

AuboStudio SDK

0.6.3

Generated by Doxygen 1.16.1

1 AuboStudio SDK	1
2 Topic Index	5
2.1 Topics	5
3 Directory Hierarchy	7
3.1 Directories	7
4 Namespace Index	21
4.1 Namespace List	21
5 Hierarchical Index	23
5.1 Class Hierarchy	23
6 Class Index	27
6.1 Class List	27
7 File Index	33
7.1 File List	33
8 Topic Documentation	37
8.1 ApplicationApi	37
8.1.1 Detailed Description	38
8.1.2 Function Documentation	38
8.1.2.1 getDeviceManager()	38
8.1.2.2 getFeatureModel()	39
8.1.2.3 getIoModel()	39
8.1.2.4 getPayloadModel()	39
8.1.2.5 getTcpModel()	39
8.1.2.6 getValueFactory()	39
8.1.2.7 getVariableModel()	40
8.2 InstallationApi	40
8.2.1 Detailed Description	40
8.2.2 Function Documentation	41
8.2.2.1 getFeatureContributionModel()	41
8.2.2.2 getFunctionModel()	41
8.2.2.3 getPayloadContributionModel()	41
8.2.2.4 getTcpContributionModel()	41
8.3 ProgramApi	41
8.3.1 Detailed Description	42
8.3.2 Function Documentation	42
8.3.2.1 getProgramModel()	42
8.3.2.2 getUndoRedoManager()	42
8.3.2.3 getWaypointModel()	43
8.4 SystemApi	43
8.4.1 Detailed Description	44

8.4.2 Function Documentation	45
8.4.2.1 getBugfixVersion()	45
8.4.2.2 getBuildNumber()	45
8.4.2.3 getConfigurableDigitalInputConfigure()	45
8.4.2.4 getConfigurableDigitalOutputConfigure()	45
8.4.2.5 getENCConfigure()	45
8.4.2.6 getHomeDefaultPosition()	46
8.4.2.7 getHomePackagePosition()	46
8.4.2.8 getJointDefaultAcceleration()	46
8.4.2.9 getJointDefaultVelocity()	46
8.4.2.10 getJointMaxAcceleration()	46
8.4.2.11 getJointMaxPositions()	47
8.4.2.12 getJointMaxVelocity()	47
8.4.2.13 getJointMinPositions()	47
8.4.2.14 getLocale()	47
8.4.2.15 getLocaleForProgrammingLanguage()	47
8.4.2.16 getMajorVersion()	48
8.4.2.17 getMaxPayloadDistance()	48
8.4.2.18 getMaxPayloadMass()	48
8.4.2.19 getMinorVersion()	48
8.4.2.20 getRobotDhParams()	48
8.4.2.21 getSerialNumber()	49
8.4.2.22 getStandardAnalogInputConfigure()	49
8.4.2.23 getStandardAnalogOutputConfigure()	49
8.4.2.24 getStandardDigitalInputConfigure()	49
8.4.2.25 getStandardDigitalOutputConfigure()	50
8.4.2.26 getStaticLinkInputConfigure()	50
8.4.2.27 getStaticLinkOutputConfigure()	50
8.4.2.28 getTcpDefaultAcceleration()	50
8.4.2.29 getTcpDefaultVelocity()	50
8.4.2.30 getTcpMaxAcceleration()	51
8.4.2.31 getTcpMaxDistance()	51
8.4.2.32 getTcpMaxVelocity()	51
8.4.2.33 getToolAnalogInputConfigure()	51
8.4.2.34 getToolAnalogOutputConfigure()	51
8.4.2.35 getToolDigitalIOConfigure()	52
8.4.2.36 getUnitType()	52
8.4.2.37 isRealRobot()	52
8.5 UserInterfaceApi	52
8.5.1 Detailed Description	53
8.5.2 Function Documentation	53
8.5.2.1 getUserInteraction()	53

9 Directory Documentation	55
9.1 include/aubo_caps Directory Reference	55
9.2 include/aubo_caps/domain/program/nodes/builtin Directory Reference	56
9.3 include/aubo_caps/contribution/driver/gripper/capability Directory Reference	57
9.4 include/aubo_caps/contribution Directory Reference	57
9.5 include/aubo_caps/domain/data Directory Reference	58
9.6 include/aubo_caps/domain/device Directory Reference	58
9.7 doc Directory Reference	59
9.8 include/aubo_caps/domain Directory Reference	59
9.9 include/aubo_caps/contribution/driver Directory Reference	60
9.10 include/aubo_caps/domain/value/expression Directory Reference	60
9.11 include/aubo_caps/domain/feature Directory Reference	61
9.12 include/aubo_caps/domain/function Directory Reference	61
9.13 include/aubo_caps/contribution/driver/gripper Directory Reference	62
9.14 include Directory Reference	62
9.15 include/aubo_caps/contribution/installation Directory Reference	63
9.16 include/aubo_caps/domain/installation Directory Reference	63
9.17 include/aubo_caps/domain/io Directory Reference	64
9.18 include/aubo_caps/domain/program/nodes Directory Reference	64
9.19 include/aubo_caps/domain/payload Directory Reference	65
9.20 include/aubo_caps/contribution/program Directory Reference	65
9.21 include/aubo_caps/domain/program Directory Reference	66
9.22 include/aubo_caps/contribution/driver/screw_driver Directory Reference	67
9.23 include/aubo_caps/domain/script Directory Reference	67
9.24 include/aubo_caps/domain/program/structure Directory Reference	68
9.25 include/aubo_caps/domain/tcp Directory Reference	68
9.26 include/aubo_caps/domain/undoredo Directory Reference	69
9.27 include/aubo_caps/domain/user_interface Directory Reference	69
9.28 include/aubo_caps/domain/value Directory Reference	70
9.29 include/aubo_caps/domain/variable Directory Reference	71
9.30 include/aubo_caps/domain/waypoint Directory Reference	71
10 Namespace Documentation	73
10.1 arcs Namespace Reference	73
10.2 arcs::aubo_scope Namespace Reference	73
10.2.1 Typedef Documentation	81
10.2.1.1 ContributionProviderPtr	81
10.2.1.2 RobotMovementCallback	81
10.2.1.3 RobotPositionCallback	81
10.2.1.4 WidgetHandle	81
10.2.2 Enumeration Type Documentation	81
10.2.2.1 ScrewdriverOperationType	81
10.2.2.2 UnitType	82

10.2.3 Function Documentation	82
10.2.3.1 ARCS_CLASS_FORWARD() [1/130]	82
10.2.3.2 ARCS_CLASS_FORWARD() [2/130]	82
10.2.3.3 ARCS_CLASS_FORWARD() [3/130]	82
10.2.3.4 ARCS_CLASS_FORWARD() [4/130]	82
10.2.3.5 ARCS_CLASS_FORWARD() [5/130]	83
10.2.3.6 ARCS_CLASS_FORWARD() [6/130]	83
10.2.3.7 ARCS_CLASS_FORWARD() [7/130]	83
10.2.3.8 ARCS_CLASS_FORWARD() [8/130]	83
10.2.3.9 ARCS_CLASS_FORWARD() [9/130]	83
10.2.3.10 ARCS_CLASS_FORWARD() [10/130]	83
10.2.3.11 ARCS_CLASS_FORWARD() [11/130]	83
10.2.3.12 ARCS_CLASS_FORWARD() [12/130]	83
10.2.3.13 ARCS_CLASS_FORWARD() [13/130]	83
10.2.3.14 ARCS_CLASS_FORWARD() [14/130]	84
10.2.3.15 ARCS_CLASS_FORWARD() [15/130]	84
10.2.3.16 ARCS_CLASS_FORWARD() [16/130]	84
10.2.3.17 ARCS_CLASS_FORWARD() [17/130]	84
10.2.3.18 ARCS_CLASS_FORWARD() [18/130]	84
10.2.3.19 ARCS_CLASS_FORWARD() [19/130]	84
10.2.3.20 ARCS_CLASS_FORWARD() [20/130]	84
10.2.3.21 ARCS_CLASS_FORWARD() [21/130]	84
10.2.3.22 ARCS_CLASS_FORWARD() [22/130]	84
10.2.3.23 ARCS_CLASS_FORWARD() [23/130]	85
10.2.3.24 ARCS_CLASS_FORWARD() [24/130]	85
10.2.3.25 ARCS_CLASS_FORWARD() [25/130]	85
10.2.3.26 ARCS_CLASS_FORWARD() [26/130]	85
10.2.3.27 ARCS_CLASS_FORWARD() [27/130]	85
10.2.3.28 ARCS_CLASS_FORWARD() [28/130]	85
10.2.3.29 ARCS_CLASS_FORWARD() [29/130]	85
10.2.3.30 ARCS_CLASS_FORWARD() [30/130]	85
10.2.3.31 ARCS_CLASS_FORWARD() [31/130]	85
10.2.3.32 ARCS_CLASS_FORWARD() [32/130]	86
10.2.3.33 ARCS_CLASS_FORWARD() [33/130]	86
10.2.3.34 ARCS_CLASS_FORWARD() [34/130]	86
10.2.3.35 ARCS_CLASS_FORWARD() [35/130]	86
10.2.3.36 ARCS_CLASS_FORWARD() [36/130]	86
10.2.3.37 ARCS_CLASS_FORWARD() [37/130]	86
10.2.3.38 ARCS_CLASS_FORWARD() [38/130]	86
10.2.3.39 ARCS_CLASS_FORWARD() [39/130]	86
10.2.3.40 ARCS_CLASS_FORWARD() [40/130]	86
10.2.3.41 ARCS_CLASS_FORWARD() [41/130]	87
10.2.3.42 ARCS_CLASS_FORWARD() [42/130]	87

10.2.3.43 ARCS_CLASS_FORWARD()	[43/130]	87
10.2.3.44 ARCS_CLASS_FORWARD()	[44/130]	87
10.2.3.45 ARCS_CLASS_FORWARD()	[45/130]	87
10.2.3.46 ARCS_CLASS_FORWARD()	[46/130]	87
10.2.3.47 ARCS_CLASS_FORWARD()	[47/130]	87
10.2.3.48 ARCS_CLASS_FORWARD()	[48/130]	87
10.2.3.49 ARCS_CLASS_FORWARD()	[49/130]	87
10.2.3.50 ARCS_CLASS_FORWARD()	[50/130]	88
10.2.3.51 ARCS_CLASS_FORWARD()	[51/130]	88
10.2.3.52 ARCS_CLASS_FORWARD()	[52/130]	88
10.2.3.53 ARCS_CLASS_FORWARD()	[53/130]	88
10.2.3.54 ARCS_CLASS_FORWARD()	[54/130]	88
10.2.3.55 ARCS_CLASS_FORWARD()	[55/130]	88
10.2.3.56 ARCS_CLASS_FORWARD()	[56/130]	88
10.2.3.57 ARCS_CLASS_FORWARD()	[57/130]	88
10.2.3.58 ARCS_CLASS_FORWARD()	[58/130]	88
10.2.3.59 ARCS_CLASS_FORWARD()	[59/130]	89
10.2.3.60 ARCS_CLASS_FORWARD()	[60/130]	89
10.2.3.61 ARCS_CLASS_FORWARD()	[61/130]	89
10.2.3.62 ARCS_CLASS_FORWARD()	[62/130]	89
10.2.3.63 ARCS_CLASS_FORWARD()	[63/130]	89
10.2.3.64 ARCS_CLASS_FORWARD()	[64/130]	89
10.2.3.65 ARCS_CLASS_FORWARD()	[65/130]	89
10.2.3.66 ARCS_CLASS_FORWARD()	[66/130]	89
10.2.3.67 ARCS_CLASS_FORWARD()	[67/130]	89
10.2.3.68 ARCS_CLASS_FORWARD()	[68/130]	90
10.2.3.69 ARCS_CLASS_FORWARD()	[69/130]	90
10.2.3.70 ARCS_CLASS_FORWARD()	[70/130]	90
10.2.3.71 ARCS_CLASS_FORWARD()	[71/130]	90
10.2.3.72 ARCS_CLASS_FORWARD()	[72/130]	90
10.2.3.73 ARCS_CLASS_FORWARD()	[73/130]	90
10.2.3.74 ARCS_CLASS_FORWARD()	[74/130]	90
10.2.3.75 ARCS_CLASS_FORWARD()	[75/130]	90
10.2.3.76 ARCS_CLASS_FORWARD()	[76/130]	90
10.2.3.77 ARCS_CLASS_FORWARD()	[77/130]	91
10.2.3.78 ARCS_CLASS_FORWARD()	[78/130]	91
10.2.3.79 ARCS_CLASS_FORWARD()	[79/130]	91
10.2.3.80 ARCS_CLASS_FORWARD()	[80/130]	91
10.2.3.81 ARCS_CLASS_FORWARD()	[81/130]	91
10.2.3.82 ARCS_CLASS_FORWARD()	[82/130]	91
10.2.3.83 ARCS_CLASS_FORWARD()	[83/130]	91
10.2.3.84 ARCS_CLASS_FORWARD()	[84/130]	91
10.2.3.85 ARCS_CLASS_FORWARD()	[85/130]	91

10.2.3.86 ARCS_CLASS_FORWARD() [86/130]	92
10.2.3.87 ARCS_CLASS_FORWARD() [87/130]	92
10.2.3.88 ARCS_CLASS_FORWARD() [88/130]	92
10.2.3.89 ARCS_CLASS_FORWARD() [89/130]	92
10.2.3.90 ARCS_CLASS_FORWARD() [90/130]	92
10.2.3.91 ARCS_CLASS_FORWARD() [91/130]	92
10.2.3.92 ARCS_CLASS_FORWARD() [92/130]	92
10.2.3.93 ARCS_CLASS_FORWARD() [93/130]	92
10.2.3.94 ARCS_CLASS_FORWARD() [94/130]	92
10.2.3.95 ARCS_CLASS_FORWARD() [95/130]	93
10.2.3.96 ARCS_CLASS_FORWARD() [96/130]	93
10.2.3.97 ARCS_CLASS_FORWARD() [97/130]	93
10.2.3.98 ARCS_CLASS_FORWARD() [98/130]	93
10.2.3.99 ARCS_CLASS_FORWARD() [99/130]	93
10.2.3.100 ARCS_CLASS_FORWARD() [100/130]	93
10.2.3.101 ARCS_CLASS_FORWARD() [101/130]	93
10.2.3.102 ARCS_CLASS_FORWARD() [102/130]	93
10.2.3.103 ARCS_CLASS_FORWARD() [103/130]	93
10.2.3.104 ARCS_CLASS_FORWARD() [104/130]	94
10.2.3.105 ARCS_CLASS_FORWARD() [105/130]	94
10.2.3.106 ARCS_CLASS_FORWARD() [106/130]	94
10.2.3.107 ARCS_CLASS_FORWARD() [107/130]	94
10.2.3.108 ARCS_CLASS_FORWARD() [108/130]	94
10.2.3.109 ARCS_CLASS_FORWARD() [109/130]	94
10.2.3.110 ARCS_CLASS_FORWARD() [110/130]	94
10.2.3.111 ARCS_CLASS_FORWARD() [111/130]	94
10.2.3.112 ARCS_CLASS_FORWARD() [112/130]	94
10.2.3.113 ARCS_CLASS_FORWARD() [113/130]	95
10.2.3.114 ARCS_CLASS_FORWARD() [114/130]	95
10.2.3.115 ARCS_CLASS_FORWARD() [115/130]	95
10.2.3.116 ARCS_CLASS_FORWARD() [116/130]	95
10.2.3.117 ARCS_CLASS_FORWARD() [117/130]	95
10.2.3.118 ARCS_CLASS_FORWARD() [118/130]	95
10.2.3.119 ARCS_CLASS_FORWARD() [119/130]	95
10.2.3.120 ARCS_CLASS_FORWARD() [120/130]	95
10.2.3.121 ARCS_CLASS_FORWARD() [121/130]	95
10.2.3.122 ARCS_CLASS_FORWARD() [122/130]	96
10.2.3.123 ARCS_CLASS_FORWARD() [123/130]	96
10.2.3.124 ARCS_CLASS_FORWARD() [124/130]	96
10.2.3.125 ARCS_CLASS_FORWARD() [125/130]	96
10.2.3.126 ARCS_CLASS_FORWARD() [126/130]	96
10.2.3.127 ARCS_CLASS_FORWARD() [127/130]	96
10.2.3.128 ARCS_CLASS_FORWARD() [128/130]	96

10.2.3.129 ARCS_CLASS_FORWARD() [129/130]	96
10.2.3.130 ARCS_CLASS_FORWARD() [130/130]	96
10.3 arcs::aubo_sdk Namespace Reference	97
10.3.1 Function Documentation	97
10.3.1.1 ARCS_CLASS_FORWARD()	97
11 Class Documentation	99
11.1 arcs::aubo_scope::AnalogIo Class Reference	99
11.1.1 Detailed Description	101
11.1.2 Constructor & Destructor Documentation	101
11.1.2.1 AnalogIo() [1/3]	101
11.1.2.2 AnalogIo() [2/3]	102
11.1.2.3 ~AnalogIo()	102
11.1.2.4 AnalogIo() [3/3]	102
11.1.3 Member Function Documentation	102
11.1.3.1 getMaxRangeValue()	102
11.1.3.2 getMinRangeValue()	102
11.1.3.3 getValue()	102
11.1.3.4 isCurrent()	103
11.1.3.5 isVoltage()	103
11.1.3.6 setValue()	103
11.1.4 Friends And Related Symbol Documentation	103
11.1.4.1 DataSwitch	103
11.1.5 Member Data Documentation	103
11.1.5.1 d_	103
11.2 arcs::aubo_scope::AnalogIoFilter Struct Reference	104
11.2.1 Detailed Description	104
11.2.2 Constructor & Destructor Documentation	104
11.2.2.1 AnalogIoFilter()	104
11.2.3 Member Function Documentation	104
11.2.3.1 operator()	104
11.2.4 Member Data Documentation	104
11.2.4.1 interface_	104
11.2.4.2 is_input_	105
11.3 arcs::aubo_scope::ApplicationApi Class Reference	105
11.3.1 Detailed Description	106
11.3.2 Constructor & Destructor Documentation	106
11.3.2.1 ApplicationApi() [1/3]	106
11.3.2.2 ApplicationApi() [2/3]	107
11.3.2.3 ~ApplicationApi()	107
11.3.2.4 ApplicationApi() [3/3]	107
11.3.3 Member Function Documentation	107
11.3.3.1 getInstallationNode() [1/2]	107

11.3.3.2	getInstallationNode() [2/2]	108
11.3.3.3	getRobotProxy()	108
11.3.4	Friends And Related Symbol Documentation	109
11.3.4.1	DataSwitch	109
11.3.5	Member Data Documentation	109
11.3.5.1	d__	109
11.4	arcs::aubo_scope::AssemblyNode Class Reference	109
11.4.1	Detailed Description	110
11.4.2	Member Enumeration Documentation	110
11.4.2.1	ConfigType	110
11.4.3	Constructor & Destructor Documentation	111
11.4.3.1	AssemblyNode() [1/3]	111
11.4.3.2	AssemblyNode() [2/3]	112
11.4.3.3	~AssemblyNode()	112
11.4.3.4	AssemblyNode() [3/3]	112
11.4.4	Member Function Documentation	112
11.4.4.1	addAssemblyUntilNode()	112
11.4.5	Friends And Related Symbol Documentation	112
11.4.5.1	DataSwitch	112
11.4.6	Member Data Documentation	113
11.4.6.1	d__	113
11.5	arcs::aubo_scope::AssignmentNode Class Reference	113
11.5.1	Detailed Description	114
11.5.2	Member Enumeration Documentation	114
11.5.2.1	ConfigType	114
11.5.2.2	InputMode	115
11.5.3	Constructor & Destructor Documentation	115
11.5.3.1	AssignmentNode() [1/3]	115
11.5.3.2	AssignmentNode() [2/3]	116
11.5.3.3	~AssignmentNode()	116
11.5.3.4	AssignmentNode() [3/3]	116
11.5.4	Member Function Documentation	116
11.5.4.1	getConfigType()	116
11.5.4.2	getExpression()	117
11.5.4.3	getVariable()	117
11.5.4.4	setConfigType()	117
11.5.4.5	setExpression()	117
11.5.4.6	setVariable()	117
11.5.5	Friends And Related Symbol Documentation	117
11.5.5.1	DataSwitch	117
11.5.6	Member Data Documentation	118
11.5.6.1	d__	118
11.6	arcs::aubo_scope::AuboCapInfo Class Reference	118

11.6.1 Detailed Description	118
11.6.2 Constructor & Destructor Documentation	118
11.6.2.1 ~AuboCapInfo()	118
11.6.3 Member Function Documentation	119
11.6.3.1 getName()	119
11.6.3.2 getSymbolicName()	119
11.6.3.3 getVendor()	119
11.7 arcs::aubo_scope::BaseFeature Class Reference	119
11.7.1 Detailed Description	121
11.7.2 Constructor & Destructor Documentation	121
11.7.2.1 BaseFeature() [1/3]	121
11.7.2.2 BaseFeature() [2/3]	122
11.7.2.3 ~BaseFeature()	122
11.7.2.4 BaseFeature() [3/3]	122
11.7.3 Friends And Related Symbol Documentation	122
11.7.3.1 DataSwitch	122
11.7.4 Member Data Documentation	122
11.7.4.1 d_	122
11.8 arcs::aubo_scope::BoolRegister Class Reference	123
11.8.1 Detailed Description	124
11.8.2 Constructor & Destructor Documentation	125
11.8.2.1 ~BoolRegister()	125
11.9 arcs::aubo_scope::BreakNode Class Reference	125
11.9.1 Detailed Description	126
11.9.2 Constructor & Destructor Documentation	126
11.9.2.1 BreakNode() [1/3]	126
11.9.2.2 BreakNode() [2/3]	127
11.9.2.3 ~BreakNode()	127
11.9.2.4 BreakNode() [3/3]	128
11.9.3 Friends And Related Symbol Documentation	128
11.9.3.1 DataSwitch	128
11.9.4 Member Data Documentation	128
11.9.4.1 d_	128
11.10 arcs::aubo_scope::CallNode Class Reference	128
11.10.1 Detailed Description	130
11.10.2 Constructor & Destructor Documentation	130
11.10.2.1 CallNode() [1/3]	130
11.10.2.2 CallNode() [2/3]	130
11.10.2.3 ~CallNode()	131
11.10.2.4 CallNode() [3/3]	131
11.10.3 Friends And Related Symbol Documentation	131
11.10.3.1 DataSwitch	131
11.10.4 Member Data Documentation	131

