

## AMORE++

pre-alpha (active development aiming to release a beta version this summer (2011) )

Generated by Doxygen 1.7.4

Mon Jun 20 2011 15:56:11



# Contents

|          |  |          |
|----------|--|----------|
| <b>1</b> | <b>The AMORE++ package</b>                           | <b>1</b> |
| 1.1      | Introduction . . . . .                               | 1        |
| 1.2      | Motivation . . . . .                                 | 1        |
| 1.3      | Road Map . . . . .                                   | 1        |
| <b>2</b> | <b>Class Index</b>                                   | <b>3</b> |
| 2.1      | Class Hierarchy . . . . .                            | 3        |
| <b>3</b> | <b>Class Index</b>                                   | <b>5</b> |
| 3.1      | Class List . . . . .                                 | 5        |
| <b>4</b> | <b>File Index</b>                                    | <b>7</b> |
| 4.1      | File List . . . . .                                  | 7        |
| <b>5</b> | <b>Class Documentation</b>                           | <b>9</b> |
| 5.1      | ADAPTgdTrainingVariables Class Reference . . . . .   | 9        |
| 5.1.1    | Detailed Description . . . . .                       | 10       |
| 5.1.2    | Member Data Documentation . . . . .                  | 10       |
| 5.1.2.1  | output . . . . .                                     | 10       |
| 5.1.2.2  | outputDerivative . . . . .                           | 10       |
| 5.2      | ADAPTgdwmTrainingVariables Class Reference . . . . . | 11       |
| 5.2.1    | Detailed Description . . . . .                       | 12       |
| 5.2.2    | Member Data Documentation . . . . .                  | 12       |
| 5.2.2.1  | output . . . . .                                     | 12       |
| 5.2.2.2  | outputDerivative . . . . .                           | 12       |
| 5.3      | BATCHgdTrainingVariables Class Reference . . . . .   | 13       |
| 5.3.1    | Detailed Description . . . . .                       | 14       |

|         |  |    |
|---------|--|----|
| 5.3.2   | Member Data Documentation . . . . .                  | 14 |
| 5.3.2.1 | output . . . . .                                     | 14 |
| 5.3.2.2 | outputDerivative . . . . .                           | 14 |
| 5.4     | BATCHgdwmTrainingVariables Class Reference . . . . . | 15 |
| 5.4.1   | Detailed Description . . . . .                       | 16 |
| 5.4.2   | Member Data Documentation . . . . .                  | 16 |
| 5.4.2.1 | output . . . . .                                     | 16 |
| 5.4.2.2 | outputDerivative . . . . .                           | 16 |
| 5.5     | CompareId Struct Reference . . . . .                 | 17 |
| 5.5.1   | Detailed Description . . . . .                       | 17 |
| 5.5.2   | Member Function Documentation . . . . .              | 17 |
| 5.5.2.1 | operator() . . . . .                                 | 17 |
| 5.5.2.2 | operator() . . . . .                                 | 17 |
| 5.5.2.3 | operator() . . . . .                                 | 17 |
| 5.5.2.4 | operator() . . . . .                                 | 18 |
| 5.6     | Con Class Reference . . . . .                        | 18 |
| 5.6.1   | Detailed Description . . . . .                       | 19 |
| 5.6.2   | Constructor & Destructor Documentation . . . . .     | 19 |
| 5.6.2.1 | Con . . . . .  | 19 |
| 5.6.2.2 | Con . . . . .  | 20 |
| 5.6.2.3 | Con . . . . .  | 20 |
| 5.6.2.4 | Con . . . . .  | 20 |
| 5.6.2.5 | Con . . . . .  | 20 |
| 5.6.2.6 | ~Con . . . . .                                       | 20 |
| 5.6.3   | Member Function Documentation . . . . .              | 21 |
| 5.6.3.1 | getFrom . . . . .                                    | 21 |
| 5.6.3.2 | getId . . . . .                                      | 21 |
| 5.6.3.3 | getWeight . . . . .                                  | 22 |
| 5.6.3.4 | Id . . . . .   | 23 |
| 5.6.3.5 | neuron . . . . .                                     | 24 |
| 5.6.3.6 | setFrom . . . . .                                    | 25 |
| 5.6.3.7 | setWeight . . . . .                                  | 25 |
| 5.6.3.8 | show . . . . .                                       | 26 |
| 5.6.3.9 | show . . . . .                                       | 26 |

|          |   |    |
|----------|---|----|
| 5.6.3.10 | <a href="#">validate</a>                                      | 27 |
| 5.6.3.11 | <a href="#">validate</a>                                      | 28 |
| 5.6.3.12 | <a href="#">weight</a>  | 28 |
| 5.6.4    | <a href="#">Member Data Documentation</a>                     | 28 |
| 5.6.4.1  | <a href="#">d_neuron</a>                                      | 29 |
| 5.6.4.2  | <a href="#">d_weight</a>                                      | 29 |
| 5.6.4.3  | <a href="#">from</a>  | 29 |
| 5.6.4.4  | <a href="#">weight</a>  | 29 |
| 5.7      | <a href="#">ConContainer Class Reference</a>                  | 30 |
| 5.7.1    | <a href="#">Detailed Description</a>                          | 33 |
| 5.7.2    | <a href="#">Member Typedef Documentation</a>                  | 33 |
| 5.7.2.1  | <a href="#">const_iterator</a>                                | 33 |
| 5.7.2.2  | <a href="#">const_reference</a>                               | 34 |
| 5.7.2.3  | <a href="#">iterator</a>                                      | 34 |
| 5.7.2.4  | <a href="#">value_type</a>                                    | 34 |
| 5.7.3    | <a href="#">Constructor &amp; Destructor Documentation</a>    | 34 |
| 5.7.3.1  | <a href="#">ConContainer</a>                                  | 34 |
| 5.7.3.2  | <a href="#">ConContainer</a>                                  | 34 |
| 5.7.4    | <a href="#">Member Function Documentation</a>                 | 34 |
| 5.7.4.1  | <a href="#">erase</a>   | 34 |
| 5.7.4.2  | <a href="#">getId</a>   | 36 |
| 5.7.4.3  | <a href="#">numOfCons</a>                                     | 38 |
| 5.7.4.4  | <a href="#">select</a>  | 39 |
| 5.7.4.5  | <a href="#">setFrom</a>                                       | 41 |
| 5.7.4.6  | <a href="#">setWeight</a>                                     | 43 |
| 5.7.4.7  | <a href="#">setWeight</a>                                     | 45 |
| 5.7.4.8  | <a href="#">validate</a>                                      | 46 |
| 5.8      | <a href="#">Container&lt; T &gt; Class Template Reference</a> | 47 |
| 5.8.1    | <a href="#">Detailed Description</a>                          | 51 |
| 5.8.2    | <a href="#">Member Typedef Documentation</a>                  | 51 |
| 5.8.2.1  | <a href="#">const_iterator</a>                                | 51 |
| 5.8.2.2  | <a href="#">const_reference</a>                               | 51 |
| 5.8.2.3  | <a href="#">iterator</a>                                      | 51 |
| 5.8.2.4  | <a href="#">value_type</a>                                    | 51 |

|          |  |    |
|----------|--|----|
| 5.8.3    | Constructor & Destructor Documentation . . . . .           | 52 |
| 5.8.3.1  | Container . . . . .  | 52 |
| 5.8.3.2  | ~Container . . . . .                                       | 52 |
| 5.8.3.3  | Container . . . . .  | 52 |
| 5.8.3.4  | Container . . . . .  | 52 |
| 5.8.4    | Member Function Documentation . . . . .                    | 52 |
| 5.8.4.1  | append . . . . .   | 52 |
| 5.8.4.2  | begin . . . . .  | 54 |
| 5.8.4.3  | clear . . . . .  | 54 |
| 5.8.4.4  | clear . . . . .  | 55 |
| 5.8.4.5  | createIterator . . . . .                                   | 55 |
| 5.8.4.6  | empty . . . . .  | 55 |
| 5.8.4.7  | empty . . . . .  | 56 |
| 5.8.4.8  | end . . . . .  | 56 |
| 5.8.4.9  | load . . . . .   | 56 |
| 5.8.4.10 | operator[] . . . . .                                       | 57 |
| 5.8.4.11 | push_back . . . . .  | 57 |
| 5.8.4.12 | push_back . . . . .  | 58 |
| 5.8.4.13 | reserve . . . . .  | 59 |
| 5.8.4.14 | reserve . . . . .  | 59 |
| 5.8.4.15 | resize . . . . .   | 60 |
| 5.8.4.16 | show . . . . .   | 60 |
| 5.8.4.17 | show . . . . .   | 60 |
| 5.8.4.18 | size . . . . .   | 61 |
| 5.8.4.19 | size . . . . .   | 61 |
| 5.8.4.20 | store . . . . .  | 62 |
| 5.8.4.21 | validate . . . . .   | 62 |
| 5.8.4.22 | validate . . . . .   | 62 |
| 5.8.5    | Friends And Related Function Documentation . . . . .       | 63 |
| 5.8.5.1  | ContainerIterator< T > . . . . .                           | 63 |
| 5.8.6    | Member Data Documentation . . . . .                        | 63 |
| 5.8.6.1  | collection . . . . .                                       | 63 |
| 5.8.6.2  | collection . . . . .                                       | 63 |
| 5.9      | ContainerInterface< T > Class Template Reference . . . . . | 63 |

|          |   |    |
|----------|---|----|
| 5.9.1    | Detailed Description                            | 65 |
| 5.9.2    | Constructor & Destructor Documentation          | 65 |
| 5.9.2.1  | ~ContainerInterface                             | 65 |
| 5.9.2.2  | ContainerInterface                              | 65 |
| 5.9.3    | Member Function Documentation                   | 66 |
| 5.9.3.1  | clear   | 66 |
| 5.9.3.2  | createIterator                                  | 66 |
| 5.9.3.3  | empty   | 66 |
| 5.9.3.4  | push_back                                       | 66 |
| 5.9.3.5  | reserve   | 66 |
| 5.9.3.6  | show  | 66 |
| 5.9.3.7  | size  | 67 |
| 5.9.3.8  | validate  | 67 |
| 5.10     | ContainerIterator< T > Class Template Reference | 67 |
| 5.10.1   | Detailed Description                            | 70 |
| 5.10.2   | Constructor & Destructor Documentation          | 70 |
| 5.10.2.1 | ContainerIterator                               | 70 |
| 5.10.2.2 | ~ContainerIterator                              | 70 |
| 5.10.3   | Member Function Documentation                   | 70 |
| 5.10.3.1 | currentItem                                     | 70 |
| 5.10.3.2 | first   | 71 |
| 5.10.3.3 | isDone  | 71 |
| 5.10.3.4 | next  | 71 |
| 5.10.4   | Friends And Related Function Documentation      | 71 |
| 5.10.4.1 | Container< T >                                  | 71 |
| 5.10.5   | Member Data Documentation                       | 71 |
| 5.10.5.1 | d_container                                     | 71 |
| 5.10.5.2 | d_iterator                                      | 72 |
| 5.11     | IteratorInterface< T > Class Template Reference | 72 |
| 5.11.1   | Detailed Description                            | 73 |
| 5.11.2   | Constructor & Destructor Documentation          | 74 |
| 5.11.2.1 | ~IteratorInterface                              | 74 |
| 5.11.2.2 | IteratorInterface                               | 74 |
| 5.11.3   | Member Function Documentation                   | 74 |

|          |                                    |    |
|----------|------------------------------------|----|
| 5.11.3.1 | currentItem                        | 74 |
| 5.11.3.2 | first                              | 74 |
| 5.11.3.3 | isDone                             | 74 |
| 5.11.3.4 | next                               | 74 |
| 5.12     | Layer Class Reference              | 75 |
| 5.12.1   | Detailed Description               | 77 |
| 5.12.2   | Member Function Documentation      | 77 |
| 5.12.2.1 | show                               | 77 |
| 5.12.2.2 | validate                           | 77 |
| 5.12.3   | Member Data Documentation          | 77 |
| 5.12.3.1 | nNeurons                           | 77 |
| 5.13     | MLPlayer Class Reference           | 77 |
| 5.13.1   | Detailed Description               | 80 |
| 5.14     | MLPlayerContainer Class Reference  | 80 |
| 5.14.1   | Detailed Description               | 83 |
| 5.15     | MLPneuralNet Class Reference       | 83 |
| 5.15.1   | Detailed Description               | 85 |
| 5.15.2   | Member Function Documentation      | 85 |
| 5.15.2.1 | show                               | 85 |
| 5.15.2.2 | validate                           | 85 |
| 5.15.3   | Member Data Documentation          | 85 |
| 5.15.3.1 | nLayers                            | 85 |
| 5.15.3.2 | nLayers                            | 85 |
| 5.16     | MLPneuron Class Reference          | 86 |
| 5.16.1   | Detailed Description               | 88 |
| 5.16.2   | Member Function Documentation      | 88 |
| 5.16.2.1 | show                               | 88 |
| 5.16.2.2 | validate                           | 88 |
| 5.16.3   | Member Data Documentation          | 88 |
| 5.16.3.1 | bias                               | 88 |
| 5.16.3.2 | bias                               | 88 |
| 5.17     | MLPneuronContainer Class Reference | 89 |
| 5.17.1   | Detailed Description               | 92 |
| 5.17.2   | Member Function Documentation      | 92 |



|           |  |     |
|-----------|--|-----|
| 5.17.2.1  | buildAndAppend                         | 92  |
| 5.17.2.2  | getId                                  | 92  |
| 5.18      | NeuralNet Class Reference              | 92  |
| 5.18.1    | Detailed Description                   | 93  |
| 5.18.2    | Member Function Documentation          | 93  |
| 5.18.2.1  | show                                   | 93  |
| 5.18.2.2  | train                                  | 93  |
| 5.18.2.3  | validate                               | 94  |
| 5.19      | Neuron Class Reference                 | 94  |
| 5.19.1    | Detailed Description                   | 97  |
| 5.19.2    | Constructor & Destructor Documentation | 98  |
| 5.19.2.1  | Neuron                                 | 98  |
| 5.19.2.2  | Neuron                                 | 98  |
| 5.19.2.3  | Neuron                                 | 98  |
| 5.19.2.4  | Neuron                                 | 98  |
| 5.19.2.5  | Neuron                                 | 98  |
| 5.19.2.6  | ~Neuron                                | 98  |
| 5.19.3    | Member Function Documentation          | 98  |
| 5.19.3.1  | getConId                               | 98  |
| 5.19.3.2  | getId                                  | 98  |
| 5.19.3.3  | getWeight                              | 99  |
| 5.19.3.4  | Id                                     | 99  |
| 5.19.3.5  | numOfCons                              | 99  |
| 5.19.3.6  | setFrom                                | 99  |
| 5.19.3.7  | setId                                  | 99  |
| 5.19.3.8  | setWeight                              | 100 |
| 5.19.3.9  | show                                   | 100 |
| 5.19.3.10 | show                                   | 101 |
| 5.19.3.11 | validate                               | 101 |
| 5.19.3.12 | validate                               | 101 |
| 5.19.4    | Member Data Documentation              | 102 |
| 5.19.4.1  | con                                    | 102 |
| 5.19.4.2  | d_Id                                   | 102 |
| 5.19.4.3  | Id                                     | 102 |

|          |  |     |
|----------|--|-----|
| 5.19.4.4 | outputValue                            | 102 |
| 5.20     | NeuronContainer Class Reference        | 102 |
| 5.20.1   | Detailed Description                   | 105 |
| 5.20.2   | Member Typedef Documentation           | 105 |
| 5.20.2.1 | const_iterator                         | 105 |
| 5.20.2.2 | const_reference                        | 105 |
| 5.20.2.3 | iterator                               | 106 |
| 5.20.2.4 | value_type                             | 106 |
| 5.20.3   | Constructor & Destructor Documentation | 106 |
| 5.20.3.1 | NeuronContainer                        | 106 |
| 5.20.3.2 | NeuronContainer                        | 106 |
| 5.20.3.3 | ~NeuronContainer                       | 106 |
| 5.20.4   | Member Function Documentation          | 106 |
| 5.20.4.1 | getConId                               | 106 |
| 5.20.4.2 | getFrom                                | 107 |
| 5.20.4.3 | getId                                  | 107 |
| 5.20.4.4 | getWeight                              | 107 |
| 5.20.4.5 | numOfCons                              | 107 |
| 5.20.4.6 | numOfNeurons                           | 108 |
| 5.20.4.7 | setFrom                                | 108 |
| 5.20.4.8 | setId                                  | 108 |
| 5.20.4.9 | setWeight                              | 109 |
| 5.21     | RBFneuralNet Class Reference           | 109 |
| 5.21.1   | Detailed Description                   | 111 |
| 5.21.2   | Member Function Documentation          | 111 |
| 5.21.2.1 | show                                   | 111 |
| 5.21.2.2 | validate                               | 111 |
| 5.21.3   | Member Data Documentation              | 111 |
| 5.21.3.1 | nLayers                                | 111 |
| 5.22     | RBFneuron Class Reference              | 111 |
| 5.22.1   | Detailed Description                   | 114 |
| 5.22.2   | Member Function Documentation          | 114 |
| 5.22.2.1 | show                                   | 114 |
| 5.22.2.2 | validate                               | 114 |

|          |   |            |
|----------|---|------------|
| 5.22.3   | Member Data Documentation . . . . .                                     | 114        |
| 5.22.3.1 | altitude . . . . .  | 114        |
| 5.22.3.2 | width . . . . .   | 114        |
| 5.23     | SimulationVariables Class Reference . . . . .                           | 114        |
| 5.23.1   | Detailed Description . . . . .  | 115        |
| 5.23.2   | Member Data Documentation . . . . .                                     | 115        |
| 5.23.2.1 | outputValue . . . . .   | 115        |
| 5.24     | TrainingVariablesSet Class Reference . . . . .                          | 115        |
| 5.24.1   | Detailed Description . . . . .  | 115        |
| <b>6</b> | <b>File Documentation</b>   | <b>117</b> |
| 6.1      | pkg/AMORE/src/AMORE.h File Reference . . . . .                          | 117        |
| 6.1.1    | Define Documentation . . . . .  | 118        |
| 6.1.1.1  | foreach . . . . .   | 118        |
| 6.1.1.2  | size_type . . . . .   | 118        |
| 6.1.2    | Typedef Documentation . . . . .   | 118        |
| 6.1.2.1  | ConContainer . . . . .  | 118        |
| 6.1.2.2  | ConPtr . . . . .  | 118        |
| 6.1.2.3  | NeuronContainer . . . . .   | 119        |
| 6.1.2.4  | NeuronPtr . . . . .   | 119        |
| 6.1.2.5  | NeuronRef . . . . .   | 119        |
| 6.2      | pkg/AMORE/src/Con.cpp File Reference . . . . .                          | 119        |
| 6.3      | pkg/AMORE/src/old/Con.cpp File Reference . . . . .                      | 120        |
| 6.4      | pkg/AMORE/src/Container.cpp File Reference . . . . .                    | 120        |
| 6.5      | pkg/AMORE/src/old/Container.cpp File Reference . . . . .                | 121        |
| 6.6      | pkg/AMORE/src/containerInterface.cpp File Reference . . . . .           | 121        |
| 6.7      | pkg/AMORE/src/ContainerIterator.cpp File Reference . . . . .            | 122        |
| 6.8      | pkg/AMORE/src/dia/ADAPTgdTrainingVariables.h File Reference . . . . .   | 123        |
| 6.9      | pkg/AMORE/src/dia/ADAPTgdwmTrainingVariables.h File Reference . . . . . | 124        |
| 6.10     | pkg/AMORE/src/dia/BATCHgdwmTrainingVariables.h File Reference . . . . . | 125        |
| 6.11     | pkg/AMORE/src/dia/BATCHgdwmTrainingVariables.h File Reference . . . . . | 125        |
| 6.12     | pkg/AMORE/src/dia/Con.h File Reference . . . . .                        | 126        |
| 6.13     | pkg/AMORE/src/old/Con.h File Reference . . . . .                        | 127        |
| 6.14     | pkg/AMORE/src/dia/Container.h File Reference . . . . .                  | 127        |

|          |  |     |
|----------|--|-----|
| 6.15     | <a href="#">pkg/AMORE/src/old/Container.h File Reference</a>             | 128 |
| 6.16     | <a href="#">pkg/AMORE/src/dia/ContainerInterface.h File Reference</a>    | 129 |
| 6.17     | <a href="#">pkg/AMORE/src/dia/ContainerIterator.h File Reference</a>     | 129 |
| 6.18     | <a href="#">pkg/AMORE/src/dia/IteratorInterface.h File Reference</a>     | 131 |
| 6.19     | <a href="#">pkg/AMORE/src/dia/Layer.h File Reference</a>                 | 131 |
| 6.20     | <a href="#">pkg/AMORE/src/dia/MLPneuralNet.h File Reference</a>          | 131 |
| 6.21     | <a href="#">pkg/AMORE/src/old/MLPneuralNet.h File Reference</a>          | 132 |
| 6.22     | <a href="#">pkg/AMORE/src/dia/MLPneuron.h File Reference</a>             | 132 |
| 6.23     | <a href="#">pkg/AMORE/src/old/MLPneuron.h File Reference</a>             | 133 |
| 6.24     | <a href="#">pkg/AMORE/src/dia/NeuralNet.h File Reference</a>             | 134 |
| 6.25     | <a href="#">pkg/AMORE/src/old/NeuralNet.h File Reference</a>             | 134 |
| 6.26     | <a href="#">pkg/AMORE/src/dia/Neuron.h File Reference</a>                | 135 |
| 6.27     | <a href="#">pkg/AMORE/src/old/Neuron.h File Reference</a>                | 135 |
| 6.28     | <a href="#">pkg/AMORE/src/dia/RBFneuralNet.h File Reference</a>          | 136 |
| 6.29     | <a href="#">pkg/AMORE/src/old/RBFneuralNet.h File Reference</a>          | 136 |
| 6.30     | <a href="#">pkg/AMORE/src/dia/RBFneuron.h File Reference</a>             | 136 |
| 6.31     | <a href="#">pkg/AMORE/src/dia/SimulationVariables.h File Reference</a>   | 137 |
| 6.32     | <a href="#">pkg/AMORE/src/dia/TrainingVariablesSet.h File Reference</a>  | 137 |
| 6.33     | <a href="#">pkg/AMORE/src/IteratorInterface.cpp File Reference</a>       | 138 |
| 6.34     | <a href="#">pkg/AMORE/src/Neuron.cpp File Reference</a>                  | 138 |
| 6.35     | <a href="#">pkg/AMORE/src/old/Neuron.cpp File Reference</a>              | 139 |
| 6.36     | <a href="#">pkg/AMORE/src/old/ConContainer.cpp File Reference</a>        | 140 |
| 6.37     | <a href="#">pkg/AMORE/src/old/ConContainer.h File Reference</a>          | 140 |
| 6.38     | <a href="#">pkg/AMORE/src/old/MLPlayer.h File Reference</a>              | 140 |
| 6.39     | <a href="#">pkg/AMORE/src/old/MLPlayerContainer.h File Reference</a>     | 140 |
| 6.40     | <a href="#">pkg/AMORE/src/old/MLPneuralNetFactory.cpp File Reference</a> | 141 |
| 6.40.1   | <a href="#">Function Documentation</a>                                   | 141 |
| 6.40.1.1 | <a href="#">CreateMLPneuralNet</a>                                       | 141 |
| 6.41     | <a href="#">pkg/AMORE/src/old/MLPneuronContainer.h File Reference</a>    | 141 |
| 6.42     | <a href="#">pkg/AMORE/src/old/NeuronContainer.cpp File Reference</a>     | 142 |
| 6.43     | <a href="#">pkg/AMORE/src/old/NeuronContainer.h File Reference</a>       | 142 |

# Chapter 1

## The AMORE++ package

### 1.1 Introduction

Here you will find the documentation of the C++ component of the AMORE++ R package.

The AMORE++ package is a new version of the publicly available AMORE package for neural network training and simulation under R

### 1.2 Motivation

Since the release of the previous version of the AMORE many things have changed in the R programming world.

The advent of the Reference Classes and of packages like Rcpp, inline and RUnit compel us to write a better version of the package in order to provide a more useful framework for neural network training and simulation.

