContextChildren(InChildNodes);
00040     };
00041
00042     FPPhraseContextMenuNode(const TCHAR* InInputString, const float InMenuScalar, TPhraseNodeArray
InChildNodes)
00043         : FPPhraseNode(InInputString)
00044         , ContextMenuScalar(InMenuScalar)
00045     {

```

```

00046         ConstructContextChildren(InChildNodes);
00047     };
00048
00049     FPhraseContextMenuNode(const TCHAR* InInputString, const float InMenuScalar,
TDelegate<TSharedPtr<IMenu>(FParseRecord& Record)> InOnGetMenu, TPhraseNodeArray InChildNodes)
00050     : FPhraseNode(InInputString)
00051     , ContextMenuScalar(InMenuScalar)
00052     , OnGetMenu(InOnGetMenu)
00053     {
00054         ConstructContextChildren(InChildNodes);
00055     }
00056
00057     FPhraseContextMenuNode(const TCHAR* InInputString, const float InMenuScalar,
TDelegate<void(FParseRecord& Record)> InOnPhraseParsed, TPhraseNodeArray InChildNodes)
00058     : FPhraseNode(InInputString, InOnPhraseParsed)
00059     , ContextMenuScalar(InMenuScalar)
00060     {
00061         ConstructContextChildren(InChildNodes);
00062     }
00063
00064     FPhraseContextMenuNode(const TCHAR* InInputString, const float InMenuScalar,
TDelegate<TSharedPtr<IMenu>(FParseRecord& Record)> InOnGetMenu, TDelegate<void(FParseRecord& Record)>
InOnPhraseParsed, TPhraseNodeArray InChildNodes)
00065     : FPhraseNode(InInputString, InOnPhraseParsed)
00066     , ContextMenuScalar(InMenuScalar)
00067     , OnGetMenu(InOnGetMenu)
00068     {
00069         ConstructContextChildren(InChildNodes);
00070     }
00071
00072     virtual ~FPhraseContextMenuNode()
00073     {
00074     }
00075
00076
00077     // FPhraseNode Implementation
00078
00085     virtual FParseResult ParsePhrase(TArray<FString>& InPhraseWordArray, FParseRecord& InParseRecord)
override;
00086
00094     virtual FParseResult ParsePhraseAsContext(TArray<FString>& InPhraseWordArray, FParseRecord&
InParseRecord) override;
00095
00096     // End FPhraseNode Implementation
00097
00098 protected:
00099
00100     // FPhraseContextNodeBase Implementation
00101
00107     virtual bool HasContextObject(TArray<UPhraseTreeContextObject*> InContextObjects) const override;
00108
00113     virtual UPhraseTreeContextObject* CreateContextObject(FParseRecord& Record) override;
00114
00120     virtual void ConstructContextChildren(TPhraseNodeArray& InChildNodes) override;
00121
00122     // End FPhraseContextNode Implementation
00123
00124 protected:
00125
00129     const float ContextMenuScalar;
00130
00134     TDelegate<TSharedPtr<IMenu>(FParseRecord& Record)> OnGetMenu;
00135 };
00136
00137 template<typename ContextMenuType>
00138 FParseResult FPhraseContextMenuNode<ContextMenuType>::ParsePhrase(TArray<FString>& InPhraseWordArray,
FParseRecord& InParseRecord)
00139 {
00140     if (!HasContextObject(InParseRecord.GetContextStack()))
00141     {
00142         UPhraseTreeContextObject* NewObject = CreateContextObject(InParseRecord);
00143         InParseRecord.PushContextObj(NewObject);
00144     }
00145
00146     if (InPhraseWordArray.IsEmpty())
00147     {
00148         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Emptied Phrase Array ||"))
00149
00151         return FParseResult(PHRASE_REQUIRES_MORE, AsShared());
00152     }
00153
00154     InPhraseWordArray.Pop();
00155
00156     OnPhraseParsed.ExecuteIfBound(InParseRecord);
00157
00158     return ParseChildren(InPhraseWordArray, InParseRecord);

```



```

00159
00160     return FPhraseNode::ParsePhrase(InPhraseWordArray, InParseRecord);
00161 }
00162
00163 template<typename ContextMenuType>
00164 inline FParseResult FPhraseContextMenuNode<ContextMenuType>::ParsePhraseAsContext(TArray<FString>&
    InPhraseWordArray, FParseRecord& InParseRecord)
00165 {
00166     if (!HasContextObject(InParseRecord.GetContextStack()))
00167     {
00168         UPhraseTreeContextObject* NewObject = CreateContextObject(InParseRecord);
00169
00170         InParseRecord.PushContextObj(NewObject);
00171     }
00172
00173     if (InPhraseWordArray.IsEmpty())
00174     {
00175         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Emptied Phrase Array ||"))
00176
00177         return FParseResult(PHRASE_REQUIRES_MORE, AsShared());
00178     }
00179
00180     OnPhraseParsed.ExecuteIfBound(InParseRecord);
00181
00182     return ParseChildren(InPhraseWordArray, InParseRecord);
00183 }
00184
00185 template<typename ContextMenuType>
00186 bool FPhraseContextMenuNode<ContextMenuType>::HasContextObject(TArray<UPhraseTreeContextObject*>
    InContextObjects) const
00187 {
00188     for (auto& ContextObject : InContextObjects)
00189     {
00190         if (ContextObject->IsA(ContextMenuType::StaticClass()) && ContextObject->GetContextRoot() ==
            AsShared())
00191         {
00192             return true;
00193         }
00194     }
00195
00196     return false;
00197 }
00198
00199 template<typename ContextMenuType>
00200 UPhraseTreeContextObject* FPhraseContextMenuNode<ContextMenuType>::CreateContextObject(FParseRecord&
    Record)
00201 {
00202     if (!OnGetMenu.IsBound())
00203     {
00204         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("OnGetMenu Delegate Not Bound. Cannot Create Context
            Object, linked to a Menu.));
00205         return nullptr;
00206     }
00207
00208     TSharedPtr<IMenu> NewMenu = OnGetMenu.Execute(Record);
00209
00210     if (!NewMenu.IsValid())
00211     {
00212         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("OnGetMenu Delegate Returned Invalid Menu. Cannot
            Create Context Object.));
00213         return nullptr;
00214     }
00215
00216     ContextMenuType* NewContextObject = NewObject<ContextMenuType>();
00217     NewContextObject->Init(NewMenu.ToSharedRef(), this->AsShared());
00218
00219     NewContextObject->ScaleMenu(ContextMenuScalar);
00220
00221     return NewContextObject;
00222 }
00223
00224 template<typename ContextMenuType>
00225 void FPhraseContextMenuNode<ContextMenuType>::ConstructContextChildren(TPhraseNodeArray& InChildNodes)
00226 {
00227     // Construct Context Specific Children Nodes,
00228     // With Linked Functionality to the Context Menu Object and Root Node.
00229     TSharedPtr<FPhraseEventNode> CloseContextNode = MakeShared<FPhraseEventNode>();
00230     CloseContextNode->OnPhraseParsed.BindLambda(
00231         [this](FParseRecord& Record) {
00232
00233             UPhraseTreeContextMenuObject* ContextMenu =
                Record.GetContextObj<UPhraseTreeContextMenuObject>();
00234             if (ContextMenu->GetContextRoot() == this->AsShared())
00235             {
00236                 ContextMenu->Close();
00237                 ContextMenu->RemoveFromRoot();
00238

```

```

00239         Record.PopContextObj();
00240     }
00241 }
00242 };
00243
00244 this->ChildNodes = TPhraseNodeArray{
00245     MakeShared<FPhraseNode>(TEXT("CLOSE"),
00246         TPhraseNodeArray {
00247             CloseContextNode
00248         })
00249 };
00250
00251 this->ChildNodes.Append(InChildNodes);
00252 }

```

## 5.82 PhraseContextNode.h

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #pragma once
00004
00005 #include "CoreMinimal.h"
00006 #include "PhraseTree/PhraseNode.h"
00007 #include "PhraseTree/IPhraseContextNode.h"
00008 #include "PhraseTree/Containers/ContextObject.h"
00009
00010 #include "OpenAccessibilityComLogging.h"
00011 #include "PhraseEventNode.h"
00012
00013 template<class ContextType = UPhraseTreeContextObject>
00014 class FPhraseContextNode : public FPhraseNode, public IPhraseContextNodeBase
00015 {
00016 public:
00017     FPhraseContextNode(const TCHAR* InInputString)
00018         : FPhraseNode(InInputString)
00019     {
00020         static_assert(std::is_base_of<UPhraseTreeContextObject, ContextType>::value, "ContextType must
be a subclass of UPhraseTreeContextObject");
00021
00022         TPhraseNodeArray EmptyArray = TPhraseNodeArray();
00023         ConstructContextChildren(EmptyArray);
00024     }
00025
00026     FPhraseContextNode(const TCHAR* InInputString, TPhraseNodeArray InChildNodes)
00027         : FPhraseNode(InInputString, InChildNodes)
00028     {
00029         static_assert(std::is_base_of<UPhraseTreeContextObject, ContextType>::value, "ContextType must
be a subclass of UPhraseTreeContextObject");
00030
00031         ConstructContextChildren(InChildNodes);
00032     }
00033
00034     FPhraseContextNode(const TCHAR* InInputString, TDelegate<void(FParseRecord& Record)>
InOnPhraseParsed, TPhraseNodeArray InChildNodes)
00035         : FPhraseNode(InInputString, InOnPhraseParsed)
00036     {
00037         static_assert(std::is_base_of<UPhraseTreeContextObject, ContextType>::value, "ContextType must
be a subclass of UPhraseTreeContextObject");
00038
00039         ConstructContextChildren(InChildNodes);
00040     }
00041
00042     virtual ~FPhraseContextNode()
00043     {
00044     }
00045
00046     // FPhraseNode Implementation
00047
00048     virtual FParseResult ParsePhrase(TArray<FString>& InPhraseWordArray, FParseRecord& InParseRecord)
00049         override;
00050
00051     virtual FParseResult ParsePhraseAsContext(TArray<FString>& InPhraseWordArray, FParseRecord&
InParseRecord) override;
00052
00053     // End FPhraseNode Implementation
00054
00055 protected:
00056
00057     // FPhraseContextNodeBase Implementation
00058
00059     virtual bool HasContextObject(TArray<UPhraseTreeContextObject*> InContextObjects) const override;
00060

```

```

00061
00062     virtual UPhraseTreeContextObject* CreateContextObject(FParseRecord& Record) override;
00063
00064     virtual void ConstructContextChildren(TPhraseNodeArray& InChildNodes) override;
00065
00066     // End FPhraseContextNodeBase Implementation
00067
00068 };
00069
00070 template<class ContextType>
00071 FParseResult FPhraseContextNode<ContextType>::ParsePhrase(TArray<FString>& InPhraseWordArray,
00072     FParseRecord& InParseRecord)
00073 {
00074     if (!HasContextObject(InParseRecord.GetContextStack()))
00075     {
00076         UPhraseTreeContextObject* NewObject = CreateContextObject(InParseRecord);
00077         InParseRecord.PushContextObj(NewObject);
00078     }
00079     return FPhraseNode::ParsePhrase(InPhraseWordArray, InParseRecord);
00080 }
00081
00082 template<class ContextType>
00083 FParseResult FPhraseContextNode<ContextType>::ParsePhraseAsContext(TArray<FString>& InPhraseWordArray,
00084     FParseRecord& InParseRecord)
00085 {
00086     if (!HasContextObject(InParseRecord.GetContextStack()))
00087     {
00088         UPhraseTreeContextObject* NewObject = CreateContextObject(InParseRecord);
00089         InParseRecord.PushContextObj(NewObject);
00090     }
00091     if (InPhraseWordArray.IsEmpty())
00092     {
00093         UE_LOG(LogOpenAccessibilityCom, Log, TEXT("|| Emptied Phrase Array ||"))
00094         return FParseResult(PHRASE_REQUIRES_MORE, AsShared());
00095     }
00096     OnPhraseParsed.ExecuteIfBound(InParseRecord);
00097     // Pass
00098     return ParseChildren(InPhraseWordArray, InParseRecord);
00099 }
00100
00101 template<class ContextType>
00102 bool FPhraseContextNode<ContextType>::HasContextObject(TArray<UPhraseTreeContextObject*>
00103     InContextObjects) const
00104 {
00105     for (auto& ContextObject : InContextObjects)
00106     {
00107         if (ContextObject->IsA(ContextType::StaticClass()) && ContextObject->GetContextRoot() ==
00108             AsShared())
00109         {
00110             return true;
00111         }
00112     }
00113     return false;
00114 }
00115
00116 template<class ContextType>
00117 UPhraseTreeContextObject* FPhraseContextNode<ContextType>::CreateContextObject(FParseRecord& Record)
00118 {
00119     ContextType* NewContextObject = NewObject<ContextType>();
00120     NewContextObject->Init();
00121     NewContextObject->SetContextRootNode(AsShared());
00122     return NewContextObject;
00123 }
00124
00125 template<class ContextType>
00126 void FPhraseContextNode<ContextType>::ConstructContextChildren(TPhraseNodeArray& InChildNodes)
00127 {
00128     TSharedPtr<FPhraseEventNode> CloseContextNode = MakeShared<FPhraseEventNode>();
00129     CloseContextNode->OnPhraseParsed.BindLambda(
00130         [this](FParseRecord& Record) {
00131             UPhraseTreeContextObject* ContextObject = Record.GetContextObj();
00132             if (ContextObject->GetContextRoot() == this->AsShared())
00133             {
00134                 ContextObject->Close();
00135                 ContextObject->RemoveFromRoot();
00136             }
00137         }
00138     );
00139 }

```

```

00144         Record.PopContextObj();
00145     }
00146 }
00147 );
00148
00149 this->ChildNodes = TPhraseNodeArray{
00150     MakeShared<FPhraseNode>(TEXT("CLOSE"),
00151         TPhraseNodeArray {
00152             CloseContextNode
00153         })
00154 };
00155
00156 this->ChildNodes.Append(InChildNodes);
00157 }