11.10.4.1 d_	131
11.11 arcs::aubo_scope::CaseNode Class Reference	132
11.11.1 Detailed Description	133
11.11.2 Constructor & Destructor Documentation	133
11.11.2.1 CaseNode() [1/3]	133
11.11.2.2 CaseNode() [2/3]	134
11.11.2.3 ~CaseNode()	134
11.11.2.4 CaseNode() [3/3]	134
11.11.3 Friends And Related Symbol Documentation	134
11.11.3.1 DataSwitch	134
11.11.4 Member Data Documentation	134
11.11.4.1 d_	134
11.12 arcs::aubo_scope::CircleMoveNode Class Reference	135
11.12.1 Detailed Description	136
11.12.2 Constructor & Destructor Documentation	136
11.12.2.1 CircleMoveNode() [1/3]	136
11.12.2.2 CircleMoveNode() [2/3]	137
11.12.2.3 ~CircleMoveNode()	137
11.12.2.4 CircleMoveNode() [3/3]	137
11.12.3 Friends And Related Symbol Documentation	137
11.12.3.1 DataSwitch	137
11.12.4 Member Data Documentation	137
11.12.4.1 d_	137
11.13 arcs::aubo_scope::CommentNode Class Reference	138
11.13.1 Detailed Description	139
11.13.2 Constructor & Destructor Documentation	139
11.13.2.1 CommentNode() [1/3]	139
11.13.2.2 CommentNode() [2/3]	140
11.13.2.3 ~CommentNode()	140
11.13.2.4 CommentNode() [3/3]	140
11.13.3 Member Function Documentation	140
11.13.3.1 getComment()	140
11.13.3.2 setComment()	140
11.13.4 Friends And Related Symbol Documentation	141
11.13.4.1 DataSwitch	141
11.13.5 Member Data Documentation	141
11.13.5.1 d_	141
11.14 arcs::aubo_scope::ContributionProvider< T > Class Template Reference	141
11.14.1 Detailed Description	141
11.14.2 Constructor & Destructor Documentation	142
11.14.2.1 ~ContributionProvider()	142
11.14.3 Member Function Documentation	142
11.14.3.1 get()	142

11.15 arcs::aubo_scope::ConveyorTrackingNode Class Reference	143
11.15.1 Detailed Description	144
11.15.2 Member Enumeration Documentation	144
11.15.2.1 ConveyorStartActionQueueType	144
11.15.2.2 ConveyorStartActionType	144
11.15.2.3 ConveyorTrackingTargetType	145
11.15.3 Constructor & Destructor Documentation	145
11.15.3.1 ConveyorTrackingNode() [1/3]	145
11.15.3.2 ConveyorTrackingNode() [2/3]	146
11.15.3.3 ~ConveyorTrackingNode()	146
11.15.3.4 ConveyorTrackingNode() [3/3]	146
11.15.4 Friends And Related Symbol Documentation	146
11.15.4.1 DataSwitch	146
11.15.5 Member Data Documentation	146
11.15.5.1 d_	146
11.16 arcs::aubo_scope::DataModel Class Reference	147
11.16.1 Detailed Description	149
11.16.2 Constructor & Destructor Documentation	149
11.16.2.1 DataModel() [1/3]	149
11.16.2.2 DataModel() [2/3]	150
11.16.2.3 ~DataModel()	150
11.16.2.4 DataModel() [3/3]	150
11.16.3 Member Function Documentation	150
11.16.3.1 get() [1/21]	150
11.16.3.2 get() [2/21]	151
11.16.3.3 get() [3/21]	151
11.16.3.4 get() [4/21]	151
11.16.3.5 get() [5/21]	152
11.16.3.6 get() [6/21]	152
11.16.3.7 get() [7/21]	152
11.16.3.8 get() [8/21]	153
11.16.3.9 get() [9/21]	153
11.16.3.10 get() [10/21]	153
11.16.3.11 get() [11/21]	154
11.16.3.12 get() [12/21]	154
11.16.3.13 get() [13/21]	154
11.16.3.14 get() [14/21]	155
11.16.3.15 get() [15/21]	155
11.16.3.16 get() [16/21]	155
11.16.3.17 get() [17/21]	156
11.16.3.18 get() [18/21]	156
11.16.3.19 get() [19/21]	156
11.16.3.20 get() [20/21]	156

11.16.3.21	get() [21/21]	157
11.16.3.22	getKeys()	157
11.16.3.23	isSet()	157
11.16.3.24	remove()	157
11.16.3.25	set() [1/21]	158
11.16.3.26	set() [2/21]	158
11.16.3.27	set() [3/21]	159
11.16.3.28	set() [4/21]	159
11.16.3.29	set() [5/21]	160
11.16.3.30	set() [6/21]	160
11.16.3.31	set() [7/21]	161
11.16.3.32	set() [8/21]	161
11.16.3.33	set() [9/21]	162
11.16.3.34	set() [10/21]	162
11.16.3.35	set() [11/21]	163
11.16.3.36	set() [12/21]	163
11.16.3.37	set() [13/21]	163
11.16.3.38	set() [14/21]	164
11.16.3.39	set() [15/21]	164
11.16.3.40	set() [16/21]	164
11.16.3.41	set() [17/21]	165
11.16.3.42	set() [18/21]	165
11.16.3.43	set() [19/21]	166
11.16.3.44	set() [20/21]	166
11.16.3.45	set() [21/21]	166
11.16.4	Friends And Related Symbol Documentation	166
11.16.4.1	DataSwitch	166
11.16.5	Member Data Documentation	167
11.16.5.1	d__	167
11.17	arcs::aubo_scope::DefaultCaseNode Class Reference	167
11.17.1	Detailed Description	168
11.17.2	Constructor & Destructor Documentation	169
11.17.2.1	DefaultCaseNode() [1/3]	169
11.17.2.2	DefaultCaseNode() [2/3]	169
11.17.2.3	~DefaultCaseNode()	170
11.17.2.4	DefaultCaseNode() [3/3]	170
11.17.3	Friends And Related Symbol Documentation	170
11.17.3.1	DataSwitch	170
11.17.4	Member Data Documentation	170
11.17.4.1	d__	170
11.18	arcs::aubo_scope::DefaultSingletonTraits< Type > Struct Template Reference	170
11.18.1	Detailed Description	170
11.19	arcs::aubo_scope::Device Class Reference	171

11.19.1 Detailed Description	171
11.19.2 Constructor & Destructor Documentation	171
11.19.2.1 ~Device()	171
11.19.3 Member Function Documentation	172
11.19.3.1 getContributorInfo()	172
11.19.3.2 getDisplayName()	172
11.19.3.3 getId()	172
11.19.3.4 isResolvable()	172
11.20 arcs::aubo_scope::DeviceManager Class Reference	173
11.20.1 Detailed Description	173
11.20.2 Constructor & Destructor Documentation	173
11.20.2.1 ~DeviceManager()	173
11.20.3 Member Function Documentation	173
11.20.3.1 getGrippers()	173
11.20.3.2 getScrewDrivers()	174
11.21 arcs::aubo_scope::DeviceRegistrationManager Class Reference	174
11.21.1 Detailed Description	174
11.21.2 Constructor & Destructor Documentation	174
11.21.2.1 ~DeviceRegistrationManager()	174
11.21.3 Member Function Documentation	174
11.21.3.1 registerGripperContribution()	174
11.21.3.2 registerScrewDriverContribution()	175
11.22 arcs::aubo_scope::DigitalIo Class Reference	175
11.22.1 Detailed Description	177
11.22.2 Constructor & Destructor Documentation	177
11.22.2.1 DigitalIo() [1/3]	177
11.22.2.2 DigitalIo() [2/3]	178
11.22.2.3 ~DigitalIo()	178
11.22.2.4 DigitalIo() [3/3]	179
11.22.3 Member Function Documentation	179
11.22.3.1 getValue()	179
11.22.3.2 setValue()	179
11.22.4 Friends And Related Symbol Documentation	179
11.22.4.1 DataSwitch	179
11.22.5 Member Data Documentation	180
11.22.5.1 d_	180
11.23 arcs::aubo_scope::DigitalIoFilter Struct Reference	180
11.23.1 Detailed Description	180
11.23.2 Constructor & Destructor Documentation	180
11.23.2.1 DigitalIoFilter()	180
11.23.3 Member Function Documentation	181
11.23.3.1 operator>()	181
11.23.4 Member Data Documentation	181

11.23.4.1	interface__	181
11.23.4.2	is_input__	181
11.24	arcs::aubo_scope::DirectionNode Class Reference	181
11.24.1	Detailed Description	183
11.24.2	Member Enumeration Documentation	183
11.24.2.1	Axis	183
11.24.2.2	DirectionSelectionType	183
11.24.2.3	FeatureSelectionType	184
11.24.2.4	ParameterType	184
11.24.3	Constructor & Destructor Documentation	185
11.24.3.1	DirectionNode() [1/3]	185
11.24.3.2	DirectionNode() [2/3]	185
11.24.3.3	~DirectionNode()	186
11.24.3.4	DirectionNode() [3/3]	186
11.24.4	Member Function Documentation	186
11.24.4.1	getAxis()	186
11.24.4.2	getDirectionSelectionType()	186
11.24.4.3	getExpression()	186
11.24.4.4	getFeature()	186
11.24.4.5	getFeatureSelectionType()	186
11.24.4.6	getToolAcceleration()	187
11.24.4.7	getToolSpeed()	187
11.24.4.8	getType()	187
11.24.4.9	setAxis()	187
11.24.4.10	setDirectionSelectionType()	187
11.24.4.11	setExpression()	187
11.24.4.12	setFeature()	187
11.24.4.13	setFeatureSelectionType()	187
11.24.4.14	setToolAcceleration()	188
11.24.4.15	setToolSpeed()	188
11.24.4.16	setType()	188
11.24.5	Friends And Related Symbol Documentation	188
11.24.5.1	DataSwitch	188
11.24.6	Member Data Documentation	188
11.24.6.1	d__	188
11.25	arcs::aubo_scope::DoubleRangeValidator Struct Reference	189
11.25.1	Detailed Description	190
11.25.2	Constructor & Destructor Documentation	190
11.25.2.1	DoubleRangeValidator() [1/3]	190
11.25.2.2	DoubleRangeValidator() [2/3]	190
11.25.2.3	DoubleRangeValidator() [3/3]	190
11.25.3	Member Function Documentation	190
11.25.3.1	getMessage()	190

11.25.3.2	isValid()	191
11.25.3.3	updateValue()	191
11.25.4	Member Data Documentation	191
11.25.4.1	is_max_inclusive_	191
11.25.4.2	is_min_inclusive_	191
11.25.4.3	max_value_	192
11.25.4.4	min_value_	192
11.25.4.5	range_func_	192
11.25.4.6	range_func_with_flags_	192
11.26	arcs::aubo_scope::DoubleRegister Class Reference	192
11.26.1	Detailed Description	194
11.26.2	Constructor & Destructor Documentation	194
11.26.2.1	~DoubleRegister()	194
11.27	arcs::aubo_scope::DriveScrewNotOKParameters Class Reference	194
11.27.1	Detailed Description	194
11.27.2	Constructor & Destructor Documentation	195
11.27.2.1	~DriveScrewNotOKParameters()	195
11.28	arcs::aubo_scope::DriveScrewOKParameters Class Reference	195
11.28.1	Detailed Description	195
11.28.2	Constructor & Destructor Documentation	195
11.28.2.1	~DriveScrewOKParameters()	195
11.29	arcs::aubo_scope::ElseIfNode Class Reference	196
11.29.1	Detailed Description	197
11.29.2	Constructor & Destructor Documentation	197
11.29.2.1	ElseIfNode() [1/3]	197
11.29.2.2	ElseIfNode() [2/3]	198
11.29.2.3	~ElseIfNode()	198
11.29.2.4	ElseIfNode() [3/3]	198
11.29.3	Member Function Documentation	198
11.29.3.1	getExpression()	198
11.29.3.2	setExpression()	198
11.29.4	Friends And Related Symbol Documentation	199
11.29.4.1	DataSwitch	199
11.29.5	Member Data Documentation	199
11.29.5.1	d_	199
11.30	arcs::aubo_scope::ElseNode Class Reference	200
11.30.1	Detailed Description	201
11.30.2	Constructor & Destructor Documentation	201
11.30.2.1	ElseNode() [1/3]	201
11.30.2.2	ElseNode() [2/3]	202
11.30.2.3	~ElseNode()	202
11.30.2.4	ElseNode() [3/3]	202
11.30.3	Friends And Related Symbol Documentation	202

11.30.3.1 DataSwitch	202
11.30.4 Member Data Documentation	202
11.30.4.1 d_	202
11.31 arcs::aubo_scope::EtherNetIP Class Reference	203
11.31.1 Detailed Description	205
11.31.2 Constructor & Destructor Documentation	205
11.31.2.1 EtherNetIP() [1/3]	205
11.31.2.2 EtherNetIP() [2/3]	206
11.31.2.3 ~EtherNetIP()	206
11.31.2.4 EtherNetIP() [3/3]	206
11.31.3 Member Function Documentation	206
11.31.3.1 getBitOffset()	206
11.31.3.2 getByteOffset()	206
11.31.3.3 getValue()	207
11.31.3.4 isSlave()	207
11.31.3.5 setValue()	207
11.31.4 Friends And Related Symbol Documentation	207
11.31.4.1 DataSwitch	207
11.31.5 Member Data Documentation	208
11.31.5.1 d_	208
11.32 arcs::aubo_scope::EtherNetIPFilter Struct Reference	208
11.32.1 Detailed Description	208
11.32.2 Constructor & Destructor Documentation	208
11.32.2.1 EtherNetIPFilter()	208
11.32.3 Member Function Documentation	209
11.32.3.1 operator>()	209
11.32.4 Member Data Documentation	209
11.32.4.1 is_input_	209
11.32.4.2 type_	209
11.33 arcs::aubo_scope::Expression Class Reference	209
11.33.1 Detailed Description	210
11.33.2 Constructor & Destructor Documentation	210
11.33.2.1 Expression() [1/3]	210
11.33.2.2 Expression() [2/3]	211
11.33.2.3 ~Expression()	211
11.33.2.4 Expression() [3/3]	211
11.33.3 Member Function Documentation	211
11.33.3.1 append()	211
11.33.3.2 appendFeature()	211
11.33.3.3 appendIo()	212
11.33.3.4 appendVariable()	212
11.33.3.5 appendWaypoint()	212
11.33.3.6 toString()	213

11.33.4 Friends And Related Symbol Documentation	213
11.33.4.1 DataSwitch	213
11.33.5 Member Data Documentation	213
11.33.5.1 d_	213
11.34 arcs::aubo_scope::Feature Class Reference	213
11.34.1 Detailed Description	214
11.34.2 Constructor & Destructor Documentation	215
11.34.2.1 Feature() [1/3]	215
11.34.2.2 Feature() [2/3]	215
11.34.2.3 ~Feature()	216
11.34.2.4 Feature() [3/3]	216
11.34.3 Member Function Documentation	216
11.34.3.1 getDisplayName()	216
11.34.3.2 getPose()	216
11.34.3.3 hashCode()	216
11.34.3.4 isDefined()	216
11.34.3.5 isJoggable()	217
11.34.3.6 isResolvable()	217
11.34.3.7 isVariable()	217
11.34.4 Friends And Related Symbol Documentation	217
11.34.4.1 DataSwitch	217
11.34.5 Member Data Documentation	217
11.34.5.1 d_	217
11.35 arcs::aubo_scope::FeatureContributionModel Class Reference	218
11.35.1 Detailed Description	218
11.35.2 Constructor & Destructor Documentation	219
11.35.2.1 FeatureContributionModel() [1/3]	219
11.35.2.2 FeatureContributionModel() [2/3]	219
11.35.2.3 ~FeatureContributionModel()	220
11.35.2.4 FeatureContributionModel() [3/3]	220
11.35.3 Member Function Documentation	220
11.35.3.1 addFeature()	220
11.35.3.2 getFeature()	220
11.35.3.3 removeFeature()	221
11.35.3.4 renameFeature()	221
11.35.3.5 updateFeature()	221
11.35.4 Friends And Related Symbol Documentation	222
11.35.4.1 DataSwitch	222
11.35.5 Member Data Documentation	222
11.35.5.1 d_	222
11.36 arcs::aubo_scope::FeatureDefinedFilter Struct Reference	222
11.36.1 Detailed Description	223
11.36.2 Member Function Documentation	223

11.36.2.1 operator>()	223
11.37 arcs::aubo_scope::FeatureJoggableFilter Struct Reference	223
11.37.1 Detailed Description	223
11.37.2 Member Function Documentation	223
11.37.2.1 operator>()	223
11.38 arcs::aubo_scope::FeatureLine Class Reference	224
11.38.1 Detailed Description	225
11.38.2 Constructor & Destructor Documentation	225
11.38.2.1 FeatureLine() [1/3]	225
11.38.2.2 FeatureLine() [2/3]	226
11.38.2.3 ~FeatureLine()	226
11.38.2.4 FeatureLine() [3/3]	227
11.38.3 Friends And Related Symbol Documentation	227
11.38.3.1 DataSwitch	227
11.38.4 Member Data Documentation	227
11.38.4.1 d_	227
11.39 arcs::aubo_scope::FeatureModel Class Reference	227
11.39.1 Detailed Description	228
11.39.2 Constructor & Destructor Documentation	228
11.39.2.1 FeatureModel() [1/3]	228
11.39.2.2 FeatureModel() [2/3]	229
11.39.2.3 ~FeatureModel()	229
11.39.2.4 FeatureModel() [3/3]	229
11.39.3 Member Function Documentation	229
11.39.3.1 getAll()	229
11.39.3.2 getBaseFeature()	229
11.39.3.3 getByName()	229
11.39.3.4 getFeatures()	230
11.39.3.5 getGeomFeture()	230
11.39.3.6 getToolFeature()	230
11.39.4 Friends And Related Symbol Documentation	231
11.39.4.1 DataSwitch	231
11.39.5 Member Data Documentation	231
11.39.5.1 d_	231
11.40 arcs::aubo_scope::FeaturePlane Class Reference	231
11.40.1 Detailed Description	233
11.40.2 Constructor & Destructor Documentation	233
11.40.2.1 FeaturePlane() [1/3]	233
11.40.2.2 FeaturePlane() [2/3]	234
11.40.2.3 ~FeaturePlane()	234
11.40.2.4 FeaturePlane() [3/3]	234
11.40.3 Friends And Related Symbol Documentation	234
11.40.3.1 DataSwitch	234

11.40.4 Member Data Documentation	234
11.40.4.1 d_	234
11.41 arcs::aubo_scope::FeaturePoint Class Reference	235
11.41.1 Detailed Description	236
11.41.2 Constructor & Destructor Documentation	237
11.41.2.1 FeaturePoint() [1/3]	237
11.41.2.2 FeaturePoint() [2/3]	237
11.41.2.3 ~FeaturePoint()	238
11.41.2.4 FeaturePoint() [3/3]	238
11.41.3 Friends And Related Symbol Documentation	238
11.41.3.1 DataSwitch	238
11.41.4 Member Data Documentation	238
11.41.4.1 d_	238
11.42 arcs::aubo_scope::FeatureTool Class Reference	238
11.42.1 Detailed Description	240
11.42.2 Constructor & Destructor Documentation	240
11.42.2.1 FeatureTool() [1/3]	240
11.42.2.2 FeatureTool() [2/3]	241
11.42.2.3 ~FeatureTool()	241
11.42.2.4 FeatureTool() [3/3]	241
11.42.3 Friends And Related Symbol Documentation	241
11.42.3.1 DataSwitch	241
11.42.4 Member Data Documentation	241
11.42.4.1 d_	241
11.43 arcs::aubo_scope::FeatureVariable Class Reference	242
11.43.1 Detailed Description	244
11.43.2 Constructor & Destructor Documentation	244
11.43.2.1 FeatureVariable() [1/3]	244
11.43.2.2 FeatureVariable() [2/3]	245
11.43.2.3 ~FeatureVariable()	245
11.43.2.4 FeatureVariable() [3/3]	245
11.43.3 Friends And Related Symbol Documentation	245
11.43.3.1 DataSwitch	245
11.43.4 Member Data Documentation	245
11.43.4.1 d_	245
11.44 arcs::aubo_scope::FeatureVariableFilter Struct Reference	246
11.44.1 Detailed Description	246
11.44.2 Member Function Documentation	246
11.44.2.1 operator>()	246
11.45 arcs::aubo_scope::FeedScrewParameters Class Reference	246
11.45.1 Detailed Description	246
11.45.2 Constructor & Destructor Documentation	247
11.45.2.1 ~FeedScrewParameters()	247

11.46 arcs::aubo_scope::FolderNode Class Reference	247
11.46.1 Detailed Description	248
11.46.2 Constructor & Destructor Documentation	248
11.46.2.1 FolderNode() [1/3]	248
11.46.2.2 FolderNode() [2/3]	249
11.46.2.3 ~FolderNode()	249
11.46.2.4 FolderNode() [3/3]	250
11.46.3 Member Function Documentation	250
11.46.3.1 setFolderName()	250
11.46.4 Friends And Related Symbol Documentation	250
11.46.4.1 DataSwitch	250
11.46.5 Member Data Documentation	250
11.46.5.1 d_	250
11.47 arcs::aubo_scope::ForceNode Class Reference	251
11.47.1 Detailed Description	252
11.47.2 Member Enumeration Documentation	252
11.47.2.1 ConfigType	252
11.47.3 Constructor & Destructor Documentation	253
11.47.3.1 ForceNode() [1/3]	253
11.47.3.2 ForceNode() [2/3]	253
11.47.3.3 ~ForceNode()	254
11.47.3.4 ForceNode() [3/3]	254
11.47.4 Friends And Related Symbol Documentation	254
11.47.4.1 DataSwitch	254
11.47.5 Member Data Documentation	254
11.47.5.1 d_	254
11.48 arcs::aubo_scope::ForceOptionNode Class Reference	254
11.48.1 Detailed Description	256
11.48.2 Constructor & Destructor Documentation	256
11.48.2.1 ForceOptionNode() [1/3]	256
11.48.2.2 ForceOptionNode() [2/3]	256
11.48.2.3 ~ForceOptionNode()	257
11.48.2.4 ForceOptionNode() [3/3]	257
11.48.3 Friends And Related Symbol Documentation	257
11.48.3.1 DataSwitch	257
11.48.4 Member Data Documentation	257
11.48.4.1 d_	257
11.49 arcs::aubo_scope::Function Class Reference	257
11.49.1 Detailed Description	258
11.49.2 Constructor & Destructor Documentation	258
11.49.2.1 Function() [1/3]	258
11.49.2.2 Function() [2/3]	259
11.49.2.3 ~Function()	259