### 1.3 Road Map

This project is currently very active and the development team intends to provide a beta version as soon as this summer (2011)



## Chapter 2

# Class Index

### 2.1 Class Hierarchy

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

|                                   |     |
|-----------------------------------|-----|
| CompareId . . . . .               | 17  |
| Con . . . . .                     | 18  |
| ContainerInterface< T > . . . . . | 63  |
| Container< T > . . . . .          | 47  |
| Container< Con > . . . . .        | 47  |
| ConContainer . . . . .            | 30  |
| Container< MLPlayer > . . . . .   | 47  |
| MLPlayerContainer . . . . .       | 80  |
| Container< Neuron > . . . . .     | 47  |
| NeuronContainer . . . . .         | 102 |
| IteratorInterface< T > . . . . .  | 72  |
| ContainerIterator< T > . . . . .  | 67  |
| Layer . . . . .                   | 75  |
| NeuralNet . . . . .               | 92  |
| MLPneuralNet . . . . .            | 83  |
| MLPneuralNet . . . . .            | 83  |
| RBFneuralNet . . . . .            | 109 |
| RBFneuralNet . . . . .            | 109 |
| Neuron . . . . .                  | 94  |
| MLPneuron . . . . .               | 86  |
| MLPneuron . . . . .               | 86  |
| RBFneuron . . . . .               | 111 |
| NeuronContainer< MLP > . . . . .  | 102 |
| MLPneuronContainer . . . . .      | 89  |
| MLPlayer . . . . .                | 77  |
| SimulationVariables . . . . .     | 114 |
| TrainingVariablesSet . . . . .    | 115 |

|                                      |    |
|--------------------------------------|----|
| ADAPTgdTrainingVariables . . . . .   | 9  |
| ADAPTgdwmTrainingVariables . . . . . | 11 |
| BATCHgdTrainingVariables . . . . .   | 13 |
| BATCHgdwmTrainingVariables . . . . . | 15 |



## Chapter 3

# Class Index

### 3.1 Class List

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

|  |     |
|--|-----|
| <a href="#">ADAPTgdTrainingVariables</a> (Class <a href="#">ADAPTgdTrainingVariables</a> - ) . . . . .     | 9   |
| <a href="#">ADAPTgdwmTrainingVariables</a> (Class <a href="#">ADAPTgdwmTrainingVariables</a> - ) . . . . . | 11  |
| <a href="#">BATCHgdTrainingVariables</a> (Class <a href="#">BATCHgdTrainingVariables</a> - ) . . . . .     | 13  |
| <a href="#">BATCHgdwmTrainingVariables</a> (Class <a href="#">BATCHgdwmTrainingVariables</a> - ) . . . . . | 15  |
| <a href="#">CompareId</a> . . . . .  | 17  |
| <a href="#">Con</a> (Class <a href="#">Con</a> - ) . . . . .   | 18  |
| <a href="#">ConContainer</a> (A vector of connections ) . . . . .  | 30  |
| <a href="#">Container&lt; T &gt;</a> (Class <a href="#">Container</a> - ) . . . . .                        | 47  |
| <a href="#">ContainerInterface&lt; T &gt;</a> (Class <a href="#">ContainerInterface</a> - ) . . . . .      | 63  |
| <a href="#">ContainerIterator&lt; T &gt;</a> (Class <a href="#">ContainerIterator</a> - ) . . . . .        | 67  |
| <a href="#">IteratorInterface&lt; T &gt;</a> (Class <a href="#">IteratorInterface</a> - ) . . . . .        | 72  |
| <a href="#">Layer</a> (Class <a href="#">Layer</a> - ) . . . . .   | 75  |
| <a href="#">MLPlayer</a> . . . . .   | 77  |
| <a href="#">MLPlayerContainer</a> . . . . .  | 80  |
| <a href="#">MLPneuralNet</a> (Class <a href="#">MLPneuralNet</a> - ) . . . . .                             | 83  |
| <a href="#">MLPneuron</a> (Class <a href="#">MLPneuron</a> - ) . . . . .                                   | 86  |
| <a href="#">MLPneuronContainer</a> (A vector of connections ) . . . . .                                    | 89  |
| <a href="#">NeuralNet</a> (Class <a href="#">NeuralNet</a> - ) . . . . .                                   | 92  |
| <a href="#">Neuron</a> (Class <a href="#">Neuron</a> - ) . . . . .   | 94  |
| <a href="#">NeuronContainer</a> (A vector of neurons ) . . . . .   | 102 |
| <a href="#">RBFneuralNet</a> (Class <a href="#">RBFneuralNet</a> - ) . . . . .                             | 109 |
| <a href="#">RBFneuron</a> (Class <a href="#">RBFneuron</a> - ) . . . . .                                   | 111 |
| <a href="#">SimulationVariables</a> (Class <a href="#">SimulationVariables</a> - ) . . . . .               | 114 |
| <a href="#">TrainingVariablesSet</a> (Class <a href="#">TrainingVariablesSet</a> - ) . . . . .             | 115 |



## Chapter 4

# File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

|  |     |
|--|-----|
| pkg/AMORE/src/AMORE.h                          | 117 |
| pkg/AMORE/src/Con.cpp                          | 119 |
| pkg/AMORE/src/Container.cpp                    | 120 |
| pkg/AMORE/src/containerInterface.cpp           | 121 |
| pkg/AMORE/src/ContainerIterator.cpp            | 122 |
| pkg/AMORE/src/IteratorInterface.cpp            | 138 |
| pkg/AMORE/src/Neuron.cpp                       | 138 |
| pkg/AMORE/src/dia/ADAPTgdTrainingVariables.h   | 123 |
| pkg/AMORE/src/dia/ADAPTgdwmTrainingVariables.h | 124 |
| pkg/AMORE/src/dia/BATCHgdTrainingVariables.h   | 125 |
| pkg/AMORE/src/dia/BATCHgdwmTrainingVariables.h | 125 |
| pkg/AMORE/src/dia/Con.h                        | 126 |
| pkg/AMORE/src/dia/Container.h                  | 127 |
| pkg/AMORE/src/dia/ContainerInterface.h         | 129 |
| pkg/AMORE/src/dia/ContainerIterator.h          | 129 |
| pkg/AMORE/src/dia/IteratorInterface.h          | 131 |
| pkg/AMORE/src/dia/Layer.h                      | 131 |
| pkg/AMORE/src/dia/MLPNeuralNet.h               | 131 |
| pkg/AMORE/src/dia/MLPNeuron.h                  | 132 |
| pkg/AMORE/src/dia/NeuralNet.h                  | 134 |
| pkg/AMORE/src/dia/Neuron.h                     | 135 |
| pkg/AMORE/src/dia/RBFNeuralNet.h               | 136 |
| pkg/AMORE/src/dia/RBFNeuron.h                  | 136 |
| pkg/AMORE/src/dia/SimulationVariables.h        | 137 |
| pkg/AMORE/src/dia/TrainingVariablesSet.h       | 137 |
| pkg/AMORE/src/old/Con.cpp                      | 120 |
| pkg/AMORE/src/old/Con.h                        | 127 |
| pkg/AMORE/src/old/ConContainer.cpp             | 140 |
| pkg/AMORE/src/old/ConContainer.h               | 140 |

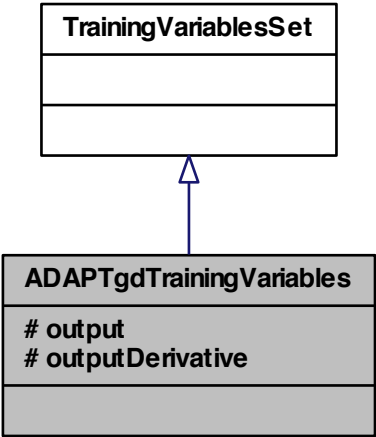
|  |     |
|--|-----|
| pkg/AMORE/src/old/ <a href="#">Container.cpp</a>           | 121 |
| pkg/AMORE/src/old/ <a href="#">Container.h</a>             | 128 |
| pkg/AMORE/src/old/ <a href="#">MLPlayer.h</a>              | 140 |
| pkg/AMORE/src/old/ <a href="#">MLPlayerContainer.h</a>     | 140 |
| pkg/AMORE/src/old/ <a href="#">MLPneuralNet.h</a>          | 132 |
| pkg/AMORE/src/old/ <a href="#">MLPneuralNetFactory.cpp</a> | 141 |
| pkg/AMORE/src/old/ <a href="#">MLPneuron.h</a>             | 133 |
| pkg/AMORE/src/old/ <a href="#">MLPneuronContainer.h</a>    | 141 |
| pkg/AMORE/src/old/ <a href="#">NeuralNet.h</a>             | 134 |
| pkg/AMORE/src/old/ <a href="#">Neuron.cpp</a>              | 139 |
| pkg/AMORE/src/old/ <a href="#">Neuron.h</a>                | 135 |
| pkg/AMORE/src/old/ <a href="#">NeuronContainer.cpp</a>     | 142 |
| pkg/AMORE/src/old/ <a href="#">NeuronContainer.h</a>       | 142 |
| pkg/AMORE/src/old/ <a href="#">RBFneuralNet.h</a>          | 136 |

# Chapter 5

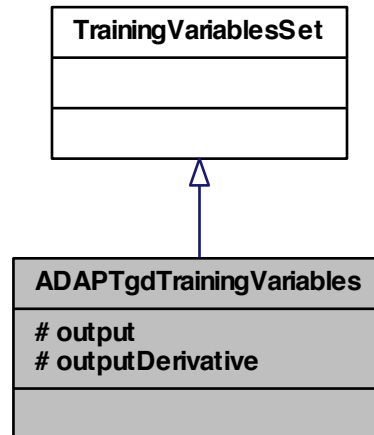
## Class Documentation

### 5.1 ADAPTgdTrainingVariables Class Reference

class [ADAPTgdTrainingVariables](#) -  
#include <ADAPTgdTrainingVariables.h>  
Inheritance diagram for ADAPTgdTrainingVariables:



Collaboration diagram for ADAPTgdTrainingVariables:



### Protected Attributes

- double [output](#)
- double [outputDerivative](#)

#### 5.1.1 Detailed Description

class [ADAPTgdTrainingVariables](#) -

Definition at line 5 of file ADAPTgdTrainingVariables.h.

#### 5.1.2 Member Data Documentation

##### 5.1.2.1 double ADAPTgdTrainingVariables::output [protected]

Definition at line 8 of file ADAPTgdTrainingVariables.h.

##### 5.1.2.2 double ADAPTgdTrainingVariables::outputDerivative [protected]

Definition at line 9 of file ADAPTgdTrainingVariables.h.

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

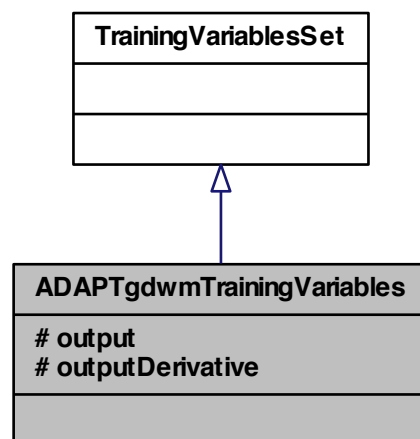
- pkg/AMORE/src/dia/[ADAPTgdTrainingVariables.h](#)

## 5.2 ADAPTgdwmTrainingVariables Class Reference

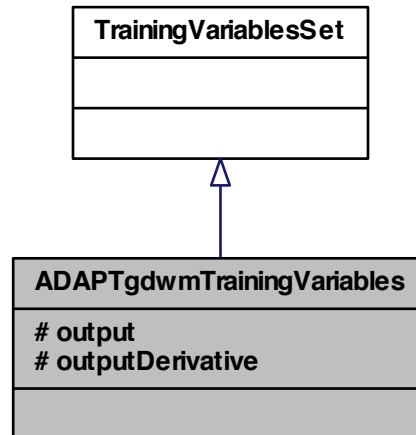
class [ADAPTgdwmTrainingVariables](#) -

```
#include <ADAPTgdwmTrainingVariables.h>
```

Inheritance diagram for ADAPTgdwmTrainingVariables:



Collaboration diagram for ADAPTgdwmTrainingVariables:



### Protected Attributes

- double [output](#)
- double [outputDerivative](#)

### 5.2.1 Detailed Description

class [ADAPTgdwmTrainingVariables](#) -

Definition at line 5 of file ADAPTgdwmTrainingVariables.h.

### 5.2.2 Member Data Documentation

#### 5.2.2.1 double ADAPTgdwmTrainingVariables::output [protected]

Definition at line 8 of file ADAPTgdwmTrainingVariables.h.

#### 5.2.2.2 double ADAPTgdwmTrainingVariables::outputDerivative [protected]

Definition at line 9 of file ADAPTgdwmTrainingVariables.h.

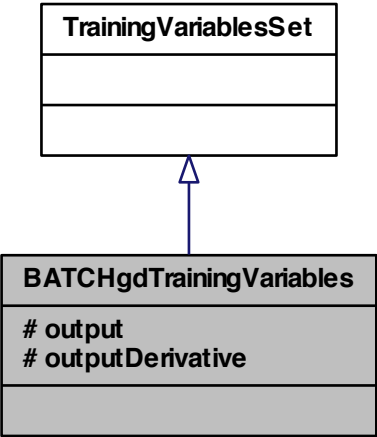
The documentation for this class was generated from the following file:



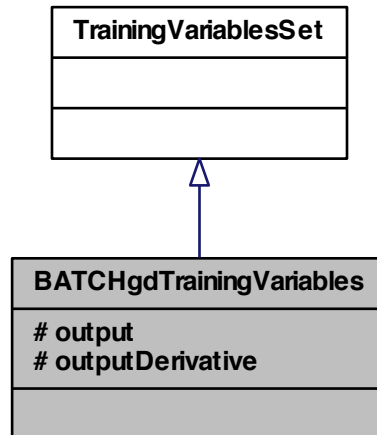
- pkg/AMORE/src/dia/[ADAPTgdwmTrainingVariables.h](#)

5.3 BATCHgdTrainingVariables Class Reference

class [BATCHgdTrainingVariables](#) -  
#include <BATCHgdTrainingVariables.h>  
Inheritance diagram for BATCHgdTrainingVariables:



Collaboration diagram for BATCHgdTrainingVariables:



### Protected Attributes

- double [output](#)
- double [outputDerivative](#)

### 5.3.1 Detailed Description

class [BATCHgdTrainingVariables](#) -

Definition at line 5 of file BATCHgdTrainingVariables.h.

### 5.3.2 Member Data Documentation

#### 5.3.2.1 double BATCHgdTrainingVariables::output [protected]

Definition at line 8 of file BATCHgdTrainingVariables.h.

#### 5.3.2.2 double BATCHgdTrainingVariables::outputDerivative [protected]

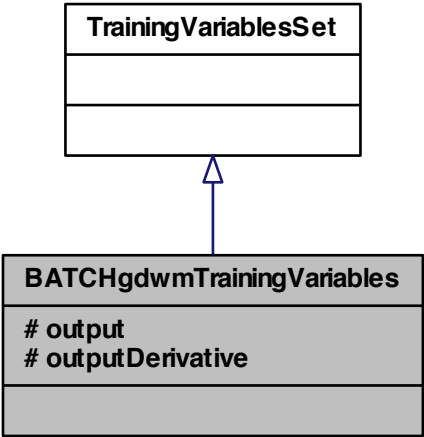
Definition at line 9 of file BATCHgdTrainingVariables.h.

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

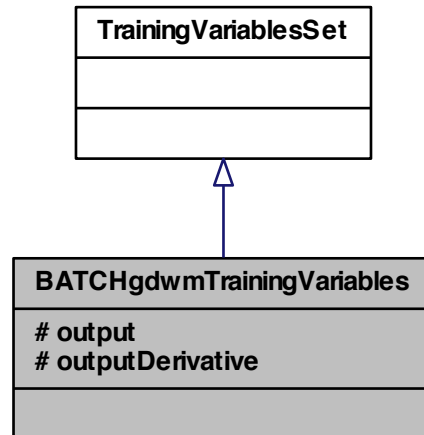
- pkg/AMORE/src/dia/BATCHgdTrainingVariables.h

5.4 BATCHgdwmTrainingVariables Class Reference

class BATCHgdwmTrainingVariables -  
#include <BATCHgdwmTrainingVariables.h>  
Inheritance diagram for BATCHgdwmTrainingVariables:



Collaboration diagram for BATCHgdwmTrainingVariables:



### Protected Attributes

- double [output](#)
- double [outputDerivative](#)

### 5.4.1 Detailed Description

class [BATCHgdwmTrainingVariables](#) -

Definition at line 5 of file BATCHgdwmTrainingVariables.h.

### 5.4.2 Member Data Documentation

5.4.2.1 double **BATCHgdwmTrainingVariables::output** protected

Definition at line 8 of file BATCHgdwmTrainingVariables.h.

5.4.2.2 double **BATCHgdwmTrainingVariables::outputDerivative** protected

Definition at line 9 of file BATCHgdwmTrainingVariables.h.

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

- [pkg/AMORE/src/dia/BATCHgdwmTrainingVariables.h](#)

## 5.5 CompareId Struct Reference

### Public Member Functions

- `bool operator() (const ConPtr a, const ConPtr b)`
- `bool operator() (const ConPtr a, const int b)`
- `bool operator() (const int a, const ConPtr b)`
- `bool operator() (const int a, const int b)`

#### 5.5.1 Detailed Description

Definition at line 352 of file ConContainer.cpp.

#### 5.5.2 Member Function Documentation

##### 5.5.2.1 `bool CompareId::operator() ( const ConPtr a, const ConPtr b )` `[inline]`

Definition at line 356 of file ConContainer.cpp.

```
{  
    return a->getId() < b->getId();  
}
```

##### 5.5.2.2 `bool CompareId::operator() ( const int a, const int b )` `[inline]`

Definition at line 377 of file ConContainer.cpp.

```
{  
    return a < b;  
}
```

##### 5.5.2.3 `bool CompareId::operator() ( const int a, const ConPtr b )` `[inline]`

Definition at line 370 of file ConContainer.cpp.

```
{  
    return a < b->getId();  
}
```

#### 5.5.2.4 `bool CompareId::operator() ( const ConPtr a, const int b )` `[inline]`

Definition at line 363 of file ConContainer.cpp.

```
{
    return a->getId() < b;
}
```

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

- `pkg/AMORE/src/old/ConContainer.cpp`

## 5.6 Con Class Reference

class `Con` -

```
#include <Con.h>
```

### Public Member Functions

- `Con (Neuron &neuron)`  
*Constructor.*
- `Con (Neuron &neuron, double weight)`  
*Constructor.*
- `Neuron & neuron ()`  
*from field accessor.*
- `int Id ()`  
*A getter of the Id of the Neuron pointed by the from field.*
- `double & weight ()`
- `void show ()`  
*Pretty print of the Con information.*
- `bool validate ()`  
*Object validator.*
- `Con ()`  
*Default Constructor.*
- `Con (NeuronPtr neuronPtr)`  
*Constructor.*
- `Con (NeuronPtr neuronPtr, double value)`  
*Constructor.*
- `~Con ()`  
*Default Destructor.*
- `NeuronPtr getFrom ()`  
*from field accessor.*
- `void setFrom (NeuronPtr neuronPtr)`

*from* field accessor.

- int `getId` ()  
*A getter of the Id of the [Neuron](#) pointed by the from field.*
- double `getWeight` ()  
*weight field accessor.*
- void `setWeight` (double value)  
*weight field accessor.*
- bool `show` ()
- bool `validate` ()

### Protected Attributes

- [NeuronRef](#) `d_neuron`
- double `d_weight`

### Private Attributes

- [NeuronWeakPtr](#) `from`  
*A smart pointer to the [Neuron](#) used as input during simulation or training.*
- double `weight`  
*A double variable that contains the weight of the connection.*

#### 5.6.1 Detailed Description

class [Con](#) -

A class to handle the information needed to describe an input connection.

The [Con](#) class provides a simple class for a connection described by a pair of values: a pointer to a [Neuron](#) object used as the `from` field and the `weight` used to propagate the value of that [Neuron](#) object.

Definition at line 3 of file `Con.h`.

#### 5.6.2 Constructor & Destructor Documentation

##### 5.6.2.1 `Con::Con ( Neuron & neuron )`

Constructor.

Definition at line 17 of file `Con.cpp`.

```

        :
    d_neuron( boost::ref(neuron) ), d_weight(0)
    {
    }

```

#### 5.6.2.2 Con::Con ( Neuron & *neuron*, double *weight* )

Constructor.

Definition at line 28 of file Con.cpp.

```
        :  
        d_neuron(boost::ref(neuron)), d_weight(weight)  
{  
}  
}
```

#### 5.6.2.3 Con::Con ( )

Default Constructor.

Definition at line 17 of file Con.cpp.

```
        :  
        weight(0), from()  
{  
}  
}
```

#### 5.6.2.4 Con::Con ( NeuronPtr *neuronPtr* )

Constructor.

Definition at line 40 of file Con.cpp.

```
        :  
        from(neuronPtr), weight(0)  
{  
}  
}
```

#### 5.6.2.5 Con::Con ( NeuronPtr *neuronPtr*, double *value* )

Constructor.

Definition at line 29 of file Con.cpp.

```
        :  
        from(neuronPtr), weight(value)  
{  
}  
}
```

#### 5.6.2.6 Con::~Con ( )

Default Destructor.

Definition at line 46 of file Con.cpp.

```
{  
}
```



### 5.6.3 Member Function Documentation

#### 5.6.3.1 NeuronPtr Con::getFrom ( )

from field accessor.

This method allows access to the address stored in the private [from](#) field (a pointer to a [Neuron](#) object).\*

#### Returns

A pointer to the [Neuron](#) object referred to by the [from](#) field.

```
//=====
//Usage example:
//=====
// Data set up
NeuronPtr ptShNeuron ( new Neuron(1) );           // Neuron
Id is set 1
ConPtr ptShCon( new Con(ptShNeuron) );           // from p
oints to ptShNeuron and weight is set to 0
// Test
ptShNeuron = ptShCon->getFrom() ;
int result = ptShNeuron->getId();

// Now, result is equal to 1.
```

#### See also

[getId](#) and the unit test files, e.g., `runit.Cpp.Con.R`, for further examples.

Definition at line 71 of file `Con.cpp`.

References from.

```
{
    return (from.lock());
}
```

#### 5.6.3.2 int Con::getId ( )

A getter of the Id of the [Neuron](#) pointed by the [from](#) field.

This method gets the Id of the [Neuron](#) referred to by the [from](#) field

#### Returns

The value of the Id (an integer).

```
//=====
//Usage example:
//=====
// Data set up
NeuronPtr ptShNeuron ( new Neuron(16) );           // Neuron
Id is set to 16
```

```

        ConPtr ptShCon( new Con(ptShNeuron) );           // from p
oints to ptShNeuron and weight is set to 0
        // Test
        int result = ptShCon->getId();

        // Now, result is equal to 16.

```

**See also**

[getFrom](#), [setFrom](#) and the unit test files, e.g., `runit.Cpp.Con.R`, for further examples.

Definition at line 123 of file `Con.cpp`.

References from.

```

{
    if (from.use_count() > 0)
    {
        NeuronPtr neuronPtr(from);
        return (neuronPtr->getId());
    }
    else
    {
        return (NA_INTEGER);
    }
}

```

**5.6.3.3 double Con::getWeight ( )**

weight field accessor.

This method allows access to the value stored in the private field [weight](#)

**Returns**

The value of [weight](#) (double)

```

//=====
//Usage example:
//=====
// Data set up
        std::vector<double> result;
        NeuronPtr ptShNeuron ( new Neuron(16) );           /
/ Neuron Id is set to 16
        ConPtr ptShCon( new Con(ptShNeuron, 12.4) ); // from poi
nts to ptShNeuron and weight is set to 12.4
        // Test
        result.push_back( ptShCon->getWeight() );
        ptShCon->setWeight(2.2);
        result.push_back( ptShCon->getWeight() );

        // Now, result is a numeric vector that contains the values 12.4 and 2.2
.

```

**See also**

[setWeight](#) and the unit test files, e.g., `runit.Cpp.Con.R`, for further examples.

Definition at line 158 of file Con.cpp.

References `weight()`.

```
{
    return (weight);
}
```

Here is the call graph for this function:



#### 5.6.3.4 `int Con::Id ( )`

A getter of the Id of the [Neuron](#) pointed by the `from` field.

This method gets the Id of the [Neuron](#) referred to by the `from` field

#### Returns

The value of the Id (an integer).

```
//=====
//Usage example:
//=====
// Data set up
NeuronPtr ptShNeuron ( new Neuron(16) );           // Neuron I
d is set to 16
ConPtr ptShCon( new Con(ptShNeuron) );             // from poi
nts to ptShNeuron and weight is set to 0
// Test
int result = ptShCon->getId();

// Now, result is equal to 16.
```

#### See also

[getFrom](#), [setFrom](#) and the unit test files, e.g., `runit.Cpp.Con.R`, for further examples.

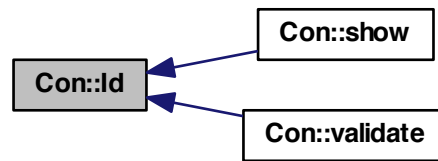
Definition at line 79 of file Con.cpp.

References `d_neuron`.

Referenced by `show()`, and `validate()`.

```
{
    return d_neuron.get().Id();
}
```

Here is the caller graph for this function:



#### 5.6.3.5 Neuron & Con::neuron ( )

from field accessor.

This method allows access to the address stored in the private [from](#) field (a pointer to a [Neuron](#) object).\*

#### Returns

A pointer to the [Neuron](#) object referred to by the [from](#) field.

```
//=====
//Usage example:
//=====
// Data set up
NeuronPtr ptShNeuron ( new Neuron(1) );           // Neuron
Id is set 1
ConPtr ptShCon( new Con(ptShNeuron) );           // from p
oints to ptShNeuron and weight is set to 0
// Test
ptShNeuron = ptShCon->getFrom() ;
int result = ptShNeuron->getId();

// Now, result is equal to 1.
```

#### See also

[getId](#) and the unit test files, e.g., `runit.Cpp.Con.R`, for further examples.

Definition at line 54 of file `Con.cpp`.

References `d_neuron`.

```
{
    return d_neuron;
}
```

#### 5.6.3.6 void Con::setFrom ( NeuronPtr neuronPtr )

from field accessor.

This method sets the value of the [from](#) field with the address used as parameter.

##### Parameters

|          |   |
|----------|---|
| <i>f</i> | A pointer to the neuron that is to be inserted in the <a href="#">from</a> field. |
|----------|---|

```
//=====
//Usage example:
//=====
// Data set up
NeuronPtr ptShNeuron ( new Neuron(1) );           // Neuron
Id is set to 1
ConPtr ptShCon( new Con() );
ptShCon->setFrom( ptShNeuron );

// Test
ptShNeuron = ptShCon->getFrom() ;
int result = ptShNeuron->getId();

// Now, result is equal to 1
```

##### See also

[getFrom](#) and [getId](#) contain usage examples. For further examples see the unit test files, e.g., `runit.Cpp.Con.R`

Definition at line 98 of file `Con.cpp`.

References from.

```
{
    from = neuronPtr;
}
```

#### 5.6.3.7 void Con::setWeight ( double value )

weight field accessor.

This method sets the value of the [weight](#) field.

##### Parameters

|          |   |
|----------|---|
| <i>w</i> | The new value (double) to be set in the <a href="#">weight</a> field. |
|----------|---|

```
//=====
//Usage example:
```

```
//=====
// Data set up
std::vector<double> result;
NeuronPtr ptShNeuron ( new Neuron(16) );
/
/ Neuron Id is set to 16
ConPtr ptShCon( new Con(ptShNeuron, 12.4) ); // from poi
nts to ptShNeuron and weight is set to 12.4
result.push_back(ptShCon->getWeight());
// Test
ptShCon->setWeight(2.2);
result.push_back(ptShCon->getWeight());

// Now, result is a numeric vector that contains the values 12.4 and 2.2
.
```

**See also**

[getWeight](#) and the unit test files (e.g. `runit.Cpp.Con.R`)

Definition at line 186 of file `Con.cpp`.

References `weight()`.

```
{
    weight = value;
}
```

Here is the call graph for this function:



5.6.3.8 `bool Con::show ( )`

5.6.3.9 `bool Con::show ( )`

Pretty print of the [Con](#) information.

This method outputs in the R terminal the contents of the [Con](#) fields.

**Returns**

true in case everything works without throwing an exception

**See also**

[setWeight](#) and the unit test files, e.g., `runit.Cpp.Con.R`, for usage examples.

Definition at line 118 of file Con.cpp.

References `d_weight`, and `Id()`.

```
{
  int id = Id();
  if (id == NA_INTEGER)
  {
    Rprintf("From: NA\t Invalid Connection \n");
  }
  else
  {
    Rprintf("From:\t %d \t Weight= \t %lf \n", id, d_weight);
  }
}
```

Here is the call graph for this function:



#### 5.6.3.10 bool Con::validate ( )

Object validator.

This method checks the object for internal coherence. A try / catch mechanism exits normal execution and returns control to the R terminal in case the contents of the [Con](#) object are identified as corrupted.

#### Returns

true in case the checks are Ok.

#### Exceptions

|           |  |
|-----------|--|
| <i>An</i> | std::range error if weight or from are not finite. |
|-----------|--|

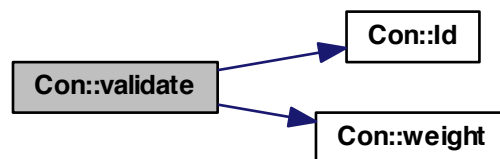
Definition at line 138 of file Con.cpp.

References `Id()`, and `weight()`.

```
{
  BEGIN_RCPP
  if (! R_FINITE(weight()) ) throw std::range_error("weight is not finite.");
  if (Id() == NA_INTEGER)
```

```
        throw std::range_error("fromId is not finite.");  
        return (true);  
    END_RCPP}
```

Here is the call graph for this function:

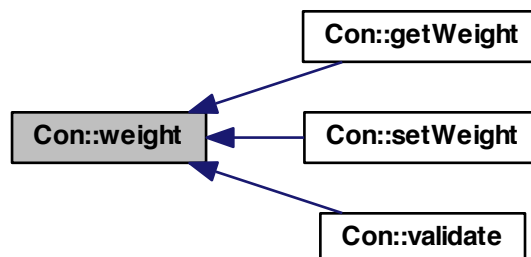


5.6.3.11 `bool Con::validate ( )`

5.6.3.12 `double& Con::weight ( )`

Referenced by `getWeight()`, `setWeight()`, and `validate()`.