```

## 5.83 PhraseDirectionalInputNode.h

```

00001 #pragma once
00002
00003 #include "CoreMinimal.h"
00004
00005 #include "PhraseEnumInputNode.h"
00006 #include "Containers/Input/InputContainers.h"
00007
00008 class OPENACCESSIBILITYCOMMUNICATION_API FPhraseDirectionalInputNode : public
    FPhraseEnumInputNode<EPhraseDirectionalInput>
00009 {
00010 public:
00011     FPhraseDirectionalInputNode(const TCHAR* NodeName)
00012         : FPhraseEnumInputNode<EPhraseDirectionalInput>(NodeName)
00013     {}
00014
00015     FPhraseDirectionalInputNode(const TCHAR* NodeName, TPhraseNodeArray InChildNodes)
00016         : FPhraseEnumInputNode<EPhraseDirectionalInput>(NodeName, InChildNodes)
00017     {}
00018
00019     FPhraseDirectionalInputNode(const TCHAR* NodeName, TDelegate<void(FParseRecord& Record)>
        InOnPhraseParsed, TPhraseNodeArray InChildNodes)
00020         : FPhraseEnumInputNode<EPhraseDirectionalInput>(NodeName, InOnPhraseParsed, InChildNodes)
00021     {}
00022
00023     FPhraseDirectionalInputNode(const TCHAR* NodeName, TPhraseNodeArray InChildNodes,
        TDelegate<void(int32 Input)> InOnInputRecieved)
00024         : FPhraseEnumInputNode<EPhraseDirectionalInput>(NodeName, InChildNodes, InOnInputRecieved)
00025     {}
00026
00027     FPhraseDirectionalInputNode(const TCHAR* NodeName, TDelegate<void(FParseRecord& Record)>
        InOnPhraseParsed, TPhraseNodeArray InChildNodes, TDelegate<void(int32 Input)> InOnInputRecieved)
00028         : FPhraseEnumInputNode<EPhraseDirectionalInput>(NodeName, InOnPhraseParsed, InChildNodes,
            InOnInputRecieved)
00029     {}
00030 };
00031
00032 class OPENACCESSIBILITYCOMMUNICATION_API FPhrase2DDirectionalInputNode : public
    FPhraseEnumInputNode<EPhrase2DDirectionalInput>
00033 {
00034 public:
00035     FPhrase2DDirectionalInputNode(const TCHAR* NodeName)
00036         : FPhraseEnumInputNode<EPhrase2DDirectionalInput>(NodeName)
00037     {}
00038
00039     FPhrase2DDirectionalInputNode(const TCHAR* NodeName, TPhraseNodeArray InChildNodes)
00040         : FPhraseEnumInputNode<EPhrase2DDirectionalInput>(NodeName, InChildNodes)
00041     {}
00042
00043     FPhrase2DDirectionalInputNode(const TCHAR* NodeName, TDelegate<void(FParseRecord& Record)>
        InOnPhraseParsed, TPhraseNodeArray InChildNodes)
00044         : FPhraseEnumInputNode<EPhrase2DDirectionalInput>(NodeName, InOnPhraseParsed, InChildNodes)
00045     {}
00046
00047     FPhrase2DDirectionalInputNode(const TCHAR* NodeName, TPhraseNodeArray InChildNodes,
        TDelegate<void(int32 Input)> InOnInputRecieved)
00048         : FPhraseEnumInputNode<EPhrase2DDirectionalInput>(NodeName, InChildNodes, InOnInputRecieved)
00049     {}
00050
00051     FPhrase2DDirectionalInputNode(const TCHAR* NodeName, TDelegate<void (FParseRecord& Record)>
        InOnPhraseParsed, TPhraseNodeArray InChildNodes, TDelegate<void(int32 Input)> InOnInputRecieved)
00052         : FPhraseEnumInputNode<EPhrase2DDirectionalInput>(NodeName, InOnPhraseParsed, InChildNodes,
            InOnInputRecieved)
00053     {}
00054 };
00055
00056 class OPENACCESSIBILITYCOMMUNICATION_API FPhraseScrollInputNode : public
    FPhraseEnumInputNode<EPhraseScrollInput>

```

```

00057 {
00058 public:
00059     FPhraseScrollInputNode(const TCHAR* NodeName)
00060         : FPhraseEnumInputNode<EPhraseScrollInput>(NodeName)
00061     {}
00062
00063     FPhraseScrollInputNode(const TCHAR* NodeName, TPhraseNodeArray InChildNodes)
00064         : FPhraseEnumInputNode<EPhraseScrollInput>(NodeName, InChildNodes)
00065     {}
00066
00067     FPhraseScrollInputNode(const TCHAR* NodeName, TDelegate<void(FParseRecord& Record)>
InOnPhraseParsed, TPhraseNodeArray InChildNodes)
00068         : FPhraseEnumInputNode<EPhraseScrollInput>(NodeName, InOnPhraseParsed, InChildNodes)
00069     {}
00070
00071     FPhraseScrollInputNode(const TCHAR* NodeName, TPhraseNodeArray InChildNodes, TDelegate<void(int32
Input)> InOnInputRecieved)
00072         : FPhraseEnumInputNode<EPhraseScrollInput>(NodeName, InChildNodes, InOnInputRecieved)
00073     {}
00074
00075     FPhraseScrollInputNode(const TCHAR* NodeName, TDelegate<void(FParseRecord& Record)>
InOnPhraseParsed, TPhraseNodeArray InChildNodes, TDelegate<void(int32 Input)> InOnInputRecieved)
00076         : FPhraseEnumInputNode<EPhraseScrollInput>(NodeName, InOnPhraseParsed, InChildNodes,
InOnInputRecieved)
00077     {}
00078 };
00079
00080 class OPENACCESSIBILITYCOMMUNICATION_API FPhrasePositionalInputNode : public
FPhraseEnumInputNode<EPhrasePositionalInput>
00081 {
00082 public:
00083     FPhrasePositionalInputNode(const TCHAR* NodeName)
00084         : FPhraseEnumInputNode<EPhrasePositionalInput>(NodeName)
00085     {}
00086
00087     FPhrasePositionalInputNode(const TCHAR* NodeName, TPhraseNodeArray InChildNodes)
00088         : FPhraseEnumInputNode<EPhrasePositionalInput>(NodeName, InChildNodes)
00089     {}
00090
00091     FPhrasePositionalInputNode(const TCHAR* NodeName, TDelegate<void(FParseRecord& Record)>
InOnPhraseParsed, TPhraseNodeArray InChildNodes)
00092         : FPhraseEnumInputNode<EPhrasePositionalInput>(NodeName, InOnPhraseParsed, InChildNodes)
00093     {}
00094
00095     FPhrasePositionalInputNode(const TCHAR* NodeName, TPhraseNodeArray InChildNodes,
TDelegate<void(int32 Input)> InOnInputRecieved)
00096         : FPhraseEnumInputNode<EPhrasePositionalInput>(NodeName, InChildNodes, InOnInputRecieved)
00097     {}
00098
00099     FPhrasePositionalInputNode(const TCHAR* NodeName, TDelegate<void(FParseRecord& Record)>
InOnPhraseParsed, TPhraseNodeArray InChildNodes, TDelegate<void(int32 Input)> InOnInputRecieved)
00100         : FPhraseEnumInputNode<EPhrasePositionalInput>(NodeName, InOnPhraseParsed, InChildNodes,
InOnInputRecieved)
00101     {}
00102 };

```

## 5.84 PhraseEnumInputNode.h

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #pragma once
00004
00005 #include "CoreMinimal.h"
00006 #include "PhraseInputNode.h"
00007 #include "Containers/Input/InputContainers.h"
00008
00012 template<typename TEnum>
00013 class OPENACCESSIBILITYCOMMUNICATION_API FPhraseEnumInputNode : public FPhraseInputNode<int32>
00014 {
00015 public:
00016
00017     FPhraseEnumInputNode(const TCHAR* InInputString);
00018     FPhraseEnumInputNode(const TCHAR* InInputString, TPhraseNodeArray InChildNodes);
00019     FPhraseEnumInputNode(const TCHAR* InInputString, TDelegate<void(FParseRecord& Record)>
InOnPhraseParsed, TPhraseNodeArray InChildNodes);
00020     FPhraseEnumInputNode(const TCHAR* InInputString, TPhraseNodeArray InChildNodes,
TDelegate<void(int32 Input)> InOnInputRecieved);
00021     FPhraseEnumInputNode(const TCHAR* InInputString, TDelegate<void(FParseRecord& Record)>
InOnPhraseParsed, TPhraseNodeArray InChildNodes, TDelegate<void(int32 Input)> InOnInputRecieved);
00022
00023     virtual ~FPhraseEnumInputNode();
00024
00025 protected:

```

```

00026
00033     virtual bool MeetsInputRequirements(const FString& InPhrase) override;
00034
00041     virtual bool RecordInput(const FString& InInput, FParseRecord& OutParseRecord) override;
00042 };

```

## 5.85 PhraseEventNode.h

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #pragma once
00004
00005 #include "CoreMinimal.h"
00006 #include "PhraseTree/PhraseNode.h"
00007
00011 class OPENACCESSIBILITYCOMMUNICATION_API FPhraseEventNode : public FPhraseNode
00012 {
00013 public:
00014     FPhraseEventNode();
00015     FPhraseEventNode(TDelegate<void(FParseRecord&)> InEvent);
00016     FPhraseEventNode(TFunction<void(FParseRecord&)> InEventFunction);
00017
00018     virtual ~FPhraseEventNode();
00019
00020     // FPhraseNode Implementation
00021     virtual bool IsLeafNode() const override { return true; }
00022
00023     virtual bool RequiresPhrase(const FString InPhrase) override;
00024     virtual bool RequiresPhrase(const FString InPhrase, int32& OutDistance) override;
00025
00026     virtual FParseResult ParsePhrase(TArray<FString>& InPhraseArray, FParseRecord& InParseRecord)
00027     override;
00028     // End FPhraseNode Implementation
00028 };

```

## 5.86 PhraseInputNode.h

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #pragma once
00004
00005 #include "CoreMinimal.h"
00006 #include "PhraseTree/PhraseNode.h"
00007
00011 template <typename InputType = int32>
00012 class OPENACCESSIBILITYCOMMUNICATION_API FPhraseInputNode : public FPhraseNode
00013 {
00014 public:
00015     FPhraseInputNode(const TCHAR* InInputString);
00016     FPhraseInputNode(const TCHAR* InInputString, TPhraseNodeArray InChildNodes);
00017     FPhraseInputNode(const TCHAR* InInputString, TDelegate<void(FParseRecord& Record)>
00018     InOnPhraseParsed, TPhraseNodeArray InChildNodes);
00018     FPhraseInputNode(const TCHAR* InInputString, TPhraseNodeArray InChildNodes, TDelegate<void
00019     (InputType Input)> InOnInputRecieved);
00019     FPhraseInputNode(const TCHAR* InInputString, TDelegate<void(FParseRecord& Record)>
00020     InOnPhraseParsed, TPhraseNodeArray InChildNodes, TDelegate<void(InputType Input)> InOnInputRecieved);
00020
00021     virtual ~FPhraseInputNode();
00022
00023     // FPhraseNode Implementation
00024
00025     virtual bool RequiresPhrase(const FString InPhrase) override;
00026
00027     virtual bool RequiresPhrase(const FString InPhrase, int32& OutDistance) override;
00028
00029     virtual FParseResult ParsePhrase(TArray<FString>& InPhraseArray, FParseRecord& InParseRecord)
00030     override;
00031     // End FPhraseNode Implementation
00032
00033     TDelegate<void(InputType ReceivedInput)> OnInputReceived;
00034
00035 protected:
00036
00043     virtual bool MeetsInputRequirements(const FString& InPhrase);
00044
00051     virtual bool RecordInput(const FString& InInput, FParseRecord& OutParseRecord);
00052 };

```

## 5.87 PhraseNode.h

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #pragma once
00004
00005 #include "CoreMinimal.h"
00006
00007 #include "PhraseTree/Containers/ParseResult.h"
00008 #include "PhraseTree/Containers/ParseRecord.h"
00009
00010 class IPhraseNodeBase
00011 {
00012 public:
00013
00018     virtual bool IsLeafNode() const = 0;
00019
00024     virtual bool HasLeafChild() const = 0;
00025
00031     virtual bool RequiresPhrase(const FString InPhrase) = 0;
00032
00039     virtual FParseResult ParsePhrase(TArray<FString>& InPhraseWordArray, FParseRecord& InParseRecord)
    = 0;
00040
00048     virtual FParseResult ParsePhraseAsContext(TArray<FString>& InPhraseWordArray, FParseRecord&
    InParseRecord) = 0;
00049 };
00050
00054 class OPENACCESSIBILITYCOMMUNICATION_API FPhraseNode : public TSharedFromThis<FPhraseNode>
00055 {
00056 public:
00057
00058     FPhraseNode(const TCHAR* InBoundPhrase);
00059     FPhraseNode(const TCHAR* InBoundPhrase, TDelegate<void (FParseRecord& Record)> InOnPhraseParsed);
00060     FPhraseNode(const TCHAR* InBoundPhrase, TPhraseNodeArray InChildNodes);
00061     FPhraseNode(const TCHAR* InBoundPhrase, TDelegate<void (FParseRecord& Record)> InOnPhraseParsed,
    TPhraseNodeArray InChildNodes);
00062
00063     virtual ~FPhraseNode();
00064
00069     virtual bool IsLeafNode() const { return false; }
00070
00071     virtual bool HasLeafChild() const;
00072
00078     virtual bool RequiresPhrase(const FString InPhrase);
00079
00086     virtual bool RequiresPhrase(const FString InPhrase, int32& OutDistance);
00087
00094     virtual FParseResult ParsePhrase(TArray<FString>& InPhraseWordArray, FParseRecord& InParseRecord);
00095
00103     virtual FParseResult ParsePhraseAsContext(TArray<FString>& InPhraseWordArray, FParseRecord&
    InParseRecord);
00104
00108     virtual FParseResult ParsePhraseIfRequired(TArray<FString>& InPhraseWordArray, FParseRecord&
    InParseRecord);
00109
00116     virtual FParseResult ParseChildren(TArray<FString>& InPhraseArray, FParseRecord& InParseRecord);
00117
00123     bool CanBindChild(TPhraseNode& InNode);
00124
00130     bool BindChildNode(TPhraseNode InNode);
00131
00137     bool BindChildNodeForce(TPhraseNode InNode);
00138
00144     bool BindChildrenNodes(TPhraseNodeArray InNodes);
00145
00151     bool BindChildrenNodesForce(TPhraseNodeArray InNodes);
00152
00153 protected:
00154
00158     bool HasLeafChild();
00159
00160 public:
00161
00165     TWeakPtr<FPhraseNode> ParentNode;
00166
00170     TPhraseNodeArray ChildNodes;
00171
00175     FString BoundPhrase;
00176
00177     // Phrase To Be Executed On the Parse Command
00178     TDelegate<void (FParseRecord& Record)> OnPhraseParsed;
00179
00180 protected:
00181
00185     bool bHasLeafChild;
00186 };