11.49.2.4 Function() [3/3]	260
11.49.3 Member Function Documentation	260
11.49.3.1 getArguments()	260
11.49.3.2 getContributorInfo()	260
11.49.3.3 getName()	260
11.49.4 Friends And Related Symbol Documentation	260
11.49.4.1 DataSwitch	260
11.49.5 Member Data Documentation	261
11.49.5.1 d_	261
11.50 arcs::aubo_scope::FunctionModel Class Reference	261
11.50.1 Detailed Description	262
11.50.2 Constructor & Destructor Documentation	262
11.50.2.1 FunctionModel() [1/3]	262
11.50.2.2 FunctionModel() [2/3]	263
11.50.2.3 ~FunctionModel()	263
11.50.2.4 FunctionModel() [3/3]	263
11.50.3 Member Function Documentation	263
11.50.3.1 addFunction()	263
11.50.3.2 getFunction()	264
11.50.3.3 getFunctions()	264
11.50.3.4 removeFunction()	264
11.50.4 Friends And Related Symbol Documentation	265
11.50.4.1 DataSwitch	265
11.50.5 Member Data Documentation	265
11.50.5.1 d_	265
11.51 arcs::aubo_scope::GlobalVariable Class Reference	266
11.51.1 Detailed Description	267
11.51.2 Constructor & Destructor Documentation	268
11.51.2.1 GlobalVariable() [1/3]	268
11.51.2.2 GlobalVariable() [2/3]	268
11.51.2.3 ~GlobalVariable()	269
11.51.2.4 GlobalVariable() [3/3]	269
11.51.3 Friends And Related Symbol Documentation	269
11.51.3.1 DataSwitch	269
11.51.4 Member Data Documentation	269
11.51.4.1 d_	269
11.52 arcs::aubo_scope::GlobalVariableFilter Struct Reference	269
11.52.1 Detailed Description	269
11.52.2 Member Function Documentation	270
11.52.2.1 operator()()	270
11.53 arcs::aubo_scope::GripActionParameters Class Reference	270
11.53.1 Detailed Description	270
11.53.2 Constructor & Destructor Documentation	270

11.53.2.1 ~GripActionParameters()	270
11.53.3 Member Function Documentation	270
11.53.3.1 getForce()	270
11.53.3.2 getSpeed()	271
11.53.3.3 getVacuum()	271
11.53.3.4 getWidth()	271
11.53.3.5 isGripDetectionEnabled()	272
11.54 arcs::aubo_scope::GripForceCapability Class Reference	273
11.54.1 Detailed Description	273
11.54.2 Constructor & Destructor Documentation	273
11.54.2.1 ~GripForceCapability()	273
11.54.3 Member Function Documentation	273
11.54.3.1 updateCapability()	273
11.55 arcs::aubo_scope::GripperApiProvider Class Reference	274
11.55.1 Detailed Description	274
11.55.2 Constructor & Destructor Documentation	274
11.55.2.1 ~GripperApiProvider()	274
11.55.3 Member Function Documentation	275
11.55.3.1 getIoModel()	275
11.55.3.2 getSystemApi()	275
11.56 arcs::aubo_scope::GripperConfiguration Class Reference	275
11.56.1 Detailed Description	276
11.56.2 Constructor & Destructor Documentation	276
11.56.2.1 ~GripperConfiguration()	276
11.56.3 Member Function Documentation	276
11.56.3.1 registerGripDetectedCapability()	276
11.56.3.2 registerGrippingForceCapability()	277
11.56.3.3 registerGrippingVacuumCapability()	278
11.56.3.4 registerReleaseDetectedCapability()	279
11.56.3.5 registerSpeedCapability()	280
11.56.3.6 registerWidthCapability()	281
11.57 arcs::aubo_scope::GripperContribution Class Reference	282
11.57.1 Detailed Description	283
11.57.2 Constructor & Destructor Documentation	283
11.57.2.1 ~GripperContribution()	283
11.57.3 Member Function Documentation	283
11.57.3.1 configureGripper()	283
11.57.3.2 generateGripActionScript()	284
11.57.3.3 generateReleaseActionScript()	284
11.57.3.4 getTitle()	285
11.58 arcs::aubo_scope::GripperDevice Class Reference	286
11.58.1 Detailed Description	287
11.58.2 Constructor & Destructor Documentation	287

11.58.2.1 ~GripperDevice()	287
11.59 arcs::aubo_scope::GripperNode Class Reference	287
11.59.1 Detailed Description	289
11.59.2 Member Enumeration Documentation	289
11.59.2.1 ConfigType	289
11.59.2.2 SettingType	289
11.59.3 Constructor & Destructor Documentation	290
11.59.3.1 GripperNode() [1/3]	290
11.59.3.2 GripperNode() [2/3]	291
11.59.3.3 ~GripperNode()	291
11.59.3.4 GripperNode() [3/3]	291
11.59.4 Member Function Documentation	291
11.59.4.1 getConfigType()	291
11.59.4.2 getGripperDevice()	291
11.59.4.3 getSettingType()	292
11.59.4.4 setConfigType()	292
11.59.5 Friends And Related Symbol Documentation	292
11.59.5.1 DataSwitch	292
11.59.6 Member Data Documentation	292
11.59.6.1 d__	292
11.60 arcs::aubo_scope::GripVacuumCapability Class Reference	292
11.60.1 Detailed Description	293
11.60.2 Constructor & Destructor Documentation	293
11.60.2.1 ~GripVacuumCapability()	293
11.60.3 Member Function Documentation	293
11.60.3.1 updateCapability()	293
11.61 arcs::aubo_scope::HaltNode Class Reference	294
11.61.1 Detailed Description	295
11.61.2 Constructor & Destructor Documentation	296
11.61.2.1 HaltNode() [1/3]	296
11.61.2.2 HaltNode() [2/3]	296
11.61.2.3 ~HaltNode()	297
11.61.2.4 HaltNode() [3/3]	297
11.61.3 Friends And Related Symbol Documentation	297
11.61.3.1 DataSwitch	297
11.61.4 Member Data Documentation	297
11.61.4.1 d__	297
11.62 arcs::aubo_scope::IfNode Class Reference	297
11.62.1 Detailed Description	299
11.62.2 Constructor & Destructor Documentation	299
11.62.2.1 IfNode() [1/3]	299
11.62.2.2 IfNode() [2/3]	299
11.62.2.3 ~IfNode()	300

11.62.2.4 IfNode() [3/3]	300
11.62.3 Member Function Documentation	300
11.62.3.1 getExpression()	300
11.62.3.2 setExpression()	300
11.62.4 Friends And Related Symbol Documentation	301
11.62.4.1 DataSwitch	301
11.62.5 Member Data Documentation	301
11.62.5.1 d_	301
11.63 arcs::aubo_scope::InputIoFilter Struct Reference	301
11.63.1 Detailed Description	301
11.63.2 Member Function Documentation	302
11.63.2.1 operator()()	302
11.64 arcs::aubo_scope::InputValidator Struct Reference	302
11.64.1 Detailed Description	302
11.64.2 Member Function Documentation	303
11.64.2.1 getMessage()	303
11.64.2.2 isValid()	303
11.65 arcs::aubo_scope::InstallationApi Class Reference	304
11.65.1 Detailed Description	305
11.65.2 Constructor & Destructor Documentation	306
11.65.2.1 InstallationApi() [1/3]	306
11.65.2.2 InstallationApi() [2/3]	306
11.65.2.3 ~InstallationApi()	307
11.65.2.4 InstallationApi() [3/3]	307
11.65.3 Friends And Related Symbol Documentation	307
11.65.3.1 DataSwitch	307
11.65.4 Member Data Documentation	307
11.65.4.1 d_	307
11.66 arcs::aubo_scope::InstallationApiProvider Class Reference	307
11.66.1 Detailed Description	308
11.66.2 Constructor & Destructor Documentation	308
11.66.2.1 InstallationApiProvider() [1/3]	308
11.66.2.2 InstallationApiProvider() [2/3]	309
11.66.2.3 ~InstallationApiProvider()	309
11.66.2.4 InstallationApiProvider() [3/3]	309
11.66.3 Member Function Documentation	309
11.66.3.1 getInstallationApi()	309
11.66.3.2 getSystemApi()	309
11.66.3.3 getUserInterfaceApi()	310
11.66.4 Friends And Related Symbol Documentation	310
11.66.4.1 DataSwitch	310
11.66.5 Member Data Documentation	310
11.66.5.1 d_	310

11.67 arcs::aubo_scope::InstallationContributionConfiguration Class Reference	310
11.67.1 Detailed Description	310
11.67.2 Constructor & Destructor Documentation	311
11.67.2.1 ~InstallationContributionConfiguration()	311
11.68 arcs::aubo_scope::InstallationCreationContext Class Reference	311
11.68.1 Detailed Description	311
11.68.2 Member Enumeration Documentation	311
11.68.2.1 NodeCreationType	311
11.68.3 Constructor & Destructor Documentation	312
11.68.3.1 InstallationCreationContext() [1/3]	312
11.68.3.2 InstallationCreationContext() [2/3]	313
11.68.3.3 ~InstallationCreationContext()	313
11.68.3.4 InstallationCreationContext() [3/3]	313
11.68.4 Member Function Documentation	313
11.68.4.1 getCreatedVersion()	313
11.68.4.2 getLastSavedVersion()	313
11.68.4.3 getNodeCreationType()	313
11.68.5 Friends And Related Symbol Documentation	313
11.68.5.1 DataSwitch	313
11.68.6 Member Data Documentation	314
11.68.6.1 d_	314
11.69 arcs::aubo_scope::InstallationNodeContribution Class Reference	314
11.69.1 Detailed Description	314
11.69.2 Constructor & Destructor Documentation	314
11.69.2.1 ~InstallationNodeContribution()	314
11.69.3 Member Function Documentation	314
11.69.3.1 closeView()	314
11.69.3.2 generateScript()	314
11.69.3.3 openView()	315
11.70 arcs::aubo_scope::InstallationNodeService Class Reference	315
11.70.1 Detailed Description	315
11.70.2 Constructor & Destructor Documentation	315
11.70.2.1 ~InstallationNodeService()	315
11.70.3 Member Function Documentation	316
11.70.3.1 configureContribution()	316
11.70.3.2 createInstallationNode()	316
11.70.3.3 createView()	317
11.70.3.4 getIcon()	317
11.70.3.5 getTitle()	317
11.71 arcs::aubo_scope::InstallationNodeView Class Reference	318
11.71.1 Detailed Description	318
11.71.2 Constructor & Destructor Documentation	318
11.71.2.1 ~InstallationNodeView()	318

11.71.3 Member Function Documentation	318
11.71.3.1 buildUi()	318
11.72 arcs::aubo_scope::IntegerRangeValidator Struct Reference	319
11.72.1 Detailed Description	320
11.72.2 Constructor & Destructor Documentation	320
11.72.2.1 IntegerRangeValidator() [1/2]	320
11.72.2.2 IntegerRangeValidator() [2/2]	320
11.72.3 Member Function Documentation	320
11.72.3.1 getMessage()	320
11.72.3.2 isValid()	321
11.72.3.3 updateValue()	321
11.72.4 Member Data Documentation	321
11.72.4.1 max_value__	321
11.72.4.2 min_value__	321
11.72.4.3 range_func__	322
11.73 arcs::aubo_scope::IntegerRegister Class Reference	322
11.73.1 Detailed Description	324
11.73.2 Constructor & Destructor Documentation	324
11.73.2.1 ~IntegerRegister()	324
11.74 arcs::aubo_scope::Io Class Reference	324
11.74.1 Detailed Description	325
11.74.2 Member Enumeration Documentation	325
11.74.2.1 InterfaceType	325
11.74.2.2 IoType	326
11.74.3 Constructor & Destructor Documentation	326
11.74.3.1 Io() [1/3]	326
11.74.3.2 Io() [2/3]	327
11.74.3.3 ~Io()	327
11.74.3.4 Io() [3/3]	328
11.74.4 Member Function Documentation	328
11.74.4.1 getAction()	328
11.74.4.2 getDefaultName()	328
11.74.4.3 getDisplayName()	328
11.74.4.4 getInterfaceType()	328
11.74.4.5 getType()	329
11.74.4.6 getValueStr()	329
11.74.4.7 hashCode()	329
11.74.4.8 isInput()	329
11.74.4.9 isResolvable()	329
11.74.5 Friends And Related Symbol Documentation	330
11.74.5.1 DataSwitch	330
11.74.6 Member Data Documentation	330
11.74.6.1 d__	330

11.75 arcs::aubo_scope::IoConfigure Struct Reference	330
11.75.1 Detailed Description	331
11.75.2 Member Data Documentation	331
11.75.2.1 function_lock	331
11.75.2.2 functions	331
11.75.2.3 label	331
11.75.2.4 maxs	331
11.75.2.5 mins	331
11.75.2.6 names	332
11.75.2.7 num	332
11.75.2.8 safety_functions	332
11.75.2.9 safety_functions_fixed	332
11.75.2.10 sequences	332
11.75.2.11 types	332
11.76 arcs::aubo_scope::IoModel Class Reference	332
11.76.1 Detailed Description	333
11.76.2 Constructor & Destructor Documentation	333
11.76.2.1 IoModel() [1/3]	333
11.76.2.2 IoModel() [2/3]	334
11.76.2.3 ~IoModel()	334
11.76.2.4 IoModel() [3/3]	335
11.76.3 Member Function Documentation	335
11.76.3.1 getAll()	335
11.76.3.2 getIo()	335
11.76.3.3 getIos()	335
11.76.4 Friends And Related Symbol Documentation	336
11.76.4.1 DataSwitch	336
11.76.5 Member Data Documentation	336
11.76.5.1 d_	336
11.77 arcs::aubo_scope::KeyboardManager Class Reference	336
11.77.1 Detailed Description	338
11.77.2 Constructor & Destructor Documentation	338
11.77.2.1 KeyboardManager() [1/3]	338
11.77.2.2 KeyboardManager() [2/3]	339
11.77.2.3 ~KeyboardManager()	339
11.77.2.4 KeyboardManager() [3/3]	339
11.77.3 Member Function Documentation	339
11.77.3.1 registerCodeEditorKeyboard() [1/2]	339
11.77.3.2 registerCodeEditorKeyboard() [2/2]	339
11.77.3.3 registerDoubleKeyboard()	340
11.77.3.4 registerExpressionKeyboard()	340
11.77.3.5 registerIntegerKeyboard()	340
11.77.3.6 registerIPAddressKeyboard()	340

11.77.3.7 registerNumberKeyboard()	340
11.77.3.8 registerPasswordKeyboard() [1/2]	340
11.77.3.9 registerPasswordKeyboard() [2/2]	341
11.77.3.10 registerPositiveDoubleKeyboard()	341
11.77.3.11 registerPositiveIntegerKeyboard()	341
11.77.3.12 registerStandardKeyboard() [1/2]	341
11.77.3.13 registerStandardKeyboard() [2/2]	341
11.77.3.14 registerStringKeyboard() [1/2]	342
11.77.3.15 registerStringKeyboard() [2/2]	342
11.77.4 Friends And Related Symbol Documentation	342
11.77.4.1 DataSwitch	342
11.77.5 Member Data Documentation	342
11.77.5.1 d__	342
11.78 arcs::aubo_scope::LoopNode Class Reference	343
11.78.1 Detailed Description	344
11.78.2 Member Enumeration Documentation	344
11.78.2.1 ConfigType	344
11.78.3 Constructor & Destructor Documentation	345
11.78.3.1 LoopNode() [1/3]	345
11.78.3.2 LoopNode() [2/3]	346
11.78.3.3 ~LoopNode()	346
11.78.3.4 LoopNode() [3/3]	346
11.78.4 Member Function Documentation	346
11.78.4.1 getConfigType()	346
11.78.4.2 getExpression()	346
11.78.4.3 getLoopCount()	346
11.78.4.4 getLoopCountVariable()	347
11.78.4.5 isEvaluateContinously()	347
11.78.4.6 setConfigType()	347
11.78.4.7 setEvaluateContinously()	347
11.78.4.8 setExpression()	347
11.78.4.9 setLoopCount()	347
11.78.4.10 setLoopCountVariable()	348
11.78.5 Friends And Related Symbol Documentation	348
11.78.5.1 DataSwitch	348
11.78.6 Member Data Documentation	348
11.78.6.1 d__	348
11.79 arcs::aubo_scope::ModbusIo Class Reference	349
11.79.1 Detailed Description	351
11.79.2 Constructor & Destructor Documentation	351
11.79.2.1 ModbusIo() [1/3]	351
11.79.2.2 ModbusIo() [2/3]	352
11.79.2.3 ~ModbusIo()	352

11.79.2.4 ModbusIo() [3/3]	352
11.79.3 Member Function Documentation	352
11.79.3.1 getIpAddress()	352
11.79.3.2 getSignalAddress()	352
11.79.3.3 getValue()	353
11.79.3.4 isSlave()	353
11.79.3.5 setValue()	353
11.79.4 Friends And Related Symbol Documentation	353
11.79.4.1 DataSwitch	353
11.79.5 Member Data Documentation	354
11.79.5.1 d_	354
11.80 arcs::aubo_scope::ModbusIoFilter Struct Reference	354
11.80.1 Detailed Description	354
11.80.2 Constructor & Destructor Documentation	354
11.80.2.1 ModbusIoFilter()	354
11.80.3 Member Function Documentation	355
11.80.3.1 operator()()	355
11.80.4 Member Data Documentation	355
11.80.4.1 is_input_	355
11.80.4.2 type_	355
11.81 arcs::aubo_scope::MoveNode Class Reference	355
11.81.1 Detailed Description	357
11.81.2 Member Enumeration Documentation	357
11.81.2.1 ConfigType	357
11.81.2.2 FeatureSelectionType	357
11.81.2.3 TcpSelectionType	358
11.81.3 Constructor & Destructor Documentation	358
11.81.3.1 MoveNode() [1/3]	358
11.81.3.2 MoveNode() [2/3]	359
11.81.3.3 ~MoveNode()	359
11.81.3.4 MoveNode() [3/3]	359
11.81.4 Member Function Documentation	359
11.81.4.1 getBlend()	359
11.81.4.2 getConfigType()	360
11.81.4.3 getFeature()	360
11.81.4.4 getJointAcceleration()	360
11.81.4.5 getJointSpeed()	360
11.81.4.6 getTCPSelection()	360
11.81.4.7 getTcpSelectionType()	360
11.81.4.8 getToolAcceleration()	360
11.81.4.9 getToolSpeed()	360
11.81.4.10 getUseJoints()	360
11.81.4.11 setBlend()	360

11.81.4.12	setConfigType()	361
11.81.4.13	setFeature() [1/2]	361
11.81.4.14	setFeature() [2/2]	361
11.81.4.15	setJointAcceleration()	361
11.81.4.16	setJointSpeed()	362
11.81.4.17	setTCPSelection()	362
11.81.4.18	setTcpSelectionType()	362
11.81.4.19	setToolAcceleration()	362
11.81.4.20	setToolSpeed()	363
11.81.4.21	setUseJoints()	363
11.81.5	Friends And Related Symbol Documentation	363
11.81.5.1	DataSwitch	363
11.81.6	Member Data Documentation	363
11.81.6.1	d__	363
11.82	arcs::aubo_scope::OfflineTrackNode Class Reference	364
11.82.1	Detailed Description	365
11.82.2	Constructor & Destructor Documentation	365
11.82.2.1	OfflineTrackNode() [1/3]	365
11.82.2.2	OfflineTrackNode() [2/3]	366
11.82.2.3	~OfflineTrackNode()	366
11.82.2.4	OfflineTrackNode() [3/3]	366
11.82.3	Friends And Related Symbol Documentation	366
11.82.3.1	DataSwitch	366
11.82.4	Member Data Documentation	366
11.82.4.1	d__	366
11.83	arcs::aubo_scope::OutputIoFilter Struct Reference	367
11.83.1	Detailed Description	367
11.83.2	Member Function Documentation	367
11.83.2.1	operator>()()	367
11.84	arcs::aubo_scope::PalletNode Class Reference	367
11.84.1	Detailed Description	369
11.84.2	Constructor & Destructor Documentation	369
11.84.2.1	PalletNode() [1/3]	369
11.84.2.2	PalletNode() [2/3]	369
11.84.2.3	~PalletNode()	370
11.84.2.4	PalletNode() [3/3]	370
11.84.3	Friends And Related Symbol Documentation	370
11.84.3.1	DataSwitch	370
11.84.4	Member Data Documentation	370
11.84.4.1	d__	370
11.85	arcs::aubo_scope::Payload Class Reference	371
11.85.1	Detailed Description	371
11.85.2	Member Enumeration Documentation	371

11.85.2.1 ParametersWarning	371
11.85.3 Constructor & Destructor Documentation	372
11.85.3.1 Payload() [1/3]	372
11.85.3.2 Payload() [2/3]	373
11.85.3.3 ~Payload()	373
11.85.3.4 Payload() [3/3]	373
11.85.4 Member Function Documentation	373
11.85.4.1 getCenterOfGravity()	373
11.85.4.2 getDisplayName()	374
11.85.4.3 getInertiaMatrix()	374
11.85.4.4 getMass()	374
11.85.4.5 getParametersWarning()	374
11.85.4.6 hashCode()	375
11.85.4.7 isResolvable()	375
11.85.4.8 isUsable()	375
11.85.5 Friends And Related Symbol Documentation	375
11.85.5.1 DataSwitch	375
11.85.6 Member Data Documentation	376
11.85.6.1 d_	376
11.86 arcs::aubo_scope::PayloadContributionModel Class Reference	376
11.86.1 Detailed Description	377
11.86.2 Constructor & Destructor Documentation	377
11.86.2.1 PayloadContributionModel() [1/3]	377
11.86.2.2 PayloadContributionModel() [2/3]	378
11.86.2.3 ~PayloadContributionModel()	378
11.86.2.4 PayloadContributionModel() [3/3]	378
11.86.3 Member Function Documentation	378
11.86.3.1 addPayload() [1/2]	378
11.86.3.2 addPayload() [2/2]	379
11.86.3.3 getPayload()	381
11.86.3.4 removePayload()	381
11.86.3.5 updatePayload() [1/2]	382
11.86.3.6 updatePayload() [2/2]	382
11.86.4 Friends And Related Symbol Documentation	383
11.86.4.1 DataSwitch	383
11.86.5 Member Data Documentation	383
11.86.5.1 d_	383
11.87 arcs::aubo_scope::PayloadModel Class Reference	384
11.87.1 Detailed Description	384
11.87.2 Constructor & Destructor Documentation	385
11.87.2.1 PayloadModel() [1/3]	385
11.87.2.2 PayloadModel() [2/3]	385
11.87.2.3 ~PayloadModel()	386