Here is the caller graph for this function:



## 5.6.4 Member Data Documentation



**5.6.4.1 NeuronRef Con::d\_neuron** [protected]

Definition at line 6 of file Con.h.

Referenced by Id(), and neuron().

**5.6.4.2 double Con::d\_weight** [protected]

Definition at line 7 of file Con.h.

Referenced by show().

**5.6.4.3 NeuronWeakPtr Con::from** [private]

A smart pointer to the [Neuron](#) used as input during simulation or training.

The [from](#) field contains the address of the [Neuron](#) whose output will be used as input by the [Neuron](#) containing the [Con](#) object.

Definition at line 22 of file Con.h.

Referenced by getFrom(), getId(), and setFrom().

**5.6.4.4 double & Con::weight** [private]

A double variable that contains the weight of the connection.

weight field accessor.

The [weight](#) field contains the factor by which the output value of the [Neuron](#) addressed by the [from](#) field is multiplied during simulation or training.

This method allows access to the value stored in the private field [weight](#)

**Returns**

The value of [weight](#) (double)

```
//=====
//Usage example:
//=====
// Data set up
std::vector<double> result;
NeuronPtr ptShNeuron ( new Neuron(16) );
/
/ Neuron Id is set to 16
ConPtr ptShCon( new Con(ptShNeuron, 12.4) ); // from poi
nts to ptShNeuron and weight is set to 12.4
// Test
result.push_back( ptShCon->getWeight() );
ptShCon->setWeight(2.2);
result.push_back( ptShCon->getWeight() );

// Now, result is a numeric vector that contains the values 12.4 and 2.2
.
```

**See also**

[setWeight](#) and the unit test files, e.g., `runit.Cpp.Con.R`, for further examples.

Definition at line 27 of file `Con.h`.

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

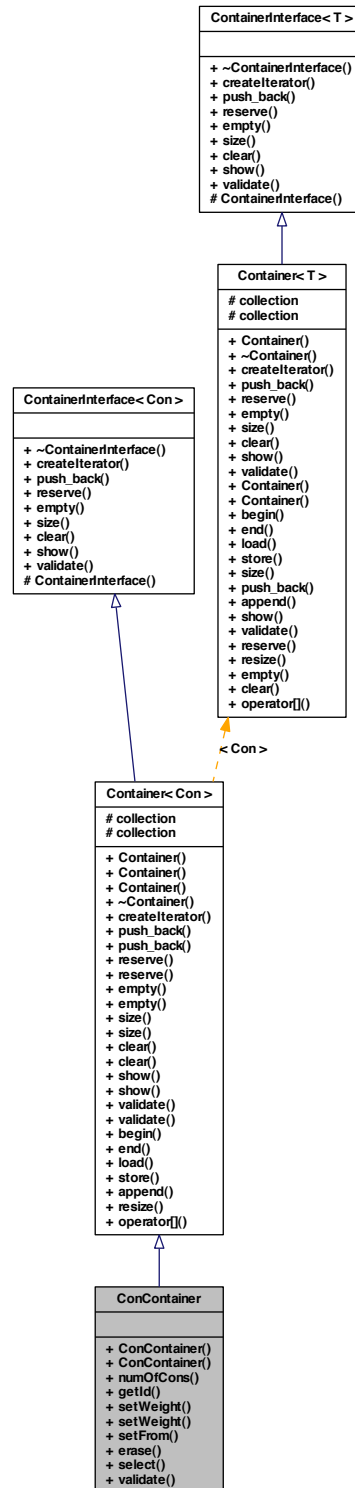
- `pkg/AMORE/src/dia/Con.h`
- `pkg/AMORE/src/old/Con.h`
- `pkg/AMORE/src/Con.cpp`
- `pkg/AMORE/src/old/Con.cpp`

## 5.7 ConContainer Class Reference

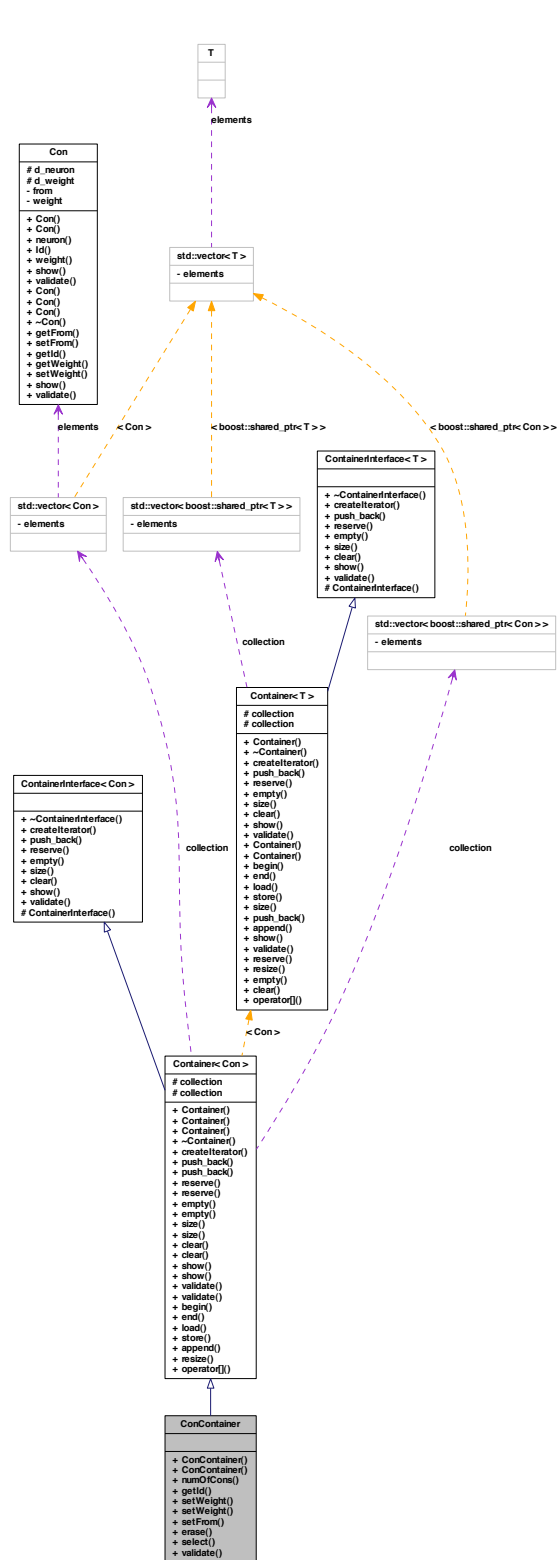
A vector of connections.

```
#include <ConContainer.h>
```

Inheritance diagram for ConContainer:



Collaboration diagram for ConContainer:



## Public Types

- typedef std::vector< boost::shared\_ptr< [Con](#) > >::iterator iterator
- typedef std::vector< boost::shared\_ptr< [Con](#) > >::const\_iterator const\_iterator
- typedef boost::shared\_ptr< [Con](#) > value\_type
- typedef [value\\_type](#) const & const\_reference

## Public Member Functions

- [ConContainer](#) ()
- [ConContainer](#) (std::vector< [ConPtr](#) > collection)
- int numOfCons ()  
*Size of the [ConContainer](#) object.*
- std::vector< int > getId ()  
*Getter of the Id values of the vector of Cons.*
- bool setWeight (std::vector< double > nWeights)  
*Setter of the weight field of the [Con](#) objects related to [ConContainer](#).*
- bool setWeight (std::vector< double > nWeights, std::vector< int > nlds)  
*Setter of the weights of the specified elements from the [ConContainer](#) object.*
- bool setFrom ([NeuronContainer](#) neuronContainer)  
*Setter of the from fields of the [Con](#) objects related to [ConContainer](#).*
- void erase (std::vector< int > nlds)  
*Erase the specified elements from the vecCom object.*
- [ConContainerPtr](#) select (std::vector< int > nlds)  
*Selects the specified elements from the vecCom object.*
- bool validate ()  
*Object validator.*

### 5.7.1 Detailed Description

A vector of connections.

The [ConContainer](#) class provides a simple class for a vector of connections. It's named after the R equivalent Reference Class.

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

### 5.7.2 Member Typedef Documentation

#### 5.7.2.1 typedef std::vector<boost::shared\_ptr<[Con](#)> >::const\_iterator [ConContainer::const\\_iterator](#)

Reimplemented from [Container< \[Con\]\(#\) >](#).

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

### 5.7.2.2 `typedef value_type const& ConContainer::const_reference`

Reimplemented from [Container< Con >](#).

Definition at line 27 of file ConContainer.h.

### 5.7.2.3 `typedef std::vector<boost::shared_ptr<Con> >::iterator ConContainer::iterator`

Reimplemented from [Container< Con >](#).

Definition at line 21 of file ConContainer.h.

### 5.7.2.4 `typedef boost::shared_ptr<Con> ConContainer::value_type`

Reimplemented from [Container< Con >](#).

Definition at line 25 of file ConContainer.h.

## 5.7.3 Constructor & Destructor Documentation

### 5.7.3.1 `ConContainer::ConContainer ( )`

Definition at line 8 of file ConContainer.cpp.

```
{
}
```

### 5.7.3.2 `ConContainer::ConContainer ( std::vector< ConPtr > collection )`

Definition at line 12 of file ConContainer.cpp.

```

        :
        Container<Con> (collection) // Call to Base constructor
    {
    }
```

## 5.7.4 Member Function Documentation

### 5.7.4.1 `void ConContainer::erase ( std::vector< int > nlds )`

Erase the specified elements from the vecCom object.

Provides a convenient way of removing some [Con](#) objects from the collection field of the [ConContainer](#) object.

#### Parameters

|              |   |
|--------------|---|
| <i>vFrom</i> | An <code>std::vector&lt;int&gt;</code> with the lds of the connections to remove. |
|--------------|---|

```

//=====
//Usage example:
//=====

// Data set up
std::vector<int> result;
std::vector<NeuronPtr> neuronContainer;
ConContainerPtr conContainerPtr( new ConContainer() );
ConContainerPtr vErased;
ConPtr ptC;
NeuronPtr ptN;
int ids[]= {11, 10, 9, 3, 4, 5, 6, 7, 8, 2, 1};
std::vector<double> nWeights;
nWeights.push_back(11.32);
nWeights.push_back(1.26);
nWeights.push_back(2.14);
nWeights.push_back(3.16);
nWeights.push_back(4.14);
nWeights.push_back(5.19);
nWeights.push_back(6.18);
nWeights.push_back(7.16);
nWeights.push_back(8.14);
nWeights.push_back(9.12);
nWeights.push_back(10.31);

for (int i=0; i<nWeights.size() ; i++) {
/ Let's create a vector with three neurons
    ptN.reset( new Neuron( ids[i] ) );
    neuronContainer.push_back(ptN);
}
conContainerPtr->buildAndAppend(neuronContainer, nWeights
);

// Test

std::vector<int> toRemove;
toRemove.push_back(1);
toRemove.push_back(3);
toRemove.push_back(5);
toRemove.push_back(7);

conContainerPtr->erase(toRemove);
conContainerPtr->show();
result=conContainerPtr->getId();

// The output at the R terminal would display :
//
// From:      2      Weight=      9.120000
// From:      4      Weight=      4.140000
// From:      6      Weight=      6.180000
// From:      8      Weight=      8.140000
// From:      9      Weight=      2.140000
// From:     10      Weight=      1.260000
// From:     11      Weight=     11.320000

```

**See also**

[select](#) and the unit test files, e.g. `runit.Cpp.ConContainer.R`, for further examples.

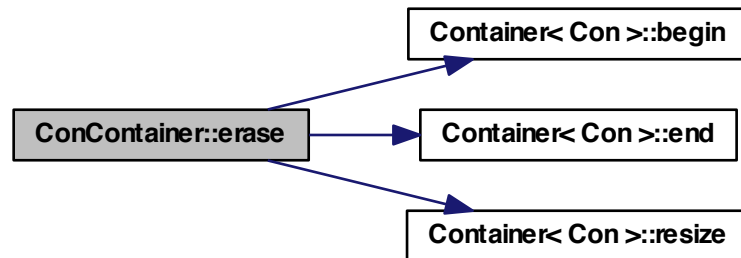
Definition at line 450 of file `ConContainer.cpp`.

References `Container< Con >::begin()`, `Container< Con >::end()`, and `Container<`

Con >::resize().

```
{
    std::vector<ConPtr>::iterator itr;
    sort(begin(), end(), CompareId());
    sort(nIds.begin(), nIds.end());
    itr = set_difference(begin(), end(), nIds.begin(), nIds.end(), begin(),
        CompareId());
    resize(itr - begin());
}
```

Here is the call graph for this function:



#### 5.7.4.2 std::vector< int > ConContainer::getId ( )

Getter of the Id values of the vector of Cons.

This function returns the Id's of the neurons referred to by the vector of Cons.

#### Returns

An std::vector<int> that contains the Ids

```
//=====
//Usage example:
//=====
// Data set up
Neuron N1, N2, N3;
ConContainer conContainer;
std::vector<int> result;

N1.setId(10);
N2.setId(20);
N3.setId(30);

ConPtr ptCon( new Con(&N1, 1.13) );    // Create new Con
```



```

    and initialize ptCon
        conContainer.push_back(ptCon);
/ push_back
        ptCon.reset( new Con(&N2, 2.22) );
new Con and assign to ptCon
        conContainer.push_back(ptCon);
/ push_back
        ptCon.reset( new Con(&N3, 3.33) );
new Con and assign to ptCon
        conContainer.push_back(ptCon);
/ push_back

// Test
        conContainer.show() ;
        conContainer.validate();
        result=conContainer.getId();

// Now result is a vector that contains the values 10, 20 and 30.

```

**See also**

getWeight and the unit test files, e.g. `runit.Cpp.ConContainer.R`, for further examples.

Definition at line 93 of file `ConContainer.cpp`.

References `numOfCons()`.

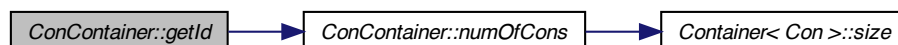
Referenced by `validate()`.

```

{
    std::vector<int> result;
    result.reserve(numOfCons());
    foreach (ConPtr itr, *this)
    {
        result.push_back(itr->getId());
    }
    return result;
}

```

Here is the call graph for this function:



Here is the caller graph for this function:



#### 5.7.4.3 int ConContainer::numOfCons ( )

Size of the [ConContainer](#) object.

This function returns the size of the [ConContainer](#) object, that is to say, the number of [Con](#) objects it contains.

#### Returns

The size of the vector

```

//=====
//Usage example:
//=====
// Data set up

Container<Neuron>( ) );
ConContainer( ) );

std::vector<int> result;
std::vector<ConPtr> vcA, vcB;
ContainerNeuronPtr neuronContainerPtr( new
ConContainerPtr conContainerPtr( new
ConPtr ptC;
NeuronPtr ptN;
int ids[] = {10, 20, 30};
double weights[] = {1.13, 2.22, 3.33 };
for (int i=0; i<=2 ; i++) {
    ptN.reset( new Neuron( ids[i] ) );
    neuronContainerPtr->push_back(ptN);
}

// Test
for (int i=0; i<=2 ; i++) {
    result.push_back(conContainerPtr->numOfCons());
    // Append numOfCons to result, create new Con and push_back into
    conContainer
    ptC.reset( new Con( neuronContainerPtr->load().at(i), weights[i] ) );
    conContainerPtr->push_back(ptC);
}

// Now, result contains a numeric vector with values 0, 1, 2, and 3.
  
```

**See also**

[Container::size](#) (alias)

Definition at line 52 of file ConContainer.cpp.

References `Container< Con >::size()`.

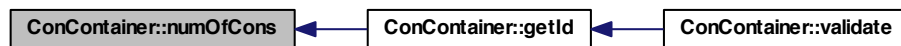
Referenced by `getId()`.

```
{
    return size();
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



#### 5.7.4.4 ConContainerPtr ConContainer::select ( `std::vector< int > nlds` )

Selects the specified elements from the `vecCom` object.

Provides a convenient way of selecting some [Con](#) objects from the collection field of the [ConContainer](#) object.

**Parameters**

|              |   |
|--------------|---|
| <i>vFrom</i> | An <code>std::vector&lt;int&gt;</code> with the Ids of the connections to select. |
|--------------|---|

```
//=====
//Usage example:
```

```

//=====

// Data set up
std::vector<int> result;
std::vector<NeuronPtr> neuronContainer;
ConContainerPtr conContainerPtr( new ConContainer() );
ConPtr ptC;
NeuronPtr ptN;
int ids[] = {11, 10, 9, 3, 4, 5, 6, 7, 8, 2, 1};
double weights[] = {11.32, 1.26, 2.14, 3.16, 4.14, 5.19, 6.18, 7.16
, 8.14, 9.12, 10.31};
std::vector<double> nWeights;
for (int i=0; i<11; i++) {
    nWeights.push_back(weights[i]);
}
for (int i=0; i<nWeights.size() ; i++) {
/ Let's create a vector with three neurons
    ptN.reset( new Neuron( ids[i] ) );
    neuronContainer.push_back(ptN);
}
conContainerPtr->buildAndAppend(neuronContainer, nWeights);
// Test
std::vector<int> toSelect;
toSelect.push_back(1);
toSelect.push_back(3);
toSelect.push_back(5);
toSelect.push_back(7);

ConContainerPtr vSelect ( conContainerPtr->select(toSelect) );

result=vSelect->getId();

// Now, result is a numeric vector with the values 1, 3, 5 and 7.

```

### See also

[erase](#) and the unit test files, e.g. `runit.Cpp.ConContainer.R`, for further examples.

Definition at line 505 of file `ConContainer.cpp`.

References `Container< Con >::begin()`, `Container< Con >::end()`, and `Container< Con >::size()`.

Referenced by `setWeight()`.

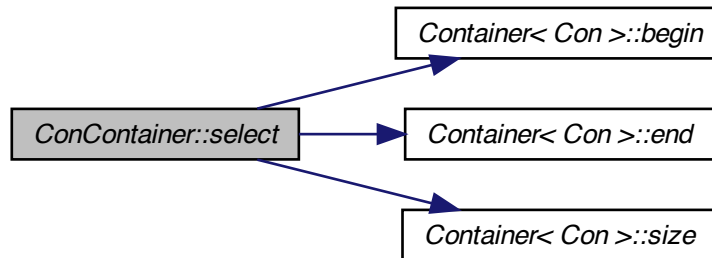
```

{
    ConContainerPtr result(new ConContainer);
    result->reserve(size());
    sort(begin(), end(), CompareId());
    sort(nIds.begin(), nIds.end());
    set_intersection(begin(), end(), nIds.begin(), nIds.end(),
        std::back_inserter(*result), CompareId());

    return result;
}

```

Here is the call graph for this function:



Here is the caller graph for this function:



#### 5.7.4.5 bool ConContainer::setFrom ( NeuronContainer neuronContainer )

Setter of the from fields of the [Con](#) objects related to [ConContainer](#).

This function provides a convenient way of getting the values of the weight field of those [Con](#) object pointed to by the smart pointer stored in the [ConContainer](#) object.

##### Parameters

|              |   |
|--------------|---|
| <i>vFrom</i> | An <code>std::vector&lt;NeuronPtr&gt;</code> with the pointers to be set in the from fields of the <a href="#">ConContainer</a> object. |
|--------------|---|

##### Returns

true if not exception is thrown

```
//=====
//Usage example:
```

```

//=====

// Data set up
std::vector<int> result;
ContainerNeuronPtr      neuronContainerPtr( new
Container<Neuron>() );
ConContainerPtr conContainerPtr( new ConContainer() );
ConPtr  ptC;
NeuronPtr ptN;

int ids[] = {10, 20, 30};
double weights[] = {1.13, 2.22, 3.33 };

    for (int i=0; i<=2 ; i++) {                                // Let's
create a vector with three neurons
        ptN.reset( new Neuron( ids[i] ) );
        neuronContainerPtr->push_back(ptN);
    }
    for (int i=0; i<=2 ; i++) {                                // and a
vector with three connections
        ptC.reset( new Con() );
        conContainerPtr->push_back(ptC);
    }
// Test
conContainerPtr->setFrom(neuronContainerPtr->load()) ;
conContainerPtr->show();
result=conContainerPtr->getId();

// Now result is a vector that contains the values 10, 20 and 30.

```

### See also

getFrom and the unit test files, e.g. runit.Cpp.ConContainer.R, for further examples.

Definition at line 333 of file ConContainer.cpp.

References `Container< T >::begin()`, `Container< T >::empty()`, `Container< Con >::size()`, and `Container< T >::size()`.

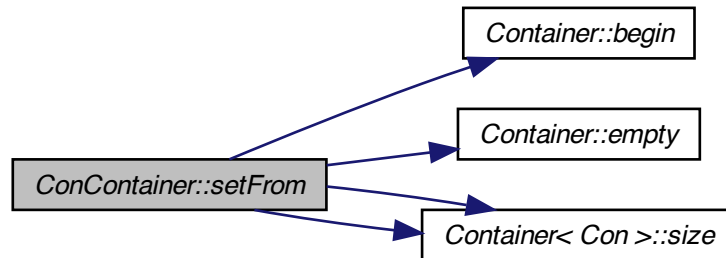
```

{
    BEGIN_RCPP
    if (neuronContainer.empty())
        { throw std::range_error("[ C++ ConContainer::setFrom]: Error, w is empty");}

    if (neuronContainer.size() != size())
        {
            throw std::range_error(
                "[C++ ConContainer::setFrom]: Error, neuronContainer.size() != collecti
on.size()");
        }
    std::vector<NeuronPtr>::iterator itrNeuron = neuronContainer.begin();
    foreach(ConPtr itr , *this)
        {
            itr->setFrom( *itrNeuron );
            itrNeuron++;
        }
    return true;
    END_RCPP
}

```

Here is the call graph for this function:



#### 5.7.4.6 bool ConContainer::setWeight ( std::vector< double > nWeights, std::vector< int > nlds )

Setter of the weights of the specified elements from the [ConContainer](#) object.

Provides a convenient way of setting the weights of some [Con](#) objects from the collection field of the [ConContainer](#) object.

##### Parameters

|                 |   |
|-----------------|---|
| <i>nWeights</i> | A numeric (double) vector with the weights to be set in the <a href="#">Con</a> objects contained in the <a href="#">ConContainer</a> object. |
| <i>vFrom</i>    | An std::vector<int> with the lds of the connections to select   |

##### Returns

true in case no exception is thrown

```

//=====
//Usage example:
//=====

// Data set up
std::vector<double> result;
std::vector<NeuronPtr> neuronContainer;
ConContainerPtr conContainerPtr( new ConContainer() );
ConPtr ptC;
NeuronPtr ptN;
int ids[]={11, 10, 9, 3, 4, 5, 6, 7, 8, 2, 1};
double weights[]={11.32, 1.26, 2.14, 3.16, 4.14, 5.19, 6.
18, 7.16, 8.14, 9.12, 10.31};
std::vector<double> nWeights;
for (int i=0; i<11; i++) {
    nWeights.push_back(weights[i]);
  
```

```

    }
    for (int i=0; i<nWeights.size() ; i++) {
/ Let's create a vector with three neurons
    ptN.reset( new Neuron( ids[i] ) );
    neuronContainer.push_back(ptN);
    }
    conContainerPtr->buildAndAppend(neuronContainer, nWeights
);

    std::vector<int> toSelect;
    std::vector<double> vNewWeights;
    toSelect.push_back(1);
    toSelect.push_back(3);
    toSelect.push_back(5);
    toSelect.push_back(7);
    vNewWeights.push_back(1000.1);
    vNewWeights.push_back(3000.3);
    vNewWeights.push_back(5000.5);
    vNewWeights.push_back(7000.7);
    conContainerPtr->setWeight(vNewWeights, toSelect);

// Test

    result = conContainerPtr->getWeight();
    return wrap(result);

// Now, result is a numeric vector with the values 1000.10, 9.12, 3000.3
0, 4.14, 5000.50, 6.18, 7000.70, 8.14, 2.14, 1.26 and 11.32 .

```

### See also

getWeigth and the unit test files, e.g. runit.Cpp.ConContainer.R, for further examples.

Definition at line 627 of file ConContainer.cpp.

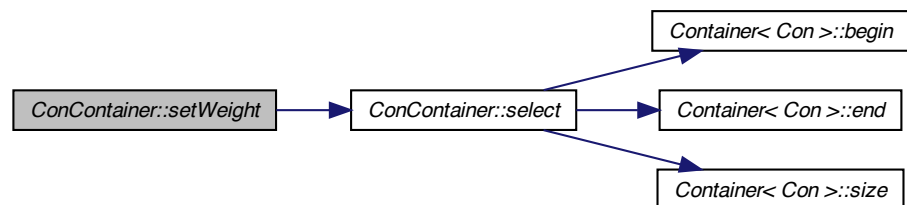
References `select()`.

```

{
BEGIN_RCPP return select(nIds)->setWeight(nWeights);
END_RCPP
}

```

Here is the call graph for this function:





## 5.7.4.7 bool ConContainer::setWeight ( std::vector&lt; double &gt; nWeights )

Setter of the weight field of the [Con](#) objects related to [ConContainer](#).

This function provides a convenient way of setting the values of the weight field of those [Con](#) objects pointed to by the smart pointer stored in the [ConContainer](#) object.

**Parameters**

|                 |   |
|-----------------|---|
| <i>nWeights</i> | A numeric (double) vector with the weights to be set in the <a href="#">Con</a> objects contained in the <a href="#">ConContainer</a> object. |
|-----------------|---|

**Returns**

true in case no exception is thrown

```
//=====
//Usage example:
//=====
// Data set up
std::vector<double> result;
int ids[]= {1, 2, 3};
double weights[] = {12.3, 1.2, 2.1 };
ConContainer conContainer;
std::vector<NeuronPtr> neuronContainer;
std::vector<double> nWeights;
NeuronPtr ptNeuron;

for (int i=0; i<=2; i++) {
ptNeuron.reset( new Neuron(ids[i]) );
neuronContainer.push_back(ptNeuron);
nWeights.push_back(0);
/
/ weights are set to 0
}
conContainer.buildAndAppend(neuronContainer, nWeights);
conContainer.show();

for (int i=0; i<=2; i++) {
nWeights.at(i)=weights[i];
}

// Test
conContainer.setWeight(nWeights);
/
/ weights are set to 12.3, 1.2 and 2.1
result=conContainer.getWeight();

// Now result is a vector that contains the values 12.3, 1.2 and 2.1 .
```

**See also**

[getWeight](#) and the unit test files, e.g. `runit.Cpp.ConContainer.R`, for further examples.

Definition at line 270 of file `ConContainer.cpp`.

References `Container< Con >::size()`.