```

## 5.88 PhraseStringInputNode.h

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #pragma once
00004
00005 #include "CoreMinimal.h"
00006 #include "PhraseInputNode.h"
00007
00011 class OPENACCESSIBILITYCOMMUNICATION_API FPhraseStringInputNode : public FPhraseInputNode<FString>
00012 {
00013 public:
00014
00015     FPhraseStringInputNode(const TCHAR* InInputString);
00016     FPhraseStringInputNode(const TCHAR* InInputString, TPhraseNodeArray InChildNodes);
00017     FPhraseStringInputNode(const TCHAR* InInputString, TDelegate<void(FParseRecord& Record)>
        InOnPhraseParsed, TPhraseNodeArray InChildNodes);
00018     FPhraseStringInputNode(const TCHAR* InInputString, TPhraseNodeArray InChildNodes,
        TDelegate<void(FString Input)> InOnInputRecieved);
00019     FPhraseStringInputNode(const TCHAR* InInputString, TDelegate<void(FParseRecord& Record)>
        InOnPhraseParsed, TPhraseNodeArray InChildNodes, TDelegate<void(FString Input)> InOnInputRecieved);
00020
00021     virtual ~FPhraseStringInputNode();
00022
00023 protected:
00024
00025     // FPhraseInputNode Implementation
00026
00027     virtual bool MeetsInputRequirements(const FString& InPhrase) override;
00028
00029     virtual bool RecordInput(const FString& InInput, FParseRecord& OutParseRecord) override;
00030
00031     // End FPhraseInputNode Implementation
00032 };

```

## 5.89 PhraseTreeFunctionLibrary.h

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #pragma once
00004
00005 #include "CoreMinimal.h"
00006 #include "UObject/ObjectMacros.h"
00007 #include "UObject/Object.h"
00008 #include "UObject/UnrealType.h"
00009 #include "UObject/ScriptMacros.h"
00010
00011 #include "PhraseTree.h"
00012
00013 #include "PhraseTree/Containers/ParseRecord.h"
00014 #include "PhraseTree/Containers/Input/UParseIntInput.h"
00015 #include "PhraseTree/Containers/Input/UParseStringInput.h"
00016 #include "PhraseTree/Containers/Input/UParseEnumInput.h"
00017
00018 #include "PhraseTreeFunctionLibrary.generated.h"
00019
00020 // Utility Definitions
00021
00022 DECLARE_LOG_CATEGORY_EXTERN(LogOpenAccessibilityPhraseEvent, Log, All);
00023
00024 DEFINE_LOG_CATEGORY(LogOpenAccessibilityPhraseEvent);
00025
00026 UCLASS(Abstract)
00027 class OPENACCESSIBILITYCOMMUNICATION_API UPhraseTreeFunctionLibrary : public UObject
00028 {
00029     GENERATED_BODY()
00030
00031 public:
00032
00033     virtual void BindBranches(TSharedRef<FPhraseTree> PhraseTree) {};
00034
00035 };

```

## 5.90 Utils.h

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #pragma once
00004

```



```

00005 #include "CoreMinimal.h"
00006
00007 #include "OpenAccessibility.h"
00008 #include "OpenAccessibilityCommunication.h"
00009
00010 // Utility Macros
00011
00012 #define EMPTY_ARG
00013
00014
00020 #define GET_ACTIVE_REGULAR_WINDOW_RETURN(ActiveContainerName, ReturnObject) \
00021     TSharedPtr<SWindow> ActiveContainerName; \
00022     { \
00023         ActiveContainerName = FSlateApplication::Get().GetActiveTopLevelRegularWindow(); \
00024         if (!ActiveContainerName.IsValid()) \
00025         { \
00026             UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, \
00027                 TEXT("GET_ACTIVE_WINDOW: No Active Window Was Found")) \
00028             return ReturnObject; \
00029         } \
00030     }; \
00031
00036 #define GET_ACTIVE_REGULAR_WINDOW(ActiveContainerName) \
00037     GET_ACTIVE_REGULAR_WINDOW_RETURN(ActiveContainerName, EMPTY_ARG)
00038
00039
00045 #define GET_ACTIVE_WINDOW_RETURN(ActiveContainerName, ReturnObject) \
00046     TSharedPtr<SWindow> ActiveContainerName; \
00047     { \
00048         ActiveContainerName = FSlateApplication::Get().GetActiveTopLevelWindow(); \
00049         if (!ActiveContainerName.IsValid()) \
00050         { \
00051             UE_LOG(LogOpenAccessibilityPhraseEvent, Warning, \
00052                 TEXT("GET_ACTIVE_WINDOW: No Active Window Was Found")) \
00053             return ReturnObject; \
00054         } \
00055     }; \
00056
00061 #define GET_ACTIVE_WINDOW(ActiveContainerName) \
00062     GET_ACTIVE_WINDOW_RETURN(ActiveContainerName, EMPTY_ARG)
00063
00064
00070 #define GET_ACTIVE_TAB_RETURN(ActiveContainerName, ReturnObject) \
00071     TSharedPtr<SDockTab> ActiveContainerName; \
00072     { \
00073         ActiveContainerName = FGlobalTabmanager::Get()->GetActiveTab(); \
00074         if (!ActiveContainerName.IsValid()) \
00075         { \
00076             UE_LOG(LogOpenAccessibilityPhraseEvent, Display, \
00077                 TEXT("GET_ACTIVE_TAB: Not Active Tab Was Found")); \
00078             return ReturnObject; \
00079         } \
00080     }; \
00081
00086 #define GET_ACTIVE_TAB(ActiveContainerName) \
00087     GET_ACTIVE_TAB_RETURN(ActiveContainerName, EMPTY_ARG)
00088
00089
00095 #define GET_ACTIVE_TAB_CONTENT_RETURN(ActiveContainerName, ReturnObject) \
00096     TSharedPtr<SWidget> ActiveContainerName; \
00097     { \
00098         GET_ACTIVE_TAB_RETURN(_AT, ReturnObject) \
00099         ActiveContainerName = _AT->GetContent(); \
00100         if (_AT == nullptr || !ActiveContainerName.IsValid()) { \
00101             UE_LOG(LogOpenAccessibilityPhraseEvent, Display, \
00102                 TEXT("GET_ACTIVE_TAB_CONTENT: FOUND ACTIVE TAB IS NOT VALID")); \
00103             return ReturnObject; \
00104         } \
00105     }; \
00106
00112 #define GET_ACTIVE_TAB_CONTENT(ActiveContainerName) \
00113     GET_ACTIVE_TAB_CONTENT_RETURN(ActiveContainerName, EMPTY_ARG)
00114
00115
00122 #define GET_CAST_ACTIVE_TAB_CONTENT_RETURN(ActiveContainerName, ActiveTabType, ReturnObject) \
00123     static_assert(TIsDerivedFrom<ActiveTabType, SWidget>::IsDerived, "Provided Type Is Not a Valid \
00124     Widget Type"); \
00125     TSharedPtr<ActiveTabType> ActiveContainerName; \
00126     { \
00127         GET_ACTIVE_TAB_CONTENT_RETURN(_PreCastContainer, ReturnObject); \
00128         ActiveContainerName = StaticCastSharedPtr<ActiveTabType>(_PreCastContainer); \

```

```

00128         if (!ActiveContainerName.IsValid() || ActiveContainerName->GetType() != #ActiveTabType) {
00129             UE_LOG(LogOpenAccessibilityPhraseEvent, Display,
00130                 TEXT("GET_ACTIVE_TAB: FOUND ACTIVE TAB IS NOT VALID"));
00131             return ReturnObject;
00132         }
00133     };
00134
00140 #define GET_CAST_ACTIVE_TAB_CONTENT(ActiveContainerName, ActiveTabType) \
00141     GET_CAST_ACTIVE_TAB_CONTENT_RETURN(ActiveContainerName, ActiveTabType, EMPTY_ARG)
00142
00143 #define GET_ACTIVE_KEYBOARD_WIDGET_RETURN(ActiveContainerName, ReturnObject) \
00144     TSharedPtr<SWidget> ActiveContainerName;
00145     {
00146         FSlateApplication &SlateApp = FSlateApplication::Get();
00147         if (!SlateApp.IsInitialized())
00148             return ReturnObject;
00149
00150         ActiveContainerName = SlateApp.GetKeyboardFocusedWidget();
00151         if (!ActiveContainerName.IsValid()) {
00152             UE_LOG(LogOpenAccessibilityPhraseEvent, Display,
00153                 TEXT("GET_ACTIVE_KEYBOARD_WIDGET: NO ACTIVE WIDGET FOUND.));
00154             return ReturnObject;
00155         }
00156     };
00157
00158 #define GET_ACTIVE_KEYBOARD_WIDGET(ActiveContainerName) \
00159     GET_ACTIVE_KEYBOARD_WIDGET_RETURN(ActiveContainerName, EMPTY_ARG)
00160
00161 #define GET_TOP_CONTEXT_RETURN(InRecord, ContextObjectName, ContextObjectType, ReturnObject) \
00162     ContextObjectType *ContextObjectName;
00163     {
00164         ContextObjectName = InRecord.GetContextObj<ContextObjectType>();
00165         if (ContextObjectName == nullptr) {
00166             UE_LOG(LogOpenAccessibilityPhraseEvent, Display,
00167                 TEXT("GET_TOP_CONTEXT: NO CONTEXT OBJECT FOUND.));
00168             return ReturnObject;
00169         }
00170     };
00171
00172 #define GET_TOP_CONTEXT(InRecord, ContextObjectName, ContextObjectType) \
00173     GET_TOP_CONTEXT_RETURN(InRecord, ContextObjectName, ContextObjectType, EMPTY_ARG)
00174
00175 // Utility Functions
00176
00177 FORCEINLINE TSharedPtr<FPhraseTree> GetPhraseTree()
00178 {
00179     FOpenAccessibilityCommunicationModule &OAComsModule =
00180         FOpenAccessibilityCommunicationModule::Get();
00181
00182     if (OAComsModule.PhraseTree.IsValid())
00183         return OAComsModule.PhraseTree.ToSharedRef();
00184
00185     return TSharedPtr<FPhraseTree>();
00186 }
00187
00188 FORCEINLINE TSharedPtr<FAssetAccessibilityRegistry> GetAssetRegistry()
00189 {
00190     FOpenAccessibilityModule &OAModule = FOpenAccessibilityModule::Get();
00191
00192     if (OAModule.AssetAccessibilityRegistry.IsValid())
00193         return OAModule.AssetAccessibilityRegistry.ToSharedRef();
00194
00195     return TSharedPtr<FAssetAccessibilityRegistry>();
00196 }
00197
00198 // Delegate Utilities
00199
00200 template<typename ObjectType>
00201 [[nodiscard]] FORCEINLINE TDelegate<void(FParseRecord&)> CreateParseDelegate(ObjectType* ObjPtr, void
00202 (ObjectType::* ObjFunction)(FParseRecord&))
00203 {
00204     return TDelegate<void(FParseRecord&)>::CreateUObject(ObjPtr, ObjFunction);
00205 }
00206
00207 template<typename ObjectType, typename InputType>
00208 [[nodiscard]] FORCEINLINE TDelegate<void(InputType)> CreateInputDelegate(ObjectType* ObjPtr, void
00209 (ObjectType::* ObjFunction)(InputType))
00210 {
00211     return TDelegate<void(InputType)>::CreateUObject(ObjPtr, ObjFunction);
00212 }

```

```

00257 }
00258
00266 template <typename ObjectType>
00267 [[nodiscard]] FORCEINLINE TDelegate<TSharedPtr<IMenu>(FParseRecord&)> CreateMenuDelegate(ObjectType*
    ObjPtr, TSharedPtr<IMenu> (ObjectType::* ObjFunction)(FParseRecord&))
00268 {
00269     return TDelegate<TSharedPtr<IMenu>(FParseRecord&)>::CreateUObject(ObjPtr, ObjFunction);
00270 }
00271
00272 // Utility Functions
00273 namespace EventUtils
00274 {
00282 [[nodiscard]] FORCEINLINE FString RemoveWordsFromEnd(const FString& InString, const int32&
    AmountToRemove)
00283 {
00284     TArray<FString> SplitTextBoxString;
00285     InString.ParseIntoArrayWS(SplitTextBoxString);
00286
00287     int RemovedAmount = 0;
00288     int CurrentIndex = SplitTextBoxString.Num() - 1;
00289     while (RemovedAmount < AmountToRemove) {
00290         if (SplitTextBoxString.IsEmpty())
00291             break;
00292
00293         SplitTextBoxString.RemoveAt(CurrentIndex--);
00294         RemovedAmount++;
00295     }
00296
00297     if (SplitTextBoxString.Num() > 0)
00298         return FString::Join(SplitTextBoxString, TEXT(" "));
00299
00300     return TEXT("");
00301 }
00302 }

```

## 5.91 Utils.h

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #pragma once
00004
00005 #include "CoreMinimal.h"
00006
00007 class OPENACCESSIBILITYCOMMUNICATION_API NumericParser
00008 {
00009 public:
00010
00017     static bool IsValidNumeric(const FString& StringToCheck, bool ConvertToUpper = true);
00018
00024     static void StringToNumeric(FString& NumericString, bool ConvertToUpper = true);
00025
00026 private:
00027     static const TMap<const FString, const FString> StringMappings;
00028 };

```

## 5.92 PhraseTreeUtils.h

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #pragma once
00004
00005 #include "CoreMinimal.h"
00006
00007 #include "PhraseTree/PhraseTreeFunctionLibrary.h"
00008
00009 #include "PhraseTreeUtils.generated.h"
00010
00011 UCLASS()
00012 class OPENACCESSIBILITYCOMMUNICATION_API UPhraseTreeUtils : public UObject
00013 {
00014     GENERATED_BODY()
00015
00016 public:
00017
00018     UPhraseTreeUtils();
00019
00020     virtual ~UPhraseTreeUtils();
00021

```