11.87.2.4 PayloadModel() [3/3]	386
11.87.3 Member Function Documentation	386
11.87.3.1 getCurrentPayload()	386
11.87.3.2 getDefaultPayload()	386
11.87.3.3 getPayload()	386
11.87.3.4 getPayloads()	387
11.87.4 Friends And Related Symbol Documentation	387
11.87.4.1 DataSwitch	387
11.87.5 Member Data Documentation	387
11.87.5.1 d_	387
11.88 arcs::aubo_scope::PersistedVariable Class Reference	388
11.88.1 Detailed Description	389
11.88.2 Constructor & Destructor Documentation	390
11.88.2.1 PersistedVariable() [1/3]	390
11.88.2.2 PersistedVariable() [2/3]	390
11.88.2.3 ~PersistedVariable()	391
11.88.2.4 PersistedVariable() [3/3]	391
11.88.3 Member Function Documentation	391
11.88.3.1 isResolvable()	391
11.88.4 Friends And Related Symbol Documentation	391
11.88.4.1 DataSwitch	391
11.88.5 Member Data Documentation	391
11.88.5.1 d_	391
11.89 arcs::aubo_scope::PersistedVariableFilter Struct Reference	392
11.89.1 Detailed Description	392
11.89.2 Member Function Documentation	392
11.89.2.1 operator>()	392
11.90 arcs::aubo_scope::PopupManager Class Reference	392
11.90.1 Detailed Description	393
11.90.2 Constructor & Destructor Documentation	393
11.90.2.1 PopupManager() [1/3]	393
11.90.2.2 PopupManager() [2/3]	394
11.90.2.3 ~PopupManager()	394
11.90.2.4 PopupManager() [3/3]	394
11.90.3 Member Function Documentation	394
11.90.3.1 isPopupCountZero()	394
11.90.3.2 notifyPopupHidden()	394
11.90.3.3 notifyPopupShown()	394
11.90.3.4 popupCount()	394
11.90.3.5 setPopupCountChangedCallback()	395
11.90.4 Friends And Related Symbol Documentation	395
11.90.4.1 DefaultSingletonTraits< PopupManager >	395
11.90.5 Member Data Documentation	395

11.90.5.1 d_	395
11.91 arcs::aubo_scope::PopupNode Class Reference	396
11.91.1 Detailed Description	397
11.91.2 Member Enumeration Documentation	397
11.91.2.1 Type	397
11.91.3 Constructor & Destructor Documentation	398
11.91.3.1 PopupNode() [1/3]	398
11.91.3.2 PopupNode() [2/3]	398
11.91.3.3 ~PopupNode()	399
11.91.3.4 PopupNode() [3/3]	399
11.91.4 Member Function Documentation	399
11.91.4.1 getMessage()	399
11.91.4.2 getType()	399
11.91.4.3 setMessage()	399
11.91.4.4 setType()	399
11.91.5 Friends And Related Symbol Documentation	400
11.91.5.1 DataSwitch	400
11.91.6 Member Data Documentation	400
11.91.6.1 d_	400
11.92 arcs::aubo_scope::PrepareToStartScrewdriverParameters Class Reference	400
11.92.1 Detailed Description	401
11.92.2 Constructor & Destructor Documentation	401
11.92.2.1 ~PrepareToStartScrewdriverParameters()	401
11.93 arcs::aubo_scope::Profinet Class Reference	401
11.93.1 Detailed Description	403
11.93.2 Constructor & Destructor Documentation	403
11.93.2.1 Profinet() [1/3]	403
11.93.2.2 Profinet() [2/3]	404
11.93.2.3 ~Profinet()	404
11.93.2.4 Profinet() [3/3]	405
11.93.3 Member Function Documentation	405
11.93.3.1 getBitOffset()	405
11.93.3.2 getByteOffset()	405
11.93.3.3 getValue()	405
11.93.3.4 isSlave()	405
11.93.3.5 setValue()	405
11.93.4 Friends And Related Symbol Documentation	406
11.93.4.1 DataSwitch	406
11.93.5 Member Data Documentation	406
11.93.5.1 d_	406
11.94 arcs::aubo_scope::ProfinetFilter Struct Reference	406
11.94.1 Detailed Description	407
11.94.2 Constructor & Destructor Documentation	407

11.94.2.1 ProfinetFilter()	407
11.94.3 Member Function Documentation	407
11.94.3.1 operator>()	407
11.94.4 Member Data Documentation	407
11.94.4.1 is_input_	407
11.94.4.2 type_	407
11.95 arcs::aubo_scope::ProgramApi Class Reference	408
11.95.1 Detailed Description	409
11.95.2 Constructor & Destructor Documentation	410
11.95.2.1 ProgramApi() [1/3]	410
11.95.2.2 ProgramApi() [2/3]	410
11.95.2.3 ~ProgramApi()	411
11.95.2.4 ProgramApi() [3/3]	411
11.95.3 Friends And Related Symbol Documentation	411
11.95.3.1 DataSwitch	411
11.95.4 Member Data Documentation	411
11.95.4.1 d_	411
11.96 arcs::aubo_scope::ProgramApiProvider Class Reference	411
11.96.1 Detailed Description	412
11.96.2 Constructor & Destructor Documentation	412
11.96.2.1 ProgramApiProvider() [1/3]	412
11.96.2.2 ProgramApiProvider() [2/3]	413
11.96.2.3 ~ProgramApiProvider()	413
11.96.2.4 ProgramApiProvider() [3/3]	413
11.96.3 Member Function Documentation	413
11.96.3.1 getProgramApi()	413
11.96.3.2 getSystemApi()	413
11.96.3.3 getUserInterfaceApi()	414
11.96.4 Friends And Related Symbol Documentation	414
11.96.4.1 DataSwitch	414
11.96.5 Member Data Documentation	414
11.96.5.1 d_	414
11.97 arcs::aubo_scope::ProgramContributionConfiguration Class Reference	414
11.97.1 Detailed Description	415
11.97.2 Constructor & Destructor Documentation	416
11.97.2.1 ProgramContributionConfiguration() [1/3]	416
11.97.2.2 ProgramContributionConfiguration() [2/3]	416
11.97.2.3 ~ProgramContributionConfiguration()	417
11.97.2.4 ProgramContributionConfiguration() [3/3]	417
11.97.3 Member Function Documentation	417
11.97.3.1 setAllowBreakpointOnChildNodesInSubtree()	417
11.97.3.2 setAllowBreakpointOnNode()	417
11.97.3.3 setAllowInsertOnlyIntoNodes()	417

11.97.3.4	setAllowStartFromChildNodesInSubtree()	418
11.97.3.5	setAllowStartFromNode()	418
11.97.3.6	setChildrenAllowed()	418
11.97.3.7	setChildrenAllowedOnly()	419
11.97.3.8	setDeprecated()	419
11.97.3.9	setDisplayOrderId()	419
11.97.3.10	setUserInsertable()	420
11.97.4	Friends And Related Symbol Documentation	420
11.97.4.1	DataSwitch	420
11.97.5	Member Data Documentation	420
11.97.5.1	d_	420
11.98	arcs::aubo_scope::ProgramCreationContext Class Reference	421
11.98.1	Detailed Description	421
11.98.2	Member Enumeration Documentation	421
11.98.2.1	NodeCreationType	421
11.98.3	Constructor & Destructor Documentation	422
11.98.3.1	ProgramCreationContext() [1/3]	422
11.98.3.2	ProgramCreationContext() [2/3]	423
11.98.3.3	~ProgramCreationContext()	423
11.98.3.4	ProgramCreationContext() [3/3]	423
11.98.4	Member Function Documentation	423
11.98.4.1	getCreatedVersion()	423
11.98.4.2	getLastSavedVersion()	423
11.98.4.3	getNodeCreationType()	423
11.98.5	Friends And Related Symbol Documentation	424
11.98.5.1	DataSwitch	424
11.98.6	Member Data Documentation	424
11.98.6.1	d_	424
11.99	arcs::aubo_scope::ProgramHomeNode Class Reference	424
11.99.1	Detailed Description	426
11.99.2	Constructor & Destructor Documentation	426
11.99.2.1	ProgramHomeNode() [1/3]	426
11.99.2.2	ProgramHomeNode() [2/3]	426
11.99.2.3	~ProgramHomeNode()	427
11.99.2.4	ProgramHomeNode() [3/3]	427
11.99.3	Friends And Related Symbol Documentation	427
11.99.3.1	DataSwitch	427
11.99.4	Member Data Documentation	427
11.99.4.1	d_	427
11.100	arcs::aubo_scope::ProgramModel Class Reference	427
11.100.1	Detailed Description	428
11.100.2	Constructor & Destructor Documentation	429
11.100.2.1	ProgramModel() [1/3]	429

11.100.2.2 ProgramModel() [2/3]	429
11.100.2.3 ~ProgramModel()	430
11.100.2.4 ProgramModel() [3/3]	430
11.100.3 Member Function Documentation	430
11.100.3.1 getProgramNodeFactory()	430
11.100.3.2 getTreeNode()	430
11.100.4 Friends And Related Symbol Documentation	430
11.100.4.1 DataSwitch	430
11.100.5 Member Data Documentation	431
11.100.5.1 d_	431
11.101 arcs::aubo_scope::ProgramNode Class Reference	431
11.101.1 Detailed Description	433
11.101.2 Constructor & Destructor Documentation	433
11.101.2.1 ProgramNode() [1/3]	433
11.101.2.2 ProgramNode() [2/3]	434
11.101.2.3 ~ProgramNode()	434
11.101.2.4 ProgramNode() [3/3]	434
11.101.3 Member Function Documentation	434
11.101.3.1 getProgramNodeContribution()	434
11.101.4 Friends And Related Symbol Documentation	434
11.101.4.1 DataSwitch	434
11.101.5 Member Data Documentation	435
11.101.5.1 d_	435
11.102 arcs::aubo_scope::ProgramNodeContribution Class Reference	435
11.102.1 Detailed Description	435
11.102.2 Constructor & Destructor Documentation	435
11.102.2.1 ~ProgramNodeContribution()	435
11.102.3 Member Function Documentation	436
11.102.3.1 closeView()	436
11.102.3.2 generateScript()	436
11.102.3.3 getAction()	436
11.102.3.4 getTitle()	436
11.102.3.5 isDefined()	437
11.102.3.6 openView()	437
11.102.3.7 postAction()	437
11.103 arcs::aubo_scope::ProgramNodeFactory Class Reference	438
11.103.1 Detailed Description	439
11.103.2 Constructor & Destructor Documentation	440
11.103.2.1 ProgramNodeFactory() [1/3]	440
11.103.2.2 ProgramNodeFactory() [2/3]	440
11.103.2.3 ~ProgramNodeFactory()	441
11.103.2.4 ProgramNodeFactory() [3/3]	441
11.103.3 Member Function Documentation	441

11.103.3.1	createAssemblyNode()	441
11.103.3.2	createAssignmentNode()	441
11.103.3.3	createBreakNode()	441
11.103.3.4	createCallNode()	441
11.103.3.5	createCaseNode()	441
11.103.3.6	createCircleMoveNode()	441
11.103.3.7	createCommentNode()	441
11.103.3.8	createConveyorTrackingNode()	441
11.103.3.9	createDefaultCaseNode()	442
11.103.3.10	createDirectionNode()	442
11.103.3.11	createElseIfNode()	442
11.103.3.12	createElseNode()	442
11.103.3.13	createFolderNode()	442
11.103.3.14	createForceNode()	442
11.103.3.15	createForceOptionNode()	442
11.103.3.16	createGripperNode()	442
11.103.3.17	createHaltNode()	443
11.103.3.18	createIfNode()	443
11.103.3.19	createLoopNode()	443
11.103.3.20	createMoveNode()	443
11.103.3.21	createMoveNodeNoTemplate()	444
11.103.3.22	createOfflineTrackNode()	444
11.103.3.23	createPalletNode()	444
11.103.3.24	createPopupNode()	444
11.103.3.25	createProgramHomeNode()	444
11.103.3.26	createRecordTrackNode()	444
11.103.3.27	createScrewdrivingNode()	444
11.103.3.28	createScriptNode()	444
11.103.3.29	createSeekNode()	444
11.103.3.30	createSetNode()	445
11.103.3.31	createSetPayloadNode()	445
11.103.3.32	createSubProgNode()	445
11.103.3.33	createSwitchNode()	445
11.103.3.34	createThreadNode()	445
11.103.3.35	createTimerNode()	445
11.103.3.36	createUntilNode()	445
11.103.3.37	createUserProgramNode()	445
11.103.3.38	createVisualLocalizationProgramNode()	446
11.103.3.39	createWaitNode()	446
11.103.3.40	createWaypointNode() [1/2]	446
11.103.3.41	createWaypointNode() [2/2]	446
11.103.4	Friends And Related Symbol Documentation	447
11.103.4.1	DataSwitch	447

11.103.5 Member Data Documentation	447
11.103.5.1 d_	447
11.104 arcs::aubo_scope::ProgramNodeService Class Reference	447
11.104.1 Detailed Description	448
11.104.2 Constructor & Destructor Documentation	448
11.104.2.1 ~ProgramNodeService()	448
11.104.3 Member Function Documentation	448
11.104.3.1 configureContribution()	448
11.104.3.2 createNode()	448
11.104.3.3 createView()	449
11.104.3.4 getIcon()	449
11.104.3.5 getId()	450
11.104.3.6 getTitle()	450
11.105 arcs::aubo_scope::ProgramNodeView Class Reference	450
11.105.1 Detailed Description	450
11.105.2 Constructor & Destructor Documentation	451
11.105.2.1 ~ProgramNodeView()	451
11.105.3 Member Function Documentation	451
11.105.3.1 buildUi()	451
11.106 arcs::aubo_scope::ProgramSelectionParameters Class Reference	451
11.106.1 Detailed Description	451
11.106.2 Constructor & Destructor Documentation	452
11.106.2.1 ~ProgramSelectionParameters()	452
11.106.3 Member Function Documentation	452
11.106.3.1 getScrewdriverOperationType()	452
11.106.3.2 getScrewdriverProgram()	452
11.107 arcs::aubo_scope::RecordTrackNode Class Reference	453
11.107.1 Detailed Description	454
11.107.2 Constructor & Destructor Documentation	454
11.107.2.1 RecordTrackNode() [1/3]	454
11.107.2.2 RecordTrackNode() [2/3]	455
11.107.2.3 ~RecordTrackNode()	455
11.107.2.4 RecordTrackNode() [3/3]	455
11.107.3 Friends And Related Symbol Documentation	455
11.107.3.1 DataSwitch	455
11.107.4 Member Data Documentation	455
11.107.4.1 d_	455
11.108 arcs::aubo_scope::Register Class Reference	456
11.108.1 Detailed Description	457
11.108.2 Constructor & Destructor Documentation	457
11.108.2.1 ~Register()	457
11.109 arcs::aubo_scope::RegisteredCapabilities Class Reference	457
11.109.1 Detailed Description	458

11.109.2 Constructor & Destructor Documentation	458
11.109.2.1 ~RegisteredCapabilities()	458
11.109.3 Member Function Documentation	458
11.109.3.1 getGripForceCapability()	458
11.109.3.2 getGripVacuumCapability()	459
11.109.3.3 getSpeedCapability()	459
11.109.3.4 getWidthCapability()	459
11.110 arcs::aubo_scope::ReleaseActionParameters Class Reference	460
11.110.1 Detailed Description	460
11.110.2 Constructor & Destructor Documentation	460
11.110.2.1 ~ReleaseActionParameters()	460
11.110.3 Member Function Documentation	460
11.110.3.1 getSpeed()	460
11.110.3.2 getWidth()	461
11.110.3.3 isReleaseDetectionEnabled()	461
11.111 arcs::aubo_scope::RenderInterface Class Reference	462
11.111.1 Detailed Description	464
11.111.2 Member Enumeration Documentation	464
11.111.2.1 ShowOptions	464
11.111.3 Constructor & Destructor Documentation	465
11.111.3.1 RenderInterface() [1/3]	465
11.111.3.2 RenderInterface() [2/3]	466
11.111.3.3 ~RenderInterface()	466
11.111.3.4 RenderInterface() [3/3]	466
11.111.4 Member Function Documentation	466
11.111.4.1 addCoordinate()	466
11.111.4.2 addCube() [1/4]	466
11.111.4.3 addCube() [2/4]	467
11.111.4.4 addCube() [3/4]	468
11.111.4.5 addCube() [4/4]	468
11.111.4.6 addLine() [1/2]	469
11.111.4.7 addLine() [2/2]	469
11.111.4.8 addMesh()	470
11.111.4.9 addPlane() [1/2]	470
11.111.4.10 addPlane() [2/2]	471
11.111.4.11 addPointCloud()	471
11.111.4.12 addWaypoint()	471
11.111.4.13 clearCubes()	472
11.111.4.14 clearLines()	472
11.111.4.15 clearMeshes()	472
11.111.4.16 clearPlanes()	473
11.111.4.17 clearPointCloud()	473
11.111.4.18 clearWaypoints()	473

11.111.4.19	getMeshes()	473
11.111.4.20	hasCoordinate()	473
11.111.4.21	hasPointCloud()	474
11.111.4.22	hasWaypoint()	474
11.111.4.23	removeCoordinate()	475
11.111.4.24	removeCube()	475
11.111.4.25	removeLine()	475
11.111.4.26	removeMesh()	476
11.111.4.27	removePlane()	476
11.111.4.28	removePointCloud()	476
11.111.4.29	removeWaypoint()	477
11.111.4.30	setBlendTrack() [1/2]	477
11.111.4.31	setBlendTrack() [2/2]	477
11.111.4.32	setMoveTrack() [1/2]	478
11.111.4.33	setMoveTrack() [2/2]	478
11.111.4.34	setRenderEmbedded()	478
11.111.4.35	setShowOptions()	479
11.111.5	Friends And Related Symbol Documentation	479
11.111.5.1	DataSwitch	479
11.111.6	Member Data Documentation	479
11.111.6.1	d_	479
11.112	arcs::aubo_scope::ScrewdriverApiProvider Class Reference	480
11.112.1	Detailed Description	480
11.112.2	Constructor & Destructor Documentation	480
11.112.2.1	~ScrewdriverApiProvider()	480
11.112.3	Member Function Documentation	480
11.112.3.1	getIOModel()	480
11.112.3.2	getSystemAPI()	480
11.113	arcs::aubo_scope::ScrewdriverConfiguration Class Reference	481
11.113.1	Detailed Description	481
11.113.2	Constructor & Destructor Documentation	481
11.113.2.1	~ScrewdriverConfiguration()	481
11.113.3	Member Function Documentation	481
11.113.3.1	addScrewdriverProgram()	481
11.113.3.2	addScrewdriverPrograms()	482
11.113.3.3	registerDriveScrewNotOKCapability()	483
11.113.3.4	registerDriveScrewOKCapability()	484
11.113.3.5	registerFeedScrewCapability()	485
11.113.3.6	registerOperationTypeCapability()	485
11.113.3.7	registerPrepareToStartScrewdriverCapability()	486
11.113.3.8	registerProgramSelectionCapability()	487
11.113.3.9	registerScrewdriverReadyCapability()	487
11.114	arcs::aubo_scope::ScrewdriverContribution Class Reference	488

11.114.1 Detailed Description	489
11.114.2 Constructor & Destructor Documentation	489
11.114.2.1 ~ScrewdriverContribution()	489
11.114.3 Member Function Documentation	489
11.114.3.1 configureScrewdriver()	489
11.114.3.2 generateStartScrewdriverScript()	490
11.114.3.3 generateStopScrewdriverScript()	490
11.114.3.4 getTitle()	491
11.115 arcs::aubo_scope::ScrewdriverParameters Class Reference	491
11.115.1 Detailed Description	491
11.115.2 Constructor & Destructor Documentation	492
11.115.2.1 ~ScrewdriverParameters()	492
11.115.3 Member Function Documentation	492
11.115.3.1 getScrewdriverOperationType()	492
11.115.3.2 getScrewdriverProgram()	492
11.116 arcs::aubo_scope::ScrewdriverProgram Class Reference	493
11.116.1 Detailed Description	493
11.116.2 Constructor & Destructor Documentation	493
11.116.2.1 ~ScrewdriverProgram()	493
11.116.3 Member Function Documentation	493
11.116.3.1 getDisplayName()	493
11.116.3.2 getId()	494
11.117 arcs::aubo_scope::ScrewdriverReadyParameters Class Reference	494
11.117.1 Detailed Description	494
11.117.2 Constructor & Destructor Documentation	494
11.117.2.1 ~ScrewdriverReadyParameters()	494
11.118 arcs::aubo_scope::ScrewdrivingNode Class Reference	495
11.118.1 Detailed Description	496
11.118.2 Constructor & Destructor Documentation	496
11.118.2.1 ScrewdrivingNode() [1/3]	496
11.118.2.2 ScrewdrivingNode() [2/3]	497
11.118.2.3 ~ScrewdrivingNode()	497
11.118.2.4 ScrewdrivingNode() [3/3]	497
11.118.3 Friends And Related Symbol Documentation	497
11.118.3.1 DataSwitch	497
11.118.4 Member Data Documentation	497
11.118.4.1 d_	497
11.119 arcs::aubo_scope::ScriptNode Class Reference	498
11.119.1 Detailed Description	499
11.119.2 Constructor & Destructor Documentation	499
11.119.2.1 ScriptNode() [1/3]	499
11.119.2.2 ScriptNode() [2/3]	500
11.119.2.3 ~ScriptNode()	500

11.119.2.4 ScriptNode() [3/3]	500
11.119.3 Friends And Related Symbol Documentation	500
11.119.3.1 DataSwitch	500
11.119.4 Member Data Documentation	501
11.119.4.1 d_	501
11.120 arcs::aubo_scope::ScriptWriter Class Reference	501
11.120.1 Detailed Description	503
11.120.2 Constructor & Destructor Documentation	503
11.120.2.1 ScriptWriter() [1/3]	503
11.120.2.2 ScriptWriter() [2/3]	504
11.120.2.3 ~ScriptWriter()	504
11.120.2.4 ScriptWriter() [3/3]	504
11.120.3 Member Function Documentation	504
11.120.3.1 anonyFunction()	504
11.120.3.2 appendLine()	504
11.120.3.3 appendRaw()	504
11.120.3.4 appendVectorDouble()	505
11.120.3.5 assign() [1/3]	505
11.120.3.6 assign() [2/3]	505
11.120.3.7 assign() [3/3]	506
11.120.3.8 decreaseIndent()	506
11.120.3.9 defineFunction()	506
11.120.3.10 defineThread()	507
11.120.3.11 elseCondition()	507
11.120.3.12 elseifCondition()	507
11.120.3.13 end()	507
11.120.3.14 forCondition()	507
11.120.3.15 generateScript()	508
11.120.3.16 getResolvedVariable()	508
11.120.3.17 ifCondition()	508
11.120.3.18 ifNotCondition()	508
11.120.3.19 increaseIndent()	509
11.120.3.20 incrementVariable()	509
11.120.3.21 killThread()	509
11.120.3.22 lineFeed()	509
11.120.3.23 note()	509
11.120.3.24 returnMethod()	510
11.120.3.25 runThread()	510
11.120.3.26 setLabel()	510
11.120.3.27 setPayload()	510
11.120.3.28 setRobotIndex()	510
11.120.3.29 setTcp()	511
11.120.3.30 sleep()	511