```
{
BEGIN_RCPP
```

```

if (nWeights.empty())
{ throw std::range_error("[ C++ ConContainer::setWeight]: Error, nWeights is
  empty");}
if (nWeights.size() != size())
{
  throw std::range_error(
    "[C++ ConContainer::setWeight]: Error, nWeights.size() != collection.si
    ze()");
}
std::vector<double>::iterator itrWeight = nWeights.begin();
foreach (ConPtr itr, *this)
{
  itr->setWeight( *itrWeight );
  itrWeight++;
}
return true;
END_RCPP}

```

Here is the call graph for this function:



#### 5.7.4.8 `bool ConContainer::validate ( ) [virtual]`

Object validator.

This method checks the object for internal coherence. A try / catch mechanism exits normal execution and returns control to the R terminal in case the contents of the [ConContainer](#) object are identified as corrupted.

#### Returns

true in case the checks are Ok.

#### Exceptions

|           |  |
|-----------|--|
| <i>An</i> | std::range error if weight or from are not finite. |
|-----------|--|

#### See also

The unit test files, e.g., `runit.Cpp.ConContainer.R`, for usage examples.

Reimplemented from [Container< Con >](#).

Definition at line 645 of file `ConContainer.cpp`.

References getId().

```
{
    BEGIN_RCPP

    std::vector<int>::iterator itr;
    std::vector<int> vIds = getId();
    sort(vIds.begin(), vIds.end());
    itr = adjacent_find(vIds.begin(), vIds.end());
    if (itr != vIds.end())
        throw std::range_error(
            "[C++ ConContainer::validate]: Error, duplicated Id.");
    Container<Con>::validate();
    return (true);
END_RCPP};
```

Here is the call graph for this function:



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

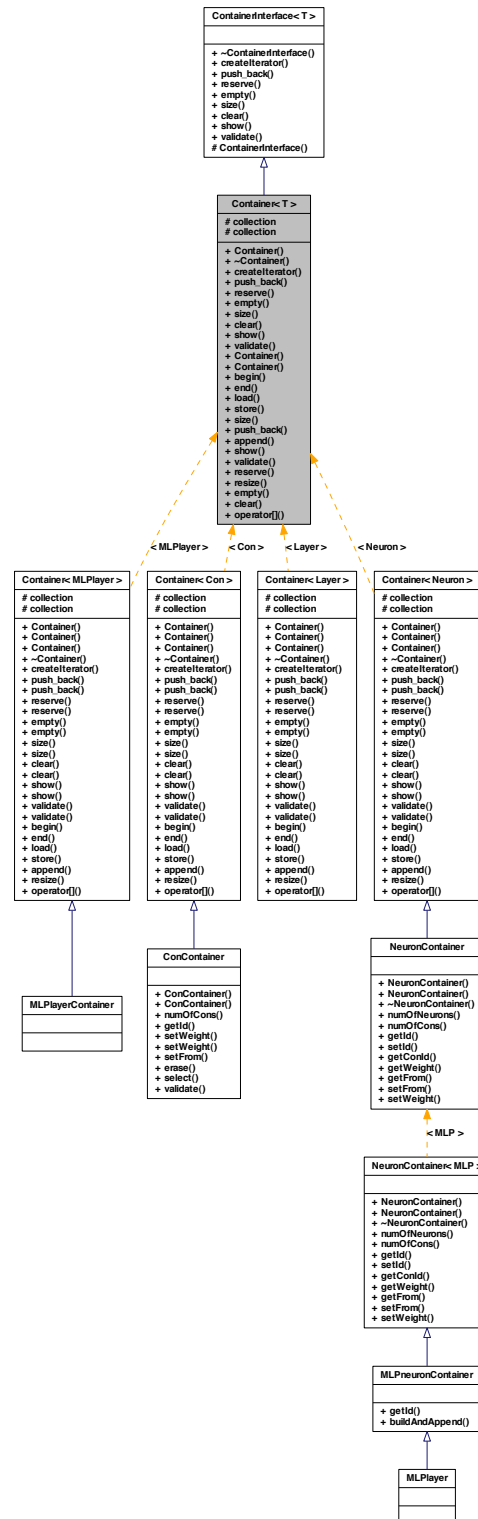
- pkg/AMORE/src/old/[ConContainer.h](#)
- pkg/AMORE/src/old/[ConContainer.cpp](#)

## 5.8 Container< T > Class Template Reference

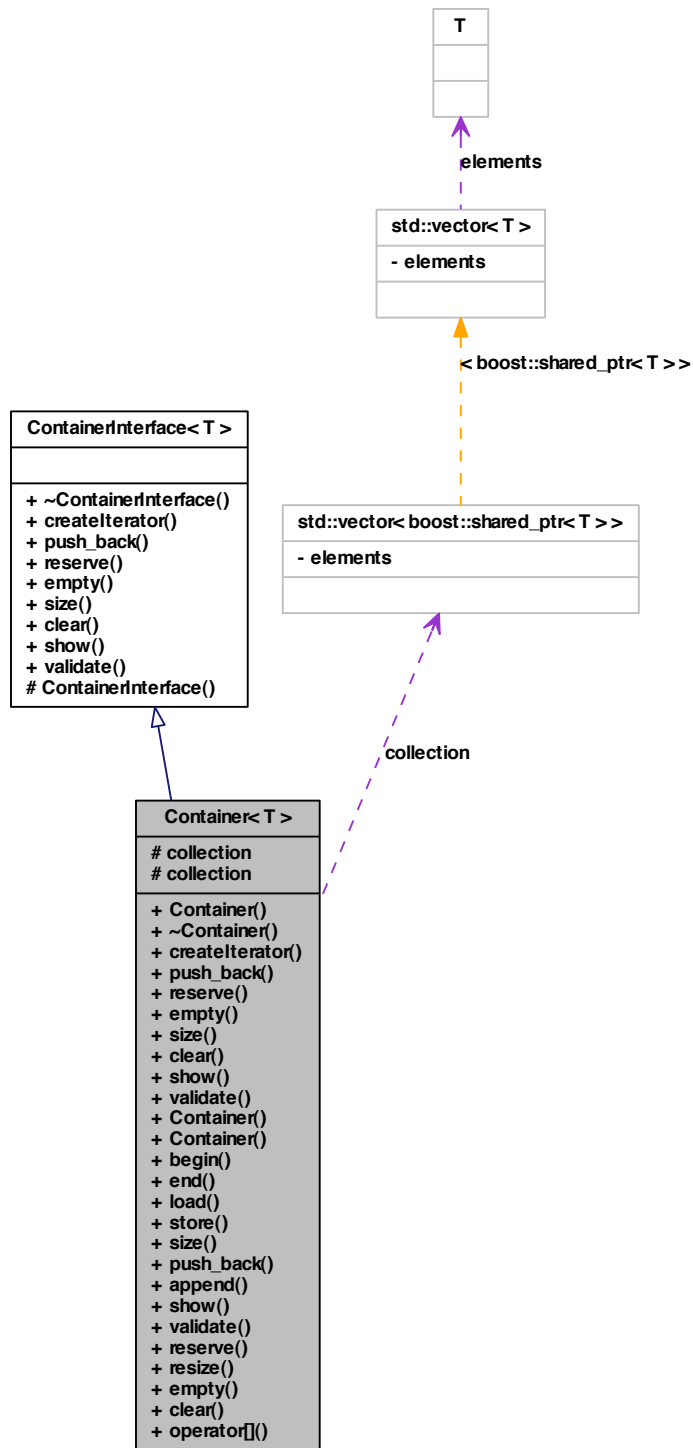
class [Container](#) -

```
#include <Container.h>
```

Inheritance diagram for Container< T >:



Collaboration diagram for Container< T >:



## Public Types

- typedef std::vector< boost::shared\_ptr< T > >::iterator iterator
- typedef std::vector< boost::shared\_ptr< T > >::const\_iterator const\_iterator
- typedef boost::shared\_ptr< T > value\_type
- typedef value\_type const & const\_reference

## Public Member Functions

- Container ()
- ~Container ()
- boost::shared\_ptr< IteratorInterface< T > > createIterator ()
- void push\_back (T const &const\_reference)  
*Append a shared\_ptr at the end of collection.*
- void reserve (int n)
- bool empty ()
- size\_type size ()  
*Returns the size or length of the vector.*
- void clear ()
- void show ()  
*Pretty print of the Container< T >*
- bool validate ()  
*Object validator.*
- Container ()
- Container (std::vector< value > first, std::vector< value > last)
- iterator begin ()
- iterator end ()
- std::vector< boost::shared\_ptr< T > > load ()  
*collection field accessor function*
- void store (typename std::vector< boost::shared\_ptr< T > > collectionT)  
*collection field accessor function*
- size\_type size ()
- void push\_back (boost::shared\_ptr< T > const &const\_reference)  
*Append a shared\_ptr at the end of collection.*
- void append (Container< T > containerT)  
*Appends a Container< T > object.*
- bool show ()
- bool validate ()
- void reserve (int n)
- void resize (int n)
- bool empty ()
- void clear ()
- boost::shared\_ptr< T > & operator[] (size\_type offset)

### Protected Attributes

- `std::vector< T >` [collection](#)
- `std::vector< boost::shared_ptr< T > >` [collection](#)

### Friends

- class [ContainerIterator< T >](#)

### 5.8.1 Detailed Description

`template<typename T>class Container< T >`

class [Container](#) -

Definition at line 6 of file `Container.h`.

### 5.8.2 Member Typedef Documentation

**5.8.2.1** `template<typename T> typedef std::vector<boost::shared_ptr<T>  
>::const_iterator Container< T >::const_iterator`

Reimplemented in [ConContainer](#), and [NeuronContainer](#).

Definition at line 22 of file `Container.h`.

**5.8.2.2** `template<typename T> typedef value_type const& Container< T  
>::const_reference`

Reimplemented in [ConContainer](#), and [NeuronContainer](#).

Definition at line 26 of file `Container.h`.

**5.8.2.3** `template<typename T> typedef std::vector<boost::shared_ptr<T> >::iterator  
Container< T >::iterator`

Reimplemented in [ConContainer](#), and [NeuronContainer](#).

Definition at line 19 of file `Container.h`.

**5.8.2.4** `template<typename T> typedef boost::shared_ptr<T> Container< T  
>::value_type`

Reimplemented in [ConContainer](#), and [NeuronContainer](#).

Definition at line 24 of file `Container.h`.

### 5.8.3 Constructor & Destructor Documentation

#### 5.8.3.1 `template<typename T> Container< T>::Container ( )`

Definition at line 11 of file Container.cpp.

```
{
}
```

#### 5.8.3.2 `template<typename T> Container< T>::~~Container ( )`

Definition at line 17 of file Container.cpp.

```
{
}
```

#### 5.8.3.3 `template<typename T> Container< T>::Container ( )`

#### 5.8.3.4 `template<typename T> Container< T>::Container ( std::vector< value > first, std::vector< value > last )`

### 5.8.4 Member Function Documentation

#### 5.8.4.1 `template<typename T> void Container< T>::append ( Container< T> v )`

Appends a Container<T> object.

This method inserts the collection field of a second object at the end of the collection field of the calling object.

#### Parameters

|   |  |
|---|--|
| v | The Container<T> object to be added to the current one |
|---|--|

#### See also

The unit test files, e.g., `runit.Cpp.Container.R`, for usage examples.

```
//=====
//Usage example:
//=====
// Data set up

Container<Neuron>() );
Container<Con>() );
Container<Con>() );

std::vector<int> result;
std::vector<ConPtr> vcA, vcB;
ContainerNeuronPtr neuronContainerPtr( new
ContainerConPtr conContainerPtrA( new
ContainerConPtr conContainerPtrB( new
```



```

        ConPtr ptC;
        NeuronPtr ptN;
        int ids[] = {1, 2, 3, 4, 5, 6};
        double weights[] = {1.13, 2.22, 3.33, 5.6, 4.2, 3
.6 };
        for (int i=0; i<=5 ; i++) {
/
/ Let's create a vector with six neurons
            ptN.reset( new Neuron( ids[i] ) );
            neuronContainerPtr->push_back(ptN);
        }
        for (int i=0; i<=2 ; i++) {
/
/ A vector with three connections
            ptC.reset( new Con( neuronContainerPtr->load().at(i), weights[i]) );
            conContainerPtrA->push_back(ptC);
        }
        for (int i=3; i<=5 ; i++) {
/
/ Another vector with three connections
            ptC.reset( new Con( neuronContainerPtr->load().at(i), weights[i]) );
            conContainerPtrB->push_back(ptC);
        }

        // Test
        conContainerPtrA->append(*conContainerPtrB);
        conContainerPtrA->validate();
        conContainerPtrA->show() ;

        // After execution of the code above, the output at the R terminal would
        display:
        //
        // From:      1      Weight=      1.130000
        // From:      2      Weight=      2.220000
        // From:      3      Weight=      3.330000
        // From:      4      Weight=      5.600000
        // From:      5      Weight=      4.200000
        // From:      6      Weight=      3.600000

```

**See also**

[Container::store](#) , [Container::push\\_back](#) and the unit test files, e.g., `runit.Cpp.Container.R`, for usage examples.

Definition at line 207 of file `Container.cpp`.

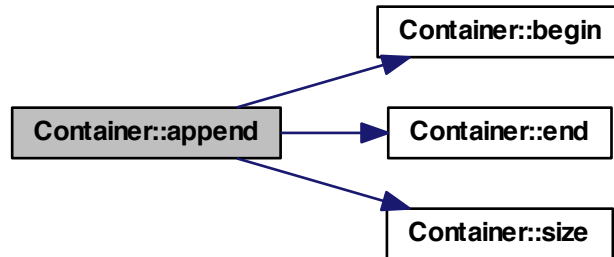
References `Container< T >::begin()`, `Container< T >::end()`, and `Container< T >::size()`.

```

{
    reserve(size() + v.size());
    collection.insert(end(), v.begin(), v.end());
}

```

Here is the call graph for this function:



#### 5.8.4.2 `template<typename T> std::vector< boost::shared_ptr< T > >::iterator Container< T >::begin ( )`

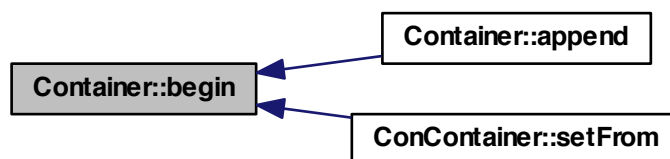
Definition at line 22 of file `Container.cpp`.

Referenced by `Container< T >::append()`, and `ConContainer::setFrom()`.

```

{
    return collection.begin();
}
  
```

Here is the caller graph for this function:



#### 5.8.4.3 `template<typename T> void Container< T >::clear ( ) [virtual]`

Implements [ContainerInterface< T >](#).

Definition at line 177 of file Container.cpp.

```
{  
    collection.clear();  
}
```

**5.8.4.4** `template<typename T> void Container< T >::clear ( ) [virtual]`

Implements [ContainerInterface< T >](#).

**5.8.4.5** `template<typename T > boost::shared_ptr< IteratorInterface< T > >  
Container< T >::createIterator ( ) [virtual]`

Implements [ContainerInterface< T >](#).

Definition at line 23 of file Container.cpp.

```
{  
    boost::shared_ptr< ContainerIterator<T> > containerIteratorPtr( new  
        ContainerIterator<T> ());  
    containerIteratorPtr->d_container = this;  
    containerIteratorPtr->d_iterator = collection.begin();  
    return containerIteratorPtr;  
}
```

**5.8.4.6** `template<typename T > bool Container< T >::empty ( ) [virtual]`

Implements [ContainerInterface< T >](#).

Definition at line 163 of file Container.cpp.

Referenced by `ConContainer::setFrom()`.

```
{  
    return (collection.empty());  
}
```

Here is the caller graph for this function:



5.8.4.7 `template<typename T> bool Container< T >::empty ( )` [virtual]

Implements [ContainerInterface< T >](#).

5.8.4.8 `template<typename T > std::vector< boost::shared_ptr< T > >::iterator  
Container< T >::end ( )`

Definition at line 29 of file Container.cpp.

Referenced by `Container< T >::append()`.

```
{
    return collection.end();
}
```

Here is the caller graph for this function:



5.8.4.9 `template<typename T > std::vector< boost::shared_ptr< T > > Container< T  
>::load ( )`

collection field accessor function

This method allows access to the data stored in the [collection](#) field.

### Returns

The collection vector.

```
//=====
//Usage example:
//=====
// Data set up
std::vector<int> result;
std::vector<ConPtr> vcA, vcB;
ContainerNeuronPtr neuronContainerPtr( new
ContainerConPtr conContainerPtr( new
ConPtr ptC;
NeuronPtr ptN;
int ids[] = {10, 20, 30};
Container<Neuron>() );
Container<Con>() );
```

```

        double weights[] = {1.13, 2.22, 3.33 };
        for (int i=0; i<=2 ; i++) {
            /
            / Let's create a vector with three neurons
            ptN.reset( new Neuron( ids[i] ) );
            neuronContainerPtr->push_back(ptN);
        }
        for (int i=0; i<=2 ; i++) {
            /
            / and a vector with three connections
            ptC.reset( new Con( neuronContainerPtr->load().at(i), weights[i]) );
            vcA.push_back(ptC);
        }
        // Test
        conContainerPtr->store(vcA);
        vcB = conContainerPtr->load();
        for (int i=0; i<=2 ; i++) {
            /
            / get Ids. Container does not have getId defined
            result.push_back( vcB.at(i)->getId());
        }

        // Now, result is an integer vector with values 10, 20, 30.

```

**See also**

[store](#) and the unit test files, e.g., `runit.Cpp.Container.R`, for usage examples.

Definition at line 254 of file `Container.cpp`.

```

{
    return collection;
}

```

#### 5.8.4.10 `template<typename T> boost::shared_ptr< T > & Container< T >::operator[] ( size_type offset )`

Definition at line 317 of file `Container.cpp`.

```

{
    return collection[offset];
}

```

#### 5.8.4.11 `template<typename T> void Container< T >::push_back ( T const & reference ) [virtual]`

Append a `shared_ptr` at the end of collection.

Implements `push_back` for the [Container](#) class

**Parameters**

|                   |   |
|-------------------|---|
| <i>TsharedPtr</i> | A <code>shared_ptr</code> pointer to be inserted at the end of collection |
|-------------------|---|

```
//=====
```

```

//Usage example:
//=====
// Data set up
    Neuron N1, N2, N3;
    Container<Con> conContainer;
    std::vector<ConPtr> vc;
    std::vector<int> result;
    N1.setId(10);
    N2.setId(20);
    N3.setId(30);

// Test
    ConPtr ptCon( new Con(&N1, 1.13) );      // Create new Con
and initialize ptCon
    conContainer.push_back(ptCon);           /
/ push_back
    ptCon.reset( new Con(&N2, 2.22) );      // create
new Con and assign to ptCon
    conContainer.push_back(ptCon);           /
/ push_back
    ptCon.reset( new Con(&N3, 3.33) );      // create
new Con and assign to ptCon
    conContainer.push_back(ptCon);           /
/ push_back

    vc = conContainer.load();

    result.push_back(vc.at(0)->getId());
    result.push_back(vc.at(1)->getId());
    result.push_back(vc.at(2)->getId());
// After execution of this code, result contains a numeric vector with va
lues 10, 20 and 30.

```

### See also

C++ documentation for `std::vector::push_back` and the unit test files, e.g., `runit.Cpp.Container.R`, for usage examples.

Implements [ContainerInterface< T >](#).

Definition at line 68 of file `Container.cpp`.

```

{
    collection.push_back(reference);
}

```

**5.8.4.12** `template<typename T> void Container< T >::push_back ( boost::shared_ptr< T > const & const.reference )`

Append a `shared_ptr` at the end of collection.

Implements `push_back` for the [Container](#) class

### Parameters

|                   |   |
|-------------------|---|
| <i>TsharedPtr</i> | A <code>shared_ptr</code> pointer to be inserted at the end of collection |
|-------------------|---|

```

//=====

```

```

//Usage example:
//=====
// Data set up
    Neuron N1, N2, N3;
    Container<Con> conContainer;
    std::vector<ConPtr> vc;
    std::vector<int> result;
    N1.setId(10);
    N2.setId(20);
    N3.setId(30);

// Test
    ConPtr ptCon( new Con(&N1, 1.13) );      // Create new Con
and initialize ptCon
    conContainer.push_back(ptCon);           /
/ push_back
    ptCon.reset( new Con(&N2, 2.22) );      // create
new Con and assign to ptCon
    conContainer.push_back(ptCon);           /
/ push_back
    ptCon.reset( new Con(&N3, 3.33) );      // create
new Con and assign to ptCon
    conContainer.push_back(ptCon);           /
/ push_back

    vc = conContainer.load();

    result.push_back(vc.at(0)->getId());
    result.push_back(vc.at(1)->getId());
    result.push_back(vc.at(2)->getId());

// After execution of this code, result contains a numeric vector with va
lues 10, 20 and 30.

```

**See also**

C++ documentation for `std::vector::push_back` and the unit test files, e.g., `runit.Cpp.Container.R`, for usage examples.

Definition at line 71 of file `Container.cpp`.

```

{
    collection.push_back(const_reference);
}

```

**5.8.4.13** `template<typename T> void Container< T >::reserve ( int n )` [virtual]

Implements [ContainerInterface< T >](#).

**5.8.4.14** `template<typename T> void Container< T >::reserve ( int n )` [virtual]

Implements [ContainerInterface< T >](#).

Definition at line 170 of file `Container.cpp`.

```

{
    collection.reserve(n);
}

```

#### 5.8.4.15 `template<typename T> void Container<T>::resize ( int n )`

Definition at line 289 of file Container.cpp.

```
{
    collection.resize(n);
}
```

#### 5.8.4.16 `template<typename T> bool Container<T>::show ( ) [virtual]`

Implements [ContainerInterface<T>](#).

#### 5.8.4.17 `template<typename T> bool Container<T>::show ( ) [virtual]`

Pretty print of the Container<T>

This method outputs in the R terminal the contents of [Container::collection](#).

#### Returns

true in case everything works without throwing an exception

\*

```
//=====
//Usage example:
//=====
// Data set up
ContainerNeuronPtr      neuronContainerPtr( new
Container<Neuron>() );
ContainerConPtr conContainerPtr( new Container<Con>() );
ConPtr ptC;
NeuronPtr ptN;
int ids[] = {10, 20, 30};
double weights[] = {1.13, 2.22, 3.33 };

    for (int i=0; i<=2 ; i++) {
/ Let's create a vector with three neurons
        ptN.reset( new Neuron( ids[i] ) );
        neuronContainerPtr->push_back(ptN);
    }

    for (int i=0; i<=2 ; i++) {
/ and a vector with three connections
        ptC.reset( new Con( neuronContainerPtr->load().at
(i), weights[i]) );
        conContainerPtr->push_back(ptC);
    }

// Test
    conContainerPtr->show() ;

// The output at the R terminal would display:
//
//      # From:  10      Weight=      1.130000
```



```
//      # From:  20      Weight=      2.220000
//      # From:  30      Weight=      3.330000
//
```

**See also**

The unit test files, e.g., `runit.Cpp.Container.R`, for usage examples.

Implements [ContainerInterface< T >](#).

Definition at line 118 of file `Container.cpp`.

```
{
    for (typename std::vector<T>::iterator itr(collection.begin()); itr
        != collection.end(); ++itr)
    {
        itr->show();
    }
}
```

**5.8.4.18** `template<typename T> size_type Container< T >::size ( )` [virtual]

Implements [ContainerInterface< T >](#).

**5.8.4.19** `template<typename T> size_type Container< T >::size ( )` [virtual]

Returns the size or length of the vector.

This method returns the size of the vector. In the classes derived from `Container<T>` this is aliased as `numOfCons`, `numOfNeurons` and `numOfLayers`. The unit test files, e.g., `runit.Cpp.Container.R`, for usage examples.

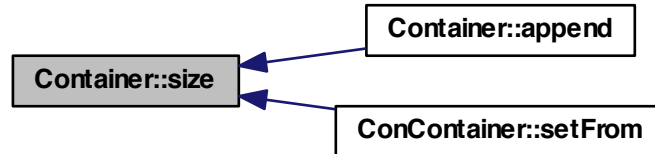
Implements [ContainerInterface< T >](#).

Definition at line 155 of file `Container.cpp`.

Referenced by `Container< T >::append()`, and `ConContainer::setFrom()`.

```
{
    return collection.size();
}
```

Here is the caller graph for this function:



**5.8.4.20** `template<typename T> void Container< T >::store ( typename std::vector< boost::shared_ptr< T > > collectionT )`

collection field accessor function

This method sets the value of the data stored in the [collection](#) field.

#### Parameters

|   |   |
|---|---|
| v | The vector of smart pointers to be stored in the collection field |
|---|---|

#### See also

[load](#) and the unit test files, e.g., `runit.Cpp.Container.R`, for usage examples.

Definition at line 268 of file `Container.cpp`.

```

{
    collection = collectionT;
}
  
```

**5.8.4.21** `template<typename T> bool Container< T >::validate ( ) [virtual]`

Implements [ContainerInterface< T >](#).

Reimplemented in [ConContainer](#).

**5.8.4.22** `template<typename T> bool Container< T >::validate ( ) [virtual]`

Object validator.

This method checks the object for internal coherence. This method calls the `validate` method for each element in collection,

**See also**

The unit test files, e.g., `runit.Cpp.Container.R`, for usage examples.

Implements [ContainerInterface< T >](#).

Reimplemented in [ConContainer](#).

Definition at line 136 of file `Container.cpp`.

```
{
    for (typename std::vector<T>::iterator itr(collection.begin()); itr
        != collection.end(); ++itr)
    {
        itr->validate();
    }
    return true;
}
```

**5.8.5 Friends And Related Function Documentation**

**5.8.5.1** `template<typename T> friend class ContainerIterator< T >` `[friend]`

Definition at line 12 of file `Container.h`.

**5.8.6 Member Data Documentation**

**5.8.6.1** `template<typename T> std::vector< T > Container< T >::collection`  
`[protected]`

Definition at line 9 of file `Container.h`.

**5.8.6.2** `template<typename T> std::vector<boost::shared_ptr<T> > Container< T`  
`>::collection` `[protected]`

Definition at line 15 of file `Container.h`.

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

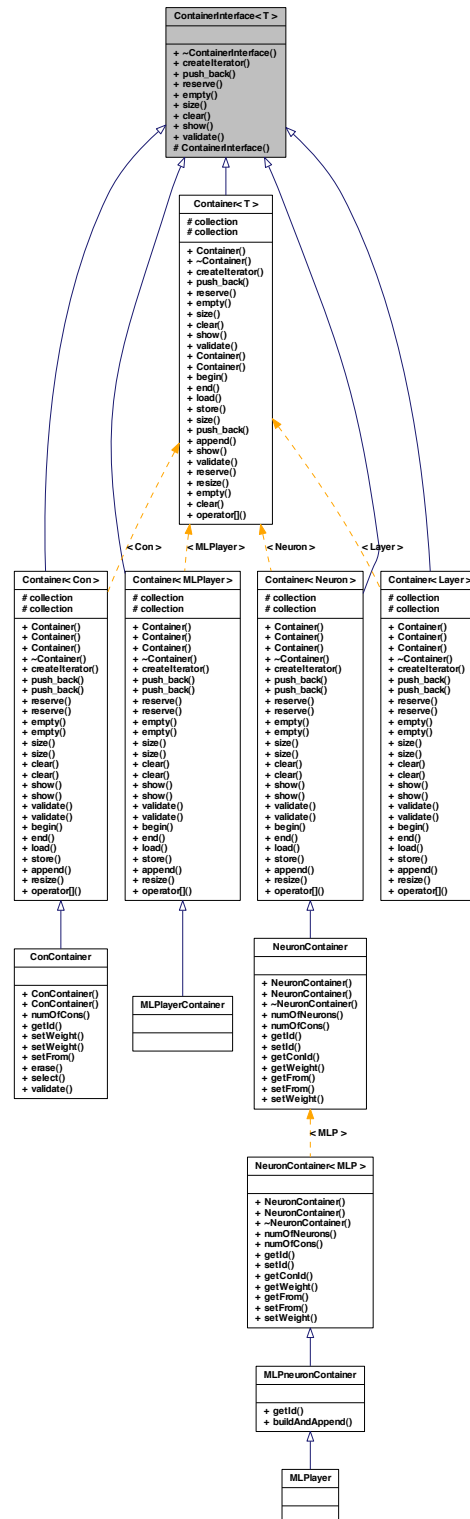
- `pkg/AMORE/src/dia/Container.h`
- `pkg/AMORE/src/old/Container.h`
- `pkg/AMORE/src/Container.cpp`
- `pkg/AMORE/src/old/Container.cpp`

**5.9 ContainerInterface< T > Class Template Reference**

class [ContainerInterface](#) -

`#include <ContainerInterface.h>`

Inheritance diagram for ContainerInterface< T >:



## Public Member Functions

- virtual [~ContainerInterface](#) ()
- virtual boost::shared\_ptr< [IteratorInterface](#)< T > > [createIterator](#) ()=0
- virtual void [push\\_back](#) (T const &const\_reference)=0
- virtual void [reserve](#) (int n)=0
- virtual bool [empty](#) ()=0
- virtual size\_type [size](#) ()=0
- virtual void [clear](#) ()=0
- virtual void [show](#) ()=0
- virtual bool [validate](#) ()=0

## Protected Member Functions

- [ContainerInterface](#) ()

### 5.9.1 Detailed Description

template<typename T>class ContainerInterface< T >

class [ContainerInterface](#) -

Definition at line 4 of file ContainerInterface.h.

### 5.9.2 Constructor & Destructor Documentation

5.9.2.1 template<typename T > ContainerInterface< T >::~~ContainerInterface ( )  
[virtual]

Definition at line 10 of file containerInterface.cpp.