```

00022 // Function Library Methods
00023
00028 void RegisterFunctionLibrary(UPhraseTreeFunctionLibrary* LibraryToRegister);
00029
00034 void SetPhraseTree(TSharedRef<FPhraseTree> NewPhraseTree)
00035 {
00036     this->PhraseTree = NewPhraseTree;
00037 }
00038
00039 protected:
00040
00044 UPROPERTY(EditAnywhere)
00045 TArray<UPhraseTreeFunctionLibrary*> RegisteredLibraries;
00046
00047
00051 TWeakPtr<FPhraseTree> PhraseTree;
00052 };
00053

```

## 5.93 SocketCommunicationServer.h

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #pragma once
00004
00005 #include "CoreMinimal.h"
00006
00007 #ifdef WITH_ZEROMQ
00008 #include "zmq.hpp"
00009 #include "zmq_addon.hpp"
00010 #else
00011 #error "ZeroMQ Could not be found. Please Make Sure ZEROMQ is Installed Correctly, and the WITH_ZEROMQ
    Definition is Valid."
00012 #endif // WITH_ZEROMQ
00013
00014 class FJsonObject;
00015
00016 typedef zmq::send_flags ComSendFlags;
00017 typedef zmq::recv_flags ComRecvFlags;
00018
00022 class OPENACCESSIBILITYCOMMUNICATION_API FSocketCommunicationServer
00023 {
00024 public:
00025
00026     FSocketCommunicationServer(const std::string SendAddress = "tcp://127.0.0.1:5555", const
        std::string RecvAddress = "tcp://127.0.0.1:5556", const int PollTimeout = 10);
00027     ~FSocketCommunicationServer();
00028
00033     bool EventOccured();
00034
00042     bool SendArrayBuffer(const float* MessageData, size_t Size, ComSendFlags SendFlags =
        ComSendFlags::none);
00043
00050     bool SendArrayBuffer(const float MessageData[], ComSendFlags SendFlags = ComSendFlags::none);
00051
00058     bool SendArrayBuffer(const TArray<float>& ArrayMessage, ComSendFlags SendFlags =
        ComSendFlags::none);
00059
00067     bool SendArrayMessage(const float* MessageData, size_t Size, ComSendFlags SendFlags =
        ComSendFlags::none);
00068
00075     bool SendArrayMessage(const float MessageData[], ComSendFlags SendFlags = ComSendFlags::none);
00076
00083     bool SendArrayMessage(const TArray<float>& ArrayMessage, ComSendFlags SendFlags =
        ComSendFlags::none);
00084
00093     bool SendArrayMessageWithMeta(const float* MessageData, size_t Size, const
        TSharedRef<FJsonObject>& Metadata, ComSendFlags SendFlags = ComSendFlags::none);
00094
00102     bool SendArrayMessageWithMeta(const float MessageData[], const TSharedRef<FJsonObject>& Metadata,
        ComSendFlags SendFlags = ComSendFlags::none);
00103
00111     bool SendArrayMessageWithMeta(const TArray<float>& ArrayMessage, const TSharedRef<FJsonObject>&
        Metadata, ComSendFlags SendFlags = ComSendFlags::none);
00112
00119     bool SendStringBuffer(const std::string StringMessage, ComSendFlags SendFlags =
        ComSendFlags::none);
00120
00127     bool SendJsonBuffer(const std::string JsonMessage, ComSendFlags SendFlags = ComSendFlags::none);
00128
00137     template <typename T>
00138     bool RecvArray(TArray<T>& OutArrayData, size_t Size, ComRecvFlags RecvFlag = ComRecvFlags::none);
00139

```

```

00146     bool RecvString(FString& OutStringMessage, ComRecvFlags RecvFlag = ComRecvFlags::none);
00147
00154     bool RecvJson(FString& OutJsonMessage, ComRecvFlags RecvFlag = ComRecvFlags::none);
00155
00162     bool RecvStringMultipart(TArray<FString>& OutMessages, ComRecvFlags RecvFlag =
ComRecvFlags::none);
00163
00171     bool RecvStringMultipartWithMeta(TArray<FString>& OutMessages, TSharedPtr<FJsonObject>&
OutMetadata, ComRecvFlags RecvFlag = ComRecvFlags::none);
00172
00173 protected:
00174
00182     bool RecvMultipartWithMeta(std::vector<zmq::message_t>& OutMultipartMessages,
TSharedPtr<FJsonObject>& OutMetadata, ComRecvFlags RecvFlags);
00183
00190     bool SerializeJSON(const TSharedRef<FJsonObject>& InJsonObject, FString& OutJsonString);
00191
00198     bool DeserializeJSON(const FString& InJsonString, TSharedPtr<FJsonObject>& OutJsonObject);
00199
00200 protected:
00201
00205     zmq::context_t* Context;
00206
00210     zmq::socket_t* SendSocket;
00211
00215     zmq::socket_t* RecvSocket;
00216
00220     zmq::poller_t<int>* Poller;
00221
00222     std::string SendAddress;
00223     std::string RecvAddress;
00224
00228     int PollTimeout;
00229 };

```

## 5.94 UBAudioCapture.h

```

00001 // Copyright F-Dudley. All Rights Reserved.
00002
00003 #pragma once
00004
00005 #include "CoreMinimal.h"
00006 #include "AudioCapture.h"
00007
00011 class OPENACCESSIBILITYCOMMUNICATION_API UBAudioCapture : public UAudioCapture
00012 {
00013
00014 public:
00015     UBAudioCapture();
00016     virtual ~UBAudioCapture();
00017
00024     bool OpenDefaultAudioStream(int32 OverrideSampleRate, int32 OverrideInputChannels);
00025 };

```



# Index

- `__del__`
  - `OpenAccessibilityPy.Audio.AudioResampler`, 9
  - `OpenAccessibilityPy.CommunicationServer.CommunicationServer`, 12
  - `OpenAccessibilityPy.OpenAccessibilityPy`, 185
  - `OpenAccessibilityPy.WhisperInterface.WhisperInterface`, 314
- `__init__`
- `OpenAccessibilityPy.Audio.AudioResampler`, 9
- `OpenAccessibilityPy.CommunicationServer.CommunicationServer`, 12
- `OpenAccessibilityPy.OpenAccessibilityPy`, 184
- `OpenAccessibilityPy.WhisperInterface.WhisperInterface`, 314
- `~FAccessibilityNodeFactory`
  - `FAccessibilityNodeFactory`, 21
- `~FAssetAccessibilityRegistry`
  - `FAssetAccessibilityRegistry`, 26
- `~FGraphIndexer`
  - `FGraphIndexer`, 36
- `~FIndexer`
  - `FIndexer< KeyType, ValueType >`, 50
- `~FParseRecord`
  - `FParseRecord`, 76
- `~FPhraseContextMenuNode`
  - `FPhraseContextMenuNode< ContextMenuType >`, 93
- `~FPhraseContextNode`
  - `FPhraseContextNode< ContextType >`, 99
- `~FPhraseEnumInputNode`
  - `FPhraseEnumInputNode< TEnum >`, 106
- `~FPhraseEventNode`
  - `FPhraseEventNode`, 109
- `~FPhraseInputNode`
  - `FPhraseInputNode< InputType >`, 114
- `~FPhraseNode`
  - `FPhraseNode`, 120
- `~FPhraseStringInputNode`
  - `FPhraseStringInputNode`, 134
- `~FPhraseTree`
  - `FPhraseTree`, 137
- `~FPhraseTreeBranchBind`
  - `FPhraseTreeBranchBind`, 142
- `~FPhraseTreeContextManager`
  - `FPhraseTreeContextManager`, 144
- `~FSocketCommunicationServer`
  - `FSocketCommunicationServer`, 149
- `~FTranscriptionVisualizer`
  - `FTranscriptionVisualizer`, 166
- `~OAEEditorAccessibilityManager`
  - `OAEEditorAccessibilityManager`, 179
- `~SAAccessibilityTranscriptionVis`
  - `SAAccessibilityTranscriptionVis`, 189
- `~SContentIndexer`
  - `SContentIndexer`, 193
- `~SIndexer`
  - `SIndexer`, 200
- `~TGraphAccessibilityNodeFactory`
  - `TGraphAccessibilityNodeFactory< T >`, 204
- `~UAccessibilityAddNodeContextMenu`
  - `UAccessibilityAddNodeContextMenu`, 211
- `~UAccessibilityGraphLocomotionContext`
  - `UAccessibilityGraphLocomotionContext`, 237
- `~UAccessibilityWindowToolbar`
  - `UAccessibilityWindowToolbar`, 247
- `~UAudioManager`
  - `UAudioManager`, 250
- `~UBAudioCapture`
  - `UBAudioCapture`, 255
- `~ULocalizedInputLibrary`
  - `ULocalizedInputLibrary`, 257
- `~UNodeInteractionLibrary`
  - `UNodeInteractionLibrary`, 263
- `~UParseEnumInput`
  - `UParseEnumInput`, 282
- `~UParseInput`
  - `UParseInput`, 284
- `~UParseIntInput`
  - `UParseIntInput`, 285
- `~UParseStringInput`
  - `UParseStringInput`, 287
- `~UPhraseTreeContextMenuObject`
  - `UPhraseTreeContextMenuObject`, 290
- `~UPhraseTreeContextObject`
  - `UPhraseTreeContextObject`, 296
- `~UPhraseTreeUtils`
  - `UPhraseTreeUtils`, 299
- `~UViewInteractionLibrary`
  - `UViewInteractionLibrary`, 302
- `~UWindowInteractionLibrary`
  - `UWindowInteractionLibrary`, 307
- `AccessibilityNodeFactory`
  - `FOpenAccessibilityModule`, 69
- `AccessibilityRegistry`
  - `TGraphAccessibilityNodeFactory< T >`, 207
- `AddNode`
  - `FGraphIndexer`, 36, 37
- `AddPhraseInput`

- FParseRecord, 76
- AddPhraseString
  - FParseRecord, 77
- AddValue
  - FIndexer< KeyType, ValueType >, 51
- AppendFilterText
  - UAccessibilityAddNodeContextMenu, 211
  - UAccessibilityGraphEditorContext, 224
- AppendScrollDistance
  - UAccessibilityAddNodeContextMenu, 211
  - UAccessibilityGraphEditorContext, 225
- ApplyAccessibilityWidget
  - UAccessibilityAddNodeContextMenu, 212
- AssetAccessibilityRegistry
  - FOpenAccessibilityModule, 69
- audio\_resampler
  - OpenAccessibilityPy.OpenAccessibilityPy, 187
- AudioManager
  - FOpenAccessibilityCommunicationModule, 66
- AvailableIndexes
  - FIndexer< KeyType, ValueType >, 59
- AvailableIndices
  - FGraphIndexer, 45
- beam\_size
  - OpenAccessibilityPy.WhisperInterface.WhisperInterface, 316
- bHasLeafChild
  - FPhraseNode, 128
- BindBranch
  - FPhraseTree, 137
- BindBranches
  - FPhraseTree, 138
  - ULocalizedInputLibrary, 257
  - UNodeInteractionLibrary, 263
  - UPhraseTreeFunctionLibrary, 298
  - UViewInteractionLibrary, 303
  - UWindowInteractionLibrary, 307
- BindChildNode
  - FPhraseNode, 121
- BindChildNodeForce
  - FPhraseNode, 121
- BindChildrenNodes
  - FPhraseNode, 122
- BindChildrenNodesForce
  - FPhraseNode, 122
- BindFocusChangedEvent
  - UAccessibilityGraphLocomotionContext, 237
- BindMenuDismissed
  - UPhraseTreeContextMenuObject, 290
- BindTickDelegate
  - UPhraseTreeContextMenuObject, 291
- blsActive
  - UPhraseTreeContextObject, 297
- BlueprintCompile
  - UNodeInteractionLibrary, 267
- BotRight
  - FPanelViewPosition, 73
- BottomRight
  - FGraphLocomotionChunk, 48
- BoundPhrase
  - FPhraseNode, 128
- BranchRoot
  - FPhraseTreeBranchBind, 142
- CalculateVisualChunksBounds
  - UAccessibilityGraphLocomotionContext, 237
- CanBindChild
  - FPhraseNode, 123
- CancelLocomotion
  - UAccessibilityGraphLocomotionContext, 238
- ChangeChunkVis
  - UAccessibilityGraphLocomotionContext, 238
- CheckBoxes
  - UAccessibilityGraphEditorContext, 234
- ChildNodes
  - FPhraseNode, 128
- ChunkArray
  - UAccessibilityGraphLocomotionContext, 244
- ChunkIndexer
  - FGraphLocomotionChunk, 48
- ChunkSize
  - UAccessibilityGraphLocomotionContext, 244
- ChunkVisWidget
  - FGraphLocomotionChunk, 48
- ChunkWidget
  - FGraphLocomotionChunk, 49
- Close
  - UAccessibilityAddNodeContextMenu, 212
  - UAccessibilityGraphEditorContext, 225
  - UAccessibilityGraphLocomotionContext, 238
  - UPhraseTreeContextMenuObject, 291
  - UPhraseTreeContextObject, 296
- CloseActiveWindow
  - UWindowInteractionLibrary, 308
- com\_server
  - OpenAccessibilityPy.OpenAccessibilityPy, 187
- ConfirmSelection
  - UAccessibilityGraphLocomotionContext, 239
- Construct
  - SAccessibilityTranscriptionVis, 189
  - SContentIndexer, 193
  - SIndexer, 201
- ConstructBottomIndexer
  - SContentIndexer, 193
- ConstructContentContainer
  - SContentIndexer, 194
- ConstructContextChildren
  - FPhraseContextMenuNode< ContextMenuType >, 93
  - FPhraseContextNode< ContextType >, 99
  - IPhraseContextNodeBase, 173
- ConstructIndexContainer
  - SContentIndexer, 195
- ConstructIndexText
  - SContentIndexer, 195
- ConstructLeftIndexer
  - SContentIndexer, 195