11.120.3.31	sync()	511
11.120.3.32	whileCondition()	511
11.120.3.33	whileNot()	512
11.120.3.34	whileTrue()	512
11.120.3.35	writeChildren()	512
11.120.4	Friends And Related Symbol Documentation	512
11.120.4.1	DataSwitch	512
11.120.5	Member Data Documentation	513
11.120.5.1	d_	513
11.121	arcs::aubo_scope::SeekNode Class Reference	513
11.121.1	Detailed Description	514
11.121.2	Constructor & Destructor Documentation	514
11.121.2.1	SeekNode() [1/3]	514
11.121.2.2	SeekNode() [2/3]	515
11.121.2.3	~SeekNode()	515
11.121.2.4	SeekNode() [3/3]	516
11.121.3	Friends And Related Symbol Documentation	516
11.121.3.1	DataSwitch	516
11.121.4	Member Data Documentation	516
11.121.4.1	d_	516
11.122	arcs::aubo_scope::SetNode Class Reference	516
11.122.1	Detailed Description	518
11.122.2	Member Enumeration Documentation	518
11.122.2.1	ConfigType	518
11.122.2.2	OutputSelectionType	519
11.122.3	Constructor & Destructor Documentation	520
11.122.3.1	SetNode() [1/3]	520
11.122.3.2	SetNode() [2/3]	520
11.122.3.3	~SetNode()	521
11.122.3.4	SetNode() [3/3]	521
11.122.4	Member Function Documentation	521
11.122.4.1	getAnalogOutput()	521
11.122.4.2	getBoolValueToSet()	521
11.122.4.3	getConfigType()	521
11.122.4.4	getCurrent()	522
11.122.4.5	getExpression()	522
11.122.4.6	getFloatValueToSet()	522
11.122.4.7	getOutput()	522
11.122.4.8	getOutputSelectionType()	522
11.122.4.9	getPulseTime()	522
11.122.4.10	getVoltage()	523
11.122.4.11	setAnalogOutput()	523
11.122.4.12	setBoolValueToSet()	523

11.122.4.13	setConfigType()	523
11.122.4.14	setCurrent()	523
11.122.4.15	setExpression()	523
11.122.4.16	setFloatValueToSet()	523
11.122.4.17	setOutput()	523
11.122.4.18	setPulseTime()	524
11.122.4.19	setVoltage()	524
11.122.5	Friends And Related Symbol Documentation	524
11.122.5.1	DataSwitch	524
11.122.6	Member Data Documentation	524
11.122.6.1	d_	524
11.123	arcs::aubo_scope::SetPayloadNode Class Reference	525
11.123.1	Detailed Description	526
11.123.2	Member Enumeration Documentation	526
11.123.2.1	ConfigType	526
11.123.3	Constructor & Destructor Documentation	527
11.123.3.1	SetPayloadNode() [1/3]	527
11.123.3.2	SetPayloadNode() [2/3]	528
11.123.3.3	~SetPayloadNode()	528
11.123.3.4	SetPayloadNode() [3/3]	528
11.123.4	Member Function Documentation	528
11.123.4.1	getCenterOfGravity()	528
11.123.4.2	getConfigType()	528
11.123.4.3	getMass()	529
11.123.4.4	setCenterOfGravity()	529
11.123.4.5	setConfigType()	529
11.123.4.6	setMass()	529
11.123.5	Friends And Related Symbol Documentation	529
11.123.5.1	DataSwitch	529
11.123.6	Member Data Documentation	529
11.123.6.1	d_	529
11.124	arcs::aubo_scope::SpeedCapability Class Reference	530
11.124.1	Detailed Description	530
11.124.2	Constructor & Destructor Documentation	530
11.124.2.1	~SpeedCapability()	530
11.124.3	Member Function Documentation	530
11.124.3.1	updateCapability()	530
11.125	arcs::aubo_scope::StringLengthValidator Struct Reference	531
11.125.1	Detailed Description	532
11.125.2	Constructor & Destructor Documentation	532
11.125.2.1	StringLengthValidator() [1/2]	532
11.125.2.2	StringLengthValidator() [2/2]	533
11.125.3	Member Function Documentation	533

11.125.3.1	getMessage()	533
11.125.3.2	isValid()	533
11.125.3.3	updateValue()	534
11.125.4	Member Data Documentation	534
11.125.4.1	max_length	534
11.125.4.2	min_length	534
11.125.4.3	range_func	534
11.126	arcs::aubo_scope::SubProgNode Class Reference	534
11.126.1	Detailed Description	536
11.126.2	Constructor & Destructor Documentation	536
11.126.2.1	SubProgNode() [1/3]	536
11.126.2.2	SubProgNode() [2/3]	536
11.126.2.3	~SubProgNode()	537
11.126.2.4	SubProgNode() [3/3]	537
11.126.3	Friends And Related Symbol Documentation	537
11.126.3.1	DataSwitch	537
11.126.4	Member Data Documentation	537
11.126.4.1	d_	537
11.127	arcs::aubo_scope::SwitchNode Class Reference	538
11.127.1	Detailed Description	539
11.127.2	Constructor & Destructor Documentation	539
11.127.2.1	SwitchNode() [1/3]	539
11.127.2.2	SwitchNode() [2/3]	540
11.127.2.3	~SwitchNode()	540
11.127.2.4	SwitchNode() [3/3]	540
11.127.3	Friends And Related Symbol Documentation	540
11.127.3.1	DataSwitch	540
11.127.4	Member Data Documentation	541
11.127.4.1	d_	541
11.128	arcs::aubo_scope::SystemApi Class Reference	541
11.128.1	Detailed Description	543
11.128.2	Constructor & Destructor Documentation	543
11.128.2.1	SystemApi() [1/3]	543
11.128.2.2	SystemApi() [2/3]	544
11.128.2.3	~SystemApi()	544
11.128.2.4	SystemApi() [3/3]	544
11.128.3	Friends And Related Symbol Documentation	544
11.128.3.1	DataSwitch	544
11.128.4	Member Data Documentation	544
11.128.4.1	d_	544
11.129	arcs::aubo_scope::TCP Class Reference	545
11.129.1	Detailed Description	545
11.129.2	Constructor & Destructor Documentation	546

11.129.2.1 TCP() [1/3]	546
11.129.2.2 TCP() [2/3]	546
11.129.2.3 ~TCP()	547
11.129.2.4 TCP() [3/3]	547
11.129.3 Member Function Documentation	547
11.129.3.1 getDisplayName()	547
11.129.3.2 getOffset()	547
11.129.3.3 hashCode()	547
11.129.3.4 isResolvable()	548
11.129.4 Friends And Related Symbol Documentation	548
11.129.4.1 DataSwitch	548
11.129.5 Member Data Documentation	548
11.129.5.1 d_	548
11.130 arcs::aubo_scope::TcpContributionModel Class Reference	548
11.130.1 Detailed Description	549
11.130.2 Constructor & Destructor Documentation	549
11.130.2.1 TcpContributionModel() [1/3]	549
11.130.2.2 TcpContributionModel() [2/3]	550
11.130.2.3 ~TcpContributionModel()	550
11.130.2.4 TcpContributionModel() [3/3]	550
11.130.3 Member Function Documentation	551
11.130.3.1 addTcp()	551
11.130.3.2 getTcp()	551
11.130.3.3 removeTcp()	551
11.130.3.4 updateTcp()	552
11.130.4 Friends And Related Symbol Documentation	552
11.130.4.1 DataSwitch	552
11.130.5 Member Data Documentation	553
11.130.5.1 d_	553
11.131 arcs::aubo_scope::TcpModel Class Reference	553
11.131.1 Detailed Description	554
11.131.2 Constructor & Destructor Documentation	554
11.131.2.1 TcpModel() [1/3]	554
11.131.2.2 TcpModel() [2/3]	555
11.131.2.3 ~TcpModel()	555
11.131.2.4 TcpModel() [3/3]	555
11.131.3 Member Function Documentation	555
11.131.3.1 connectedTo()	555
11.131.3.2 disconnectFrom()	556
11.131.3.3 getAll()	556
11.131.3.4 getDefaultTcp()	556
11.131.3.5 getTcp()	556
11.131.4 Friends And Related Symbol Documentation	557

11.131.4.1 DataSwitch	557
11.131.5 Member Data Documentation	557
11.131.5.1 d_	557
11.132 arcs::aubo_scope::ThreadNode Class Reference	558
11.132.1 Detailed Description	559
11.132.2 Constructor & Destructor Documentation	559
11.132.2.1 ThreadNode() [1/3]	559
11.132.2.2 ThreadNode() [2/3]	560
11.132.2.3 ~ThreadNode()	560
11.132.2.4 ThreadNode() [3/3]	560
11.132.3 Friends And Related Symbol Documentation	560
11.132.3.1 DataSwitch	560
11.132.4 Member Data Documentation	561
11.132.4.1 d_	561
11.133 arcs::aubo_scope::TimerNode Class Reference	561
11.133.1 Detailed Description	562
11.133.2 Constructor & Destructor Documentation	562
11.133.2.1 TimerNode() [1/3]	562
11.133.2.2 TimerNode() [2/3]	563
11.133.2.3 ~TimerNode()	563
11.133.2.4 TimerNode() [3/3]	564
11.133.3 Friends And Related Symbol Documentation	564
11.133.3.1 DataSwitch	564
11.133.4 Member Data Documentation	564
11.133.4.1 d_	564
11.134 arcs::aubo_scope::TreeNode Class Reference	564
11.134.1 Detailed Description	565
11.134.2 Constructor & Destructor Documentation	566
11.134.2.1 TreeNode() [1/3]	566
11.134.2.2 TreeNode() [2/3]	566
11.134.2.3 ~TreeNode()	567
11.134.2.4 TreeNode() [3/3]	567
11.134.3 Member Function Documentation	567
11.134.3.1 addChild()	567
11.134.3.2 cutChildNode()	567
11.134.3.3 getChildren()	568
11.134.3.4 getChildrenNotSuppressed()	568
11.134.3.5 getParent()	568
11.134.3.6 getProgramNode()	568
11.134.3.7 insertChildAfter()	568
11.134.3.8 insertChildBefore()	569
11.134.3.9 isInProgramTree()	570
11.134.3.10 isSuppressed()	570

11.134.3.11	locateDescendantTreeNode()	570
11.134.3.12	removeChild()	571
11.134.3.13	setChildSequenceLocked()	571
11.134.3.14	traverse()	572
11.134.4	Friends And Related Symbol Documentation	572
11.134.4.1	DataSwitch	572
11.134.5	Member Data Documentation	573
11.134.5.1	d_	573
11.135	arcs::aubo_scope::UIntegerRangeValidator Struct Reference	573
11.135.1	Detailed Description	574
11.135.2	Constructor & Destructor Documentation	574
11.135.2.1	UIntegerRangeValidator() [1/2]	574
11.135.2.2	UIntegerRangeValidator() [2/2]	574
11.135.3	Member Function Documentation	574
11.135.3.1	getMessage()	574
11.135.3.2	isValid()	575
11.135.3.3	updateValue()	575
11.135.4	Member Data Documentation	575
11.135.4.1	max_value_	575
11.135.4.2	min_value_	575
11.135.4.3	range_func_	576
11.136	arcs::aubo_scope::UndoRedoManager Class Reference	576
11.136.1	Detailed Description	576
11.136.2	Constructor & Destructor Documentation	577
11.136.2.1	UndoRedoManager() [1/3]	577
11.136.2.2	UndoRedoManager() [2/3]	577
11.136.2.3	~UndoRedoManager()	578
11.136.2.4	UndoRedoManager() [3/3]	578
11.136.3	Member Function Documentation	578
11.136.3.1	recordChanges()	578
11.136.4	Friends And Related Symbol Documentation	578
11.136.4.1	DataSwitch	578
11.136.5	Member Data Documentation	578
11.136.5.1	d_	578
11.137	arcs::aubo_scope::UntilNode Class Reference	579
11.137.1	Detailed Description	581
11.137.2	Member Enumeration Documentation	581
11.137.2.1	BlendParameterType	581
11.137.2.2	CompareOperator	581
11.137.2.3	ConfigType	582
11.137.2.4	CustomDecelerationParameterType	582
11.137.2.5	DecelerationParameterType	583
11.137.2.6	IOInputSelectionType	583

11.137.3 Constructor & Destructor Documentation	584
11.137.3.1 UntilNode() [1/3]	584
11.137.3.2 UntilNode() [2/3]	584
11.137.3.3 ~UntilNode()	585
11.137.3.4 UntilNode() [3/3]	585
11.137.4 Member Function Documentation	585
11.137.4.1 getAnalogInputValueToProceed()	585
11.137.4.2 getAngularDeceleration()	585
11.137.4.3 getBlendParameterType()	585
11.137.4.4 getCartesianDeceleration()	585
11.137.4.5 getCompareOperator()	585
11.137.4.6 getConfigType()	586
11.137.4.7 getCustomDecelerationParameterType()	586
11.137.4.8 getDecelerationParameters()	586
11.137.4.9 getDecelerationParameterType()	586
11.137.4.10 getDigitalInputValueToProceed()	586
11.137.4.11 getDistance()	586
11.137.4.12 getExpressionConfig()	586
11.137.4.13 getIOInputSelectionType()	587
11.137.4.14 getWaypointBlendParameter()	587
11.137.4.15 setAnalogInputValueToProceed()	587
11.137.4.16 setAngularDeceleration()	588
11.137.4.17 setBlendParameterType()	588
11.137.4.18 setCartesianDeceleration()	588
11.137.4.19 setCompareOperator()	588
11.137.4.20 setConfigType()	588
11.137.4.21 setCustomDecelerationParameterType()	588
11.137.4.22 setDecelerationParameters()	588
11.137.4.23 setDecelerationParameterType()	589
11.137.4.24 setDigitalInputValueToProceed()	589
11.137.4.25 setDistance()	590
11.137.4.26 setExpressionConfig()	590
11.137.4.27 setIOInputSelectionType()	590
11.137.4.28 setWaypointBlendParameter()	590
11.137.5 Friends And Related Symbol Documentation	591
11.137.5.1 DataSwitch	591
11.137.6 Member Data Documentation	591
11.137.6.1 d_	591
11.138 arcs::aubo_scope::UserInteraction Class Reference	591
11.138.1 Detailed Description	592
11.138.2 Constructor & Destructor Documentation	592
11.138.2.1 UserInteraction() [1/3]	592
11.138.2.2 UserInteraction() [2/3]	593

11.138.2.3	~UserInteraction()	593
11.138.2.4	UserInteraction() [3/3]	593
11.138.3	Member Function Documentation	593
11.138.3.1	getKeyboardManager()	593
11.138.3.2	getPopupManager()	593
11.138.3.3	getRenderInterface()	594
11.138.3.4	getRobotThumb()	594
11.138.3.5	getUserDefinedRobotPosition() [1/4]	594
11.138.3.6	getUserDefinedRobotPosition() [2/4]	594
11.138.3.7	getUserDefinedRobotPosition() [3/4]	595
11.138.3.8	getUserDefinedRobotPosition() [4/4]	595
11.138.3.9	requestUserToMoveJoint() [1/3]	595
11.138.3.10	requestUserToMoveJoint() [2/3]	596
11.138.3.11	requestUserToMoveJoint() [3/3]	596
11.138.3.12	requestUserToMoveLine() [1/2]	597
11.138.3.13	requestUserToMoveLine() [2/2]	598
11.138.3.14	setOperationLogDisplayName()	598
11.138.4	Friends And Related Symbol Documentation	599
11.138.4.1	DataSwitch	599
11.138.5	Member Data Documentation	599
11.138.5.1	d_	599
11.139	arcs::aubo_scope::UserInterfaceApi Class Reference	599
11.139.1	Detailed Description	600
11.139.2	Constructor & Destructor Documentation	600
11.139.2.1	UserInterfaceApi() [1/3]	600
11.139.2.2	UserInterfaceApi() [2/3]	600
11.139.2.3	~UserInterfaceApi()	601
11.139.2.4	UserInterfaceApi() [3/3]	601
11.139.3	Friends And Related Symbol Documentation	601
11.139.3.1	DataSwitch	601
11.139.4	Member Data Documentation	601
11.139.4.1	d_	601
11.140	arcs::aubo_scope::ValueFactory Class Reference	601
11.140.1	Detailed Description	602
11.140.2	Constructor & Destructor Documentation	602
11.140.2.1	ValueFactory() [1/3]	602
11.140.2.2	ValueFactory() [2/3]	603
11.140.2.3	~ValueFactory()	603
11.140.2.4	ValueFactory() [3/3]	604
11.140.3	Member Function Documentation	604
11.140.3.1	createExpression() [1/2]	604
11.140.3.2	createExpression() [2/2]	604
11.140.4	Friends And Related Symbol Documentation	604

11.140.4.1 DataSwitch	604
11.140.5 Member Data Documentation	604
11.140.5.1 d_	604
11.141 arcs::aubo_scope::Variable Class Reference	605
11.141.1 Detailed Description	606
11.141.2 Member Enumeration Documentation	606
11.141.2.1 Type	606
11.141.3 Constructor & Destructor Documentation	607
11.141.3.1 Variable() [1/3]	607
11.141.3.2 Variable() [2/3]	607
11.141.3.3 ~Variable()	608
11.141.3.4 Variable() [3/3]	608
11.141.4 Member Function Documentation	608
11.141.4.1 equals()	608
11.141.4.2 getDisplayName()	608
11.141.4.3 getType()	608
11.141.4.4 hashCode()	608
11.141.4.5 toString()	609
11.141.5 Friends And Related Symbol Documentation	609
11.141.5.1 DataSwitch	609
11.141.6 Member Data Documentation	609
11.141.6.1 d_	609
11.142 arcs::aubo_scope::VariableFactory Class Reference	609
11.142.1 Detailed Description	610
11.142.2 Constructor & Destructor Documentation	610
11.142.2.1 VariableFactory() [1/3]	610
11.142.2.2 VariableFactory() [2/3]	611
11.142.2.3 ~VariableFactory()	611
11.142.2.4 VariableFactory() [3/3]	611
11.142.3 Member Function Documentation	611
11.142.3.1 createFeatureVariable()	611
11.142.3.2 createGlobalVariable() [1/2]	612
11.142.3.3 createGlobalVariable() [2/2]	612
11.142.4 Friends And Related Symbol Documentation	613
11.142.4.1 DataSwitch	613
11.142.5 Member Data Documentation	613
11.142.5.1 d_	613
11.143 arcs::aubo_scope::VariableModel Class Reference	613
11.143.1 Detailed Description	614
11.143.2 Constructor & Destructor Documentation	614
11.143.2.1 VariableModel() [1/3]	614
11.143.2.2 VariableModel() [2/3]	615
11.143.2.3 ~VariableModel()	615

11.143.2.4	VariableModel() [3/3]	615
11.143.3	Member Function Documentation	615
11.143.3.1	connectedTo()	615
11.143.3.2	disconnectFrom()	616
11.143.3.3	get() [1/2]	616
11.143.3.4	get() [2/2]	616
11.143.3.5	getAll()	617
11.143.3.6	getFeatureVariable()	617
11.143.3.7	getVariableFactory()	618
11.143.4	Friends And Related Symbol Documentation	618
11.143.4.1	DataSwitch	618
11.143.5	Member Data Documentation	618
11.143.5.1	d_	618
11.144	arcs::aubo_scope::ViewApiProvider Class Reference	618
11.144.1	Detailed Description	619
11.144.2	Constructor & Destructor Documentation	619
11.144.2.1	ViewApiProvider() [1/3]	619
11.144.2.2	ViewApiProvider() [2/3]	619
11.144.2.3	ViewApiProvider() [3/3]	620
11.144.2.4	~ViewApiProvider()	620
11.144.3	Member Function Documentation	620
11.144.3.1	getSystemApi()	620
11.144.3.2	getUserInterfaceApi()	620
11.144.4	Friends And Related Symbol Documentation	621
11.144.4.1	DataSwitch	621
11.144.5	Member Data Documentation	621
11.144.5.1	d_	621
11.145	arcs::aubo_scope::VisualLocalizationProgramNode Class Reference	621
11.145.1	Detailed Description	623
11.145.2	Constructor & Destructor Documentation	623
11.145.2.1	VisualLocalizationProgramNode() [1/3]	623
11.145.2.2	VisualLocalizationProgramNode() [2/3]	623
11.145.2.3	~VisualLocalizationProgramNode()	624
11.145.2.4	VisualLocalizationProgramNode() [3/3]	624
11.145.3	Friends And Related Symbol Documentation	624
11.145.3.1	DataSwitch	624
11.145.4	Member Data Documentation	624
11.145.4.1	d_	624
11.146	arcs::aubo_scope::WaitNode Class Reference	625
11.146.1	Detailed Description	627
11.146.2	Member Enumeration Documentation	627
11.146.2.1	CompareOperator	627
11.146.2.2	ConfigType	627

11.146.3 Constructor & Destructor Documentation	628
11.146.3.1 WaitNode() [1/3]	628
11.146.3.2 WaitNode() [2/3]	629
11.146.3.3 ~WaitNode()	629
11.146.3.4 WaitNode() [3/3]	629
11.146.4 Member Function Documentation	629
11.146.4.1 getAnalogInput()	629
11.146.4.2 getBoolRegister()	629
11.146.4.3 getBoolValueToWaitFor()	629
11.146.4.4 getCompareOperator()	630
11.146.4.5 getConfigType()	630
11.146.4.6 getDigitalInput()	630
11.146.4.7 getDoubleRegisterInput()	630
11.146.4.8 getExpressionInputConfig()	630
11.146.4.9 getFloatValueToWaitFor()	630
11.146.4.10 getModbusInput()	630
11.146.4.11 getWaitTime()	630
11.146.4.12 setAnalogInput()	630
11.146.4.13 setBoolRegister()	631
11.146.4.14 setBoolValueToWaitFor()	632
11.146.4.15 setCompareOperator()	632
11.146.4.16 setConfigType()	632
11.146.4.17 setDigitalInput()	632
11.146.4.18 setDoubleRegisterInput()	633
11.146.4.19 setExpressionInputConfig()	633
11.146.4.20 setFloatValueToWaitFor()	634
11.146.4.21 setModbusInput()	634
11.146.4.22 setWaitTime()	634
11.146.5 Friends And Related Symbol Documentation	635
11.146.5.1 DataSwitch	635
11.146.6 Member Data Documentation	635
11.146.6.1 d_	635
11.147 arcs::aubo_scope::Waypoint Class Reference	635
11.147.1 Detailed Description	636
11.147.2 Constructor & Destructor Documentation	636
11.147.2.1 Waypoint() [1/3]	636
11.147.2.2 Waypoint() [2/3]	637
11.147.2.3 ~Waypoint()	637
11.147.2.4 Waypoint() [3/3]	637
11.147.3 Member Function Documentation	637
11.147.3.1 getDisplayName()	637
11.147.3.2 getJointPositions()	637
11.147.3.3 getPose()	638