```
{  
}
```

5.9.2.2 template<typename T > ContainerInterface< T >::ContainerInterface ( )  
[protected]

Definition at line 4 of file containerInterface.cpp.

```
{  
}
```

### 5.9.3 Member Function Documentation

5.9.3.1 `template<typename T> virtual void ContainerInterface< T >::clear ( )` [pure virtual]

Implemented in [Container< T >](#), [Container< T >](#), [Container< MLPlayer >](#), [Container< MLPlayer >](#), [Container< Con >](#), [Container< Con >](#), [Container< Layer >](#), [Container< Layer >](#), [Container< Neuron >](#), and [Container< Neuron >](#).

5.9.3.2 `template<typename T> virtual boost::shared_ptr< IteratorInterface<T> > ContainerInterface< T >::createIterator ( )` [pure virtual]

Implemented in [Container< T >](#), [Container< MLPlayer >](#), [Container< Con >](#), [Container< Layer >](#), and [Container< Neuron >](#).

5.9.3.3 `template<typename T> virtual bool ContainerInterface< T >::empty ( )` [pure virtual]

Implemented in [Container< T >](#), [Container< T >](#), [Container< MLPlayer >](#), [Container< MLPlayer >](#), [Container< Con >](#), [Container< Con >](#), [Container< Layer >](#), [Container< Layer >](#), [Container< Neuron >](#), and [Container< Neuron >](#).

5.9.3.4 `template<typename T> virtual void ContainerInterface< T >::push_back ( T const & const_reference )` [pure virtual]

Implemented in [Container< T >](#).

5.9.3.5 `template<typename T> virtual void ContainerInterface< T >::reserve ( int n )` [pure virtual]

Implemented in [Container< T >](#), [Container< T >](#), [Container< MLPlayer >](#), [Container< MLPlayer >](#), [Container< Con >](#), [Container< Con >](#), [Container< Layer >](#), [Container< Layer >](#), [Container< Neuron >](#), and [Container< Neuron >](#).

5.9.3.6 `template<typename T> virtual void ContainerInterface< T >::show ( )` [pure virtual]

Implemented in [Container< T >](#), [Container< T >](#), [Container< MLPlayer >](#), [Container< MLPlayer >](#), [Container< Con >](#), [Container< Con >](#), [Container< Layer >](#), [Container< Layer >](#), [Container< Neuron >](#), and [Container< Neuron >](#).

5.9.3.7 `template<typename T> virtual size_type ContainerInterface< T >::size ( )`  
`[pure virtual]`

Implemented in [Container< T >](#), [Container< T >](#), [Container< MLPlayer >](#), [Container< MLPlayer >](#), [Container< Con >](#), [Container< Con >](#), [Container< Layer >](#), [Container< Layer >](#), [Container< Neuron >](#), and [Container< Neuron >](#).

5.9.3.8 `template<typename T> virtual bool ContainerInterface< T >::validate ( )`  
`[pure virtual]`

Implemented in [Container< T >](#), [ConContainer](#), [Container< T >](#), [Container< MLPlayer >](#), [Container< MLPlayer >](#), [Container< Con >](#), [Container< Con >](#), [Container< Layer >](#), [Container< Layer >](#), [Container< Neuron >](#), and [Container< Neuron >](#).

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

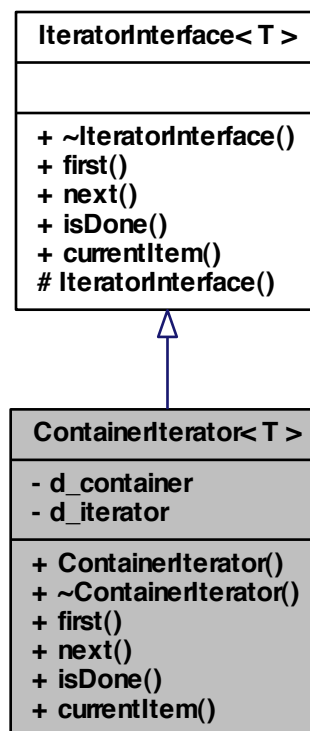
- [pkg/AMORE/src/dia/ContainerInterface.h](#)
- [pkg/AMORE/src/containerInterface.cpp](#)

## 5.10 ContainerIterator< T > Class Template Reference

class [ContainerIterator](#) -

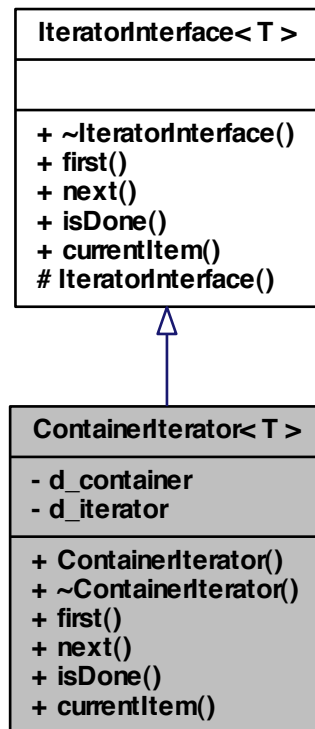
`#include <ContainerIterator.h>`

Inheritance diagram for ContainerIterator< T >:





Collaboration diagram for ContainerIterator< T >:



### Public Member Functions

- [ContainerIterator](#) ()
- [~ContainerIterator](#) ()
- void [first](#) ()
- void [next](#) ()
- bool [isDone](#) ()
- T [currentItem](#) ()

### Private Attributes

- [Container](#)< T > \* [d\\_container](#)
- `std::vector< T >::iterator` [d\\_iterator](#)

## Friends

- class [Container< T >](#)

### 5.10.1 Detailed Description

`template<typename T>class ContainerIterator< T >`

class [ContainerIterator](#) -

Definition at line 6 of file ContainerIterator.h.

### 5.10.2 Constructor & Destructor Documentation

5.10.2.1 `template<typename T > ContainerIterator< T >::ContainerIterator ( )`

Definition at line 4 of file ContainerIterator.cpp.

```
{  
}
```

5.10.2.2 `template<typename T > ContainerIterator< T >::~~ContainerIterator ( )`

Definition at line 9 of file ContainerIterator.cpp.

```
{  
}
```

### 5.10.3 Member Function Documentation

5.10.3.1 `template<typename T > T ContainerIterator< T >::currentItem ( )`  
[virtual]

Implements [IteratorInterface< T >](#).

Definition at line 37 of file ContainerIterator.cpp.

References `ContainerIterator< T >::d_iterator`.

```
{  
    return *d_iterator;  
}
```

**5.10.3.2** `template<typename T> void ContainerIterator< T >::first ( ) [virtual]`

Implements [IteratorInterface< T >](#).

Definition at line 15 of file ContainerIterator.cpp.

References `ContainerIterator< T >::d_container`, and `ContainerIterator< T >::d_iterator`.

```
{
    d_iterator = d_container->collection.begin();
}
```

**5.10.3.3** `template<typename T> bool ContainerIterator< T >::isDone ( ) [virtual]`

Implements [IteratorInterface< T >](#).

Definition at line 29 of file ContainerIterator.cpp.

References `ContainerIterator< T >::d_container`, and `ContainerIterator< T >::d_iterator`.

```
{
    bool IteratorIsDone(d_iterator == d_container->collection.end());
    return IteratorIsDone;
}
```

**5.10.3.4** `template<typename T> void ContainerIterator< T >::next ( ) [virtual]`

Implements [IteratorInterface< T >](#).

Definition at line 22 of file ContainerIterator.cpp.

References `ContainerIterator< T >::d_iterator`.

```
{
    ++d_iterator;
}
```

## 5.10.4 Friends And Related Function Documentation

**5.10.4.1** `template<typename T> friend class Container< T > [friend]`

Definition at line 13 of file ContainerIterator.h.

## 5.10.5 Member Data Documentation

**5.10.5.1** `template<typename T> Container<T>* ContainerIterator< T >::d_container [private]`

Definition at line 9 of file ContainerIterator.h.

Referenced by `ContainerIterator< T >::first()`, and `ContainerIterator< T >::isDone()`.

**5.10.5.2** `template<typename T> std::vector<T>::iterator ContainerIterator< T >::d_iterator [private]`

Definition at line 10 of file `ContainerIterator.h`.

Referenced by `ContainerIterator< T >::currentItem()`, `ContainerIterator< T >::first()`, `ContainerIterator< T >::isDone()`, and `ContainerIterator< T >::next()`.

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

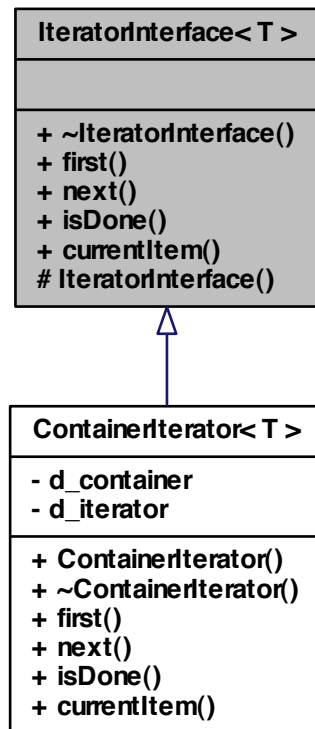
- `pkg/AMORE/src/dia/ContainerIterator.h`
- `pkg/AMORE/src/ContainerIterator.cpp`

## 5.11 `IteratorInterface< T >` Class Template Reference

class `IteratorInterface` -

```
#include <IteratorInterface.h>
```

Inheritance diagram for IteratorInterface< T >:



## Public Member Functions

- virtual `~IteratorInterface()`
- virtual void `first()`=0
- virtual void `next()`=0
- virtual bool `isDone()`=0
- virtual T `currentItem()`=0

## Protected Member Functions

- `IteratorInterface()`

### 5.11.1 Detailed Description

```
template<typename T>class IteratorInterface< T >
```

class [IteratorInterface](#) -

Definition at line 4 of file IteratorInterface.h.

### 5.11.2 Constructor & Destructor Documentation

5.11.2.1 `template<typename T > IteratorInterface< T >::~IteratorInterface ( )`  
[virtual]

Definition at line 10 of file IteratorInterface.cpp.

```
{  
}
```

5.11.2.2 `template<typename T > IteratorInterface< T >::~IteratorInterface ( )`  
[protected]

Definition at line 4 of file IteratorInterface.cpp.

```
{  
}
```

### 5.11.3 Member Function Documentation

5.11.3.1 `template<typename T > virtual T IteratorInterface< T >::currentItem ( )`  
[pure virtual]

Implemented in [ContainerIterator< T >](#).

5.11.3.2 `template<typename T > virtual void IteratorInterface< T >::first ( )` [pure virtual]

Implemented in [ContainerIterator< T >](#).

5.11.3.3 `template<typename T > virtual bool IteratorInterface< T >::isDone ( )` [pure virtual]

Implemented in [ContainerIterator< T >](#).

5.11.3.4 `template<typename T > virtual void IteratorInterface< T >::next ( )` [pure virtual]

Implemented in [ContainerIterator< T >](#).

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

- pkg/AMORE/src/dia/[IteratorInterface.h](#)

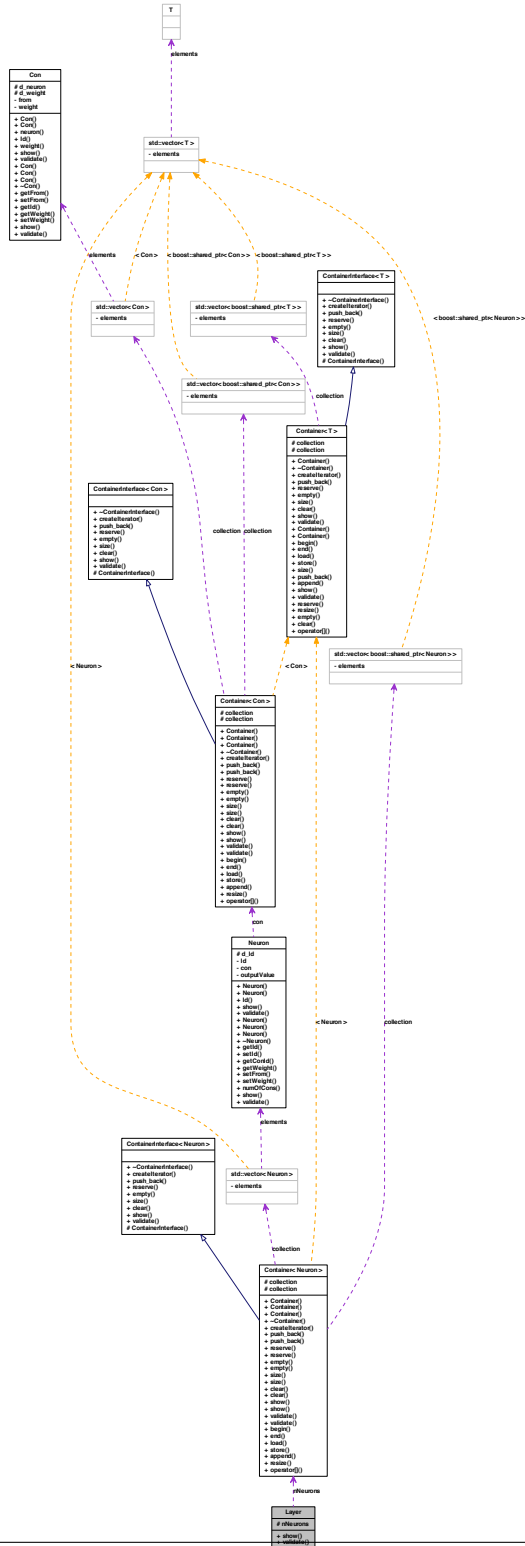
- pkg/AMORE/src/[IteratorInterface.cpp](#)

## 5.12 Layer Class Reference

class [Layer](#) -

```
#include <Layer.h>
```

Collaboration diagram for Layer:





## Public Member Functions

- void [show](#) ()
- bool [validate](#) ()

## Protected Attributes

- [Container](#)< [Neuron](#) > [nNeurons](#)

### 5.12.1 Detailed Description

class [Layer](#) -

Definition at line 3 of file [Layer.h](#).

### 5.12.2 Member Function Documentation

5.12.2.1 void [Layer::show](#) ( )

5.12.2.2 bool [Layer::validate](#) ( )

### 5.12.3 Member Data Documentation

5.12.3.1 [Container](#)<[Neuron](#)> [Layer::nNeurons](#) [protected]

Definition at line 6 of file [Layer.h](#).

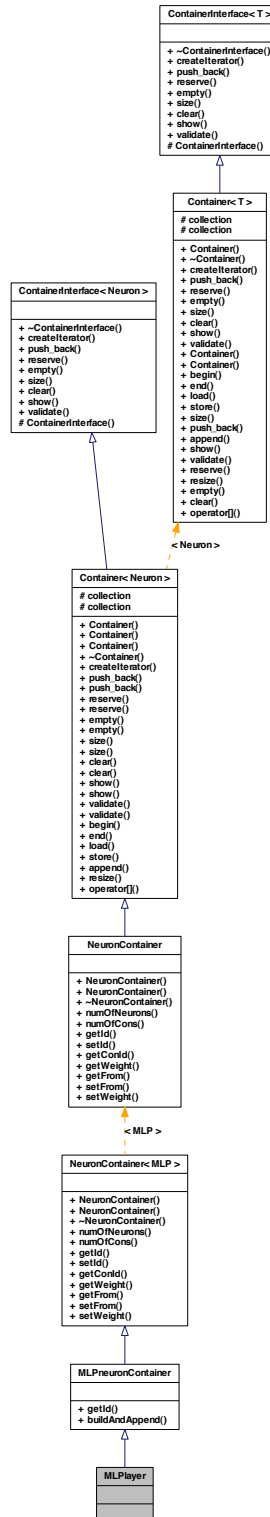
The documentation for this class was generated from the following file:

- [pkg/AMORE/src/dia/Layer.h](#)

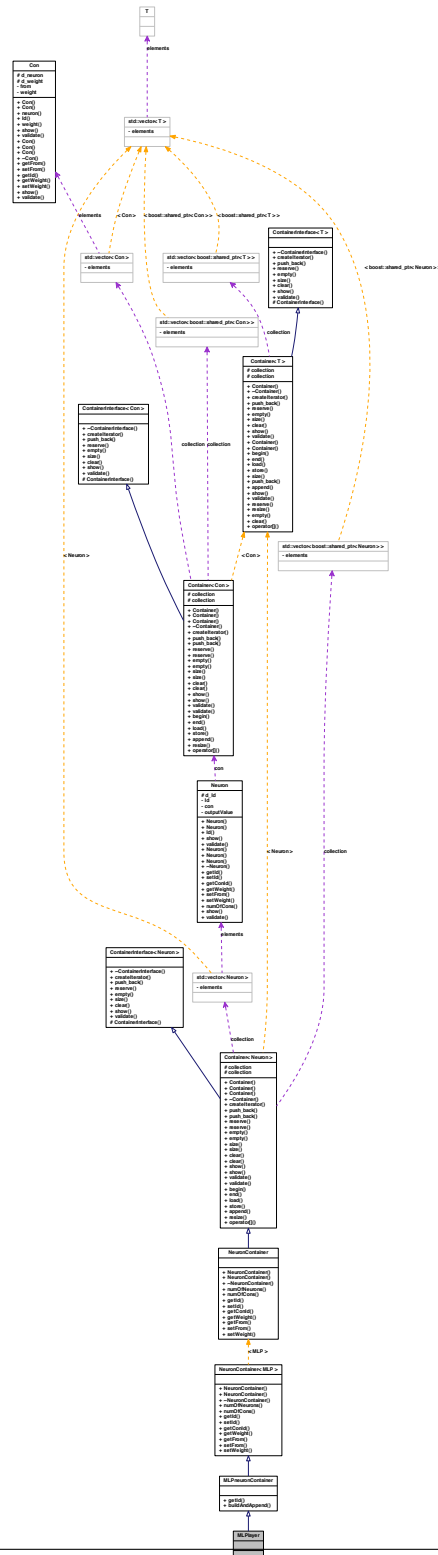
## 5.13 MLPlayer Class Reference

```
#include <MLPlayer.h>
```

Inheritance diagram for MLPPlayer:



Generated on Mon Jun 20 2011 15:56:10 for AMORE++ by Doxygen



### 5.13.1 Detailed Description

Definition at line 1 of file MLPlayer.h.

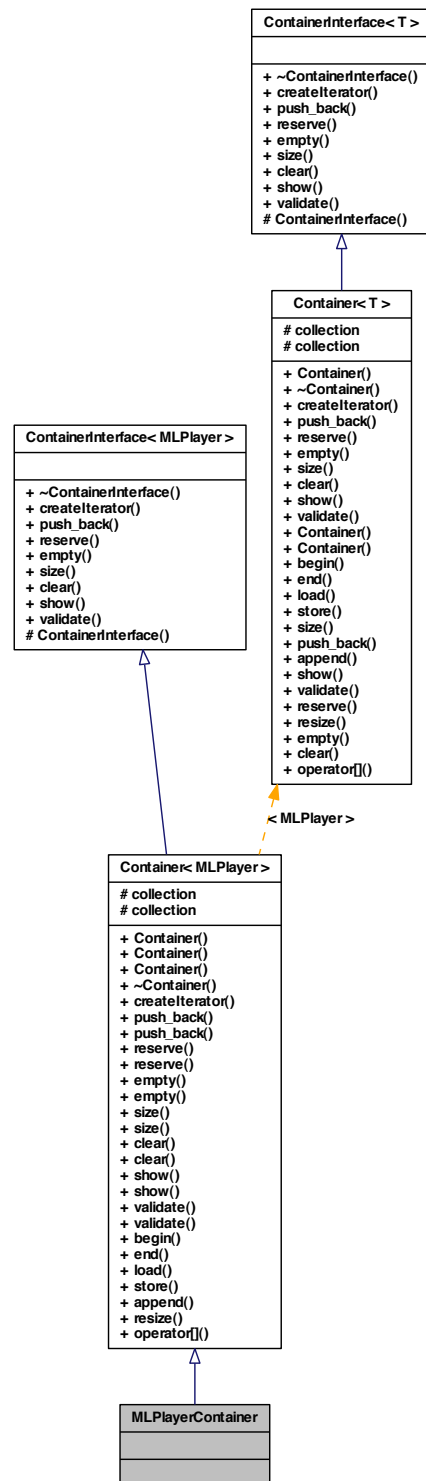
The documentation for this class was generated from the following file:

- pkg/AMORE/src/old/[MLPlayer.h](#)

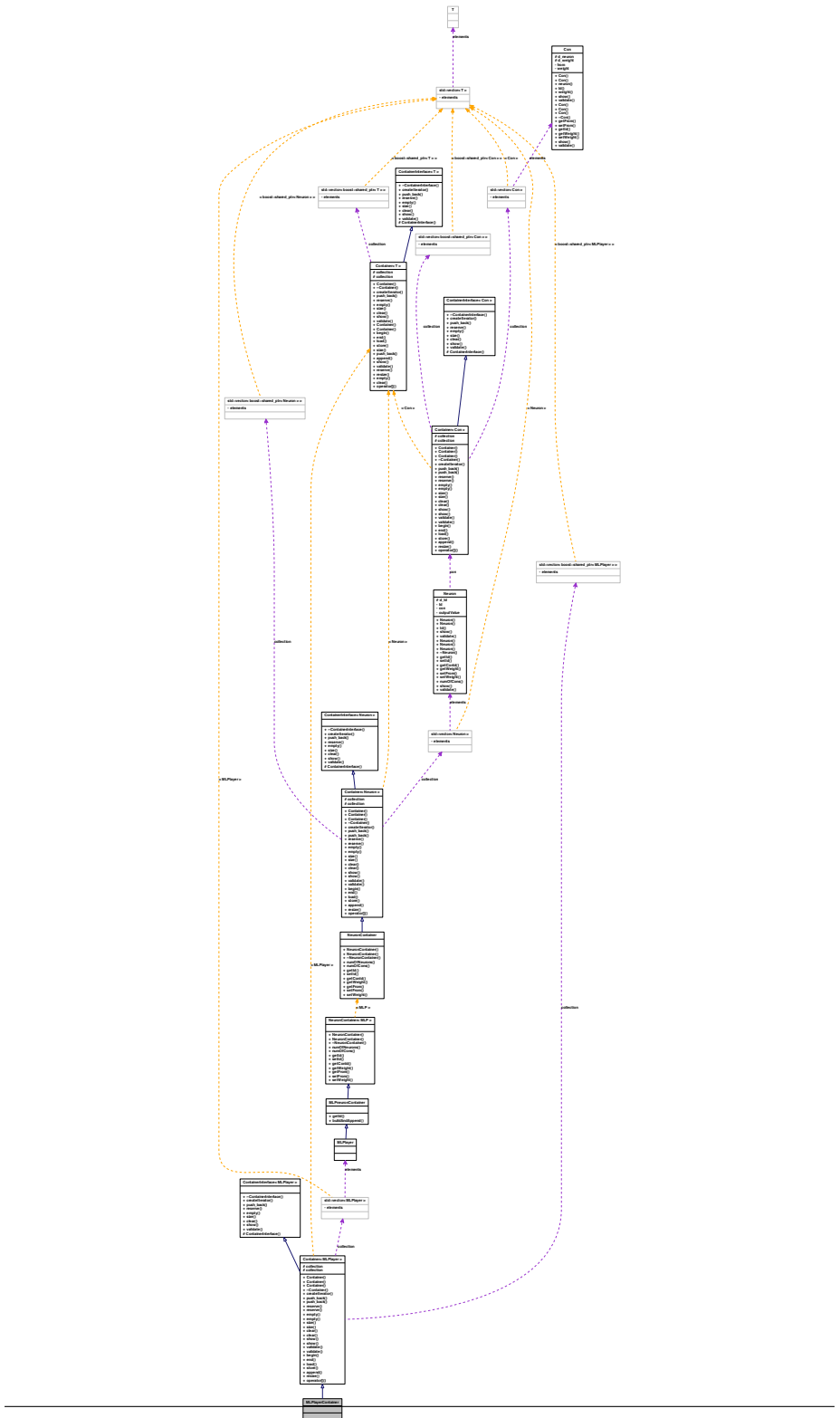
## 5.14 MLPlayerContainer Class Reference