- ConstructRightIndexer
  - SContentIndexer, [196](#)
- ConstructTopIndexer
  - SContentIndexer, [197](#)
- ConstructVisualizer
  - FTranscriptionVisualizer, [166](#)
- ContainsKey
  - FGraphIndexer, [37](#)
  - FIndexer< KeyType, ValueType >, [52](#)
- ContainsNode
  - FGraphIndexer, [37](#), [38](#)
- ContainsValue
  - FIndexer< KeyType, ValueType >, [52](#)
- Content/Python/init\_unreal.py, [317](#)
- Content/Python/OpenAccessibilityPy/\_\_init\_\_.py, [318](#)
- Content/Python/OpenAccessibilityPy/\_\_main\_\_.py, [320](#)
- Content/Python/OpenAccessibilityPy/Audio.py, [322](#)
- Content/Python/OpenAccessibilityPy/CommunicationServer.py, [324](#)
- Content/Python/OpenAccessibilityPy/LibUtils.py, [327](#)
- Content/Python/OpenAccessibilityPy/Logging.py, [327](#)
- Content/Python/OpenAccessibilityPy/WhisperInterface.py, [328](#)
- Context
  - FSocketCommunicationServer, [163](#)
- context
  - OpenAccessibilityPy.CommunicationServer.CommunicationServer, [20](#)
- ContextAwarenessCheckBox
  - UAccessibilityAddNodeContextMenu, [221](#)
- ContextMenuScalar
  - FPhraseContextMenuNode< ContextMenuType >, [97](#)
- ContextObjectStack
  - FParseRecord, [85](#)
- ContextRoot
  - UPhraseTreeContextObject, [297](#)
- CreateAccessibilityWrapper
  - UAccessibilityGraphEditorContext, [225](#)
- CreateContextObject
  - FPhraseContextMenuNode< ContextMenuType >, [94](#)
  - FPhraseContextNode< ContextType >, [99](#)
  - IPhraseContextNodeBase, [173](#)
- CreateNode
  - FAccessibilityNodeFactory, [22](#)
- CreateNodeWidget
  - TGraphAccessibilityNodeFactory< T >, [205](#)
- CreatePinWidget
  - TGraphAccessibilityNodeFactory< T >, [206](#)
- CreateVisualGrid
  - UAccessibilityGraphLocomotionContext, [239](#)
- CurrentViewPosition
  - UAccessibilityGraphLocomotionContext, [244](#)
- DeleteNode
  - UNodeInteractionLibrary, [268](#)
- DeserializeJSON
  - FSocketCommunicationServer, [150](#)
- DoesItemsRequireRefresh
  - UAccessibilityAddNodeContextMenu, [213](#)
- DumpTick
  - FOpenAccessibilityAnalyticsModule, [60](#)
- Empty
  - FIndexer< KeyType, ValueType >, [53](#)
- EnumType
  - UParseEnumInput, [283](#)
- ERROR
  - OpenAccessibilityPy.Logging.LogLevel, [176](#)
- EventOccured
  - FSocketCommunicationServer, [150](#)
  - OpenAccessibilityPy.CommunicationServer.CommunicationServer, [13](#)
- FAccessibilityNodeFactory, [21](#)
  - ~FAccessibilityNodeFactory, [21](#)
  - CreateNode, [22](#)
  - FAccessibilityNodeFactory, [21](#)
  - SetSharedPtr, [22](#)
  - WrapNodeWidget, [23](#)
  - WrapPinWidget, [24](#)
- FAssetAccessibilityRegistry, [25](#)
  - ~FAssetAccessibilityRegistry, [26](#)
- FAssetAccessibilityRegistry, [26](#)
  - GetAllGraphIndexes, [26](#), [27](#)
  - GetAllGraphKeyIndexes, [27](#)
  - GetGraphIndexer, [28](#)
  - GraphAssetIndex, [32](#)
  - IsGameWorldAssetRegistered, [28](#)
  - IsGraphAssetRegistered, [29](#)
  - RegisterGameWorldAsset, [29](#)
  - RegisterGraphAsset, [29](#), [30](#)
  - UnregisterGameWorldAsset, [30](#)
  - UnregisterGraphAsset, [32](#)
- FAudioManagerSettings, [33](#)
  - FAudioManagerSettings, [33](#)
  - LevelThreshold, [33](#)
  - SaveName, [34](#)
  - SavePath, [34](#)
- FGraphIndexer, [34](#)
  - ~FGraphIndexer, [36](#)
  - AddNode, [36](#), [37](#)
  - AvailableIndices, [45](#)
  - ContainsKey, [37](#)
  - ContainsNode, [37](#), [38](#)
  - FGraphIndexer, [35](#)
  - GetKey, [38](#), [40](#)
  - GetNode, [40](#), [41](#)
  - GetOrAddNode, [41](#), [42](#)
  - GetPin, [42](#), [43](#)
  - IndexMap, [45](#)
  - LinkedGraph, [46](#)
  - NodeSet, [46](#)
  - OnGraphChangedHandle, [46](#)
  - OnGraphEvent, [43](#)
  - OnGraphRebuild, [44](#)
  - RemoveNode, [44](#), [45](#)

- FGraphLocomotionChunk, 46
  - BottomRight, 48
  - ChunkIndexer, 48
  - ChunkVisWidget, 48
  - ChunkWidget, 49
  - GetChunkBottomRight, 47
  - GetChunkBounds, 47
  - GetChunkTopLeft, 47
  - SetChunkBounds, 47
  - SetVisColor, 48
  - TopLeft, 49
- FilterTextBox
  - UAccessibilityAddNodeContextMenu, 221
  - UAccessibilityGraphEditorContext, 235
- FIndexer
  - FIndexer< KeyType, ValueType >, 50
- FIndexer< KeyType, ValueType >, 49
  - ~FIndexer, 50
  - AddValue, 51
  - AvailableIndexes, 59
  - ContainsKey, 52
  - ContainsValue, 52
  - Empty, 53
  - FIndexer, 50
  - GetAvailableKey, 53
  - GetKey, 54
  - GetKeyOrAddValue, 55
  - GetValue, 56
  - IndexMap, 59
  - IsEmpty, 57
  - Num, 57
  - RemoveValue, 58
  - Reset, 59
- FindGraphActionMenu
  - UAccessibilityGraphEditorContext, 226
- FindStaticComponents
  - UAccessibilityGraphEditorContext, 226
- FindTreeView
  - UAccessibilityGraphEditorContext, 227
- FOpenAccessibilityAnalyticsModule, 60
  - DumpTick, 60
  - Get, 61
  - LogEvent, 61
  - ShutdownModule, 61
  - StartupModule, 62
  - SupportsDynamicReloading, 62
- FOpenAccessibilityCommunicationModule, 62
  - AudioManager, 66
  - Get, 63
  - HandleKeyDownEvent, 63
  - OnTranscriptionRecieved, 66
  - PhraseTree, 66
  - PhraseTreeUtils, 67
  - ShutdownModule, 64
  - SocketServer, 67
  - StartupModule, 64
  - SupportsDynamicReloading, 65
  - Tick, 65
  - TranscribeWaveForm, 65
- FOpenAccessibilityModule, 67
  - AccessibilityNodeFactory, 69
  - AssetAccessibilityRegistry, 69
  - Get, 68
  - ShutdownModule, 68
  - StartupModule, 68
  - SupportsDynamicReloading, 69
- FPanelViewPosition, 72
  - BotRight, 73
  - FPanelViewPosition, 72, 73
  - operator!=, 73
  - TopLeft, 74
- FParseRecord, 74
  - ~FParseRecord, 76
  - AddPhraseInput, 76
  - AddPhraseString, 77
  - ContextObjectStack, 85
  - FParseRecord, 75, 76
  - FPhraseTree, 85
  - GetContextObj, 77, 78
  - GetContextStack, 79
  - GetPhraseInput, 79–81
  - GetPhraseInputs, 81, 82
  - GetPhraseString, 82
  - HasContextObj, 83
  - PhraseInputs, 85
  - PhraseRecord, 86
  - PopContextObj, 83, 84
  - PushContextObj, 84
  - RemoveContextObj, 84
  - RemovePhraseInput, 85
- FParseResult, 86
  - FParseResult, 87
  - ReachedNode, 87
  - Result, 87
- FPhrase2DDirectionalInputNode, 88
  - FPhrase2DDirectionalInputNode, 88, 89
- FPhraseContextMenuNode
  - FPhraseContextMenuNode< ContextMenuType >, 91–93
- FPhraseContextMenuNode< ContextMenuType >, 90
  - ~FPhraseContextMenuNode, 93
  - ConstructContextChildren, 93
  - ContextMenuScalar, 97
  - CreateContextObject, 94
  - FPhraseContextMenuNode, 91–93
  - HasContextObject, 94
  - OnGetMenu, 97
  - ParsePhrase, 95
  - ParsePhraseAsContext, 96
- FPhraseContextNode
  - FPhraseContextNode< ContextType >, 98
- FPhraseContextNode< ContextType >, 97
  - ~FPhraseContextNode, 99
  - ConstructContextChildren, 99
  - CreateContextObject, 99
  - FPhraseContextNode, 98

- HasContextObject, 100
- ParsePhrase, 100
- ParsePhraseAsContext, 101
- FPhraseDirectionalInputNode, 102
  - FPhraseDirectionalInputNode, 103
- FPhraseEnumInputNode
  - FPhraseEnumInputNode< TEnum >, 105, 106
- FPhraseEnumInputNode< TEnum >, 104
  - ~FPhraseEnumInputNode, 106
  - FPhraseEnumInputNode, 105, 106
  - MeetsInputRequirements, 106
  - RecordInput, 107
- FPhraseEventNode, 108
  - ~FPhraseEventNode, 109
  - FPhraseEventNode, 108, 109
  - IsLeafNode, 109
  - ParsePhrase, 110
  - RequiresPhrase, 110, 111
- FPhraseInputNode
  - FPhraseInputNode< InputType >, 113, 114
- FPhraseInputNode< InputType >, 111
  - ~FPhraseInputNode, 114
  - FPhraseInputNode, 113, 114
  - MeetsInputRequirements, 114
  - OnInputReceived, 117
  - ParsePhrase, 115
  - RecordInput, 116
  - RequiresPhrase, 116, 117
- FPhraseNode, 118
  - ~FPhraseNode, 120
  - bHasLeafChild, 128
  - BindChildNode, 121
  - BindChildNodeForce, 121
  - BindChildrenNodes, 122
  - BindChildrenNodesForce, 122
  - BoundPhrase, 128
  - CanBindChild, 123
  - ChildNodes, 128
  - FPhraseNode, 119, 120
  - HasLeafChild, 123
  - IsLeafNode, 124
  - OnPhraseParsed, 128
  - ParentNode, 128
  - ParseChildren, 124
  - ParsePhrase, 125
  - ParsePhraseAsContext, 126
  - ParsePhraselfRequired, 126
  - RequiresPhrase, 126, 127
- FPhrasePositionalInputNode, 129
  - FPhrasePositionalInputNode, 129, 130
- FPhraseScrollInputNode, 131
  - FPhraseScrollInputNode, 131, 132
- FPhraseStringInputNode, 133
  - ~FPhraseStringInputNode, 134
  - FPhraseStringInputNode, 133, 134
  - MeetsInputRequirements, 135
  - RecordInput, 135
- FPhraseTree, 136
  - ~FPhraseTree, 137
  - BindBranch, 137
  - BindBranches, 138
  - FParseRecord, 85
  - FPhraseTree, 137
  - FPhraseTreeContextManager, 147
  - GetContextManager, 138
  - ParsePhrase, 138
  - ParseTranscription, 139
  - Tick, 141
- FPhraseTreeBranchBind, 141
  - ~FPhraseTreeBranchBind, 142
  - BranchRoot, 142
  - FPhraseTreeBranchBind, 142
  - StartNode, 143
- FPhraseTreeContextManager, 143
  - ~FPhraseTreeContextManager, 144
  - FPhraseTree, 147
  - FPhraseTreeContextManager, 144
  - GetContextStack, 144
  - HasContextObject, 144
  - HasContextObjects, 145
  - IsEmpty, 145
  - PeekContextObject, 145
  - PopContextObject, 146
  - PushContextObject, 147
- FSocketCommunicationServer, 147
  - ~FSocketCommunicationServer, 149
  - Context, 163
  - DeserializeJSON, 150
  - EventOccured, 150
  - FSocketCommunicationServer, 149
  - Poller, 163
  - PollTimeout, 163
  - RecvAddress, 164
  - RecvArray, 151
  - RecvJson, 152
  - RecvMultipartWithMeta, 152
  - RecvSocket, 164
  - RecvString, 153
  - RecvStringMultipart, 154
  - RecvStringMultipartWithMeta, 154
  - SendAddress, 164
  - SendArrayBuffer, 155, 156
  - SendArrayMessage, 157, 158
  - SendArrayMessageWithMeta, 159, 160
  - SendJsonBuffer, 161
  - SendSocket, 164
  - SendStringBuffer, 162
  - SerializeJSON, 163
- FTranscriptionVisualizer, 165
  - ~FTranscriptionVisualizer, 166
  - ConstructVisualizer, 166
  - FTranscriptionVisualizer, 165
  - GetDisplayVisualizerPosition, 166
  - GetTopScreenVisualizerPosition, 167
  - MoveVisualizer, 167
  - OnTranscriptionRecieved, 168

- RegisterTicker, 168
- ReparentWindow, 168
- Tick, 169
- TickDelegateHandle, 170
- UnregisterTicker, 169
- UpdateVisualizer, 169
- VisContent, 170
- VisWindow, 170
- FTreeViewTickRequirements
  - UAccessibilityGraphEditorContext::FTreeViewTickRequirements, 171
- GenerateVisualChunks
  - UAccessibilityGraphLocomotionContext, 239
- Get
  - FOpenAccessibilityAnalyticsModule, 61
  - FOpenAccessibilityCommunicationModule, 63
  - FOpenAccessibilityModule, 68
- GetActiveToolkitWidget
  - UAccessibilityWindowToolbar, 247
- GetAllGraphIndexes
  - FAssetAccessibilityRegistry, 26, 27
- GetAllGraphKeyIndexes
  - FAssetAccessibilityRegistry, 27
- GetAudioCaptureNumChannels
  - UAudioManager, 251
- GetAudioCaptureSampleRate
  - UAudioManager, 251
- GetAvailableKey
  - FIndexer< KeyType, ValueType >, 53
- GetChildNodes
  - TabUtils::FOpenArea, 70
  - TabUtils::FOpenSplitter, 71
- GetChunkBottomRight
  - FGraphLocomotionChunk, 47
- GetChunkBounds
  - FGraphLocomotionChunk, 47
- GetChunkTopLeft
  - FGraphLocomotionChunk, 47
- GetContent
  - SContentIndexer, 197, 198
- GetContextManager
  - FPhraseTree, 138
- GetContextObj
  - FParseRecord, 77, 78
- GetContextRoot
  - UPhraseTreeContextObject, 296
- GetContextStack
  - FParseRecord, 79
  - FPhraseTreeContextManager, 144
- GetDisplayVisualizerPosition
  - FTranscriptionVisualizer, 166
- GetEnumType
  - UParseEnumInput, 282
- GetFilterText
  - UAccessibilityAddNodeContextMenu, 213
  - UAccessibilityGraphEditorContext, 228
- GetGraphActionFromIndex
  - UAccessibilityAddNodeContextMenu, 213, 214
- GetGraphActionFromIndexSP
  - UAccessibilityAddNodeContextMenu, 214
- GetGraphIndexer
  - FAssetAccessibilityRegistry, 28
- GetIndexText
  - SIndexer, 201
- GetIsActive
  - UPhraseTreeContextObject, 296
- GetKey
  - FGraphIndexer, 38, 40
  - FIndexer< KeyType, ValueType >, 54
- GetKeyOrAddValue
  - FIndexer< KeyType, ValueType >, 55
- GetNode
  - FGraphIndexer, 40, 41
- GetOrAddNode
  - FGraphIndexer, 41, 42
- GetPhraseInput
  - FParseRecord, 79–81
- GetPhraseInputs
  - FParseRecord, 81, 82
- GetPhraseString
  - FParseRecord, 82
- GetPin
  - FGraphIndexer, 42, 43
- GetStaticIndexOffset
  - UAccessibilityGraphEditorContext, 228
- GetTabs
  - TabUtils::FOpenStack, 72
- GetTopScreenVisualizerPosition
  - FTranscriptionVisualizer, 167
- GetTreeViewAction
  - UAccessibilityGraphEditorContext, 228
- GetValue
  - FIndexer< KeyType, ValueType >, 56
  - UParseIntInput, 285
  - UParseStringInput, 287
- GetWindow
  - UPhraseTreeContextMenuObject, 291
- GraphAssetIndex
  - FAssetAccessibilityRegistry, 32
- GraphMenu
  - UAccessibilityAddNodeContextMenu, 221
  - UAccessibilityGraphEditorContext, 235
- GridContainer
  - UAccessibilityGraphLocomotionContext, 245
- GridParent
  - UAccessibilityGraphLocomotionContext, 245
- HandleKeyDownEvent
  - FOpenAccessibilityCommunicationModule, 63
- HandleTranscriptionRequest
  - OpenAccessibilityPy.OpenAccessibilityPy, 185
- HasContextObj
  - FParseRecord, 83
- HasContextObject
  - FPhraseContextMenuNode< ContextMenuType >, 94
  - FPhraseContextNode< ContextType >, 100