11.147.3.4	getTcpOffset()	638
11.147.3.5	hashCode()	638
11.147.3.6	isDefined()	638
11.147.4	Friends And Related Symbol Documentation	638
11.147.4.1	DataSwitch	638
11.147.5	Member Data Documentation	639
11.147.5.1	d_	639
11.148	arcs::aubo_scope::WaypointModel Class Reference	639
11.148.1	Detailed Description	640
11.148.2	Constructor & Destructor Documentation	640
11.148.2.1	WaypointModel() [1/3]	640
11.148.2.2	WaypointModel() [2/3]	641
11.148.2.3	~WaypointModel()	641
11.148.2.4	WaypointModel() [3/3]	641
11.148.3	Member Function Documentation	641
11.148.3.1	addWaypoint() [1/2]	641
11.148.3.2	addWaypoint() [2/2]	642
11.148.3.3	getByName()	642
11.148.3.4	getWaypoints()	643
11.148.3.5	removeWaypoint()	643
11.148.3.6	renameWaypoint()	643
11.148.4	Friends And Related Symbol Documentation	644
11.148.4.1	DataSwitch	644
11.148.5	Member Data Documentation	644
11.148.5.1	d_	644
11.149	arcs::aubo_scope::WaypointNode Class Reference	644
11.149.1	Detailed Description	646
11.149.2	Member Enumeration Documentation	646
11.149.2.1	BlendType	646
11.149.2.2	ConfigType	647
11.149.2.3	MotionType	647
11.149.2.4	PositionDefinition	647
11.149.2.5	RelativeOffsetType	648
11.149.2.6	RelativePositionType	648
11.149.2.7	VariableDefinition	648
11.149.2.8	VariableType	649
11.149.3	Constructor & Destructor Documentation	649
11.149.3.1	WaypointNode() [1/3]	649
11.149.3.2	WaypointNode() [2/3]	650
11.149.3.3	~WaypointNode()	650
11.149.3.4	WaypointNode() [3/3]	650
11.149.4	Member Function Documentation	650
11.149.4.1	getBlend()	650

11.149.4.2	getBlendType()	651
11.149.4.3	getConfigType()	651
11.149.4.4	getJointAcceleration()	651
11.149.4.5	getJointPositions()	651
11.149.4.6	getJointSpeed()	651
11.149.4.7	getMotionType()	652
11.149.4.8	getPose()	652
11.149.4.9	getPositionDefinition()	652
11.149.4.10	getTcpOffset()	652
11.149.4.11	getTime()	652
11.149.4.12	getToolAcceleration()	652
11.149.4.13	getToolSpeed()	653
11.149.4.14	getVariable()	653
11.149.4.15	getVariableDefinition()	653
11.149.4.16	setFixedPositionConfig() [1/2]	653
11.149.4.17	setFixedPositionConfig() [2/2]	653
11.149.4.18	setJointMotionParameters()	654
11.149.4.19	setSharedMotionParameters()	654
11.149.4.20	setTimeMotionParameters()	655
11.149.4.21	setToolMotionParameters()	655
11.149.4.22	setVariablePositionConfig() [1/2]	656
11.149.4.23	setVariablePositionConfig() [2/2]	656
11.149.5	Friends And Related Symbol Documentation	656
11.149.5.1	DataSwitch	656
11.149.6	Member Data Documentation	657
11.149.6.1	d_	657
11.150	arcs::aubo_scope::WidthCapability Class Reference	657
11.150.1	Detailed Description	657
11.150.2	Constructor & Destructor Documentation	657
11.150.2.1	~WidthCapability()	657
11.150.3	Member Function Documentation	657
11.150.3.1	updateCapability()	657
12	File Documentation	659
12.1	doc/SDK_overview_en.md File Reference	659
12.2	include/aubo_caps/class_forward.h File Reference	659
12.2.1	Macro Definition Documentation	660
12.2.1.1	ARCS_ABI_EXPORT	660
12.2.1.2	ARCS_ABI_IMPORT	660
12.2.1.3	ARCS_ABI_LOCAL	660
12.2.1.4	ARCS_CLASS_FORWARD	660
12.2.1.5	ARCS_DECLARE_PTR	660
12.2.1.6	ARCS_DECLARE_PTR_MEMBER	661

12.2.1.7 ARCS_STRUCT_FORWARD	661
12.3 class_forward.h	661
12.4 include/aubo_caps/contribution/contribution_provider.h File Reference	662
12.5 contribution_provider.h	663
12.6 include/aubo_caps/contribution/driver/gripper/capability/grip_force_capability.h File Reference	664
12.7 grip_force_capability.h	665
12.8 include/aubo_caps/contribution/driver/gripper/capability/grip_vacuum_capability.h File Reference	667
12.9 grip_vacuum_capability.h	668
12.10 include/aubo_caps/contribution/driver/gripper/capability/registered_cpabilities.h File Reference	669
12.11 registered_cpabilities.h	670
12.12 include/aubo_caps/contribution/driver/gripper/capability/speed_capability.h File Ref- erence	672
12.13 speed_capability.h	673
12.14 include/aubo_caps/contribution/driver/gripper/capability/width_capability.h File Ref- erence	674
12.15 width_capability.h	675
12.16 include/aubo_caps/contribution/driver/gripper/grip_action_parameters.h File Reference	677
12.17 grip_action_parameters.h	678
12.18 include/aubo_caps/contribution/driver/gripper/gripper_api_provider.h File Reference	679
12.19 gripper_api_provider.h	681
12.20 include/aubo_caps/contribution/driver/gripper/gripper_configuration.h File Reference	681
12.21 gripper_configuration.h	683
12.22 include/aubo_caps/contribution/driver/gripper/gripper_contribution.h File Reference	686
12.23 gripper_contribution.h	687
12.24 include/aubo_caps/contribution/driver/gripper/release_action_parameters.h File Refer- ence	690
12.25 release_action_parameters.h	691
12.26 include/aubo_caps/contribution/driver/screw_driver/screw_driver_api_provider.h File Reference	692
12.27 screw_driver_api_provider.h	694
12.28 include/aubo_caps/contribution/driver/screw_driver/screw_driver_configuration.h File Reference	694
12.29 screw_driver_configuration.h	696
12.30 include/aubo_caps/contribution/driver/screw_driver/screw_driver_contribution.h File Reference	702
12.31 screw_driver_contribution.h	703
12.32 include/aubo_caps/contribution/driver/screw_driver/screw_driver_parameters.h File Reference	705
12.33 screw_driver_parameters.h	707
12.34 include/aubo_caps/contribution/installation/installation_api_provider.h File Reference	709
12.35 installation_api_provider.h	710
12.36 include/aubo_caps/contribution/installation/installation_contribution_configuration.h File Reference	711

12.37	installation_contribution_configuration.h	712
12.38	include/aubo_caps/contribution/installation/installation_creation_context.h File Reference	712
12.39	installation_creation_context.h	713
12.40	include/aubo_caps/contribution/installation/installation_node_contribution.h File Reference	714
12.41	installation_node_contribution.h	715
12.42	include/aubo_caps/contribution/installation/installation_node_service.h File Reference	716
12.43	installation_node_service.h	717
12.44	include/aubo_caps/contribution/installation/installation_node_view.h File Reference	719
12.44.1	Typedef Documentation	720
12.44.1.1	WidgetHandle	720
12.45	installation_node_view.h	721
12.46	include/aubo_caps/contribution/program/program_api_provider.h File Reference	721
12.47	program_api_provider.h	722
12.48	include/aubo_caps/contribution/program/program_contribution_configuration.h File Reference	724
12.49	program_contribution_configuration.h	725
12.50	include/aubo_caps/contribution/program/program_creation_context.h File Reference	727
12.51	program_creation_context.h	728
12.52	include/aubo_caps/contribution/program/program_node_contribution.h File Reference	729
12.53	program_node_contribution.h	730
12.54	include/aubo_caps/contribution/program/program_node_service.h File Reference	732
12.55	program_node_service.h	732
12.56	include/aubo_caps/contribution/program/program_node_view.h File Reference	734
12.57	program_node_view.h	735
12.58	include/aubo_caps/contribution/view_api_provider.h File Reference	736
12.59	view_api_provider.h	737
12.60	include/aubo_caps/domain/application_api.h File Reference	738
12.61	application_api.h	739
12.62	include/aubo_caps/domain/data/data_model.h File Reference	742
12.63	data_model.h	743
12.64	include/aubo_caps/domain/device/device.h File Reference	752
12.65	device.h	754
12.66	include/aubo_caps/domain/device/device_manager.h File Reference	755
12.67	device_manager.h	757
12.68	include/aubo_caps/domain/device/device_registration_manager.h File Reference	758
12.69	device_registration_manager.h	759
12.70	include/aubo_caps/domain/device/gripper_device.h File Reference	760
12.71	gripper_device.h	762
12.72	include/aubo_caps/domain/extension_info.h File Reference	762
12.73	extension_info.h	764
12.74	include/aubo_caps/domain/feature/base_feature.h File Reference	765
12.75	base_feature.h	766

12.76	include/aubo_caps/domain/feature/feature.h File Reference	766
12.77	feature.h	767
12.78	include/aubo_caps/domain/feature/feature_contribution_model.h File Reference	769
12.79	feature_contribution_model.h	771
12.80	include/aubo_caps/domain/feature/feature_line.h File Reference	772
12.81	feature_line.h	773
12.82	include/aubo_caps/domain/feature/feature_model.h File Reference	774
12.83	feature_model.h	775
12.84	include/aubo_caps/domain/feature/feature_plane.h File Reference	777
12.85	feature_plane.h	778
12.86	include/aubo_caps/domain/feature/feature_point.h File Reference	779
12.87	feature_point.h	780
12.88	include/aubo_caps/domain/feature/feature_tool.h File Reference	781
12.89	feature_tool.h	782
12.90	include/aubo_caps/domain/function/function.h File Reference	782
12.91	function.h	784
12.92	include/aubo_caps/domain/function/function_model.h File Reference	785
12.93	function_model.h	786
12.94	include/aubo_caps/domain/installation/installation_api.h File Reference	788
12.95	installation_api.h	789
12.96	include/aubo_caps/domain/io/analog_io.h File Reference	791
12.97	analog_io.h	793
12.98	include/aubo_caps/domain/io/boolean_register.h File Reference	794
12.99	boolean_register.h	796
12.100	include/aubo_caps/domain/io/digital_io.h File Reference	796
12.101	digital_io.h	798
12.102	include/aubo_caps/domain/io/double_register.h File Reference	799
12.103	double_register.h	801
12.104	include/aubo_caps/domain/io/ethernet_ip.h File Reference	801
12.105	ethernet_ip.h	802
12.106	include/aubo_caps/domain/io/integer_register.h File Reference	804
12.107	integer_register.h	805
12.108	include/aubo_caps/domain/io/io.h File Reference	805
12.109	io.h	806
12.110	include/aubo_caps/domain/io/io_model.h File Reference	808
12.111	io_model.h	809
12.112	include/aubo_caps/domain/io/modbus_io.h File Reference	811
12.113	modbus_io.h	813
12.114	include/aubo_caps/domain/io/profinet.h File Reference	815
12.115	profinet.h	816
12.116	include/aubo_caps/domain/io/register.h File Reference	818
12.117	register.h	820
12.118	include/aubo_caps/domain/payload/payload.h File Reference	820

12.119	payload.h	821
12.120	include/aubo_caps/domain/payload/payload_contribution_model.h File Reference	824
12.121	payload_contribution_model.h	825
12.122	include/aubo_caps/domain/payload/payload_model.h File Reference	830
12.123	payload_model.h	832
12.124	include/aubo_caps/domain/program/nodes/builtin/assembly_node.h File Reference	833
12.125	assembly_node.h	835
12.126	include/aubo_caps/domain/program/nodes/builtin/assignment_node.h File Reference	835
12.127	assignment_node.h	837
12.128	include/aubo_caps/domain/program/nodes/builtin/break_node.h File Reference	838
12.129	break_node.h	840
12.130	include/aubo_caps/domain/program/nodes/builtin/call_node.h File Reference	840
12.131	call_node.h	842
12.132	include/aubo_caps/domain/program/nodes/builtin/case_node.h File Reference	842
12.133	case_node.h	844
12.134	include/aubo_caps/domain/program/nodes/builtin/circle_move_node.h File Reference	844
12.135	circle_move_node.h	846
12.136	include/aubo_caps/domain/program/nodes/builtin/comment_node.h File Reference	846
12.137	comment_node.h	848
12.138	include/aubo_caps/domain/program/nodes/builtin/conveyor_tracking_node.h File Reference	848
12.139	conveyor_tracking_node.h	850
12.140	include/aubo_caps/domain/program/nodes/builtin/default_case_node.h File Reference	851
12.141	default_case_node.h	853
12.142	include/aubo_caps/domain/program/nodes/builtin/direction_node.h File Reference	853
12.143	direction_node.h	855
12.144	include/aubo_caps/domain/program/nodes/builtin/else_if_node.h File Reference	856
12.145	else_if_node.h	859
12.146	include/aubo_caps/domain/program/nodes/builtin/else_node.h File Reference	859
12.147	else_node.h	861
12.148	include/aubo_caps/domain/program/nodes/builtin/folder_node.h File Reference	861
12.149	folder_node.h	863
12.150	include/aubo_caps/domain/program/nodes/builtin/force_node.h File Reference	863
12.151	force_node.h	865
12.152	include/aubo_caps/domain/program/nodes/builtin/force_option_node.h File Reference	866
12.153	force_option_node.h	868
12.154	include/aubo_caps/domain/program/nodes/builtin/gripper_node.h File Reference	868
12.155	gripper_node.h	870
12.156	include/aubo_caps/domain/program/nodes/builtin/halt_node.h File Reference	872
12.157	halt_node.h	874
12.158	include/aubo_caps/domain/program/nodes/builtin/if_node.h File Reference	874
12.159	if_node.h	876
12.160	include/aubo_caps/domain/program/nodes/builtin/loop_node.h File Reference	876

12.161	loop_node.h	879
12.162	include/aubo_caps/domain/program/nodes/builtin/move_node.h File Reference	880
12.163	move_node.h	882
12.164	include/aubo_caps/domain/program/nodes/builtin/offline_track_node.h File Reference	885
12.165	offline_track_node.h	887
12.166	include/aubo_caps/domain/program/nodes/builtin/pallet_node.h File Reference	887
12.167	pallet_node.h	889
12.168	include/aubo_caps/domain/program/nodes/builtin/popup_node.h File Reference	889
12.169	popup_node.h	891
12.170	include/aubo_caps/domain/program/nodes/builtin/program_home_node.h File Reference	891
12.171	program_home_node.h	894
12.172	include/aubo_caps/domain/program/nodes/builtin/record_track_node.h File Reference	894
12.173	record_track_node.h	896
12.174	include/aubo_caps/domain/program/nodes/builtin/screw_driving_node.h File Reference	896
12.175	screw_driving_node.h	898
12.176	include/aubo_caps/domain/program/nodes/builtin/script_node.h File Reference	898
12.177	script_node.h	900
12.178	include/aubo_caps/domain/program/nodes/builtin/seek_node.h File Reference	900
12.179	seek_node.h	902
12.180	include/aubo_caps/domain/program/nodes/builtin/set_node.h File Reference	902
12.181	set_node.h	904
12.182	include/aubo_caps/domain/program/nodes/builtin/set_payload_node.h File Reference	906
12.183	set_payload_node.h	909
12.184	include/aubo_caps/domain/program/nodes/builtin/sub_prog_node.h File Reference	910
12.185	sub_prog_node.h	912
12.186	include/aubo_caps/domain/program/nodes/builtin/switch_node.h File Reference	912
12.187	switch_node.h	914
12.188	include/aubo_caps/domain/program/nodes/builtin/thread_node.h File Reference	914
12.189	thread_node.h	916
12.190	include/aubo_caps/domain/program/nodes/builtin/timer_node.h File Reference	916
12.191	timer_node.h	918
12.192	include/aubo_caps/domain/program/nodes/builtin/until_node.h File Reference	918
12.193	until_node.h	920
12.194	include/aubo_caps/domain/program/nodes/builtin/visual_localization_program_node.h File Reference	924
12.195	visual_localization_program_node.h	926
12.196	include/aubo_caps/domain/program/nodes/builtin/wait_node.h File Reference	926
12.197	wait_node.h	928
12.198	include/aubo_caps/domain/program/nodes/builtin/waypoint_node.h File Reference	931
12.199	waypoint_node.h	933
12.200	include/aubo_caps/domain/program/nodes/program_node.h File Reference	937
12.201	program_node.h	938
12.202	include/aubo_caps/domain/program/nodes/program_node_factory.h File Reference	939

12.203	program_node_factory.h	942
12.204	include/aubo_caps/domain/program/program_api.h File Reference	945
12.205	program_api.h	946
12.206	include/aubo_caps/domain/program/program_model.h File Reference	947
12.207	program_model.h	948
12.208	include/aubo_caps/domain/program/structure/tree_node.h File Reference	949
12.209	tree_node.h	950
12.210	include/aubo_caps/domain/script/script_writer.h File Reference	955
12.211	script_writer.h	956
12.212	include/aubo_caps/domain/system_api.h File Reference	958
12.213	system_api.h	960
12.214	include/aubo_caps/domain/tcp/tcp.h File Reference	966
12.215	tcp.h	968
12.216	include/aubo_caps/domain/tcp/tcp_contribution_model.h File Reference	969
12.217	tcp_contribution_model.h	970
12.218	include/aubo_caps/domain/tcp/tcp_model.h File Reference	972
12.219	tcp_model.h	973
12.220	include/aubo_caps/domain/undoredo/undo_redo_manager.h File Reference	975
12.221	undo_redo_manager.h	977
12.222	include/aubo_caps/domain/user_interface/input_validator.h File Reference	978
12.223	input_validator.h	979
12.224	include/aubo_caps/domain/user_interface/keyboard_manager.h File Reference	981
12.225	keyboard_manager.h	982
12.226	include/aubo_caps/domain/user_interface/popup_manager.h File Reference	984
12.227	popup_manager.h	985
12.228	include/aubo_caps/domain/user_interface/render_interface.h File Reference	986
12.229	render_interface.h	987
12.230	include/aubo_caps/domain/user_interface/user_interaction.h File Reference	995
12.231	user_interaction.h	997
12.232	include/aubo_caps/domain/user_interface/user_interface_api.h File Reference	1000
12.233	user_interface_api.h	1001
12.234	include/aubo_caps/domain/value/expression/expression.h File Reference	1002
12.235	expression.h	1003
12.236	include/aubo_caps/domain/value/value_factory_provider.h File Reference	1004
12.237	value_factory_provider.h	1005
12.238	include/aubo_caps/domain/variable/feature_variable.h File Reference	1007
12.239	feature_variable.h	1009
12.240	include/aubo_caps/domain/variable/global_variable.h File Reference	1010
12.241	global_variable.h	1012
12.242	include/aubo_caps/domain/variable/persisted_variable.h File Reference	1013
12.243	persisted_variable.h	1015
12.244	include/aubo_caps/domain/variable/variable.h File Reference	1015
12.245	variable.h	1017

12.246	include/aubo_caps/domain/variable/variable_factory.h File Reference	1018
12.247	variable_factory.h	1020
12.248	include/aubo_caps/domain/variable/variable_model.h File Reference	1021
12.249	variable_model.h	1022
12.250	include/aubo_caps/domain/waypoint/waypoint.h File Reference	1024
12.251	waypoint.h	1025
12.252	include/aubo_caps/domain/waypoint/waypoint_model.h File Reference	1027
12.253	waypoint_model.h	1028
Index		1031

Chapter 1

AuboStudio SDK

Overview

AuboStudio SDK provides a complete set of interfaces for developing plugins and extending robot application capabilities.

Contents

- How plugins are integrated into aubo_scope
- SDK architecture
- Examples
- Namespaces

How plugins are integrated into aubo_scope

A plugin is a software package that runs as a subprocess of aubo_scope. aubo_scope registers plugins and interacts with them in different phases, such as during startup and when users operate aubo_scope.

Each plugin can contribute multiple new functions to aubo_scope. These functions are usually called services and generally fall into two categories:

Services:

1. Installation node services.
2. Program node services.

Example (aubo_caps gripper):

1. Installation node service: configure gripper installation and connection.
2. Program node service: provide program nodes such as opening and closing the gripper.

Plugin integration flow

- When `aubo_scope` starts, it searches for all installed `aubo_caps` packages.
- `aubo_scope` calls the activator of each `aubo_caps`. The activator overrides `start()` and `stop()`. During startup, `start()` registers services through `RegisterService`; when `aubo_scope` stops, `stop()` is called.
- After registration, the service provides functions for configuring node metadata, creating the user interaction view, and creating the contribution object for each node.
- The contribution class provides node-specific APIs and generates the corresponding script code.
- Service settings are usually static because they are used during startup and are not used directly while the robot program is running.

Program node integration in `aubo_scope`

A program node usually consists of a service, a view, and a contribution object.

ProgramNodeService

ProgramNodeService defines the static metadata and factory methods of a program node type. Typical responsibilities include:

- Returning the node type ID through `getId()`.
- Returning the node title through `getTitle()`.
- Returning the node icon path through `getIcon()`.
- Configuring node behavior through `configureContribution()`.
- Creating the node view through `createView()`.
- Creating contribution instances through `createNode()`.

ProgramNodeView

ProgramNodeView builds the user interface shown in the `aubo_scope` panel. There may be only one view instance for a node type, while multiple contribution instances can exist in a program.

ProgramNodeContribution

ProgramNodeContribution stores and updates node data, handles user interaction, determines whether the node is defined, and generates script code through `generateScript()`.

Installation node integration in `aubo_scope`

An installation node usually provides global installation-related settings. It also consists of a service, a view, and a contribution object.

SDK architecture

AuboStudio SDK exposes APIs under the `aubo_caps` namespace. These APIs can be used to register services, access system and UI capabilities, manage data models, and implement installation or program node behavior.

Examples

For plugin development examples and tutorials, see the documents in the `doc/` directory and the sample code in the `demo/` directory.

Namespaces

The main namespace is `aubo_caps`. Detailed classes, interfaces, and functions are documented in the API reference generated from header comments.