```
#include <MLPlayerContainer.h>
```

Inheritance diagram for MLPlayerContainer:



Collaboration diagram for MLPlayerContainer:



### 5.14.1 Detailed Description

Definition at line 1 of file MLPPlayerContainer.h.

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

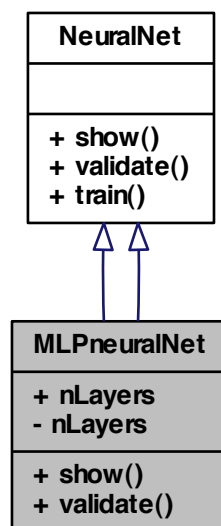
- [pkg/AMORE/src/old/MLPlayerContainer.h](#)

## 5.15 MLPneuralNet Class Reference

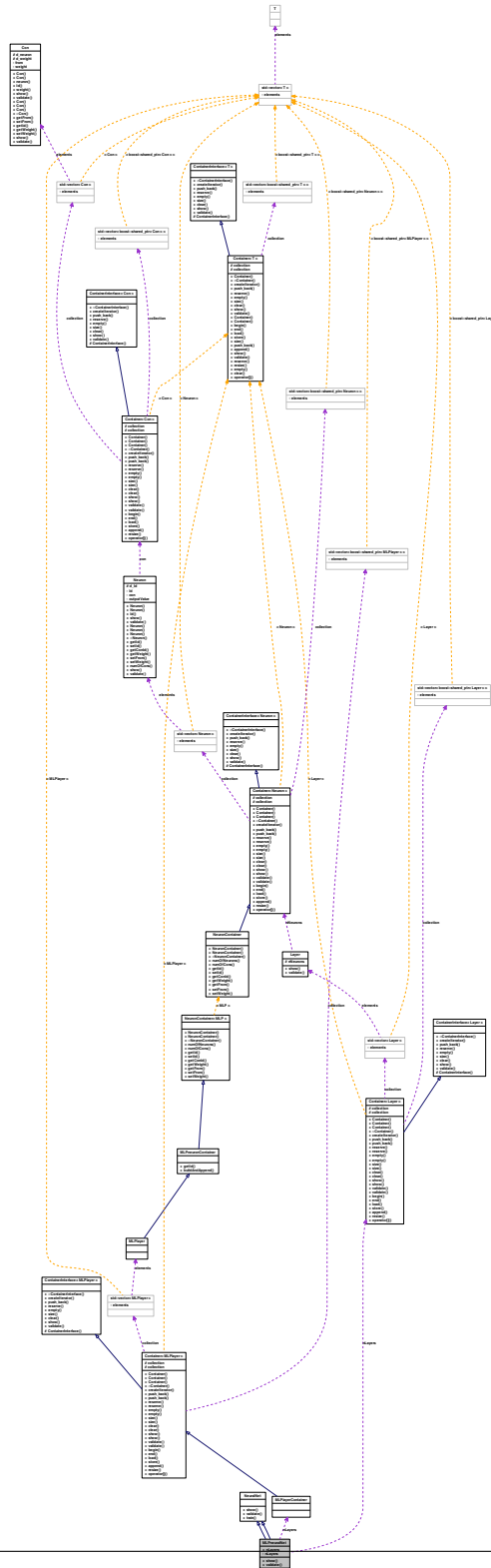
class [MLPneuralNet](#) -

```
#include <MLPneuralNet.h>
```

Inheritance diagram for MLPneuralNet:



Collaboration diagram for MLPneuralNet:





## Public Member Functions

- void [show](#) ()
- bool [validate](#) ()

## Public Attributes

- [Container](#)< [Layer](#) > [nLayers](#)

## Private Attributes

- [MLPlayerContainer](#) [nLayers](#)

### 5.15.1 Detailed Description

class [MLPneuralNet](#) -

Definition at line 5 of file MLPneuralNet.h.

### 5.15.2 Member Function Documentation

#### 5.15.2.1 void MLPneuralNet::show ( )

Reimplemented from [NeuralNet](#).

#### 5.15.2.2 bool MLPneuralNet::validate ( )

Reimplemented from [NeuralNet](#).

### 5.15.3 Member Data Documentation

#### 5.15.3.1 [Container](#)<[Layer](#)> MLPneuralNet::nLayers

Definition at line 8 of file MLPneuralNet.h.

#### 5.15.3.2 [MLPlayerContainer](#) MLPneuralNet::nLayers [\[private\]](#)

Definition at line 2 of file MLPneuralNet.h.

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

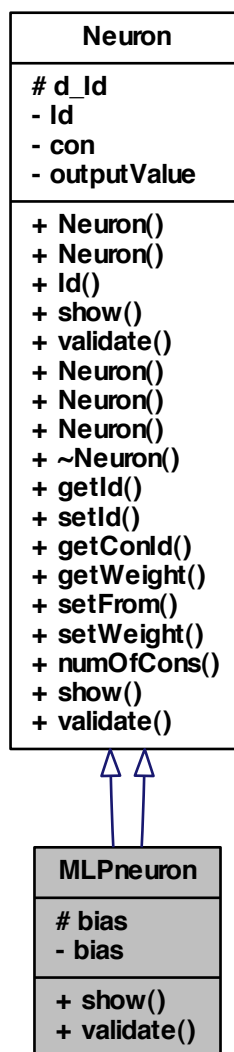
- pkg/AMORE/src/dia/[MLPneuralNet.h](#)
- pkg/AMORE/src/old/[MLPneuralNet.h](#)

## 5.16 MLPneuron Class Reference

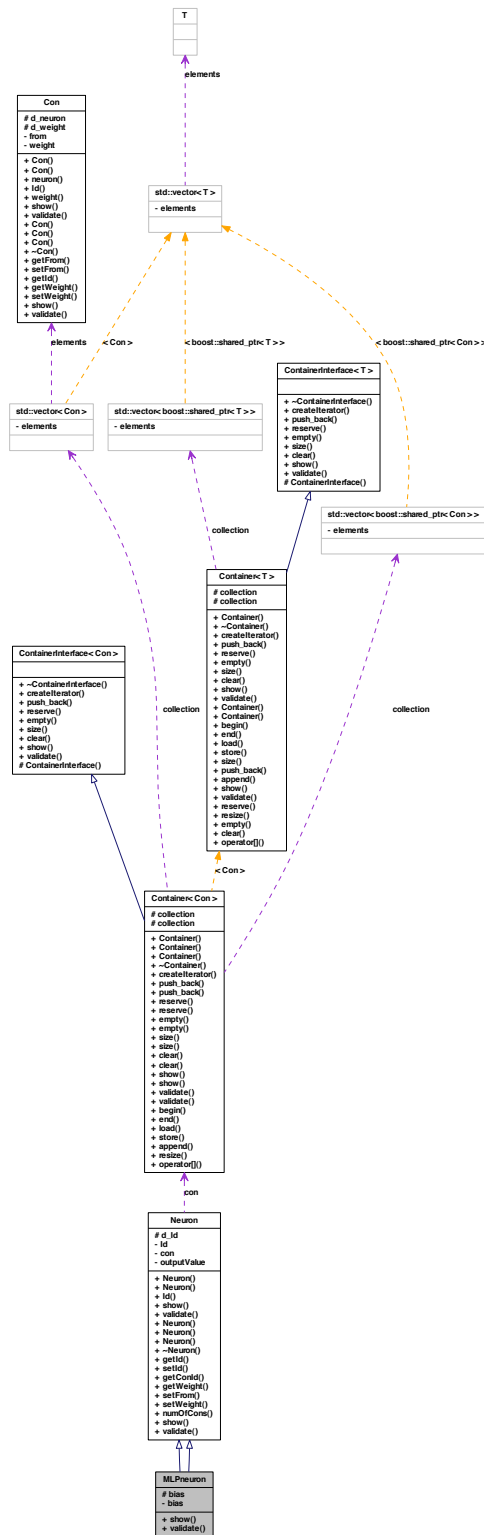
class [MLPneuron](#) -

```
#include <MLPneuron.h>
```

Inheritance diagram for MLPneuron:



Collaboration diagram for MLPneuron:



### Public Member Functions

- void [show](#) ()
- bool [validate](#) ()

### Protected Attributes

- double [bias](#)

### Private Attributes

- int [bias](#)

## 5.16.1 Detailed Description

class [MLPneuron](#) -

Definition at line 5 of file MLPneuron.h.

## 5.16.2 Member Function Documentation

### 5.16.2.1 void MLPneuron::show ( )

Reimplemented from [Neuron](#).

### 5.16.2.2 bool MLPneuron::validate ( )

Reimplemented from [Neuron](#).

## 5.16.3 Member Data Documentation

### 5.16.3.1 double MLPneuron::bias [protected]

Definition at line 8 of file MLPneuron.h.

### 5.16.3.2 int MLPneuron::bias [private]

Definition at line 2 of file MLPneuron.h.

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

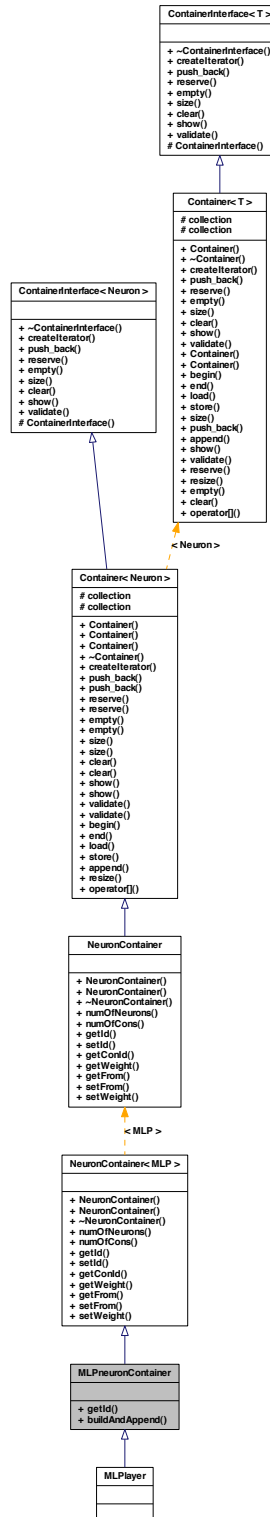
- pkg/AMORE/src/dia/[MLPneuron.h](#)
- pkg/AMORE/src/old/[MLPneuron.h](#)

## 5.17 MLPneuronContainer Class Reference

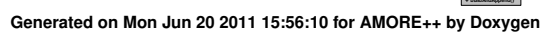
A vector of connections.

```
#include <MLPneuronContainer.h>
```

Inheritance diagram for MLPNeuronContainer:



Collaboration diagram for MLPneuronContainer:



## Public Member Functions

- `std::vector< int > getId ()`
- `bool buildAndAppend (std::vector< int > IDS, std::vector< int > BIAS, ConContainer VC)`

### 5.17.1 Detailed Description

A vector of connections.

The [ConContainer](#) class provides a simple class for a vector of connections. It's named after the R equivalent Reference Class.

Definition at line 16 of file `MLPneuronContainer.h`.

### 5.17.2 Member Function Documentation

**5.17.2.1** `bool MLPneuronContainer::buildAndAppend ( std::vector< int > IDS, std::vector< int > BIAS, ConContainer VC )`

**5.17.2.2** `std::vector<int> MLPneuronContainer::getId ( )`

Reimplemented from [NeuronContainer< MLP >](#).

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

- `pkg/AMORE/src/old/MLPneuronContainer.h`

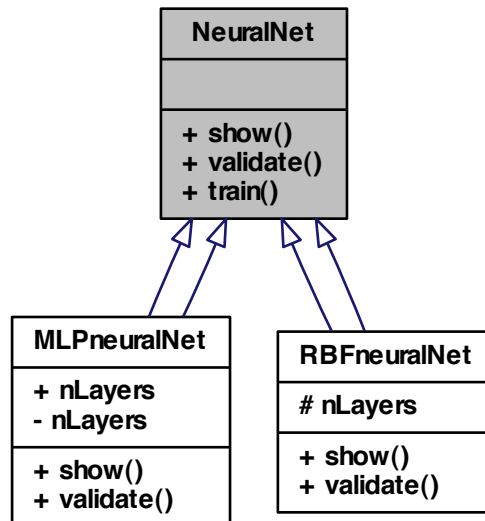
## 5.18 NeuralNet Class Reference

class [NeuralNet](#) -

```
#include <NeuralNet.h>
```



Inheritance diagram for NeuralNet:



### Public Member Functions

- void [show](#) ()
- bool [validate](#) ()
- virtual void [train](#) ()=0

#### 5.18.1 Detailed Description

class [NeuralNet](#) -

Definition at line 3 of file NeuralNet.h.

#### 5.18.2 Member Function Documentation

##### 5.18.2.1 void NeuralNet::show ( )

Reimplemented in [MLPneuralNet](#), and [RBFneuralNet](#).

##### 5.18.2.2 virtual void NeuralNet::train ( ) [pure virtual]

### 5.18.2.3 bool NeuralNet::validate ( )

Reimplemented in [MLPneuralNet](#), and [RBFneuralNet](#).

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

- pkg/AMORE/src/dia/[NeuralNet.h](#)

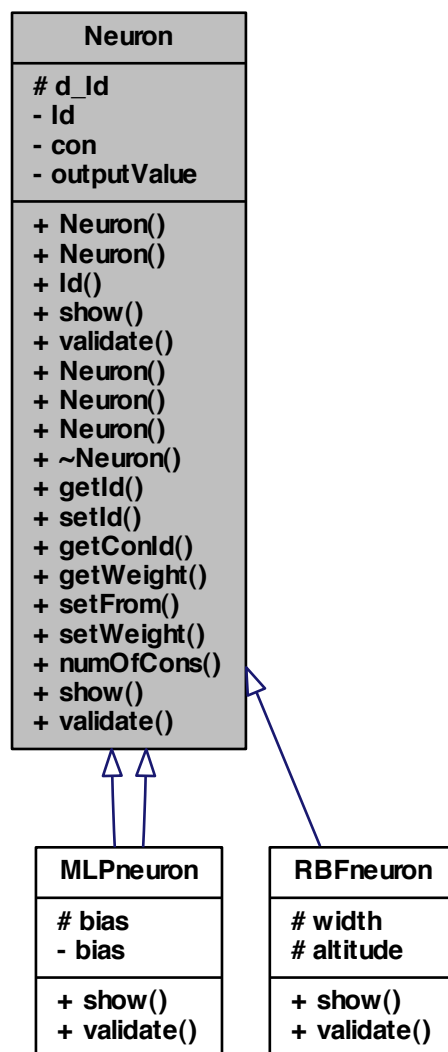
- pkg/AMORE/src/old/[NeuralNet.h](#)

## 5.19 Neuron Class Reference

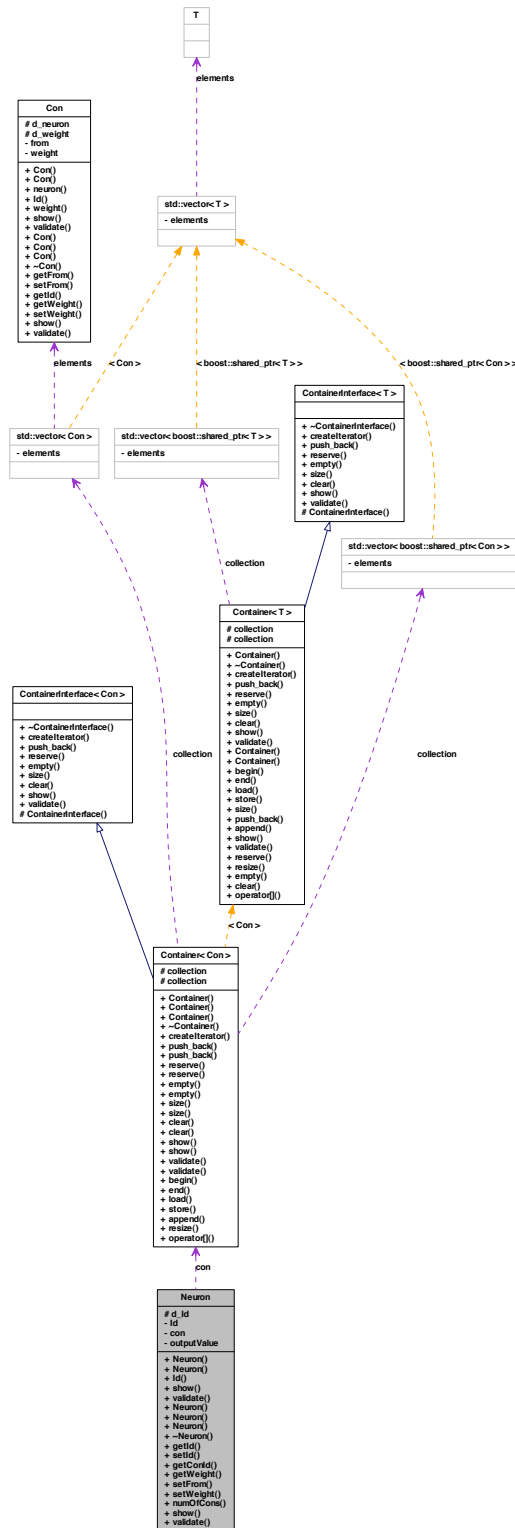
class [Neuron](#) -

```
#include <Neuron.h>
```

Inheritance diagram for Neuron:



Collaboration diagram for Neuron:



### Public Member Functions

- [Neuron](#) ()
- [Neuron](#) (int [ld](#))
- int [ld](#) ()
- void [show](#) ()
- bool [validate](#) ()
- [Neuron](#) ()
- [Neuron](#) (int [ld](#))
- [Neuron](#) (int [ld](#), [ConContainer](#) [con](#))
- [~Neuron](#) ()
- int [getId](#) ()
- void [setId](#) (int value)
- std::vector< int > [getConId](#) ()
- std::vector< double > [getWeight](#) ()
- bool [setFrom](#) ([NeuronContainer](#) [neuronContainer](#))
- bool [setWeight](#) (std::vector< double > [nWeights](#))
- int [numOfCons](#) ()
- bool [show](#) ()
- bool [validate](#) ()

### Protected Attributes

- int [d\\_ld](#)

### Private Attributes

- int [ld](#)  
*An integer variable with the [Neuron](#) ld.*
- [ConContainer](#) [con](#)  
*A vector of input connections.*
- double [outputValue](#)

#### 5.19.1 Detailed Description

class [Neuron](#) -

A class to handle the information contained in a general [Neuron](#).

A general class for neurons. The [MLPneuron](#) and [RBFneuron](#) classes will specialize this general class

Definition at line 3 of file [Neuron.h](#).

## 5.19.2 Constructor & Destructor Documentation

### 5.19.2.1 Neuron::Neuron ( )

Definition at line 10 of file Neuron.cpp.

```
        :  
        d_Id (NA_INTEGER) //, nCons ()  
    {  
    }
```

### 5.19.2.2 Neuron::Neuron ( int Id )

Definition at line 15 of file Neuron.cpp.

```
        :  
        d_Id (Id) //, nCons ()  
    {  
    }
```

### 5.19.2.3 Neuron::Neuron ( )

### 5.19.2.4 Neuron::Neuron ( int Id )

### 5.19.2.5 Neuron::Neuron ( int Id, ConContainer con )

### 5.19.2.6 Neuron::~Neuron ( )

## 5.19.3 Member Function Documentation

### 5.19.3.1 std::vector<int> Neuron::getConId ( )

### 5.19.3.2 int Neuron::getId ( )

Definition at line 26 of file Neuron.cpp.

References `Id()`.

```
{  
    return Id;  
}
```

Here is the call graph for this function:

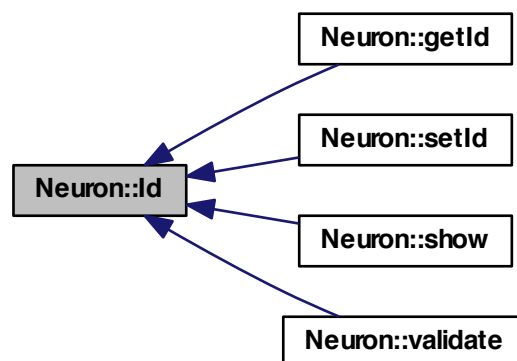


5.19.3.3 `std::vector<double> Neuron::getWeight ( )`

5.19.3.4 `int Neuron::Id ( )`

Referenced by `getId()`, `setId()`, `show()`, and `validate()`.

Here is the caller graph for this function:



5.19.3.5 `int Neuron::numOfCons ( )`

5.19.3.6 `bool Neuron::setFrom ( NeuronContainer neuronContainer )`

5.19.3.7 `void Neuron::setId ( int value )`

Definition at line 32 of file `Neuron.cpp`.

References `Id()`.

```
{
    Id = value;
}
```

Here is the call graph for this function:



**5.19.3.8** `bool Neuron::setWeight ( std::vector< double > nWeights )`

**5.19.3.9** `bool Neuron::show ( )`

Reimplemented in [MLPneuron](#), and [RBFneuron](#).

Definition at line 54 of file `Neuron.cpp`.

References `Id()`.

```
{
    int id = Id();
    Rprintf("\n-----\n");
    if (id == NA_INTEGER)
    {
        Rprintf("\n Id: NA, Invalid neuron Id");
    }
    else
    {
        Rprintf("\n Id: %d", id);
    }
    Rprintf("\n-----\n");
    #if 0

        if (nCons.size() == 0)
        {
            Rprintf("\n No connections defined");
        }
        else
        {
            nCons.show();
        }
        Rprintf("\n-----\n");
    #endif
}
```



Here is the call graph for this function:



#### 5.19.3.10 bool Neuron::show ( )

Reimplemented in [MLPNeuron](#), and [RBFNeuron](#).

#### 5.19.3.11 bool Neuron::validate ( )

Reimplemented in [MLPNeuron](#), and [RBFNeuron](#).

#### 5.19.3.12 bool Neuron::validate ( )

Reimplemented in [MLPNeuron](#), and [RBFNeuron](#).

Definition at line 82 of file Neuron.cpp.

References [Id\(\)](#).

```
{  
    BEGIN_RCPP  
    if (Id() == NA_INTEGER ) throw std::range_error("[C++ Neuron::validate]: Error,  
        Id is NA.");  
    // nCons.validate();  
    return (TRUE);  
END_RCPP}
```

Here is the call graph for this function:



### 5.19.4 Member Data Documentation

#### 5.19.4.1 ConContainer Neuron::con [private]

A vector of input connections.

Definition at line 29 of file Neuron.h.

#### 5.19.4.2 int Neuron::d\_Id [protected]

Definition at line 6 of file Neuron.h.

#### 5.19.4.3 int Neuron::ld [private]

An integer variable with the [Neuron](#) Id.

The [Neuron](#) Id provides a name to the neuron. This value is not expected to be used neither during simulation nor training but it provides an easy reference for human readers.

Definition at line 22 of file Neuron.h.

#### 5.19.4.4 double Neuron::outputValue [private]

Definition at line 30 of file Neuron.h.

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

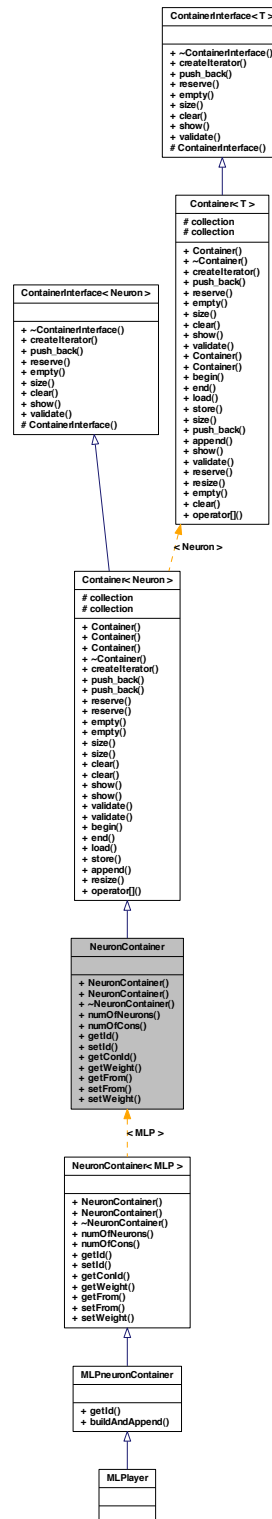
- pkg/AMORE/src/dia/[Neuron.h](#)
- pkg/AMORE/src/old/[Neuron.h](#)
- pkg/AMORE/src/[Neuron.cpp](#)
- pkg/AMORE/src/old/[Neuron.cpp](#)

## 5.20 NeuronContainer Class Reference

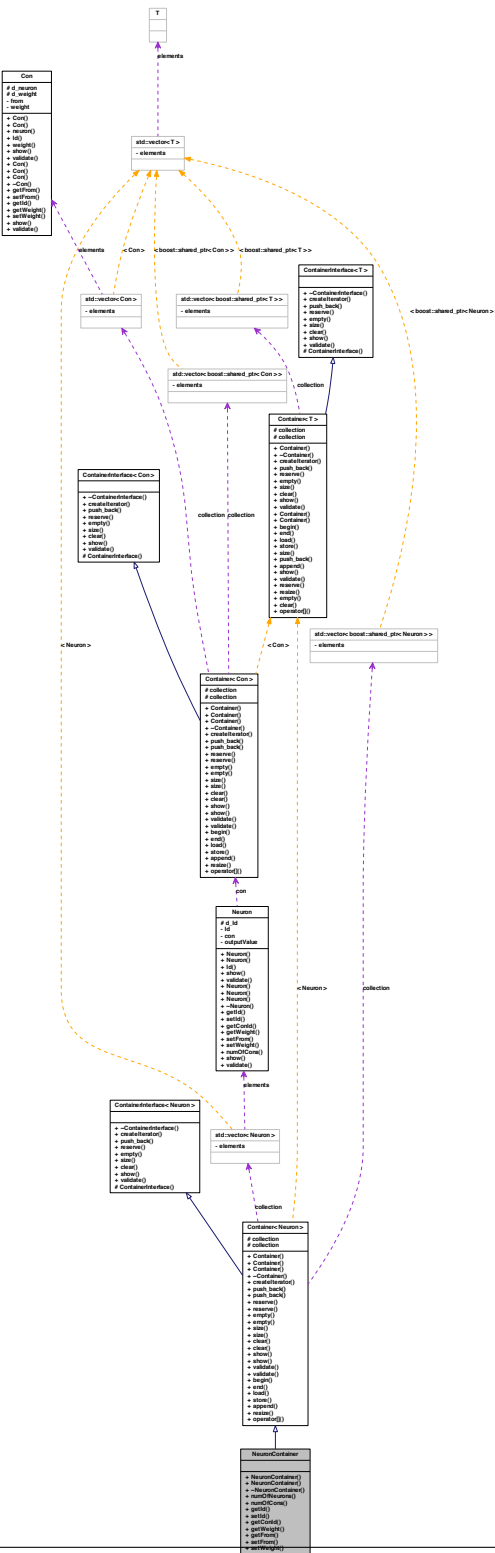
A vector of neurons.

```
#include <NeuronContainer.h>
```

Inheritance diagram for NeuronContainer:



Collaboration diagram for NeuronContainer:



## Public Types

- typedef NeuronContainer\_iterator [iterator](#)
- typedef NeuronContainer\_const\_iterator [const\\_iterator](#)
- typedef boost::shared\_ptr< [Neuron](#) > [value\\_type](#)
- typedef [value\\_type](#) const & [const\\_reference](#)

## Public Member Functions

- [NeuronContainer](#) ()
- [NeuronContainer](#) (std::vector< [NeuronPtr](#) > neuronContainer)
- [~NeuronContainer](#) ()
- int [numOfNeurons](#) ()
- std::vector< int > [numOfCons](#) ()
- std::vector< int > [getId](#) ()
- void [setId](#) (std::vector< int > nlds)
- std::vector< std::vector< int > > [getConId](#) ()
- std::vector< std::vector< double > > [getWeight](#) ()
- std::vector< [NeuronContainer](#) > [getFrom](#) ()
- void [setFrom](#) (std::vector< [NeuronContainer](#) > neuronArray)
- void [setWeight](#) (std::vector< std::vector< double > > value)

### 5.20.1 Detailed Description

A vector of neurons.

The vecNeuron class provides a simple class for a vector of neurons. It's named after the R equivalent Reference Class.

Definition at line 17 of file NeuronContainer.h.

### 5.20.2 Member Typedef Documentation

#### 5.20.2.1 typedef NeuronContainer\_const\_iterator NeuronContainer::const\_iterator

Reimplemented from [Container< Neuron >](#).

Definition at line 23 of file NeuronContainer.h.

#### 5.20.2.2 typedef value\_type const& NeuronContainer::const\_reference

Reimplemented from [Container< Neuron >](#).

Definition at line 27 of file NeuronContainer.h.

### 5.20.2.3 `typedef NeuronContainer_iterator NeuronContainer::iterator`

Reimplemented from [Container< Neuron >](#).

Definition at line 21 of file NeuronContainer.h.

### 5.20.2.4 `typedef boost::shared_ptr<Neuron> NeuronContainer::value_type`

Reimplemented from [Container< Neuron >](#).

Definition at line 25 of file NeuronContainer.h.

## 5.20.3 Constructor & Destructor Documentation

### 5.20.3.1 `NeuronContainer::NeuronContainer ( )`

Definition at line 8 of file NeuronContainer.cpp.