- FPhraseTreeContextManager, [144](#)
  - IPhraseContextNodeBase, [173](#)
- HasContextObjects
  - FPhraseTreeContextManager, [145](#)
- HasLeafChild
  - FPhraseNode, [123](#)
  - IPhraseNodeBase, [174](#)
- HideNativeVisuals
  - UAccessibilityGraphLocomotionContext, [240](#)
- Implementation
  - TGraphAccessibilityNodeFactory< T >, [207](#)
- IndexedContent
  - SContentIndexer, [199](#)
- IndexerWidget
  - SContentIndexer, [199](#)
- IndexFocus
  - UViewInteractionLibrary, [304](#)
- IndexMap
  - FGraphIndexer, [45](#)
  - FIndexer< KeyType, ValueType >, [59](#)
- IndexTextBlock
  - SIndexer, [203](#)
- INFO
  - OpenAccessibilityPy.Logging.LogLevel, [177](#)
- Init
  - UAccessibilityAddNodeContextMenu, [215](#), [216](#)
  - UAccessibilityGraphEditorContext, [229](#)
  - UAccessibilityGraphLocomotionContext, [240](#), [241](#)
  - UPhraseTreeContextMenuObject, [291](#), [292](#)
- IPhraseContextNodeBase, [172](#)
  - ConstructContextChildren, [173](#)
  - CreateContextObject, [173](#)
  - HasContextObject, [173](#)
- IPhraseNodeBase, [174](#)
  - HasLeafChild, [174](#)
  - IsLeafNode, [174](#)
  - ParsePhrase, [175](#)
  - ParsePhraseAsContext, [175](#)
  - RequiresPhrase, [175](#)
- IsActiveToolbar
  - UAccessibilityWindowToolbar, [247](#)
- IsCapturingAudio
  - UAudioManager, [251](#)
- IsEmpty
  - FIndexer< KeyType, ValueType >, [57](#)
  - FPhraseTreeContextManager, [145](#)
- IsGameWorldAssetRegistered
  - FAssetAccessibilityRegistry, [28](#)
- IsGraphAssetRegistered
  - FAssetAccessibilityRegistry, [29](#)
- IsLeafNode
  - FPhraseEventNode, [109](#)
  - FPhraseNode, [124](#)
  - IPhraseNodeBase, [174](#)
- IsValidNumeric
  - NumericParser, [177](#)
- KeyboardInputAdd
  - ULocalizedInputLibrary, [258](#)
- KeyboardInputConfirm
  - ULocalizedInputLibrary, [259](#)
- KeyboardInputExit
  - ULocalizedInputLibrary, [260](#)
- KeyboardInputRemove
  - ULocalizedInputLibrary, [260](#)
- KeyboardInputReset
  - ULocalizedInputLibrary, [261](#)
- LevelThreshold
  - FAudioManagerSettings, [33](#)
- LinkedEditor
  - UAccessibilityGraphLocomotionContext, [245](#)
- LinkedGraph
  - FGraphIndexer, [46](#)
- LocomotionCancel
  - UNodeInteractionLibrary, [268](#)
- LocomotionConfirm
  - UNodeInteractionLibrary, [269](#)
- LocomotionRevert
  - UNodeInteractionLibrary, [269](#)
- LocomotionSelect
  - UNodeInteractionLibrary, [269](#)
- LogEvent
  - FOpenAccessibilityAnalyticsModule, [61](#)
- MeetsInputRequirements
  - FPhraseEnumInputNode< TEnum >, [106](#)
  - FPhraseInputNode< InputType >, [114](#)
  - FPhraseStringInputNode, [135](#)
- Menu
  - UPhraseTreeContextMenuObject, [294](#)
- MoveNode
  - UNodeInteractionLibrary, [270](#)
- MoveViewport
  - UAccessibilityGraphLocomotionContext, [241](#)
  - UViewInteractionLibrary, [304](#)
- MoveVisualizer
  - FTranscriptionVisualizer, [167](#)
- NodeAddMenu
  - UNodeInteractionLibrary, [271](#)
- NodeAddPinMenu
  - UNodeInteractionLibrary, [272](#)
- NodeAddScroll
  - UNodeInteractionLibrary, [273](#)
- NodeAddSearchAdd
  - UNodeInteractionLibrary, [274](#)
- NodeAddSearchRemove
  - UNodeInteractionLibrary, [275](#)
- NodeAddSearchReset
  - UNodeInteractionLibrary, [275](#)
- NodeAddSelect
  - UNodeInteractionLibrary, [275](#)
- NodeIndexFocus
  - UNodeInteractionLibrary, [276](#)
- NodeSet
  - FGraphIndexer, [46](#)

- Num
  - FIndexer< KeyType, ValueType >, 57
- NumericParser, 177
  - IsValidNumeric, 177
  - StringToNumeric, 178
- OAEditorAccessibilityManager, 178
  - ~OAEditorAccessibilityManager, 179
  - OAEditorAccessibilityManager, 179
- OnAudioReadyForTranscription
  - UAudioManager, 254
- OnDefaultDeviceChanged
  - UAudioManager, 252
- OnFocusChanged
  - UAccessibilityGraphLocomotionContext, 242
- OnGetMenu
  - FPhraseContextMenuNode< ContextMenuType >, 97
- OnGraphChangedHandle
  - FGraphIndexer, 46
- OnGraphEvent
  - FGraphIndexer, 43
- OnGraphRebuild
  - FGraphIndexer, 44
- OnInputReceived
  - FPhraseInputNode< InputType >, 117
- OnMenuDismissed
  - UPhraseTreeContextMenuObject, 292
- OnPhraseParsed
  - FPhraseNode, 128
- OnTranscriptionRecieved
  - FOpenAccessibilityCommunicationModule, 66
  - FTranscriptionVisualizer, 168
- OpenAccessibility, 179
  - OpenAccessibility, 179
- OpenAccessibilityAnalytics, 181
  - OpenAccessibilityAnalytics, 181
- OpenAccessibilityCommunication, 182
  - OpenAccessibilityCommunication, 182
- OpenAccessibilityPy.Audio.AudioResampler, 9
  - \_\_del\_\_, 9
  - \_\_init\_\_, 9
  - resample, 10
- OpenAccessibilityPy.CommunicationServer.CommunicationServer
  - 11
  - \_\_del\_\_, 12
  - \_\_init\_\_, 12
  - context, 20
  - EventOccured, 13
  - poller, 20
  - poller\_timeout\_time, 20
  - ReceiveJSON, 13
  - ReceiveMultipart, 14
  - ReceiveNDArray, 14
  - ReceiveNDArrayWithMeta, 15
  - ReceiveString, 15
  - RecieveRaw, 16
  - recv\_socket, 20
  - recv\_socket\_context, 20
  - send\_socket\_context, 20
  - SendJSON, 16
  - SendMultipart, 17
  - SendMultipartWithMeta, 17
  - SendNDArray, 18
  - SendNDArrayWithMeta, 18
  - SendString, 19
- OpenAccessibilityPy.Logging.LogLevel, 176
  - ERROR, 176
  - INFO, 177
  - WARNING, 177
- OpenAccessibilityPy.OpenAccessibilityPy, 184
  - \_\_del\_\_, 185
  - \_\_init\_\_, 184
  - audio\_resampler, 187
  - com\_server, 187
  - HandleTranscriptionRequest, 185
  - pyshutdown\_handle, 188
  - Shutdown, 186
  - Tick, 187
  - tick\_handle, 188
  - whisper\_interface, 188
  - worker\_pool, 188
- OpenAccessibilityPy.WhisperInterface.WhisperInterface, 313
  - \_\_del\_\_, 314
  - \_\_init\_\_, 314
  - beam\_size, 316
  - process\_audio\_buffer, 315
  - process\_file\_from\_dir, 315
  - whisper\_model, 316
- OpenDefaultAudioStream
  - UBAudioCapture, 255
- operator!=
  - FPanelViewPosition, 73
- ParentNode
  - FPhraseNode, 128
- ParseChildren
  - FPhraseNode, 124
- ParsePhrase
  - FPhraseContextMenuNode< ContextMenuType >, 95
- FPhraseContextNode< ContextType >, 100
  - FPhraseEventNode, 110
  - FPhraseInputNode< InputType >, 115
  - FPhraseNode, 125
  - FPhraseTree, 138
  - IPhraseNodeBase, 175
- ParsePhraseAsContext
  - FPhraseContextMenuNode< ContextMenuType >, 96
- FPhraseContextNode< ContextType >, 101
  - FPhraseNode, 126
  - IPhraseNodeBase, 175
- ParsePhraseIfRequired
  - FPhraseNode, 126
- ParseTranscription
  - FPhraseTree, 139



- PeekContextObject
  - FPhraseTreeContextManager, [145](#)
- PerformGraphAction
  - UAccessibilityAddNodeContextMenu, [217](#)
- PhraseInputs
  - FParseRecord, [85](#)
- PhraseRecord
  - FParseRecord, [86](#)
- PhraseTree
  - FOpenAccessibilityCommunicationModule, [66](#)
  - UPhraseTreeUtils, [301](#)
- PhraseTreeUtils
  - FOpenAccessibilityCommunicationModule, [67](#)
- PinConnect
  - UNodeInteractionLibrary, [276](#)
- PinDisconnect
  - UNodeInteractionLibrary, [277](#)
- Poller
  - FSocketCommunicationServer, [163](#)
- poller
  - OpenAccessibilityPy.CommunicationServer.CommunicationServer, [20](#)
- poller\_timeout\_time
  - OpenAccessibilityPy.CommunicationServer.CommunicationServer, [20](#)
- PollTimeout
  - FSocketCommunicationServer, [163](#)
- PopContextObj
  - FParseRecord, [83](#), [84](#)
- PopContextObject
  - FPhraseTreeContextManager, [146](#)
- PrevFilterString
  - UAccessibilityAddNodeContextMenu, [222](#)
- PreviousPositions
  - UAccessibilityGraphLocomotionContext, [245](#)
- PrevNumGeneratedChildren
  - UAccessibilityAddNodeContextMenu, [222](#)
  - UAccessibilityGraphEditorContext::FTreeViewTickRequirements, [171](#)
- PrevNumItemsBeingObserved
  - UAccessibilityAddNodeContextMenu, [222](#)
  - UAccessibilityGraphEditorContext::FTreeViewTickRequirements, [171](#)
- PrevScrollDistance
  - UAccessibilityAddNodeContextMenu, [222](#)
  - UAccessibilityGraphEditorContext::FTreeViewTickRequirements, [171](#)
- PrevSearchText
  - UAccessibilityGraphEditorContext::FTreeViewTickRequirements, [172](#)
- PRIVATE\_OnAudioGenerate
  - UAudioManager, [252](#)
- process\_audio\_buffer
  - OpenAccessibilityPy.WhisperInterface.WhisperInterface, [315](#)
- process\_file\_from\_dir
  - OpenAccessibilityPy.WhisperInterface.WhisperInterface, [315](#)
- PushContextObj
  - FParseRecord, [84](#)
- PushContextObject
  - FPhraseTreeContextManager, [147](#)
- pyshutdown\_handle
  - OpenAccessibilityPy.OpenAccessibilityPy, [188](#)
- ReachedNode
  - FParseResult, [87](#)
- ReceiveJSON
  - OpenAccessibilityPy.CommunicationServer.CommunicationServer, [13](#)
- ReceiveMultipart
  - OpenAccessibilityPy.CommunicationServer.CommunicationServer, [14](#)
- ReceiveNDArray
  - OpenAccessibilityPy.CommunicationServer.CommunicationServer, [14](#)
- ReceiveNDArrayWithMeta
  - OpenAccessibilityPy.CommunicationServer.CommunicationServer, [15](#)
- ReceiveString
  - OpenAccessibilityPy.CommunicationServer.CommunicationServer, [15](#)
- RecieveRaw
  - OpenAccessibilityPy.CommunicationServer.CommunicationServer, [16](#)
- RecordInput
  - FPhraseEnumInputNode< TEnum >, [107](#)
  - FPhraseInputNode< InputType >, [116](#)
  - FPhraseStringInputNode, [135](#)
- recv\_socket
  - OpenAccessibilityPy.CommunicationServer.CommunicationServer, [20](#)
- recv\_socket\_context
  - OpenAccessibilityPy.CommunicationServer.CommunicationServer, [20](#)
- RecvAddress
  - FSocketCommunicationServer, [164](#)
- RecvArray
  - FSocketCommunicationServer, [151](#)
- RecvJson
  - FSocketCommunicationServer, [152](#)
- RecvMultipartWithMeta
  - FSocketCommunicationServer, [152](#)
- RecvSocket
  - FSocketCommunicationServer, [164](#)
- RecvString
  - FSocketCommunicationServer, [153](#)
- RecvStringMultipart
  - FSocketCommunicationServer, [154](#)
- RecvStringMultipartWithMeta
  - FSocketCommunicationServer, [154](#)
- RefreshAccessibilityWidgets
  - UAccessibilityAddNodeContextMenu, [217](#)
- RegisteredLibraries
  - UPhraseTreeUtils, [301](#)
- RegisterFunctionLibrary
  - UPhraseTreeUtils, [300](#)