Chapter 2

Topic Index

2.1 Topics

Here is a list of all topics with brief descriptions:

ApplicationApi	37
InstallationApi	40
ProgramApi	41
SystemApi	43
UserInterfaceApi	52

Chapter 3

Directory Hierarchy

3.1 Directories

aubo_caps	55
contribution	57
driver	60
gripper	62
capability	57
grip_force_capability.h	664
grip_vacuum_capability.h	667
registered_cpabilities.h	669
speed_capability.h	672
width_capability.h	674
grip_action_parameters.h	677
gripper_api_provider.h	679
gripper_configuration.h	681
gripper_contribution.h	686
release_action_parameters.h	690
screw_driver	67
screw_driver_api_provider.h	692
screw_driver_configuration.h	694
screw_driver_contribution.h	702
screw_driver_parameters.h	705
installation	63
installation_api_provider.h	709
installation_contribution_configuration.h	711
installation_creation_context.h	712
installation_node_contribution.h	714
installation_node_service.h	716
installation_node_view.h	719
program	65
program_api_provider.h	721
program_contribution_configuration.h	724
program_creation_context.h	727
program_node_contribution.h	729
program_node_service.h	732
program_node_view.h	734
contribution_provider.h	662
view_api_provider.h	736
domain	59

data	58
data_model.h	742
device	58
device.h	752
device_manager.h	755
device_registration_manager.h	758
gripper_device.h	760
feature	61
base_feature.h	765
feature.h	766
feature_contribution_model.h	769
feature_line.h	772
feature_model.h	774
feature_plane.h	777
feature_point.h	779
feature_tool.h	781
function	61
function.h	782
function_model.h	785
installation	63
installation_api.h	788
io	64
analog_io.h	791
boolean_register.h	794
digital_io.h	796
double_register.h	799
ethernet_ip.h	801
integer_register.h	804
io.h	805
io_model.h	808
modbus_io.h	811
profinet.h	815
register.h	818
payload	65
payload.h	820
payload_contribution_model.h	824
payload_model.h	830
program	66
nodes	64
builtin	56
assembly_node.h	833
assignment_node.h	835
break_node.h	838
call_node.h	840
case_node.h	842
circle_move_node.h	844
comment_node.h	846
conveyor_tracking_node.h	848
default_case_node.h	851
direction_node.h	853
else_if_node.h	856
else_node.h	859
folder_node.h	861
force_node.h	863
force_option_node.h	866
gripper_node.h	868
halt_node.h	872

if_node.h	874
loop_node.h	876
move_node.h	880
offline_track_node.h	885
pallet_node.h	887
popup_node.h	889
program_home_node.h	891
record_track_node.h	894
screw_driving_node.h	896
script_node.h	898
seek_node.h	900
set_node.h	902
set_payload_node.h	906
sub_prog_node.h	910
switch_node.h	912
thread_node.h	914
timer_node.h	916
until_node.h	918
visual_localization_program_node.h	924
wait_node.h	926
waypoint_node.h	931
program_node.h	937
program_node_factory.h	939
structure	68
tree_node.h	949
program_api.h	945
program_model.h	947
script	67
script_writer.h	955
tcp	68
tcp.h	966
tcp_contribution_model.h	969
tcp_model.h	972
undoredo	69
undo_redo_manager.h	975
user_interface	69
input_validator.h	978
keyboard_manager.h	981
popup_manager.h	984
render_interface.h	986
user_interaction.h	995
user_interface_api.h	1000
value	70
expression	60
expression.h	1002
value_factory_provider.h	1004
variable	71
feature_variable.h	1007
global_variable.h	1010
persisted_variable.h	1013
variable.h	1015
variable_factory.h	1018
variable_model.h	1021
waypoint	71
waypoint.h	1024
waypoint_model.h	1027
application_api.h	738

extension_info.h	762
system_api.h	958
class_forward.h	659
builtin	56
assembly_node.h	833
assignment_node.h	835
break_node.h	838
call_node.h	840
case_node.h	842
circle_move_node.h	844
comment_node.h	846
conveyor_tracking_node.h	848
default_case_node.h	851
direction_node.h	853
else_if_node.h	856
else_node.h	859
folder_node.h	861
force_node.h	863
force_option_node.h	866
grripper_node.h	868
halt_node.h	872
if_node.h	874
loop_node.h	876
move_node.h	880
offline_track_node.h	885
pallet_node.h	887
popup_node.h	889
program_home_node.h	891
record_track_node.h	894
screw_driving_node.h	896
script_node.h	898
seek_node.h	900
set_node.h	902
set_payload_node.h	906
sub_prog_node.h	910
switch_node.h	912
thread_node.h	914
timer_node.h	916
until_node.h	918
visual_localization_program_node.h	924
wait_node.h	926
waypoint_node.h	931
capability	57
grip_force_capability.h	664
grip_vacuum_capability.h	667
registered_cpabilities.h	669
speed_capability.h	672
width_capability.h	674
contribution	57
driver	60
grripper	62
capability	57
grip_force_capability.h	664
grip_vacuum_capability.h	667
registered_cpabilities.h	669
speed_capability.h	672
width_capability.h	674

grip_action_parameters.h	677
gripper_api_provider.h	679
gripper_configuration.h	681
gripper_contribution.h	686
release_action_parameters.h	690
screw_driver	67
screw_driver_api_provider.h	692
screw_driver_configuration.h	694
screw_driver_contribution.h	702
screw_driver_parameters.h	705
installation	63
installation_api_provider.h	709
installation_contribution_configuration.h	711
installation_creation_context.h	712
installation_node_contribution.h	714
installation_node_service.h	716
installation_node_view.h	719
program	65
program_api_provider.h	721
program_contribution_configuration.h	724
program_creation_context.h	727
program_node_contribution.h	729
program_node_service.h	732
program_node_view.h	734
contribution_provider.h	662
view_api_provider.h	736
data	58
data_model.h	742
device	58
device.h	752
device_manager.h	755
device_registration_manager.h	758
gripper_device.h	760
doc	59
domain	59
data	58
data_model.h	742
device	58
device.h	752
device_manager.h	755
device_registration_manager.h	758
gripper_device.h	760
feature	61
base_feature.h	765
feature.h	766
feature_contribution_model.h	769
feature_line.h	772
feature_model.h	774
feature_plane.h	777
feature_point.h	779
feature_tool.h	781
function	61
function.h	782
function_model.h	785
installation	63
installation_api.h	788

io	64
analog_io.h	791
boolean_register.h	794
digital_io.h	796
double_register.h	799
ethernet_ip.h	801
integer_register.h	804
io.h	805
io_model.h	808
modbus_io.h	811
profinet.h	815
register.h	818
payload	65
payload.h	820
payload_contribution_model.h	824
payload_model.h	830
program	66
nodes	64
builtin	56
assembly_node.h	833
assignment_node.h	835
break_node.h	838
call_node.h	840
case_node.h	842
circle_move_node.h	844
comment_node.h	846
conveyor_tracking_node.h	848
default_case_node.h	851
direction_node.h	853
else_if_node.h	856
else_node.h	859
folder_node.h	861
force_node.h	863
force_option_node.h	866
gripper_node.h	868
halt_node.h	872
if_node.h	874
loop_node.h	876
move_node.h	880
offline_track_node.h	885
pallet_node.h	887
popup_node.h	889
program_home_node.h	891
record_track_node.h	894
screw_driving_node.h	896
script_node.h	898
seek_node.h	900
set_node.h	902
set_payload_node.h	906
sub_prog_node.h	910
switch_node.h	912
thread_node.h	914
timer_node.h	916
until_node.h	918
visual_localization_program_node.h	924
wait_node.h	926
waypoint_node.h	931
program_node.h	937

program_node_factory.h	939
structure	68
tree_node.h	949
program_api.h	945
program_model.h	947
script	67
script_writer.h	955
tcp	68
tcp.h	966
tcp_contribution_model.h	969
tcp_model.h	972
undoredo	69
undo_redo_manager.h	975
user_interface	69
input_validator.h	978
keyboard_manager.h	981
popup_manager.h	984
render_interface.h	986
user_interaction.h	995
user_interface_api.h	1000
value	70
expression	60
expression.h	1002
value_factory_provider.h	1004
variable	71
feature_variable.h	1007
global_variable.h	1010
persisted_variable.h	1013
variable.h	1015
variable_factory.h	1018
variable_model.h	1021
waypoint	71
waypoint.h	1024
waypoint_model.h	1027
application_api.h	738
extension_info.h	762
system_api.h	958
driver	60
gripper	62
capability	57
grip_force_capability.h	664
grip_vacuum_capability.h	667
registered_capabilities.h	669
speed_capability.h	672
width_capability.h	674
grip_action_parameters.h	677
gripper_api_provider.h	679
gripper_configuration.h	681
gripper_contribution.h	686
release_action_parameters.h	690
screw_driver	67
screw_driver_api_provider.h	692
screw_driver_configuration.h	694
screw_driver_contribution.h	702
screw_driver_parameters.h	705
expression	60

expression.h	1002
feature	61
base_feature.h	765
feature.h	766
feature_contribution_model.h	769
feature_line.h	772
feature_model.h	774
feature_plane.h	777
feature_point.h	779
feature_tool.h	781
function	61
function.h	782
function_model.h	785
gripper	62
capability	57
grip_force_capability.h	664
grip_vacuum_capability.h	667
registered_capabilities.h	669
speed_capability.h	672
width_capability.h	674
grip_action_parameters.h	677
gripper_api_provider.h	679
gripper_configuration.h	681
gripper_contribution.h	686
release_action_parameters.h	690
include	62
aubo_caps	55
contribution	57
driver	60
gripper	62
capability	57
grip_force_capability.h	664
grip_vacuum_capability.h	667
registered_capabilities.h	669
speed_capability.h	672
width_capability.h	674
grip_action_parameters.h	677
gripper_api_provider.h	679
gripper_configuration.h	681
gripper_contribution.h	686
release_action_parameters.h	690
screw_driver	67
screw_driver_api_provider.h	692
screw_driver_configuration.h	694
screw_driver_contribution.h	702
screw_driver_parameters.h	705
installation	63
installation_api_provider.h	709
installation_contribution_configuration.h	711
installation_creation_context.h	712
installation_node_contribution.h	714
installation_node_service.h	716
installation_node_view.h	719
program	65
program_api_provider.h	721
program_contribution_configuration.h	724
program_creation_context.h	727

program_node_contribution.h	729
program_node_service.h	732
program_node_view.h	734
contribution_provider.h	662
view_api_provider.h	736
domain	59
data	58
data_model.h	742
device	58
device.h	752
device_manager.h	755
device_registration_manager.h	758
gripper_device.h	760
feature	61
base_feature.h	765
feature.h	766
feature_contribution_model.h	769
feature_line.h	772
feature_model.h	774
feature_plane.h	777
feature_point.h	779
feature_tool.h	781
function	61
function.h	782
function_model.h	785
installation	63
installation_api.h	788
io	64
analog_io.h	791
boolean_register.h	794
digital_io.h	796
double_register.h	799
ethernet_ip.h	801
integer_register.h	804
io.h	805
io_model.h	808
modbus_io.h	811
profinet.h	815
register.h	818
payload	65
payload.h	820
payload_contribution_model.h	824
payload_model.h	830
program	66
nodes	64
builtin	56
assembly_node.h	833
assignment_node.h	835
break_node.h	838
call_node.h	840
case_node.h	842
circle_move_node.h	844
comment_node.h	846
conveyor_tracking_node.h	848
default_case_node.h	851
direction_node.h	853
else_if_node.h	856

else_node.h	859
folder_node.h	861
force_node.h	863
force_option_node.h	866
gripper_node.h	868
halt_node.h	872
if_node.h	874
loop_node.h	876
move_node.h	880
offline_track_node.h	885
pallet_node.h	887
popup_node.h	889
program_home_node.h	891
record_track_node.h	894
screw_driving_node.h	896
script_node.h	898
seek_node.h	900
set_node.h	902
set_payload_node.h	906
sub_prog_node.h	910
switch_node.h	912
thread_node.h	914
timer_node.h	916
until_node.h	918
visual_localization_program_node.h	924
wait_node.h	926
waypoint_node.h	931
program_node.h	937
program_node_factory.h	939
structure	68
tree_node.h	949
program_api.h	945
program_model.h	947
script	67
script_writer.h	955
tcp	68
tcp.h	966
tcp_contribution_model.h	969
tcp_model.h	972
undoredo	69
undo_redo_manager.h	975
user_interface	69
input_validator.h	978
keyboard_manager.h	981
popup_manager.h	984
render_interface.h	986
user_interaction.h	995
user_interface_api.h	1000
value	70
expression	60
expression.h	1002
value_factory_provider.h	1004
variable	71
feature_variable.h	1007
global_variable.h	1010
persisted_variable.h	1013
variable.h	1015
variable_factory.h	1018

variable_model.h	1021
waypoint	71
waypoint.h	1024
waypoint_model.h	1027
application_api.h	738
extension_info.h	762
system_api.h	958
class_forward.h	659
installation	63
installation_api_provider.h	709
installation_contribution_configuration.h	711
installation_creation_context.h	712
installation_node_contribution.h	714
installation_node_service.h	716
installation_node_view.h	719
installation	63
installation_api.h	788
io	64
analog_io.h	791
boolean_register.h	794
digital_io.h	796
double_register.h	799
ethernet_ip.h	801
integer_register.h	804
io.h	805
io_model.h	808
modbus_io.h	811
profinet.h	815
register.h	818
nodes	64
builtin	56
assembly_node.h	833
assignment_node.h	835
break_node.h	838
call_node.h	840
case_node.h	842
circle_move_node.h	844
comment_node.h	846
conveyor_tracking_node.h	848
default_case_node.h	851
direction_node.h	853
else_if_node.h	856
else_node.h	859
folder_node.h	861
force_node.h	863
force_option_node.h	866
gripper_node.h	868
halt_node.h	872
if_node.h	874
loop_node.h	876
move_node.h	880
offline_track_node.h	885
pallet_node.h	887
popup_node.h	889
program_home_node.h	891
record_track_node.h	894
screw_driving_node.h	896

script_node.h	898
seek_node.h	900
set_node.h	902
set_payload_node.h	906
sub_prog_node.h	910
switch_node.h	912
thread_node.h	914
timer_node.h	916
until_node.h	918
visual_localization_program_node.h	924
wait_node.h	926
waypoint_node.h	931
program_node.h	937
program_node_factory.h	939
payload	65
payload.h	820
payload_contribution_model.h	824
payload_model.h	830
program	65
program_api_provider.h	721
program_contribution_configuration.h	724
program_creation_context.h	727
program_node_contribution.h	729
program_node_service.h	732
program_node_view.h	734
program	66
nodes	64
builtin	56
assembly_node.h	833
assignment_node.h	835
break_node.h	838
call_node.h	840
case_node.h	842
circle_move_node.h	844
comment_node.h	846
conveyor_tracking_node.h	848
default_case_node.h	851
direction_node.h	853
else_if_node.h	856
else_node.h	859
folder_node.h	861
force_node.h	863
force_option_node.h	866
gripper_node.h	868
halt_node.h	872
if_node.h	874
loop_node.h	876
move_node.h	880
offline_track_node.h	885
pallet_node.h	887
popup_node.h	889
program_home_node.h	891
record_track_node.h	894
screw_driving_node.h	896
script_node.h	898
seek_node.h	900
set_node.h	902

set_payload_node.h	906
sub_prog_node.h	910
switch_node.h	912
thread_node.h	914
timer_node.h	916
until_node.h	918
visual_localization_program_node.h	924
wait_node.h	926
waypoint_node.h	931
program_node.h	937
program_node_factory.h	939
structure	68
tree_node.h	949
program_api.h	945
program_model.h	947
screw_driver	67
screw_driver_api_provider.h	692
screw_driver_configuration.h	694
screw_driver_contribution.h	702
screw_driver_parameters.h	705
script	67
script_writer.h	955
structure	68
tree_node.h	949
tcp	68
tcp.h	966
tcp_contribution_model.h	969
tcp_model.h	972
undoredo	69
undo_redo_manager.h	975
user_interface	69
input_validator.h	978
keyboard_manager.h	981
popup_manager.h	984
render_interface.h	986
user_interaction.h	995
user_interface_api.h	1000
value	70
expression	60
expression.h	1002
value_factory_provider.h	1004
variable	71
feature_variable.h	1007
global_variable.h	1010
persisted_variable.h	1013
variable.h	1015
variable_factory.h	1018
variable_model.h	1021
waypoint	71
waypoint.h	1024
waypoint_model.h	1027

Chapter 4

Namespace Index

4.1 Namespace List

Here is a list of all namespaces with brief descriptions:

arcs	73
arcs::aubo_scope	73
arcs::aubo_sdk	97

Chapter 5

Hierarchical Index

5.1 Class Hierarchy

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

arcs::aubo_scope::AnalogIoFilter	104
arcs::aubo_scope::ApplicationApi	105
arcs::aubo_scope::InstallationApi	304
arcs::aubo_scope::ProgramApi	408
arcs::aubo_scope::AuboCapInfo	118
arcs::aubo_scope::ContributionProvider< T >	141
arcs::aubo_scope::DataModel	147
arcs::aubo_scope::DefaultSingletonTraits< Type >	170
arcs::aubo_scope::Device	171
arcs::aubo_scope::GripperDevice	286
arcs::aubo_scope::DeviceManager	173
arcs::aubo_scope::DeviceRegistrationManager	174
arcs::aubo_scope::DigitalIoFilter	180
arcs::aubo_scope::DriveScrewNotOKParameters	194
arcs::aubo_scope::DriveScrewOKParameters	195
arcs::aubo_scope::EtherNetIPFilter	208
arcs::aubo_scope::Expression	209
arcs::aubo_scope::Feature	213
arcs::aubo_scope::BaseFeature	119
arcs::aubo_scope::FeatureLine	224
arcs::aubo_scope::FeaturePlane	231
arcs::aubo_scope::FeaturePoint	235
arcs::aubo_scope::FeatureTool	238
arcs::aubo_scope::FeatureContributionModel	218
arcs::aubo_scope::FeatureDefinedFilter	222
arcs::aubo_scope::FeatureJoggableFilter	223
arcs::aubo_scope::FeatureModel	227
arcs::aubo_scope::FeatureVariableFilter	246
arcs::aubo_scope::FeedScrewParameters	246
arcs::aubo_scope::Function	257
arcs::aubo_scope::FunctionModel	261
arcs::aubo_scope::GlobalVariableFilter	269
arcs::aubo_scope::GripActionParameters	270
arcs::aubo_scope::GripForceCapability	273

arcs::aubo_scope::GripperApiProvider	274
arcs::aubo_scope::GripperConfiguration	275
arcs::aubo_scope::GripperContribution	282
arcs::aubo_scope::GripVacuumCapability	292
arcs::aubo_scope::InputIoFilter	301
arcs::aubo_scope::InputValidator	302
arcs::aubo_scope::DoubleRangeValidator	189
arcs::aubo_scope::IntegerRangeValidator	319
arcs::aubo_scope::StringLengthValidator	531
arcs::aubo_scope::UIntegerRangeValidator	573
arcs::aubo_scope::InstallationApiProvider	307
arcs::aubo_scope::InstallationContributionConfiguration	310
arcs::aubo_scope::InstallationCreationContext	311
arcs::aubo_scope::InstallationNodeContribution	314
arcs::aubo_scope::InstallationNodeService	315
arcs::aubo_scope::InstallationNodeView	318
arcs::aubo_scope::Io	324
arcs::aubo_scope::AnalogIo	99
arcs::aubo_scope::DigitalIo	175
arcs::aubo_scope::EtherNetIP	203
arcs::aubo_scope::ModbusIo	349
arcs::aubo_scope::Profinet	401
arcs::aubo_scope::Register	456
arcs::aubo_scope::BoolRegister	123
arcs::aubo_scope::DoubleRegister	192
arcs::aubo_scope::IntegerRegister	322
arcs::aubo_scope::IoConfigure	330
arcs::aubo_scope::IoModel	332
arcs::aubo_scope::KeyboardManager	336
arcs::aubo_scope::ModbusIoFilter	354
arcs::aubo_scope::OutputIoFilter	367
arcs::aubo_scope::Payload	371
arcs::aubo_scope::PayloadContributionModel	376
arcs::aubo_scope::PayloadModel	384
arcs::aubo_scope::PersistedVariableFilter	392
arcs::aubo_scope::PopupManager	392
arcs::aubo_scope::PrepareToStartScrewdriverParameters	400
arcs::aubo_scope::ProfinetFilter	406
arcs::aubo_scope::ProgramApiProvider	411
arcs::aubo_scope::ProgramContributionConfiguration	414
arcs::aubo_scope::ProgramCreationContext	421
arcs::aubo_scope::ProgramModel	427
arcs::aubo_scope::ProgramNode	431
arcs::aubo_scope::AssemblyNode	109
arcs::aubo_scope::AssignmentNode	113
arcs::aubo_scope::BreakNode	125
arcs::aubo_scope::CallNode	128
arcs::aubo_scope::CaseNode	132
arcs::aubo_scope::CircleMoveNode	135
arcs::aubo_scope::CommentNode	138
arcs::aubo_scope::ConveyorTrackingNode	143
arcs::aubo_scope::DefaultCaseNode	167
arcs::aubo_scope::DirectionNode	181
arcs::aubo_scope::ElseIfNode	196
arcs::aubo_scope::ElseNode	200
arcs::aubo_scope::FolderNode	247
arcs::aubo_scope::ForceNode	251
arcs::aubo_scope::ForceOptionNode	254

arcs::aubo_scope::GripperNode	287
arcs::aubo_scope::HaltNode	294
arcs::aubo_scope::IfNode	297
arcs::aubo_scope::LoopNode	343
arcs::aubo_scope::MoveNode	355
arcs::aubo_scope::OfflineTrackNode	364
arcs::aubo_scope::PalletNode	367
arcs::aubo_scope::PopupNode	396
arcs::aubo_scope::ProgramHomeNode	424
arcs::aubo_scope::RecordTrackNode	453
arcs::aubo_scope::ScrewdrivingNode	495
arcs::aubo_scope::ScriptNode	498
arcs::aubo_scope::SeekNode	513
arcs::aubo_scope::SetNode	516
arcs::aubo_scope::SetPayloadNode	525
arcs::aubo_scope::SubProgNode	534
arcs::aubo_scope::SwitchNode	538
arcs::aubo_scope::ThreadNode	558
arcs::aubo_scope::TimerNode	561
arcs::aubo_scope::UntilNode	579
arcs::aubo_scope::VisualLocalizationProgramNode	621
arcs::aubo_scope::WaitNode	625
arcs::aubo_scope::WaypointNode	644
arcs::aubo_scope::ProgramNodeContribution	435
arcs::aubo_scope::ProgramNodeFactory	438
arcs::aubo_scope::ProgramNodeService	447
arcs::aubo_scope::ProgramNodeView	450
arcs::aubo_scope::ProgramSelectionParameters	451
arcs::aubo_scope::RegisteredCapabilities	457
arcs::aubo_scope::ReleaseActionParameters	460
arcs::aubo_scope::RenderInterface	462
arcs::aubo_scope::ScrewdriverApiProvider	480
arcs::aubo_scope::ScrewdriverConfiguration	481
arcs::aubo_scope::ScrewdriverContribution	488
arcs::aubo_scope::ScrewdriverParameters	491
arcs::aubo_scope::ScrewdriverProgram	493
arcs::aubo_scope::ScrewdriverReadyParameters	494
arcs::aubo_scope::ScriptWriter	501
arcs::aubo_scope::SpeedCapability	530
arcs::aubo_scope::SystemApi	541
arcs::aubo_scope::TCP	545
arcs::aubo_scope::TcpContributionModel	548
arcs::aubo_scope::TcpModel	553
arcs::aubo_scope::TreeNode	564
arcs::aubo_scope::UndoRedoManager	576
arcs::aubo_scope::UserInteraction	591
arcs::aubo_scope::UserInterfaceApi	599
arcs::aubo_scope::ValueFactory	601
arcs::aubo_scope::Variable	605
arcs::aubo_scope::GlobalVariable	266
arcs::aubo_scope::PersistedVariable	388
arcs::aubo_scope::FeatureVariable	242
arcs::aubo_scope::VariableFactory	609
arcs::aubo_scope::VariableModel	613
arcs::aubo_scope::ViewApiProvider	618
arcs::aubo_scope::Waypoint	635
arcs::aubo_scope::WaypointModel	639
arcs::aubo_scope::WidthCapability	657