```
{
}
```

### 5.20.3.2 `NeuronContainer::NeuronContainer ( std::vector< NeuronPtr > neuronContainer )`

Definition at line 12 of file NeuronContainer.cpp.

```
Container<Neuron> (collection)
{
}
:
```

### 5.20.3.3 `NeuronContainer::~~NeuronContainer ( )`

Definition at line 17 of file NeuronContainer.cpp.

```
{
}
```

## 5.20.4 Member Function Documentation

### 5.20.4.1 `std::vector< std::vector< int > > NeuronContainer::getConld ( )`

Definition at line 60 of file NeuronContainer.cpp.

```
{
    std::vector < std::vector<int> > result;
    foreach(NeuronPtr itrNeuron, *this)
```

```
{
    result.push_back( itrNeuron->getConId() );
}
return result;
}
```

#### 5.20.4.2 `std::vector<NeuronContainer> NeuronContainer::getFrom ( )`

#### 5.20.4.3 `std::vector<int> NeuronContainer::getId ( )`

Reimplemented in [MLPNeuronContainer](#).

Definition at line 39 of file NeuronContainer.cpp.

```
{
    std::vector<int> nIds;
    foreach(NeuronPtr itrNeuron, *this)
    {
        nIds.push_back( itrNeuron->getId() );
    }
    return nIds;
}
```

#### 5.20.4.4 `std::vector<std::vector<double>> NeuronContainer::getWeight ( )`

Definition at line 71 of file NeuronContainer.cpp.

```
{
    std::vector < std::vector<double> > result;
    foreach(NeuronPtr itrNeuron, *this)
    {
        result.push_back( itrNeuron->getWeight() );
    }
    return result;
}
```

#### 5.20.4.5 `std::vector<int> NeuronContainer::numOfCons ( )`

Definition at line 28 of file NeuronContainer.cpp.

```
{
    std::vector<int> nIds;
    foreach(NeuronPtr itrNeuron, *this)
    {
        nIds.push_back( itrNeuron->numOfCons() );
    }
    return nIds;
}
```

#### 5.20.4.6 int NeuronContainer::numOfNeurons ( )

Definition at line 22 of file NeuronContainer.cpp.

References Container< Neuron >::size().

```
{
    size();
}
```

Here is the call graph for this function:



#### 5.20.4.7 void NeuronContainer::setFrom ( std::vector< NeuronContainer > neuronArray )

Definition at line 83 of file NeuronContainer.cpp.

```
{
    std::vector<NeuronContainer>::iterator itrArray(neuronArray.begin());
    foreach(NeuronPtr itrNeuron, *this)
    {
        itrNeuron->setFrom(*itrArray);
        itrArray++;
    }
}
```

#### 5.20.4.8 void NeuronContainer::setId ( std::vector< int > nIds )

Definition at line 50 of file NeuronContainer.cpp.

```
{
    std::vector<int>::iterator itrId(nIds.begin());
    foreach(NeuronPtr itrNeuron, *this)
    {
        itrNeuron->setId(*itrId);
    }
}
```



5.20.4.9 void NeuronContainer::setWeight ( std::vector< std::vector< double > > value )

Definition at line 94 of file NeuronContainer.cpp.

```
{
    std::vector<std::vector<double> >::iterator itrValue(value.begin());
    foreach(NeuronPtr itrNeuron, *this)
    {
        itrNeuron->setWeight(*itrValue);
        itrValue++;
    }
}
```

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

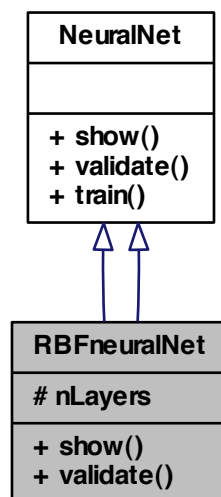
- pkg/AMORE/src/old/[NeuronContainer.h](#)
- pkg/AMORE/src/old/[NeuronContainer.cpp](#)

## 5.21 RBFneuralNet Class Reference

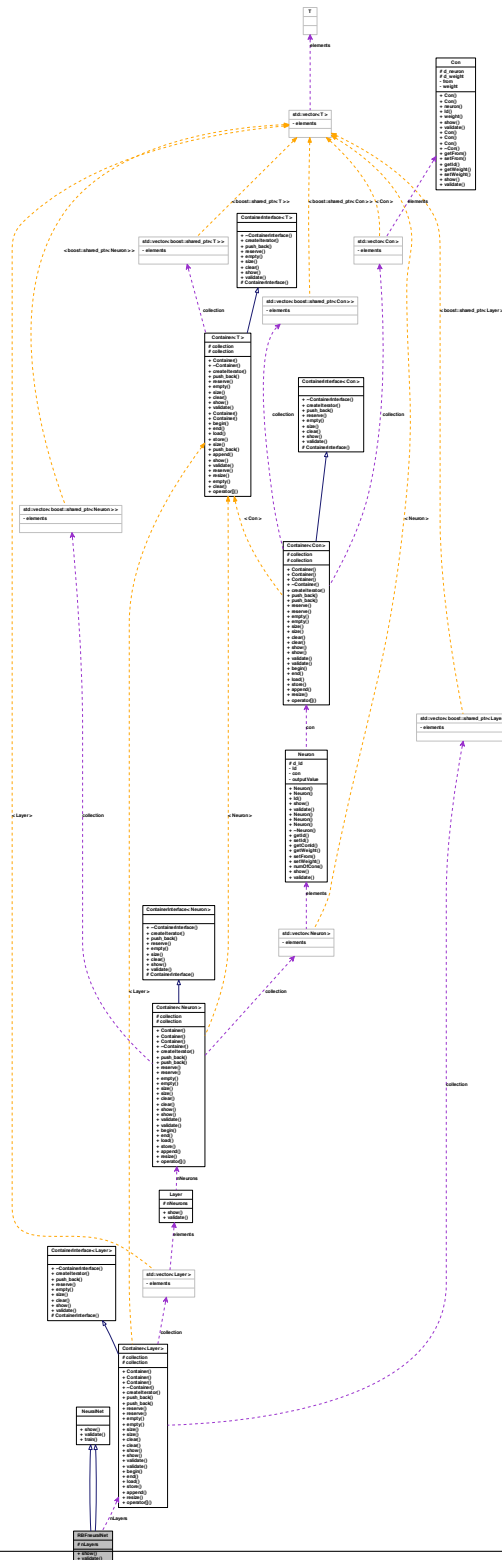
class [RBFneuralNet](#) -

```
#include <RBFneuralNet.h>
```

Inheritance diagram for RBFneuralNet:



Collaboration diagram for RBFneuralNet:



## Public Member Functions

- void [show](#) ()
- bool [validate](#) ()

## Protected Attributes

- [Container](#)< [Layer](#) > [nLayers](#)

### 5.21.1 Detailed Description

class [RBFneuralNet](#) -

Definition at line 5 of file [RBFneuralNet.h](#).

### 5.21.2 Member Function Documentation

5.21.2.1 void [RBFneuralNet::show](#) ( )

Reimplemented from [NeuralNet](#).

5.21.2.2 bool [RBFneuralNet::validate](#) ( )

Reimplemented from [NeuralNet](#).

### 5.21.3 Member Data Documentation

5.21.3.1 [Container](#)<[Layer](#)> [RBFneuralNet::nLayers](#) [protected]

Definition at line 8 of file [RBFneuralNet.h](#).

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

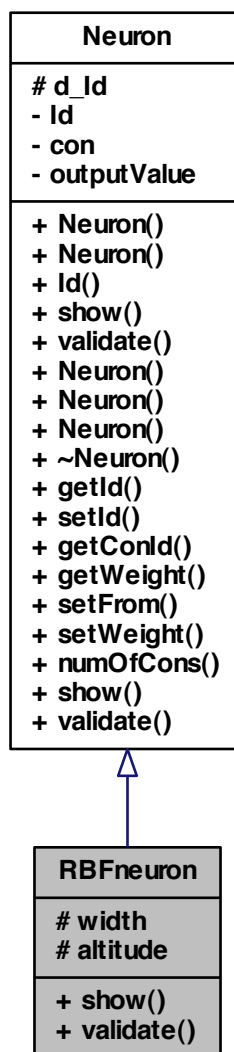
- [pkg/AMORE/src/dia/RBFneuralNet.h](#)

## 5.22 RBFneuron Class Reference

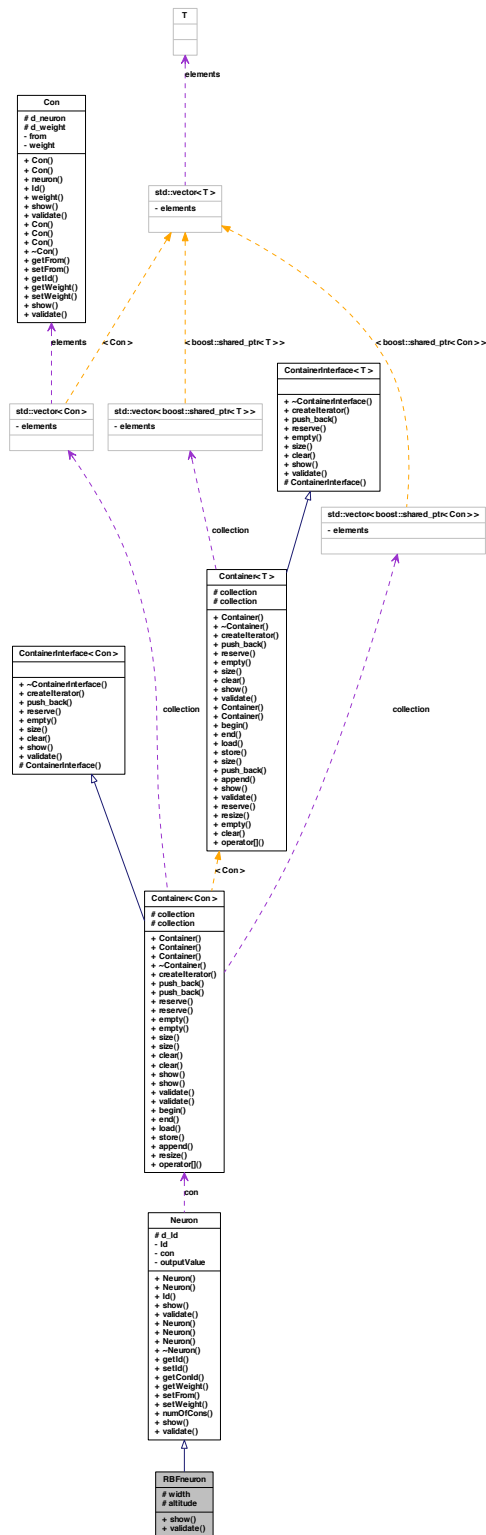
class [RBFneuron](#) -

```
#include <RBFneuron.h>
```

Inheritance diagram for RBFneuron:



Collaboration diagram for RBFneuron:



## Public Member Functions

- void [show](#) ()
- bool [validate](#) ()

## Protected Attributes

- double [width](#)
- double [altitude](#)

### 5.22.1 Detailed Description

class [RBFneuron](#) -

Definition at line 5 of file RBFneuron.h.

### 5.22.2 Member Function Documentation

#### 5.22.2.1 void RBFneuron::show ( )

Reimplemented from [Neuron](#).

#### 5.22.2.2 bool RBFneuron::validate ( )

Reimplemented from [Neuron](#).

### 5.22.3 Member Data Documentation

#### 5.22.3.1 double RBFneuron::altitude [protected]

Definition at line 9 of file RBFneuron.h.

#### 5.22.3.2 double RBFneuron::width [protected]

Definition at line 8 of file RBFneuron.h.

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

- pkg/AMORE/src/dia/[RBFneuron.h](#)

## 5.23 SimulationVariables Class Reference

class [SimulationVariables](#) -

```
#include <SimulationVariables.h>
```

### Protected Attributes

- double [outputValue](#)

#### 5.23.1 Detailed Description

class [SimulationVariables](#) -

Definition at line 3 of file SimulationVariables.h.

#### 5.23.2 Member Data Documentation

5.23.2.1 double **SimulationVariables::outputValue** `[protected]`

Definition at line 6 of file SimulationVariables.h.

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

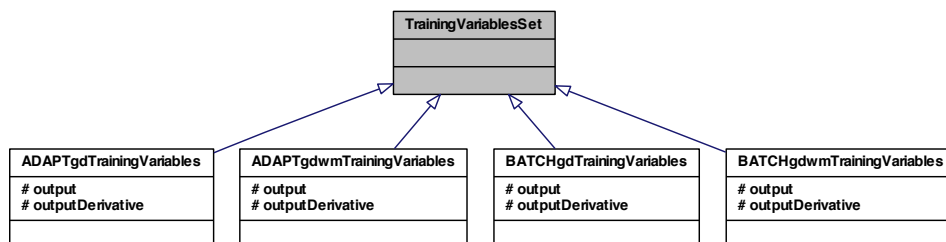
- pkg/AMORE/src/dia/[SimulationVariables.h](#)

## 5.24 TrainingVariablesSet Class Reference

class [TrainingVariablesSet](#) -

```
#include <TrainingVariablesSet.h>
```

Inheritance diagram for TrainingVariablesSet:



#### 5.24.1 Detailed Description

class [TrainingVariablesSet](#) -

Definition at line 3 of file TrainingVariablesSet.h.

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

- pkg/AMORE/src/dia/[TrainingVariablesSet.h](#)



## Chapter 6

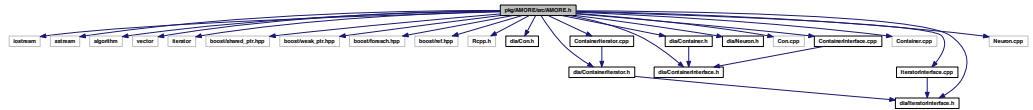
# File Documentation

### 6.1 pkg/AMORE/src/AMORE.h File Reference

```
#include <iostream>
#include <sstream>
#include <algorithm>
#include <vector>
#include <iterator>
#include <boost/shared_ptr.hpp>
#include <boost/weak_ptr.hpp>
#include <boost/foreach.hpp>
#include <boost/ref.hpp>
#include <Rcpp.h>
#include "dia/Con.h"
#include "dia/IteratorInterface.h"
#include "dia/ContainerInterface.h"
#include "dia/Container.h"
#include "dia/ContainerIterator.h"
#include "dia/Neuron.h"
#include "Con.cpp"
#include "ContainerInterface.cpp"
#include "Container.cpp"
#include "IteratorInterface.cpp"
#include "ContainerIterator.cpp"
```

```
#include "Neuron.cpp"
```

Include dependency graph for AMORE.h:



## Defines

- #define [foreach](#) BOOST\_FOREACH
- #define [size\\_type](#) unsigned int

## Typedefs

- typedef boost::reference\_wrapper< [Neuron](#) > [NeuronRef](#)
- typedef boost::shared\_ptr< [Neuron](#) > [NeuronPtr](#)
- typedef boost::shared\_ptr< [Con](#) > [ConPtr](#)
- typedef [Container](#)< [Con](#) > [ConContainer](#)
- typedef [Container](#)< [Neuron](#) > [NeuronContainer](#)

### 6.1.1 Define Documentation

#### 6.1.1.1 #define [foreach](#) BOOST\_FOREACH

Definition at line 53 of file AMORE.h.

#### 6.1.1.2 #define [size\\_type](#) unsigned int

Definition at line 58 of file AMORE.h.

### 6.1.2 Typedef Documentation

#### 6.1.2.1 typedef [Container](#)<[Con](#)> [ConContainer](#)

Definition at line 74 of file AMORE.h.

#### 6.1.2.2 typedef boost::shared\_ptr<[Con](#)> [ConPtr](#)

Definition at line 63 of file AMORE.h.

### 6.1.2.3 typedef Container<Neuron> NeuronContainer

Definition at line 75 of file AMORE.h.

### 6.1.2.4 typedef boost::shared\_ptr<Neuron> NeuronPtr

Definition at line 62 of file AMORE.h.

### 6.1.2.5 typedef boost::reference\_wrapper<Neuron> NeuronRef

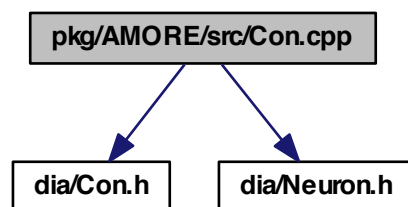
Definition at line 61 of file AMORE.h.

## 6.2 pkg/AMORE/src/Con.cpp File Reference

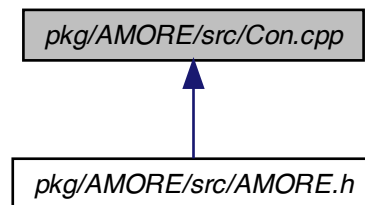
```
#include "dia/Con.h"
```

```
#include "dia/Neuron.h"
```

Include dependency graph for Con.cpp:



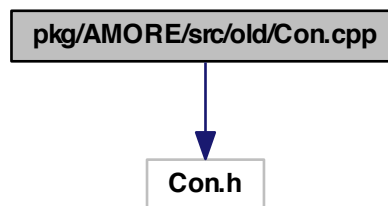
This graph shows which files directly or indirectly include this file:



### 6.3 pkg/AMORE/src/old/Con.cpp File Reference

```
#include "Con.h"
```

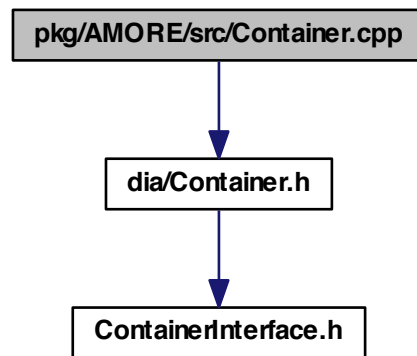
Include dependency graph for Con.cpp:



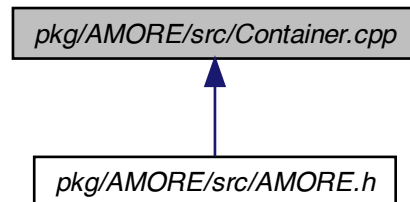
### 6.4 pkg/AMORE/src/Container.cpp File Reference

```
#include "dia/Container.h"
```

Include dependency graph for Container.cpp:



This graph shows which files directly or indirectly include this file:

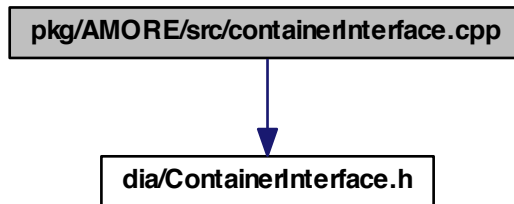


## 6.5 pkg/AMORE/src/old/Container.cpp File Reference

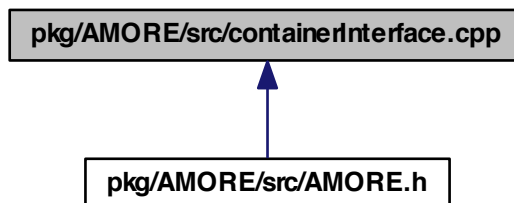
## 6.6 pkg/AMORE/src/containerInterface.cpp File Reference

```
#include "dia/ContainerInterface.h"
```

Include dependency graph for containerInterface.cpp:



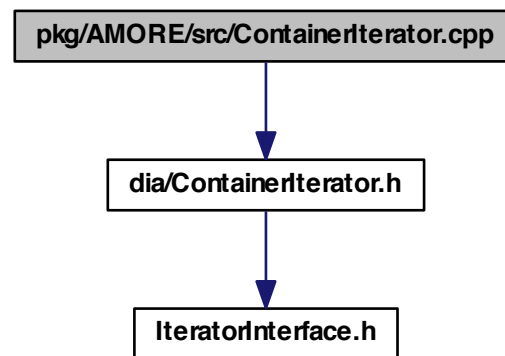
This graph shows which files directly or indirectly include this file:



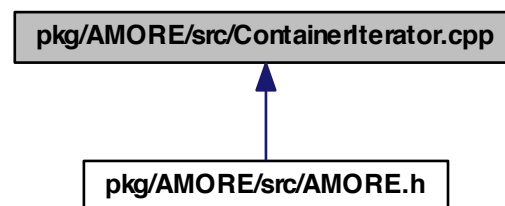
## 6.7 pkg/AMORE/src/ContainerIterator.cpp File Reference

```
#include "dia/ContainerIterator.h"
```

Include dependency graph for ContainerIterator.cpp:



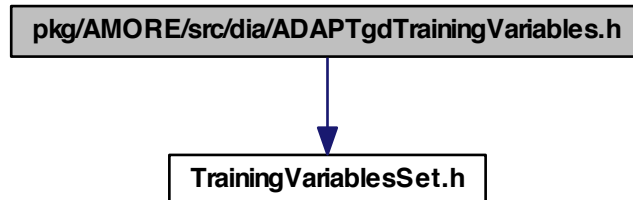
This graph shows which files directly or indirectly include this file:



## 6.8 pkg/AMORE/src/dia/ADAPTgdTrainingVariables.h File Reference

```
#include "TrainingVariablesSet.h"
```

Include dependency graph for ADAPTgdTrainingVariables.h:



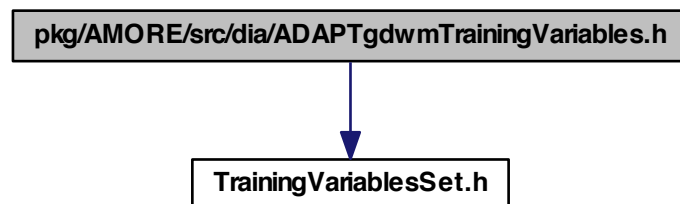
### Classes

- class [ADAPTgdTrainingVariables](#)  
*class ADAPTgdTrainingVariables -*

## 6.9 pkg/AMORE/src/dia/ADAPTgdwmTrainingVariables.h File Reference

```
#include "TrainingVariablesSet.h"
```

Include dependency graph for ADAPTgdwmTrainingVariables.h:



### Classes

- class [ADAPTgdwmTrainingVariables](#)

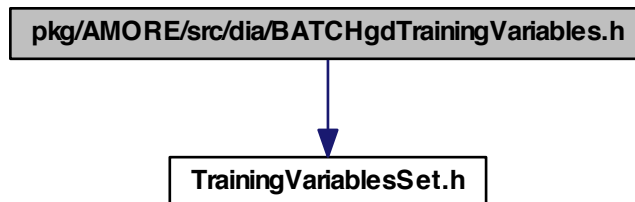


class [ADAPTgdwmTrainingVariables](#) -

## 6.10 pkg/AMORE/src/dia/BATCHgdTrainingVariables.h File Reference

```
#include "TrainingVariablesSet.h"
```

Include dependency graph for BATCHgdTrainingVariables.h:



### Classes

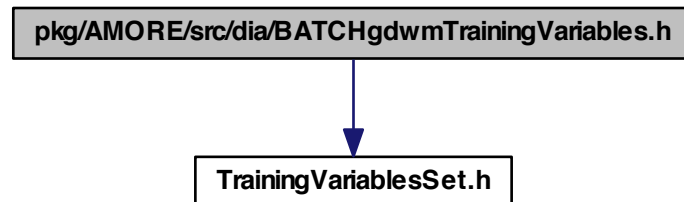
- class [BATCHgdTrainingVariables](#)

class [BATCHgdTrainingVariables](#) -

## 6.11 pkg/AMORE/src/dia/BATCHgdwmTrainingVariables.h File Reference

```
#include "TrainingVariablesSet.h"
```

Include dependency graph for BATCHgdwmTrainingVariables.h:

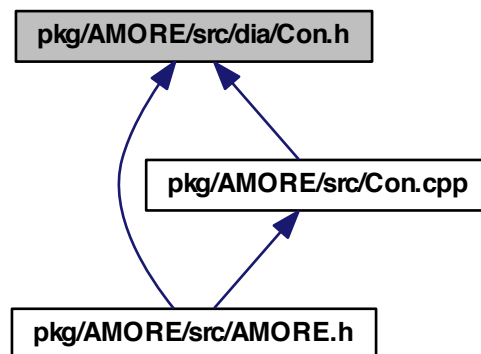


## Classes

- class [BATCHgdwmTrainingVariables](#)  
*class [BATCHgdwmTrainingVariables](#) -*

## 6.12 pkg/AMORE/src/dia/Con.h File Reference

This graph shows which files directly or indirectly include this file:



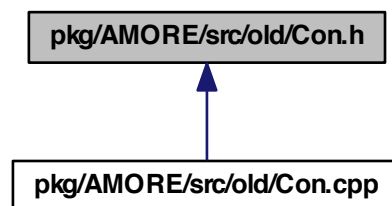
## Classes

- class [Con](#)

*class [Con](#) -*

## 6.13 pkg/AMORE/src/old/Con.h File Reference

This graph shows which files directly or indirectly include this file:



## Classes

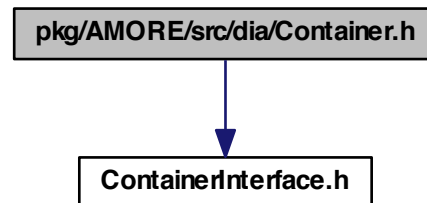
- class [Con](#)

*class [Con](#) -*

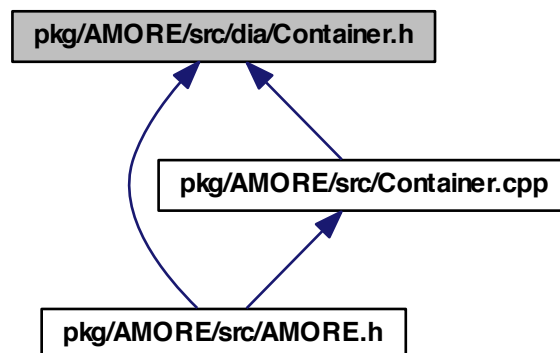
## 6.14 pkg/AMORE/src/dia/Container.h File Reference