- RegisterGameWorldAsset
  - FAssetAccessibilityRegistry, 29
- RegisterGraphAsset
  - FAssetAccessibilityRegistry, 29, 30
- RegisterTicker
  - FTranscriptionVisualizer, 168
- RemoveContextObj
  - FParseRecord, 84
- RemoveMenuDismissed
  - UPhraseTreeContextMenuObject, 293
- RemoveNode
  - FGraphIndexer, 44, 45
- RemovePhraseInput
  - FParseRecord, 85
- RemoveTickDelegate
  - UPhraseTreeContextMenuObject, 293
- RemoveValue
  - FIndexer< KeyType, ValueType >, 58
- RemoveVisualGrid
  - UAccessibilityGraphLocomotionContext, 242
- ReparentWindow
  - FTranscriptionVisualizer, 168
- RequiresPhrase
  - FPhraseEventNode, 110, 111
  - FPhraseInputNode< InputType >, 116, 117
  - FPhraseNode, 126, 127
  - IPhraseNodeBase, 175
- resample
  - OpenAccessibilityPy.Audio.AudioResampler, 10
- Reset
  - FIndexer< KeyType, ValueType >, 59
- ResetFilterText
  - UAccessibilityAddNodeContextMenu, 218
- Result
  - FParseResult, 87
- RevertToPreviousView
  - UAccessibilityGraphLocomotionContext, 242
- SAccessibilityTranscriptionVis, 188
  - ~SAccessibilityTranscriptionVis, 189
  - Construct, 189
  - SLATE\_BEGIN\_ARGS, 190
  - Tick, 190
  - TranscriptionContainer, 191
  - TranscriptionSlots, 191
  - UpdateTopTranscription, 191
- SaveAudioBufferToWAV
  - UAudioManager, 253
- SaveName
  - FAudioManagerSettings, 34
- SavePath
  - FAudioManagerSettings, 34
- ScaleMenu
  - UAccessibilityAddNodeContextMenu, 218
  - UAccessibilityGraphEditorContext, 229
  - UPhraseTreeContextMenuObject, 293
- SContentIndexer, 192
  - ~SContentIndexer, 193
  - Construct, 193
  - ConstructBottomIndexer, 193
  - ConstructContentContainer, 194
  - ConstructIndexContainer, 195
  - ConstructIndexText, 195
  - ConstructLeftIndexer, 195
  - ConstructRightIndexer, 196
  - ConstructTopIndexer, 197
  - GetContent, 197, 198
  - IndexedContent, 199
  - IndexerWidget, 199
  - SLATE\_BEGIN\_ARGS, 198
  - Tick, 198
  - UpdateIndex, 199
- SelectAction
  - UAccessibilityGraphEditorContext, 230
- SelectChunk
  - UAccessibilityGraphLocomotionContext, 243
- SelectGraphAction
  - UAccessibilityAddNodeContextMenu, 219
- SelectionAlignment
  - UNodeInteractionLibrary, 278
- SelectionComment
  - UNodeInteractionLibrary, 278
- SelectionMove
  - UNodeInteractionLibrary, 279
- SelectionNodeToggle
  - UNodeInteractionLibrary, 280
- SelectionReset
  - UNodeInteractionLibrary, 280
- SelectionStraighten
  - UNodeInteractionLibrary, 280
- SelectToolBarItem
  - UWindowInteractionLibrary, 309
- SelectToolbarItem
  - UAccessibilityWindowToolbar, 248
- send\_socket\_context
  - OpenAccessibilityPy.CommunicationServer.CommunicationServer, 20
- SendAddress
  - FSocketCommunicationServer, 164
- SendArrayBuffer
  - FSocketCommunicationServer, 155, 156
- SendArrayMessage
  - FSocketCommunicationServer, 157, 158
- SendArrayMessageWithMeta
  - FSocketCommunicationServer, 159, 160
- SendJSON
  - OpenAccessibilityPy.CommunicationServer.CommunicationServer, 16
- SendJsonBuffer
  - FSocketCommunicationServer, 161
- SendMultipart
  - OpenAccessibilityPy.CommunicationServer.CommunicationServer, 17
- SendMultipartWithMeta
  - OpenAccessibilityPy.CommunicationServer.CommunicationServer, 17
- SendNDArray



- OpenAccessibilityPy.CommunicationServer.CommunicationServerIndex, 202
- 18
- SendNDArrayWithMeta
  - OpenAccessibilityPy.CommunicationServer.CommunicationServer, 18
- SendSocket
  - FSocketCommunicationServer, 164
- SendString
  - OpenAccessibilityPy.CommunicationServer.CommunicationServer, 19
- SendStringBuffer
  - FSocketCommunicationServer, 162
- SerializeJSON
  - FSocketCommunicationServer, 163
- SetChunkBounds
  - FGraphLocomotionChunk, 47
- SetContextRootNode
  - UPhraseTreeContextObject, 297
- SetEnumType
  - UParseEnumInput, 283
- SetFilterText
  - UAccessibilityAddNodeContextMenu, 219
  - UAccessibilityGraphEditorContext, 231
- SetMenu
  - UPhraseTreeContextMenuObject, 294
- SetPhraseTree
  - UPhraseTreeUtils, 301
- SetScrollDistance
  - UAccessibilityAddNodeContextMenu, 219
  - UAccessibilityGraphEditorContext, 231
- SetScrollDistanceBottom
  - UAccessibilityAddNodeContextMenu, 220
  - UAccessibilityGraphEditorContext, 232
- SetScrollDistanceTop
  - UAccessibilityAddNodeContextMenu, 220
  - UAccessibilityGraphEditorContext, 232
- SetSharedPtr
  - FAccessibilityNodeFactory, 22
- Settings
  - UAudioManager, 254
- SetValue
  - UParseIntInput, 286
  - UParseStringInput, 288
- SetVisColor
  - FGraphLocomotionChunk, 48
- Shutdown
  - OpenAccessibilityPy.OpenAccessibilityPy, 186
- ShutdownModule
  - FOpenAccessibilityAnalyticsModule, 61
  - FOpenAccessibilityCommunicationModule, 64
  - FOpenAccessibilityModule, 68
- SIndexer, 200
  - ~SIndexer, 200
  - Construct, 201
  - GetIndexText, 201
  - IndexTextBlock, 203
  - SLATE\_BEGIN\_ARGS, 201
  - Tick, 201
- UIndexer, 202
- SLATE\_BEGIN\_ARGS
- SAccessibilityTranscriptionVis, 190
- SContentIndexer, 198
- SIndexer, 201
- SocketServer
  - FOpenAccessibilityCommunicationModule, 67
- Source/OpenAccessibility/OpenAccessibility.Build.cs, 329
- Source/OpenAccessibility/Private/AccessibilityWidgets/SAccessibilityTranscriptionVis.h, 330
- Source/OpenAccessibility/Private/AccessibilityWidgets/SContentIndexer.cpp, 331
- Source/OpenAccessibility/Private/AccessibilityWidgets/SIndexer.cpp, 333
- Source/OpenAccessibility/Private/AccessibilityWrappers/AccessibilityAddNodeContextMenu.h, 334
- Source/OpenAccessibility/Private/AccessibilityWrappers/AccessibilityGraphEditorContext.h, 338
- Source/OpenAccessibility/Private/AccessibilityWrappers/AccessibilityGraphEditorContext.h, 343
- Source/OpenAccessibility/Private/AccessibilityWrappers/AccessibilityWindowInteractionLibrary.h, 347
- Source/OpenAccessibility/Private/AssetAccessibilityRegistry.cpp, 351
- Source/OpenAccessibility/Private/GraphIndexer.cpp, 354
- Source/OpenAccessibility/Private/OAccessibilityNodeFactory.cpp, 357
- Source/OpenAccessibility/Private/OAEditorAccessibilityManager.cpp, 360
- Source/OpenAccessibility/Private/OpenAccessibility.cpp, 360
- Source/OpenAccessibility/Private/PhraseEvents/LocalizedInputLibrary.cpp, 365
- Source/OpenAccessibility/Private/PhraseEvents/NodeInteractionLibrary.cpp, 368
- Source/OpenAccessibility/Private/PhraseEvents/ViewInteractionLibrary.cpp, 379
- Source/OpenAccessibility/Private/PhraseEvents/WindowInteractionLibrary.h, 381
- Source/OpenAccessibility/Private/TranscriptionVisualizer.cpp, 386
- Source/OpenAccessibility/Private/Utils/WidgetUtils.h, 388
- Source/OpenAccessibility/Public/AccessibilityNodeFactory.h, 390
- Source/OpenAccessibility/Public/AccessibilityWidgets/SAccessibilityTranscriptionVis.h, 392
- Source/OpenAccessibility/Public/AccessibilityWidgets/SContentIndexer.h, 393
- Source/OpenAccessibility/Public/AccessibilityWidgets/SIndexer.h, 394
- Source/OpenAccessibility/Public/AccessibilityWrappers/AccessibilityAddNodeContextMenu.h, 394
- Source/OpenAccessibility/Public/AccessibilityWrappers/AccessibilityGraphEditorContext.h, 396
- Source/OpenAccessibility/Public/AccessibilityWrappers/AccessibilityGraphEditorContext.h, 396

[397](#) Source/OpenAccessibilityCommunication/Private/PhraseTree/Utils.cpp,  
 Source/OpenAccessibility/Public/AccessibilityWrappers/AccessibilityWindowToolBar.h,  
[399](#) Source/OpenAccessibilityCommunication/Private/PhraseTreeUtils.cpp,  
 Source/OpenAccessibility/Public/AssetAccessibilityRegistry.h, [431](#)  
[400](#) Source/OpenAccessibilityCommunication/Private/SocketCommunicationS  
 Source/OpenAccessibility/Public/GraphIndexer.h, [401](#) [432](#)  
 Source/OpenAccessibility/Public/Indexers/Indexer.h, Source/OpenAccessibilityCommunication/Private/UBAudioCapture.cpp,  
[402](#) [437](#)  
 Source/OpenAccessibility/Public/OAccessibilityNodeFactory.h, Source/OpenAccessibilityCommunication/Public/AudioManager.h,  
[405](#) [438](#)  
 Source/OpenAccessibility/Public/OAEditorAccessibilityManager.h, Source/OpenAccessibilityCommunication/Public/OpenAccessibilityComLo  
[405](#) [439](#)  
 Source/OpenAccessibility/Public/OpenAccessibility.h, Source/OpenAccessibilityCommunication/Public/OpenAccessibilityCommu  
[405](#) [439](#)  
 Source/OpenAccessibility/Public/OpenAccessibilityLogging.h, Source/OpenAccessibilityCommunication/Public/PhraseTree.h,  
[406](#) [440](#)  
 Source/OpenAccessibility/Public/PhraseEvents/LocalizedInput.h, Source/OpenAccessibilityCommunication/Public/PhraseTree/Containers/C  
[406](#) [443](#)  
 Source/OpenAccessibility/Public/PhraseEvents/NodeInteraction.h, Source/OpenAccessibilityCommunication/Public/PhraseTree/Containers/C  
[407](#) [444](#)  
 Source/OpenAccessibility/Public/PhraseEvents/Utils.h, Source/OpenAccessibilityCommunication/Public/PhraseTree/Containers/In  
[460](#) [444](#)  
 Source/OpenAccessibility/Public/PhraseEvents/ViewInteraction.h, Source/OpenAccessibilityCommunication/Public/PhraseTree/Containers/In  
[408](#) [445](#)  
 Source/OpenAccessibility/Public/PhraseEvents/WindowInteraction.h, Source/OpenAccessibilityCommunication/Public/PhraseTree/Containers/In  
[409](#) [446](#)  
 Source/OpenAccessibility/Public/TranscriptionVisualizer.h, Source/OpenAccessibilityCommunication/Public/PhraseTree/Containers/In  
[410](#) [446](#)  
 Source/OpenAccessibilityAnalytics/OpenAccessibilityAnalyticsBuilder.h, Source/OpenAccessibilityCommunication/Public/PhraseTree/Containers/In  
[410](#) [447](#)  
 Source/OpenAccessibilityAnalytics/Private/OpenAccessibilityAnalyticsBuilder.h, Source/OpenAccessibilityCommunication/Public/PhraseTree/Containers/P  
[411](#) [447](#)  
 Source/OpenAccessibilityAnalytics/Private/OpenAccessibilityAnalyticsBuilder.h, Source/OpenAccessibilityCommunication/Public/PhraseTree/Containers/P  
[413](#) [450](#)  
 Source/OpenAccessibilityAnalytics/Public/OpenAccessibilityAnalyticsBuilder.h, Source/OpenAccessibilityCommunication/Public/PhraseTree/IPhraseConte  
[413](#) [451](#)  
 Source/OpenAccessibilityCommunication/OpenAccessibilityCommunication.h, Source/OpenAccessibilityCommunication/Public/PhraseTree/PhraseConte  
[414](#) [451](#)  
 Source/OpenAccessibilityCommunication/Private/AudioManager.cpp, Source/OpenAccessibilityCommunication/Public/PhraseTree/PhraseConte  
[415](#) [454](#)  
 Source/OpenAccessibilityCommunication/Private/OpenAccessibilityCommunication.h, Source/OpenAccessibilityCommunication/Public/PhraseTree/PhraseDirect  
[417](#) [456](#)  
 Source/OpenAccessibilityCommunication/Private/OpenAccessibilityCommunication.h, Source/OpenAccessibilityCommunication/Public/PhraseTree/PhraseEnum  
[417](#) [457](#)  
 Source/OpenAccessibilityCommunication/Private/PhraseTree.cpp, Source/OpenAccessibilityCommunication/Public/PhraseTree/PhraseEvent  
[420](#) [458](#)  
 Source/OpenAccessibilityCommunication/Private/PhraseTree.cpp, Source/OpenAccessibilityCommunication/Public/PhraseTree/PhraseInputN  
[422](#) [458](#)  
 Source/OpenAccessibilityCommunication/Private/PhraseTree.cpp, Source/OpenAccessibilityCommunication/Public/PhraseTree/PhraseNode.  
[423](#) [459](#)  
 Source/OpenAccessibilityCommunication/Private/PhraseTree.cpp, Source/OpenAccessibilityCommunication/Public/PhraseTree/PhraseString  
[425](#) [460](#)  
 Source/OpenAccessibilityCommunication/Private/PhraseTree.cpp, Source/OpenAccessibilityCommunication/Public/PhraseTree/PhraseTreeF  
[425](#) [460](#)  
 Source/OpenAccessibilityCommunication/Private/PhraseTree.cpp, Source/OpenAccessibilityCommunication/Public/PhraseTree/Utils.h,  
[427](#) [463](#)  
 Source/OpenAccessibilityCommunication/Private/PhraseTree.cpp, Source/OpenAccessibilityCommunication/Public/PhraseTreeUtils.h,  
[430](#) [463](#)

- Source/OpenAccessibilityCommunication/Public/SocketCommunication/PhraseTreeContextMenuItemObject, 294
- 464
- Source/OpenAccessibilityCommunication/Public/UBAudioCapture/OpenAccessibilityPy.OpenAccessibilityPy, 188
- 465
- StartCapturingAudio
  - UAudioManager, 253
- StartNode
  - FPhraseTreeBranchBind, 143
- StartupModule
  - FOpenAccessibilityAnalyticsModule, 62
  - FOpenAccessibilityCommunicationModule, 64
  - FOpenAccessibilityModule, 68
- StartViewPosition
  - UAccessibilityGraphLocomotionContext, 245
- StartViewZoom
  - UAccessibilityGraphLocomotionContext, 245
- StopCapturingAudio
  - UAudioManager, 253
- StringToNumeric
  - NumericParser, 178
- SupportsDynamicReloading
  - FOpenAccessibilityAnalyticsModule, 62
  - FOpenAccessibilityCommunicationModule, 65
  - FOpenAccessibilityModule, 69
- SwitchNextActiveWindow
  - UWindowInteractionLibrary, 309
- SwitchNextTabInStack
  - UWindowInteractionLibrary, 310
- SwitchPrevActiveWindow
  - UWindowInteractionLibrary, 311
- SwitchPrevTabInStack
  - UWindowInteractionLibrary, 312
- TabUtils::FOpenArea, 70
  - GetChildNodes, 70
- TabUtils::FOpenSplitter, 71
  - GetChildNodes, 71
- TabUtils::FOpenStack, 71
  - GetTabs, 72
- TGraphAccessibilityNodeFactory
  - TGraphAccessibilityNodeFactory< T >, 204
- TGraphAccessibilityNodeFactory< T >, 203
  - ~TGraphAccessibilityNodeFactory, 204
  - AccessibilityRegistry, 207
  - CreateNodeWidget, 205
  - CreatePinWidget, 206
  - Implementation, 207
  - TGraphAccessibilityNodeFactory, 204
- Tick
  - FOpenAccessibilityCommunicationModule, 65
  - FPhraseTree, 141
  - FTranscriptionVisualizer, 169
  - OpenAccessibilityPy.OpenAccessibilityPy, 187
  - SAccessibilityTranscriptionVis, 190
  - SContentIndexer, 198
  - SIndexer, 201
  - UAccessibilityAddNodeContextMenu, 220
  - UAccessibilityGraphEditorContext, 232
  - UAccessibilityWindowToolbar, 248
  - UPhraseTreeContextMenuItemObject, 294
  - tick\_handle
  - OpenAccessibilityPy.OpenAccessibilityPy, 188
  - TickDelegateHandle
    - FTranscriptionVisualizer, 170
  - TickTreeViewAccessibility
    - UAccessibilityGraphEditorContext, 232
  - ToggleContextAwareness
    - UAccessibilityAddNodeContextMenu, 220
  - TopLeft
    - FGraphLocomotionChunk, 49
    - FPanelViewPosition, 74
  - TranscribeWaveForm
    - FOpenAccessibilityCommunicationModule, 65
  - TranscriptionContainer
    - SAccessibilityTranscriptionVis, 191
  - TranscriptionSlots
    - SAccessibilityTranscriptionVis, 191
  - TreeView
    - UAccessibilityAddNodeContextMenu, 222
    - UAccessibilityGraphEditorContext, 235
  - TreeViewCanTick
    - UAccessibilityGraphEditorContext, 233
  - TreeViewRequiresTick
    - UAccessibilityGraphEditorContext, 233
  - TreeViewTickRequirements
    - UAccessibilityGraphEditorContext, 235
  - UAccessibilityAddNodeContextMenu, 208
    - ~UAccessibilityAddNodeContextMenu, 211
    - AppendFilterText, 211
    - AppendScrollDistance, 211
    - ApplyAccessibilityWidget, 212
    - Close, 212
    - ContextAwarenessCheckBox, 221
    - DoesItemsRequireRefresh, 213
    - FilterTextBox, 221
    - GetFilterText, 213
    - GetGraphActionFromIndex, 213, 214
    - GetGraphActionFromIndexSP, 214
    - GraphMenu, 221
    - Init, 215, 216
    - PerformGraphAction, 217
    - PrevFilterString, 222
    - PrevNumGeneratedChildren, 222
    - PrevNumItemsBeingObserved, 222
    - PrevScrollDistance, 222
    - RefreshAccessibilityWidgets, 217
    - ResetFilterText, 218
    - ScaleMenu, 218
    - SelectGraphAction, 219
    - SetFilterText, 219
    - SetScrollDistance, 219
    - SetScrollDistanceBottom, 220
    - SetScrollDistanceTop, 220
    - Tick, 220
    - ToggleContextAwareness, 220
    - TreeView, 222
    - UAccessibilityAddNodeContextMenu, 210

- UpdateAccessibilityWidget, 221
- UAccessibilityGraphEditorContext, 223
  - AppendFilterText, 224
  - AppendScrollDistance, 225
  - CheckBoxes, 234
  - Close, 225
  - CreateAccessibilityWrapper, 225
  - FilterTextBox, 235
  - FindGraphActionMenu, 226
  - FindStaticComponents, 226
  - FindTreeView, 227
  - GetFilterText, 228
  - GetStaticIndexOffset, 228
  - GetTreeViewAction, 228
  - GraphMenu, 235
  - Init, 229
  - ScaleMenu, 229
  - SelectAction, 230
  - SetFilterText, 231
  - SetScrollDistance, 231
  - SetScrollDistanceBottom, 232
  - SetScrollDistanceTop, 232
  - Tick, 232
  - TickTreeViewAccessibility, 232
  - TreeView, 235
  - TreeViewCanTick, 233
  - TreeViewRequiresTick, 233
  - TreeViewTickRequirements, 235
  - UAccessibilityGraphEditorContext, 224
  - UpdateAccessibilityWidget, 234
- UAccessibilityGraphEditorContext::FTreeViewTickRequirements
  - 171
  - FTreeViewTickRequirements, 171
  - PrevNumGeneratedChildren, 171
  - PrevNumItemsBeingObserved, 171
  - PrevScrollDistance, 171
  - PrevSearchText, 172
- UAccessibilityGraphLocomotionContext, 235
  - ~UAccessibilityGraphLocomotionContext, 237
  - BindFocusChangedEvent, 237
  - CalculateVisualChunksBounds, 237
  - CancelLocomotion, 238
  - ChangeChunkVis, 238
  - ChunkArray, 244
  - ChunkSize, 244
  - Close, 238
  - ConfirmSelection, 239
  - CreateVisualGrid, 239
  - CurrentViewPosition, 244
  - GenerateVisualChunks, 239
  - GridContainer, 245
  - GridParent, 245
  - HideNativeVisuals, 240
  - Init, 240, 241
  - LinkedEditor, 245
  - MoveViewport, 241
  - OnFocusChanged, 242
  - PreviousPositions, 245
  - RemoveVisualGrid, 242
  - RevertToPreviousView, 242
  - SelectChunk, 243
  - StartViewPosition, 245
  - StartViewZoom, 245
  - UAccessibilityGraphLocomotionContext, 236
  - UnbindFocusChangedEvent, 243
  - UnHideNativeVisuals, 244
- UAccessibilityWindowToolbar, 246
  - ~UAccessibilityWindowToolbar, 247
  - GetActiveToolkitWidget, 247
  - IsActiveToolbar, 247
  - SelectToolbarItem, 248
  - Tick, 248
  - UAccessibilityWindowToolbar, 246
- UAudioManager, 249
  - ~UAudioManager, 250
  - GetAudioCaptureNumChannels, 251
  - GetAudioCaptureSampleRate, 251
  - IsCapturingAudio, 251
  - OnAudioReadyForTranscription, 254
  - OnDefaultDeviceChanged, 252
  - PRIVATE\_OnAudioGenerate, 252
  - SaveAudioBufferToWAV, 253
  - Settings, 254
  - StartCapturingAudio, 253
  - StopCapturingAudio, 253
  - UAudioManager, 250
- UBAudioCapture, 254
  - ~UBAudioCapture, 255
  - OpenDefaultAudioStream, 255
  - UBAudioCapture, 255
- ULocalizedInputLibrary, 256
  - ~ULocalizedInputLibrary, 257
  - BindBranches, 257
  - KeyboardInputAdd, 258
  - KeyboardInputConfirm, 259
  - KeyboardInputExit, 260
  - KeyboardInputRemove, 260
  - KeyboardInputReset, 261
  - ULocalizedInputLibrary, 257
- UnbindFocusChangedEvent
  - UAccessibilityGraphLocomotionContext, 243
- UnHideNativeVisuals
  - UAccessibilityGraphLocomotionContext, 244
- UNodeInteractionLibrary, 262
  - ~UNodeInteractionLibrary, 263
  - BindBranches, 263
  - BlueprintCompile, 267
  - DeleteNode, 268
  - LocomotionCancel, 268
  - LocomotionConfirm, 269
  - LocomotionRevert, 269
  - LocomotionSelect, 269
  - MoveNode, 270
  - NodeAddMenu, 271
  - NodeAddPinMenu, 272
  - NodeAddScroll, 273

- NodeAddSearchAdd, [274](#)
- NodeAddSearchRemove, [275](#)
- NodeAddSearchReset, [275](#)
- NodeAddSelect, [275](#)
- NodeIndexFocus, [276](#)
- PinConnect, [276](#)
- PinDisconnect, [277](#)
- SelectionAlignment, [278](#)
- SelectionComment, [278](#)
- SelectionMove, [279](#)
- SelectionNodeToggle, [280](#)
- SelectionReset, [280](#)
- SelectionStraighten, [280](#)
- UNodeInteractionLibrary, [263](#)
- UnregisterGameWorldAsset
  - FAssetAccessibilityRegistry, [30](#)
- UnregisterGraphAsset
  - FAssetAccessibilityRegistry, [32](#)
- UnregisterTicker
  - FTranscriptionVisualizer, [169](#)
- UParseEnumInput, [281](#)
  - ~UParseEnumInput, [282](#)
  - EnumType, [283](#)
  - GetEnumType, [282](#)
  - SetEnumType, [283](#)
- UParseInput, [283](#)
  - ~UParseInput, [284](#)
- UParseIntInput, [284](#)
  - ~UParseIntInput, [285](#)
  - GetValue, [285](#)
  - SetValue, [286](#)
  - Value, [286](#)
- UParseStringInput, [286](#)
  - ~UParseStringInput, [287](#)
  - GetValue, [287](#)
  - SetValue, [288](#)
  - Value, [288](#)
- UpdateAccessibilityWidget
  - UAccessibilityAddNodeContextMenu, [221](#)
  - UAccessibilityGraphEditorContext, [234](#)
- UpdateIndex
  - SContentIndexer, [199](#)
  - SIndexer, [202](#)
- UpdateTopTranscription
  - SAccessibilityTranscriptionVis, [191](#)
- UpdateVisualizer
  - FTranscriptionVisualizer, [169](#)
- UPhraseTreeContextMenuObject, [288](#)
  - ~UPhraseTreeContextMenuObject, [290](#)
  - BindMenuDismissed, [290](#)
  - BindTickDelegate, [291](#)
  - Close, [291](#)
  - GetWindow, [291](#)
  - Init, [291](#), [292](#)
  - Menu, [294](#)
  - OnMenuDismissed, [292](#)
  - RemoveMenuDismissed, [293](#)
  - RemoveTickDelegate, [293](#)
  - ScaleMenu, [293](#)
  - SetMenu, [294](#)
  - Tick, [294](#)
  - UPhraseTreeContextMenuObject, [289](#), [290](#)
  - Window, [294](#)
- UPhraseTreeContextObject, [295](#)
  - ~UPhraseTreeContextObject, [296](#)
  - blsActive, [297](#)
  - Close, [296](#)
  - ContextRoot, [297](#)
  - GetContextRoot, [296](#)
  - GetIsActive, [296](#)
  - SetContextRootNode, [297](#)
  - UPhraseTreeContextObject, [296](#)
- UPhraseTreeFunctionLibrary, [298](#)
  - BindBranches, [298](#)
- UPhraseTreeUtils, [299](#)
  - ~UPhraseTreeUtils, [299](#)
  - PhraseTree, [301](#)
  - RegisteredLibraries, [301](#)
  - RegisterFunctionLibrary, [300](#)
  - SetPhraseTree, [301](#)
  - UPhraseTreeUtils, [299](#)
- UViewInteractionLibrary, [302](#)
  - ~UViewInteractionLibrary, [302](#)
  - BindBranches, [303](#)
  - IndexFocus, [304](#)
  - MoveViewport, [304](#)
  - UViewInteractionLibrary, [302](#)
  - ZoomViewport, [305](#)
- UWindowInteractionLibrary, [306](#)
  - ~UWindowInteractionLibrary, [307](#)
  - BindBranches, [307](#)
  - CloseActiveWindow, [308](#)
  - SelectToolBarItem, [309](#)
  - SwitchNextActiveWindow, [309](#)
  - SwitchNextTabInStack, [310](#)
  - SwitchPrevActiveWindow, [311](#)
  - SwitchPrevTabInStack, [312](#)
  - UWindowInteractionLibrary, [307](#)
  - WindowToolBar, [313](#)
- Value
  - UParseIntInput, [286](#)
  - UParseStringInput, [288](#)
- VisContent
  - FTranscriptionVisualizer, [170](#)
- VisWindow
  - FTranscriptionVisualizer, [170](#)
- WARNING
  - OpenAccessibilityPy.Logging.LogLevel, [177](#)
- whisper\_interface
  - OpenAccessibilityPy.OpenAccessibilityPy, [188](#)
- whisper\_model
  - OpenAccessibilityPy.WhisperInterface.WhisperInterface, [316](#)
- Window
  - UPhraseTreeContextMenuObject, [294](#)

WindowToolBar  
    UWindowInteractionLibrary, [313](#)  
worker\_pool  
    OpenAccessibilityPy.OpenAccessibilityPy, [188](#)  
WrapNodeWidget  
    FAccessibilityNodeFactory, [23](#)  
WrapPinWidget  
    FAccessibilityNodeFactory, [24](#)  
  
ZoomViewport  
    UViewInteractionLibrary, [305](#)