```
#include "ContainerInterface.h"
```

Include dependency graph for Container.h:



This graph shows which files directly or indirectly include this file:



## Classes

- class [Container< T >](#)  
    class [Container](#) -

## 6.15 pkg/AMORE/src/old/Container.h File Reference

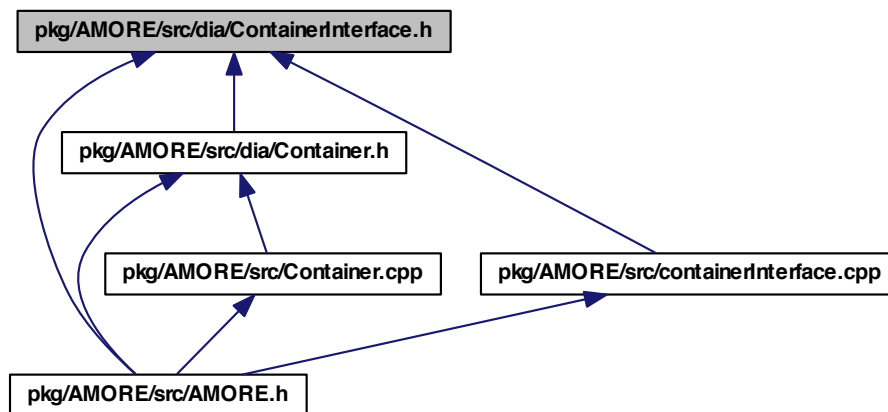
## Classes

- class [Container< T >](#)

class [Container](#) -

## 6.16 pkg/AMORE/src/dia/ContainerInterface.h File Reference

This graph shows which files directly or indirectly include this file:



## Classes

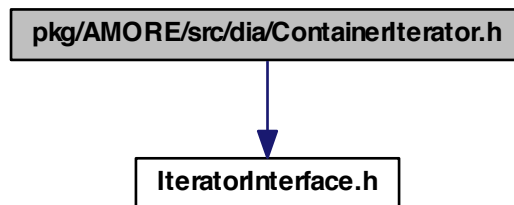
- class [ContainerInterface< T >](#)

class [ContainerInterface](#) -

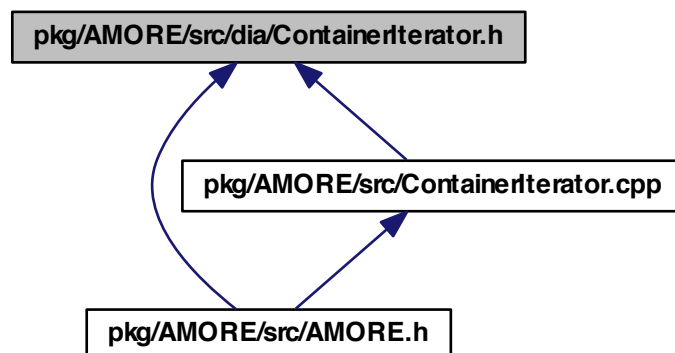
## 6.17 pkg/AMORE/src/dia/ContainerIterator.h File Reference

```
#include "IteratorInterface.h"
```

Include dependency graph for ContainerIterator.h:



This graph shows which files directly or indirectly include this file:

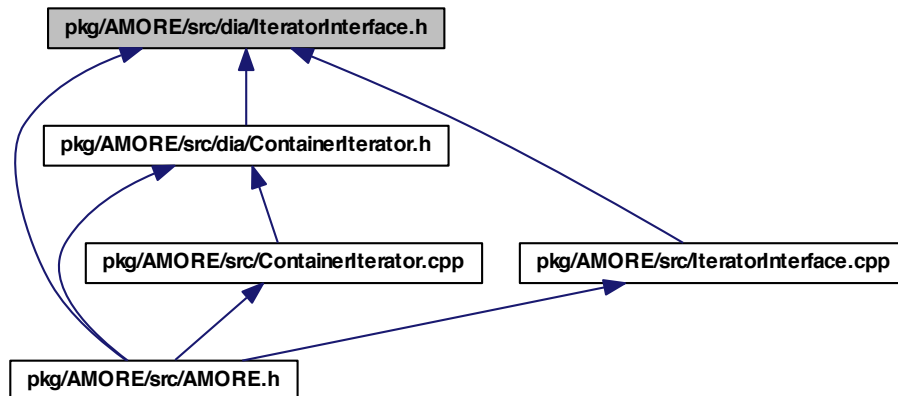


## Classes

- class `ContainerIterator< T >`  
*class `ContainerIterator` -*

## 6.18 pkg/AMORE/src/dia/IteratorInterface.h File Reference

This graph shows which files directly or indirectly include this file:



### Classes

- class [IteratorInterface< T >](#)

*class [IteratorInterface](#) -*

## 6.19 pkg/AMORE/src/dia/Layer.h File Reference

### Classes

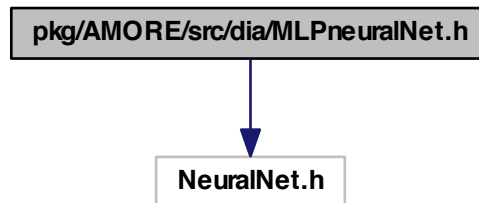
- class [Layer](#)

*class [Layer](#) -*

## 6.20 pkg/AMORE/src/dia/MLPNeuralNet.h File Reference

```
#include "NeuralNet.h"
```

Include dependency graph for MLPneuralNet.h:



## Classes

- class [MLPneuralNet](#)

*class [MLPneuralNet](#) -*

## 6.21 pkg/AMORE/src/old/MLPneuralNet.h File Reference

### Classes

- class [MLPneuralNet](#)

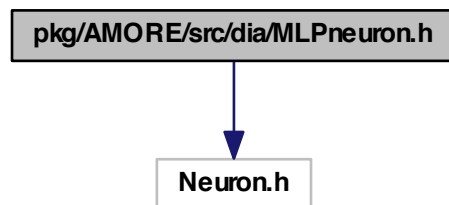
*class [MLPneuralNet](#) -*

## 6.22 pkg/AMORE/src/dia/MLPneuron.h File Reference

```
#include "Neuron.h"
```



Include dependency graph for MLPneuron.h:



## Classes

- class [MLPneuron](#)

*class [MLPneuron](#) -*

## 6.23 pkg/AMORE/src/old/MLPneuron.h File Reference

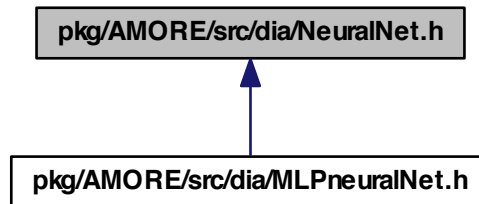
## Classes

- class [MLPneuron](#)

*class [MLPneuron](#) -*

## 6.24 pkg/AMORE/src/dia/NeuralNet.h File Reference

This graph shows which files directly or indirectly include this file:

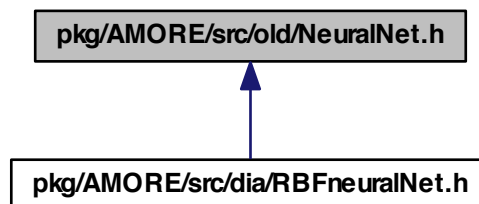


### Classes

- class [NeuralNet](#)  
*class [NeuralNet](#) -*

## 6.25 pkg/AMORE/src/old/NeuralNet.h File Reference

This graph shows which files directly or indirectly include this file:



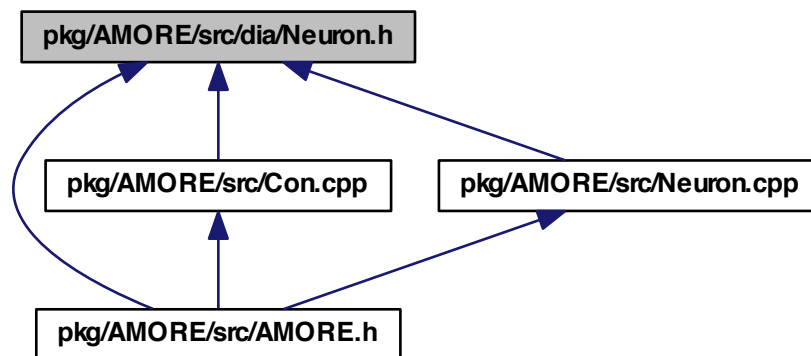
### Classes

- class [NeuralNet](#)

class [NeuralNet](#) -

## 6.26 pkg/AMORE/src/dia/Neuron.h File Reference

This graph shows which files directly or indirectly include this file:

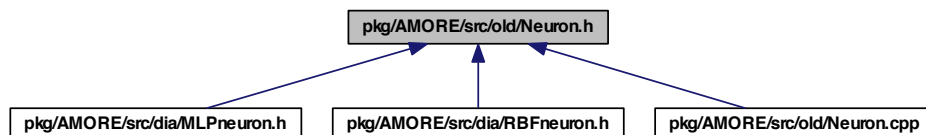


### Classes

- class [Neuron](#)  
class [Neuron](#) -

## 6.27 pkg/AMORE/src/old/Neuron.h File Reference

This graph shows which files directly or indirectly include this file:



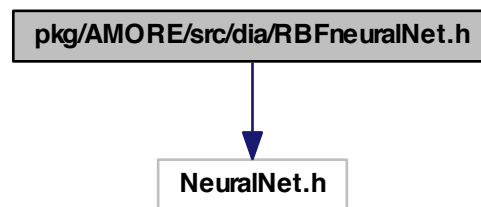
## Classes

- class [Neuron](#)  
*class [Neuron](#) -*

## 6.28 pkg/AMORE/src/dia/RBFneuralNet.h File Reference

```
#include "NeuralNet.h"
```

Include dependency graph for RBFneuralNet.h:



## Classes

- class [RBFneuralNet](#)  
*class [RBFneuralNet](#) -*

## 6.29 pkg/AMORE/src/old/RBFneuralNet.h File Reference

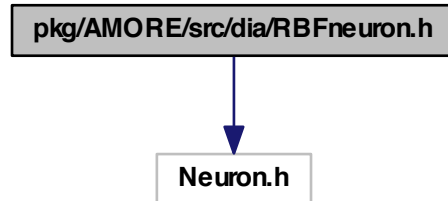
## Classes

- class [RBFneuralNet](#)  
*class [RBFneuralNet](#) -*

## 6.30 pkg/AMORE/src/dia/RBFneuron.h File Reference

```
#include "Neuron.h"
```

Include dependency graph for RBFneuron.h:



#### Classes

- class [RBFneuron](#)  
*class [RBFneuron](#) -*

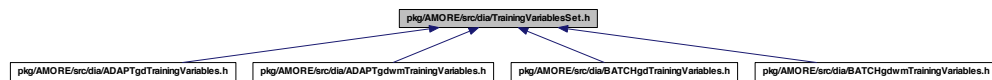
### 6.31 pkg/AMORE/src/dia/SimulationVariables.h File Reference

#### Classes

- class [SimulationVariables](#)  
*class [SimulationVariables](#) -*

### 6.32 pkg/AMORE/src/dia/TrainingVariablesSet.h File Reference

This graph shows which files directly or indirectly include this file:



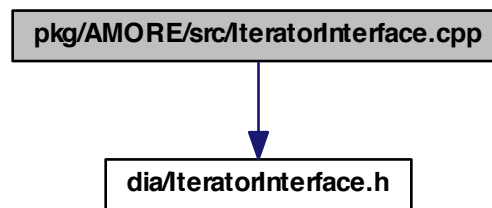
#### Classes

- class [TrainingVariablesSet](#)  
*class [TrainingVariablesSet](#) -*

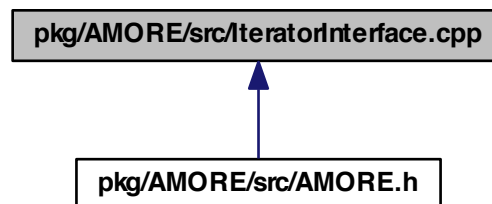
### 6.33 pkg/AMORE/src/IteratorInterface.cpp File Reference

```
#include "dia/IteratorInterface.h"
```

Include dependency graph for IteratorInterface.cpp:



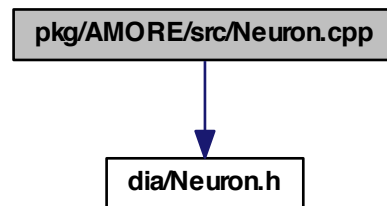
This graph shows which files directly or indirectly include this file:



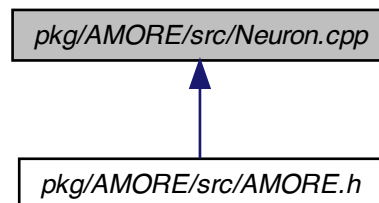
### 6.34 pkg/AMORE/src/Neuron.cpp File Reference

```
#include "dia/Neuron.h"
```

Include dependency graph for Neuron.cpp:



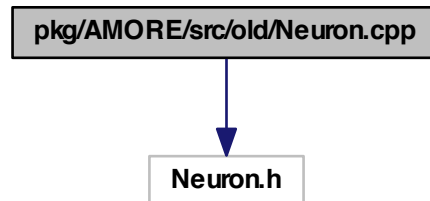
This graph shows which files directly or indirectly include this file:



## 6.35 pkg/AMORE/src/old/Neuron.cpp File Reference

```
#include "Neuron.h"
```

Include dependency graph for Neuron.cpp:



### 6.36 pkg/AMORE/src/old/ConContainer.cpp File Reference

#### Classes

- struct [CompareId](#)

### 6.37 pkg/AMORE/src/old/ConContainer.h File Reference

#### Classes

- class [ConContainer](#)  
*A vector of connections.*

### 6.38 pkg/AMORE/src/old/MLPlayer.h File Reference

#### Classes

- class [MLPlayer](#)

### 6.39 pkg/AMORE/src/old/MLPlayerContainer.h File Reference

#### Classes

- class [MLPlayerContainer](#)



## 6.40 pkg/AMORE/src/old/MLPneuralNetFactory.cpp File Reference

### Functions

- [MLPneuralNet CreateMLPneuralNet](#) (std::vector< int > numberOfNeuronsPerLayer)

### 6.40.1 Function Documentation

#### 6.40.1.1 MLPneuralNet CreateMLPneuralNet ( std::vector< int > *numberOfNeuronsPerLayer* )

Definition at line 2 of file MLPneuralNetFactory.cpp.

```
{  
  
    net = new MLPNeuralNet();  
  
    MLPPlayerPtr mlpLayerPtr;  
  
    std::vector<int> idx;  
  
    foreach (int n, numberOfNeuronsPerLayer)  
    {  
        for (int i=1; i<=n; ++i)  
        {  
            idx.push_back(i);  
        }  
        mlpLayerPtr.reset(new MLPPlayer( idx ) );  
        net.nLayers.push_back(mlpLayerPtr);  
    }  
  
    for (int i=1; i<=; ++i)  
    {  
        mlpPtr->buildAndAppend();  
    }  
}
```

## 6.41 pkg/AMORE/src/old/MLPneuronContainer.h File Reference

### Classes

- class [MLPneuronContainer](#)  
*A vector of connections.*

## 6.42 pkg/AMORE/src/old/NeuronContainer.cpp File Reference

## 6.43 pkg/AMORE/src/old/NeuronContainer.h File Reference

### Classes

- class [NeuronContainer](#)  
*A vector of neurons.*

# Index

- ~Con
  - Con, [20](#)
- ~Container
  - Container, [52](#)
- ~ContainerInterface
  - ContainerInterface, [65](#)
- ~ContainerIterator
  - ContainerIterator, [70](#)
- ~IteratorInterface
  - IteratorInterface, [74](#)
- ~Neuron
  - Neuron, [98](#)
- ~NeuronContainer
  - NeuronContainer, [106](#)
- ADAPTgdTrainingVariables, [9](#)
  - output, [10](#)
  - outputDerivative, [10](#)
- ADAPTgdwmTrainingVariables, [11](#)
  - output, [12](#)
  - outputDerivative, [12](#)
- altitude
  - RBFneuron, [114](#)
- AMORE.h
  - ConContainer, [118](#)
  - ConPtr, [118](#)
  - foreach, [118](#)
  - NeuronContainer, [118](#)
  - NeuronPtr, [119](#)
  - NeuronRef, [119](#)
  - size\_type, [118](#)
- append
  - Container, [52](#)
- BATCHgdTrainingVariables, [13](#)
  - output, [14](#)
  - outputDerivative, [14](#)
- BATCHgdwmTrainingVariables, [15](#)
  - output, [16](#)
  - outputDerivative, [16](#)
- begin
  - Container, [54](#)
- bias
  - MLPneuron, [88](#)
- buildAndAppend
  - MLPneuronContainer, [92](#)
- clear
  - Container, [54](#), [55](#)
  - ContainerInterface, [66](#)
- collection
  - Container, [63](#)
- CompareId, [17](#)
  - operator(), [17](#)
- Con, [18](#)
  - ~Con, [20](#)
  - Con, [19](#), [20](#)
  - d\_neuron, [28](#)
  - d\_weight, [29](#)
  - from, [29](#)
  - getFrom, [21](#)
  - getId, [21](#)
  - getWeight, [22](#)
  - Id, [23](#)
  - neuron, [24](#)
  - setFrom, [25](#)
  - setWeight, [25](#)
  - show, [26](#)
  - validate, [27](#), [28](#)
  - weight, [28](#), [29](#)
- con
  - Neuron, [102](#)
- ConContainer, [30](#)
  - AMORE.h, [118](#)
  - ConContainer, [34](#)
  - const\_iterator, [33](#)
  - const\_reference, [33](#)
  - erase, [34](#)
  - getId, [36](#)
  - iterator, [34](#)
  - numOfCons, [38](#)
  - select, [39](#)

- setFrom, 41
  - setWeight, 43, 44
  - validate, 46
  - value\_type, 34
- ConPtr
  - AMORE.h, 118
- const\_iterator
  - ConContainer, 33
  - Container, 51
  - NeuronContainer, 105
- const\_reference
  - ConContainer, 33
  - Container, 51
  - NeuronContainer, 105
- Container, 47
  - ~Container, 52
  - append, 52
  - begin, 54
  - clear, 54, 55
  - collection, 63
  - const\_iterator, 51
  - const\_reference, 51
  - Container, 52
  - ContainerIterator< T >, 63
  - createIterator, 55
  - empty, 55
  - end, 56
  - iterator, 51
  - load, 56
  - push\_back, 57, 58
  - reserve, 59
  - resize, 59
  - show, 60
  - size, 61
  - store, 62
  - validate, 62
  - value\_type, 51
- Container< T >
  - ContainerIterator, 71
- ContainerInterface, 63
  - ~ContainerInterface, 65
  - clear, 66
  - ContainerInterface, 65
  - createIterator, 66
  - empty, 66
  - push\_back, 66
  - reserve, 66
  - show, 66
  - size, 66
  - validate, 67
- ContainerIterator, 67
  - ~ContainerIterator, 70
  - Container< T >, 71
  - ContainerIterator, 70
  - currentItem, 70
  - d\_container, 71
  - d\_iterator, 72
  - first, 70
  - isDone, 71
  - next, 71
- ContainerIterator< T >
  - Container, 63
- createIterator
  - Container, 55
  - ContainerInterface, 66
- CreateMLPNeuralNet
  - MLPNeuralNetFactory.cpp, 141
- currentItem
  - ContainerIterator, 70
  - IteratorInterface, 74
- d\_container
  - ContainerIterator, 71
- d\_Id
  - Neuron, 102
- d\_iterator
  - ContainerIterator, 72
- d\_neuron
  - Con, 28
- d\_weight
  - Con, 29
- empty
  - Container, 55
  - ContainerInterface, 66
- end
  - Container, 56
- erase
  - ConContainer, 34
- first
  - ContainerIterator, 70
  - IteratorInterface, 74
- foreach
  - AMORE.h, 118
- from
  - Con, 29
- getConId
  - Neuron, 98

- NeuronContainer, 106
- getFrom
  - Con, 21
  - NeuronContainer, 107
- getId
  - Con, 21
  - ConContainer, 36
  - MLPneuronContainer, 92
  - Neuron, 98
  - NeuronContainer, 107
- getWeight
  - Con, 22
  - Neuron, 99
  - NeuronContainer, 107
- Id
  - Con, 23
  - Neuron, 99, 102
- isDone
  - ContainerIterator, 71
  - IteratorInterface, 74
- iterator
  - ConContainer, 34
  - Container, 51
  - NeuronContainer, 105
- IteratorInterface, 72
  - ~IteratorInterface, 74
  - currentItem, 74
  - first, 74
  - isDone, 74
  - IteratorInterface, 74
  - next, 74
- Layer, 75
  - nNeurons, 77
  - show, 77
  - validate, 77
- load
  - Container, 56
- MLPlayer, 77
- MLPlayerContainer, 80
- MLPneuralNet, 83
  - nLayers, 85
  - show, 85
  - validate, 85
- MLPneuralNetFactory.cpp
  - CreateMLPneuralNet, 141
- MLPneuron, 86
  - bias, 88
  - show, 88
  - validate, 88
- MLPneuronContainer, 89
  - buildAndAppend, 92
  - getId, 92
- NeuralNet, 92
  - show, 93
  - train, 93
  - validate, 93
- Neuron, 94
  - ~Neuron, 98
  - con, 102
  - d\_Id, 102
  - getConId, 98
  - getId, 98
  - getWeight, 99
  - Id, 99, 102
  - Neuron, 98
  - numOfCons, 99
  - outputValue, 102
  - setFrom, 99
  - setId, 99
  - setWeight, 100
  - show, 100, 101
  - validate, 101
- neuron
  - Con, 24
- NeuronContainer, 102
  - ~NeuronContainer, 106
  - AMORE.h, 118
  - const\_iterator, 105
  - const\_reference, 105
  - getConId, 106
  - getFrom, 107
  - getId, 107
  - getWeight, 107
  - iterator, 105
  - NeuronContainer, 106
  - numOfCons, 107
  - numOfNeurons, 107
  - setFrom, 108
  - setId, 108
  - setWeight, 108
  - value\_type, 106
- NeuronPtr
  - AMORE.h, 119
- NeuronRef
  - AMORE.h, 119
- next

- ContainerIterator, 71
- IteratorInterface, 74
- nLayers
  - MLPneuralNet, 85
  - RBFneuralNet, 111
- nNeurons
  - Layer, 77
- numOfCons
  - ConContainer, 38
  - Neuron, 99
  - NeuronContainer, 107
- numOfNeurons
  - NeuronContainer, 107
- operator()
  - CompareId, 17
- output
  - ADAPTgdTrainingVariables, 10
  - ADAPTgdwmTrainingVariables, 12
  - BATCHgdTrainingVariables, 14
  - BATCHgdwmTrainingVariables, 16
- outputDerivative
  - ADAPTgdTrainingVariables, 10
  - ADAPTgdwmTrainingVariables, 12
  - BATCHgdTrainingVariables, 14
  - BATCHgdwmTrainingVariables, 16
- outputValue
  - Neuron, 102
  - SimulationVariables, 115
- pkg/AMORE/src/AMORE.h, 117
- pkg/AMORE/src/Con.cpp, 119
- pkg/AMORE/src/Container.cpp, 120
- pkg/AMORE/src/containerInterface.cpp, 121
- pkg/AMORE/src/ContainerIterator.cpp, 122
- pkg/AMORE/src/dia/ADAPTgdTrainingVariables.h, 123
- pkg/AMORE/src/dia/ADAPTgdwmTrainingVariables.h, 124
- pkg/AMORE/src/dia/BATCHgdTrainingVariables.h, 125
- pkg/AMORE/src/dia/BATCHgdwmTrainingVariables.h, 125
- pkg/AMORE/src/dia/Con.h, 126
- pkg/AMORE/src/dia/Container.h, 127
- pkg/AMORE/src/dia/ContainerInterface.h, 129
- pkg/AMORE/src/dia/ContainerIterator.h, 129
- pkg/AMORE/src/dia/IteratorInterface.h, 131
- pkg/AMORE/src/dia/Layer.h, 131
- pkg/AMORE/src/dia/MLPneuralNet.h, 131
- pkg/AMORE/src/dia/MLPneuron.h, 132
- pkg/AMORE/src/dia/NeuralNet.h, 134
- pkg/AMORE/src/dia/Neuron.h, 135
- pkg/AMORE/src/dia/RBFneuralNet.h, 136
- pkg/AMORE/src/dia/RBFneuron.h, 136
- pkg/AMORE/src/dia/SimulationVariables.h, 137
- pkg/AMORE/src/dia/TrainingVariablesSet.h, 137
- pkg/AMORE/src/IteratorInterface.cpp, 138
- pkg/AMORE/src/Neuron.cpp, 138
- pkg/AMORE/src/old/Con.cpp, 120
- pkg/AMORE/src/old/Con.h, 127
- pkg/AMORE/src/old/ConContainer.cpp, 140
- pkg/AMORE/src/old/ConContainer.h, 140
- pkg/AMORE/src/old/Container.cpp, 121
- pkg/AMORE/src/old/Container.h, 128
- pkg/AMORE/src/old/MLPlayer.h, 140
- pkg/AMORE/src/old/MLPlayerContainer.h, 140
- pkg/AMORE/src/old/MLPneuralNet.h, 132
- pkg/AMORE/src/old/MLPneuralNetFactory.cpp, 141
- pkg/AMORE/src/old/MLPneuron.h, 133
- pkg/AMORE/src/old/MLPneuronContainer.h, 141
- pkg/AMORE/src/old/NeuralNet.h, 134
- pkg/AMORE/src/old/Neuron.cpp, 139
- pkg/AMORE/src/old/Neuron.h, 135
- pkg/AMORE/src/old/NeuronContainer.cpp, 142
- pkg/AMORE/src/old/NeuronContainer.h, 142
- pkg/AMORE/src/old/RBFneuralNet.h, 136
- push\_back
  - Container, 57, 58
- RBFneuralNet
  - nLayers, 111
- show, 111
  - validate, 111
- RBFneuron
  - altitude, 114
  - show, 114
  - validate, 114
  - width, 114
- reserve
  - Container, 59
- ContainerInterface, 66
  - resize

- Container, [59](#)
- select
  - ConContainer, [39](#)
- setFrom
  - Con, [25](#)
  - ConContainer, [41](#)
  - Neuron, [99](#)
  - NeuronContainer, [108](#)
- setId
  - Neuron, [99](#)
  - NeuronContainer, [108](#)
- setWeight
  - Con, [25](#)
  - ConContainer, [43](#), [44](#)
  - Neuron, [100](#)
  - NeuronContainer, [108](#)
- show
  - Con, [26](#)
  - Container, [60](#)
  - ContainerInterface, [66](#)
  - Layer, [77](#)
  - MLPneuralNet, [85](#)
  - MLPneuron, [88](#)
  - NeuralNet, [93](#)
  - Neuron, [100](#), [101](#)
  - RBFneuralNet, [111](#)
  - RBFneuron, [114](#)
- SimulationVariables, [114](#)
  - outputValue, [115](#)
- size
  - Container, [61](#)
  - ContainerInterface, [66](#)
- size\_type
  - AMORE.h, [118](#)
- store
  - Container, [62](#)
- train
  - NeuralNet, [93](#)
- TrainingVariablesSet, [115](#)
- validate
  - Con, [27](#), [28](#)
  - ConContainer, [46](#)
  - Container, [62](#)
  - ContainerInterface, [67](#)
  - Layer, [77](#)
  - MLPneuralNet, [85](#)
  - MLPneuron, [88](#)
  - NeuralNet, [93](#)
  - Neuron, [101](#)
  - RBFneuralNet, [111](#)
  - RBFneuron, [114](#)
- value\_type
  - ConContainer, [34](#)
  - Container, [51](#)
  - NeuronContainer, [106](#)
- weight
  - Con, [28](#), [29](#)
- width
  - RBFneuron, [114](#)