Chapter 6

Class Index

6.1 Class List

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

arcs::aubo_scope::AnalogIo	
AnalogIo This interface provides support for analog I/Os	99
arcs::aubo_scope::AnalogIoFilter	104
arcs::aubo_scope::ApplicationApi	105
arcs::aubo_scope::AssemblyNode	109
arcs::aubo_scope::AssignmentNode	113
arcs::aubo_scope::AuboCapInfo	
This interface provides various metadata information about an installed AuboCap, such as the name of the AuboCap	118
arcs::aubo_scope::BaseFeature	
This interface represents the Base feature in AuboScope	119
arcs::aubo_scope::BoolRegister	
This interface provides support for bool registers	123
arcs::aubo_scope::BreakNode	125
arcs::aubo_scope::CallNode	128
arcs::aubo_scope::CaseNode	132
arcs::aubo_scope::CircleMoveNode	135
arcs::aubo_scope::CommentNode	138
arcs::aubo_scope::ContributionProvider< T >	
ContributionProvider This interfaces provides access to a contribution	141
arcs::aubo_scope::ConveyorTrackingNode	143
arcs::aubo_scope::DataModel	
DataModel This interface is used for storing and managing data that represents the current configuration of, e.g	147
arcs::aubo_scope::DefaultCaseNode	167
arcs::aubo_scope::DefaultSingletonTraits< Type >	170
arcs::aubo_scope::Device	
Device This base interface represents a device supported by AuboScope, e.g	171
arcs::aubo_scope::DeviceManager	
DeviceManager Base interface for device managers	173
arcs::aubo_scope::DeviceRegistrationManager	
DeviceRegistrationManager Base interface for device registration managers	174
arcs::aubo_scope::DigitalIo	
DigitalIo This interface provides support for digital I/Os	175
arcs::aubo_scope::DigitalIoFilter	180

arcs::aubo_scope::DirectionNode	181
arcs::aubo_scope::DoubleRangeValidator	189
arcs::aubo_scope::DoubleRegister	
This interface provides support for float registers	192
arcs::aubo_scope::DriveScrewNotOKParameters	194
arcs::aubo_scope::DriveScrewOKParameters	195
arcs::aubo_scope::ElseIfNode	196
arcs::aubo_scope::ElseNode	200
arcs::aubo_scope::EtherNetIP	
EtherNetIP This interface provides support for EtherNet/IP I/Os	203
arcs::aubo_scope::EtherNetIPFilter	208
arcs::aubo_scope::Expression	
Expression An expression can be used for, e.g	209
arcs::aubo_scope::Feature	
Feature This base interface represents all types of features available in AuboScope	213
arcs::aubo_scope::FeatureContributionModel	
FeatureContributionModel This interface provides functionality for adding, updating and removing features in AuboScope	218
arcs::aubo_scope::FeatureDefinedFilter	222
arcs::aubo_scope::FeatureJoggableFilter	223
arcs::aubo_scope::FeatureLine	224
arcs::aubo_scope::FeatureModel	
FeatureModel Provides methods that returns features from the current robot installa- tion	227
arcs::aubo_scope::FeaturePlane	231
arcs::aubo_scope::FeaturePoint	
This interface represents a Point feature in AuboScope	235
arcs::aubo_scope::FeatureTool	
This interface represents the Tool feature in AuboScope	238
arcs::aubo_scope::FeatureVariable	
FeatureVariable A Feature marked as a Variable in the installation will exist even when a robot is turned off and on	242
arcs::aubo_scope::FeatureVariableFilter	246
arcs::aubo_scope::FeedScrewParameters	246
arcs::aubo_scope::FolderNode	247
arcs::aubo_scope::ForceNode	251
arcs::aubo_scope::ForceOptionNode	254
arcs::aubo_scope::Function	
Function Represents a function shown in the Expression Editor	257
arcs::aubo_scope::FunctionModel	
FunctionModel This interface provides methods to add and remove script functions in the Expression Editor accessible from Assignment and If program nodes among others	261
arcs::aubo_scope::GlobalVariable	
Global variables are available everywhere in a program	266
arcs::aubo_scope::GlobalVariableFilter	269
arcs::aubo_scope::GripActionParameters	270
arcs::aubo_scope::GripForceCapability	
GripForceCapability This interface represents a registered force capability for a gripper which supports gripping using a user configurable force	273
arcs::aubo_scope::GripperApiProvider	
GripperApiProvider This interface provides access to functionality and services avail- able from within AuboScope relevant for gripper contributions	274
arcs::aubo_scope::GripperConfiguration	
This interface provides access to register or setup various properties and capabilities of a gripper	275
arcs::aubo_scope::GripperContribution	
GripperContribution This interface defines an API for adding functionality for gripping devices to AuboScope	282

arcs::aubo_scope::GripperDevice	
GripperDevice	This interface represents a gripper device in AuboScope 286
arcs::aubo_scope::GripperNode	287
arcs::aubo_scope::GripVacuumCapability	
GripVacuumCapability	This interface represents a registered vacuum capability for a gripper which supports gripping using a user configurable vacuum level 292
arcs::aubo_scope::HaltNode	 294
arcs::aubo_scope::IfNode	 297
arcs::aubo_scope::InputIoFilter	 301
arcs::aubo_scope::InputValidator	
InputValidator	Interface representing the input validators created by InputValidationFactory 302
arcs::aubo_scope::InstallationApi	 304
arcs::aubo_scope::InstallationApiProvider	
InstallationApiProvider	Provides access to functionality and services relevant for installation nodes 307
arcs::aubo_scope::InstallationContributionConfiguration	 310
arcs::aubo_scope::InstallationCreationContext	
InstallationCreationContext	Provides information about the context in which an installation node is created 311
arcs::aubo_scope::InstallationNodeContribution	
InstallationNodeContribution	API for installation node contributions 314
arcs::aubo_scope::InstallationNodeService	
InstallationNodeService	Defines an API required for defining and adding to AuboScope an installation node and corresponding screen where the user interface is Qt-based . . 315
arcs::aubo_scope::InstallationNodeView	
InstallationNodeView	API for an installation node View which defines the UI for the corresponding InstallationNodeContribution 318
arcs::aubo_scope::IntegerRangeValidator	 319
arcs::aubo_scope::IntegerRegister	
This interface provides support for integer registers 322	
arcs::aubo_scope::Io	
Io	This is the base interface representing all types of I/Os available in AuboScope . . 324
arcs::aubo_scope::IoConfigure	 330
arcs::aubo_scope::IoModel	
IoModel	Provides methods that returns I/Os from the current robot installation . . . 332
arcs::aubo_scope::KeyboardManager	 336
arcs::aubo_scope::LoopNode	 343
arcs::aubo_scope::ModbusIo	
ModbusIo	This interface provides support for MODBUS I/Os 349
arcs::aubo_scope::ModbusIoFilter	 354
arcs::aubo_scope::MoveNode	 355
arcs::aubo_scope::OfflineTrackNode	 364
arcs::aubo_scope::OutputIoFilter	 367
arcs::aubo_scope::PalletNode	 367
arcs::aubo_scope::Payload	
API interface for payload parameters 371	
arcs::aubo_scope::PayloadContributionModel	
API interface for dynamic payload management 376	
arcs::aubo_scope::PayloadModel	
API interface for payload parameters 384	
arcs::aubo_scope::PersistedVariable	
Ingroup ApplicationApi	This interface represents variables stored with the installation 388
arcs::aubo_scope::PersistedVariableFilter	 392
arcs::aubo_scope::PopupManager	 392
arcs::aubo_scope::PopupNode	 396
arcs::aubo_scope::PrepareToStartScrewdriverParameters	 400

arcs::aubo_scope::Profinet	Profinet This interface provides support for PROFINET I/Os	401
arcs::aubo_scope::ProfinetFilter		406
arcs::aubo_scope::ProgramApi		408
arcs::aubo_scope::ProgramApiProvider	ProgramApiProvider Provides access to functionality and services relevant for program nodes	411
arcs::aubo_scope::ProgramContributionConfiguration	ProgramContributionConfiguration Provides access to configure the properties of an installation node contribution	414
arcs::aubo_scope::ProgramCreationContext	ProgramCreationContext Provides information about the context in which an installation node is created	421
arcs::aubo_scope::ProgramHomeNode		424
arcs::aubo_scope::ProgramModel	ProgramModel	427
arcs::aubo_scope::ProgramNode	ProgramNode This interface represents AuboScope program nodes	431
arcs::aubo_scope::ProgramNodeContribution	ProgramNodeContribution Defines an API required for specifying a AuboCap Program Node within AuboScope	435
arcs::aubo_scope::ProgramNodeFactory	ProgramNodeFactory This interfaces supplies methods to create different types of program nodes	438
arcs::aubo_scope::ProgramNodeService	ProgramNodeService Defines an API required for adding a new type of program node to AuboScope	447
arcs::aubo_scope::ProgramNodeView	ProgramNodeView API for a program node View which defines the UI for the corresponding ProgramNodeContribution	450
arcs::aubo_scope::ProgramSelectionParameters		451
arcs::aubo_scope::RecordTrackNode		453
arcs::aubo_scope::Register	This interface provides support for registers	456
arcs::aubo_scope::RegisteredCapabilities		457
arcs::aubo_scope::ReleaseActionParameters		460
arcs::aubo_scope::RenderInterface	RenderInterface Provides display option controls related to 3D rendering, including show/hide of ground, robot models, coordinate systems, trajectories and points	462
arcs::aubo_scope::ScrewdriverApiProvider	ScrewdriverApiProvider This interface provides access to functionality and services available from within AuboScope relevant for screwdriver contributions	480
arcs::aubo_scope::ScrewdriverConfiguration	This interface provides access to register or setup various properties and capabilities of a screwdriver	481
arcs::aubo_scope::ScrewdriverContribution	ScrewdriverContribution This interface defines an API for adding functionality for screwdrivers to AuboScope	488
arcs::aubo_scope::ScrewdriverParameters	This interface represents parameters for a screwdriver operation defined/configured by the end user	491
arcs::aubo_scope::ScrewdriverProgram	This abstract class represents a screwdriver program, typically defined on an external control box for the screwdriver	493
arcs::aubo_scope::ScrewdriverReadyParameters	This interface provides the parameters relevant for generating script for the Screwdriver Ready feedback operation capability (registered using ScrewdriverFeedbackCapabilities#registerScrewdriverR	494

arcs::aubo_scope::ScrewdrivingNode	495
arcs::aubo_scope::ScriptNode	498
arcs::aubo_scope::ScriptWriter	
ScriptWriter This interface provides support for generating Script code	501
arcs::aubo_scope::SeekNode	513
arcs::aubo_scope::SetNode	516
arcs::aubo_scope::SetPayloadNode	525
arcs::aubo_scope::SpeedCapability	
SpeedCapability This interface represents a registered speed capability for a gripper which supports operating using a user configurable speed	530
arcs::aubo_scope::StringLengthValidator	531
arcs::aubo_scope::SubProgNode	534
arcs::aubo_scope::SwitchNode	538
arcs::aubo_scope::SystemApi	541
arcs::aubo_scope::TCP	
TCP This interface represents a TCP in AuboScope	545
arcs::aubo_scope::TcpContributionModel	
TcpContributionModel This interface provides functionality for adding, updating and removing TCPs in AuboScope	548
arcs::aubo_scope::TcpModel	
TcpModel This interface provides access to the TCPs that are currently available in AuboScope	553
arcs::aubo_scope::ThreadNode	558
arcs::aubo_scope::TimerNode	561
arcs::aubo_scope::TreeNode	
TreeNode This interface represents a node in the program tree that can be used to construct a sub-tree rooted in a AuboCap program node	564
arcs::aubo_scope::UIntegerRangeValidator	573
arcs::aubo_scope::UndoRedoManager	
UndoRedoManager This interface is used to record changes triggered by an end user in the screen of a AuboCap program node contribution (e.g	576
arcs::aubo_scope::UntilNode	579
arcs::aubo_scope::UserInteraction	
UserInteraction This interface provides functionality for requesting input or actions from end users	591
arcs::aubo_scope::UserInterfaceApi	599
arcs::aubo_scope::ValueFactory	
ValueFactory	601
arcs::aubo_scope::Variable	
Variable Represents a variable in AuboScope	605
arcs::aubo_scope::VariableFactory	609
arcs::aubo_scope::VariableModel	
VariableModel Provides methods that returns the variables that currently are registered in the system	613
arcs::aubo_scope::ViewApiProvider	
ViewApiProvider Provides access to functionality relevant for the View (UI) of an installation or a program node contribution	618
arcs::aubo_scope::VisualLocalizationProgramNode	621
arcs::aubo_scope::WaitNode	625
arcs::aubo_scope::Waypoint	
Waypoint This interface composes parameters that define a position of the robot	635
arcs::aubo_scope::WaypointModel	
WaypointModel Provides methods that returns Waypoints from the current robot installation	639
arcs::aubo_scope::WaypointNode	644

[arcs::aubo_scope::WidthCapability](#)

[WidthCapability](#) This interface represents a registered width/position capability for a gripper which supports moving to a user configurable position (open/close to a configurable width) [657](#)

Chapter 7

File Index

7.1 File List

Here is a list of all files with brief descriptions:

include/aubo_caps/class_forward.h	659
include/aubo_caps/contribution/contribution_provider.h	662
include/aubo_caps/contribution/view_api_provider.h	736
include/aubo_caps/contribution/driver/gripper/grip_action_parameters.h	677
include/aubo_caps/contribution/driver/gripper/gripper_api_provider.h	679
include/aubo_caps/contribution/driver/gripper/gripper_configuration.h	681
include/aubo_caps/contribution/driver/gripper/gripper_contribution.h	686
include/aubo_caps/contribution/driver/gripper/release_action_parameters.h	690
include/aubo_caps/contribution/driver/gripper/capability/grip_force_capability.h	664
include/aubo_caps/contribution/driver/gripper/capability/grip_vacuum_capability.h	667
include/aubo_caps/contribution/driver/gripper/capability/registered_cpabilities.h	669
include/aubo_caps/contribution/driver/gripper/capability/speed_capability.h	672
include/aubo_caps/contribution/driver/gripper/capability/width_capability.h	674
include/aubo_caps/contribution/driver/screw_driver/screw_driver_api_provider.h	692
include/aubo_caps/contribution/driver/screw_driver/screw_driver_configuration.h	694
include/aubo_caps/contribution/driver/screw_driver/screw_driver_contribution.h	702
include/aubo_caps/contribution/driver/screw_driver/screw_driver_parameters.h	705
include/aubo_caps/contribution/installation/installation_api_provider.h	709
include/aubo_caps/contribution/installation/installation_contribution_configuration.h	711
include/aubo_caps/contribution/installation/installation_creation_context.h	712
include/aubo_caps/contribution/installation/installation_node_contribution.h	714
include/aubo_caps/contribution/installation/installation_node_service.h	716
include/aubo_caps/contribution/installation/installation_node_view.h	719
include/aubo_caps/contribution/program/program_api_provider.h	721
include/aubo_caps/contribution/program/program_contribution_configuration.h	724
include/aubo_caps/contribution/program/program_creation_context.h	727
include/aubo_caps/contribution/program/program_node_contribution.h	729
include/aubo_caps/contribution/program/program_node_service.h	732
include/aubo_caps/contribution/program/program_node_view.h	734
include/aubo_caps/domain/application_api.h	738
include/aubo_caps/domain/extension_info.h	762
include/aubo_caps/domain/system_api.h	958
include/aubo_caps/domain/data/data_model.h	742
include/aubo_caps/domain/device/device.h	752
include/aubo_caps/domain/device/device_manager.h	755

include/aubo_caps/domain/device/device_registration_manager.h	758
include/aubo_caps/domain/device/gripper_device.h	760
include/aubo_caps/domain/feature/base_feature.h	765
include/aubo_caps/domain/feature/feature.h	766
include/aubo_caps/domain/feature/feature_contribution_model.h	769
include/aubo_caps/domain/feature/feature_line.h	772
include/aubo_caps/domain/feature/feature_model.h	774
include/aubo_caps/domain/feature/feature_plane.h	777
include/aubo_caps/domain/feature/feature_point.h	779
include/aubo_caps/domain/feature/feature_tool.h	781
include/aubo_caps/domain/function/function.h	782
include/aubo_caps/domain/function/function_model.h	785
include/aubo_caps/domain/installation/installation_api.h	788
include/aubo_caps/domain/io/analog_io.h	791
include/aubo_caps/domain/io/boolean_register.h	794
include/aubo_caps/domain/io/digital_io.h	796
include/aubo_caps/domain/io/double_register.h	799
include/aubo_caps/domain/io/ethernet_ip.h	801
include/aubo_caps/domain/io/integer_register.h	804
include/aubo_caps/domain/io/io.h	805
include/aubo_caps/domain/io/io_model.h	808
include/aubo_caps/domain/io/modbus_io.h	811
include/aubo_caps/domain/io/profinet.h	815
include/aubo_caps/domain/io/register.h	818
include/aubo_caps/domain/payload/payload.h	820
include/aubo_caps/domain/payload/payload_contribution_model.h	824
include/aubo_caps/domain/payload/payload_model.h	830
include/aubo_caps/domain/program/program_api.h	945
include/aubo_caps/domain/program/program_model.h	947
include/aubo_caps/domain/program/nodes/program_node.h	937
include/aubo_caps/domain/program/nodes/program_node_factory.h	939
include/aubo_caps/domain/program/nodes/builtin/assembly_node.h	833
include/aubo_caps/domain/program/nodes/builtin/assignment_node.h	835
include/aubo_caps/domain/program/nodes/builtin/break_node.h	838
include/aubo_caps/domain/program/nodes/builtin/call_node.h	840
include/aubo_caps/domain/program/nodes/builtin/case_node.h	842
include/aubo_caps/domain/program/nodes/builtin/circle_move_node.h	844
include/aubo_caps/domain/program/nodes/builtin/comment_node.h	846
include/aubo_caps/domain/program/nodes/builtin/conveyor_tracking_node.h	848
include/aubo_caps/domain/program/nodes/builtin/default_case_node.h	851
include/aubo_caps/domain/program/nodes/builtin/direction_node.h	853
include/aubo_caps/domain/program/nodes/builtin/else_if_node.h	856
include/aubo_caps/domain/program/nodes/builtin/else_node.h	859
include/aubo_caps/domain/program/nodes/builtin/folder_node.h	861
include/aubo_caps/domain/program/nodes/builtin/force_node.h	863
include/aubo_caps/domain/program/nodes/builtin/force_option_node.h	866
include/aubo_caps/domain/program/nodes/builtin/gripper_node.h	868
include/aubo_caps/domain/program/nodes/builtin/halt_node.h	872
include/aubo_caps/domain/program/nodes/builtin/if_node.h	874
include/aubo_caps/domain/program/nodes/builtin/loop_node.h	876
include/aubo_caps/domain/program/nodes/builtin/move_node.h	880
include/aubo_caps/domain/program/nodes/builtin/offline_track_node.h	885
include/aubo_caps/domain/program/nodes/builtin/pallet_node.h	887
include/aubo_caps/domain/program/nodes/builtin/popup_node.h	889
include/aubo_caps/domain/program/nodes/builtin/program_home_node.h	891
include/aubo_caps/domain/program/nodes/builtin/record_track_node.h	894
include/aubo_caps/domain/program/nodes/builtin/screw_driving_node.h	896
include/aubo_caps/domain/program/nodes/builtin/script_node.h	898

include/aubo_caps/domain/program/nodes/builtin/seek_node.h	900
include/aubo_caps/domain/program/nodes/builtin/set_node.h	902
include/aubo_caps/domain/program/nodes/builtin/set_payload_node.h	906
include/aubo_caps/domain/program/nodes/builtin/sub_prog_node.h	910
include/aubo_caps/domain/program/nodes/builtin/switch_node.h	912
include/aubo_caps/domain/program/nodes/builtin/thread_node.h	914
include/aubo_caps/domain/program/nodes/builtin/timer_node.h	916
include/aubo_caps/domain/program/nodes/builtin/until_node.h	918
include/aubo_caps/domain/program/nodes/builtin/visual_localization_program_node.h . .	924
include/aubo_caps/domain/program/nodes/builtin/wait_node.h	926
include/aubo_caps/domain/program/nodes/builtin/waypoint_node.h	931
include/aubo_caps/domain/program/structure/tree_node.h	949
include/aubo_caps/domain/script/script_writer.h	955
include/aubo_caps/domain/tcp/tcp.h	966
include/aubo_caps/domain/tcp/tcp_contribution_model.h	969
include/aubo_caps/domain/tcp/tcp_model.h	972
include/aubo_caps/domain/undoredo/undo_redo_manager.h	975
include/aubo_caps/domain/user_interface/input_validator.h	978
include/aubo_caps/domain/user_interface/keyboard_manager.h	981
include/aubo_caps/domain/user_interface/popup_manager.h	984
include/aubo_caps/domain/user_interface/render_interface.h	986
include/aubo_caps/domain/user_interface/user_interaction.h	995
include/aubo_caps/domain/user_interface/user_interface_api.h	1000
include/aubo_caps/domain/value/value_factory_provider.h	1004
include/aubo_caps/domain/value/expression/expression.h	1002
include/aubo_caps/domain/variable/feature_variable.h	1007
include/aubo_caps/domain/variable/global_variable.h	1010
include/aubo_caps/domain/variable/persisted_variable.h	1013
include/aubo_caps/domain/variable/variable.h	1015
include/aubo_caps/domain/variable/variable_factory.h	1018
include/aubo_caps/domain/variable/variable_model.h	1021
include/aubo_caps/domain/waypoint/waypoint.h	1024
include/aubo_caps/domain/waypoint/waypoint_model.h	1027

Chapter 8

Topic Documentation

8.1 ApplicationApi

[ApplicationApi](#) Provides access to functionality and services which are relevant for various different applications and contexts.

Classes

- class [arcs::aubo_scope::DeviceManager](#)
[DeviceManager](#) Base interface for device managers.
- class [arcs::aubo_scope::FeatureModel](#)
[FeatureModel](#) Provides methods that returns features from the current robot installation.
- class [arcs::aubo_scope::IoModel](#)
[IoModel](#) Provides methods that returns I/Os from the current robot installation.
- class [arcs::aubo_scope::ModbusIo](#)
[ModbusIo](#) This interface provides support for MODBUS I/Os.
- class [arcs::aubo_scope::Profinet](#)
[Profinet](#) This interface provides support for PROFINET I/Os.
- class [arcs::aubo_scope::PayloadModel](#)
[PayloadModel](#) API interface for payload parameters.
- class [arcs::aubo_scope::TcpModel](#)
[TcpModel](#) This interface provides access to the TCPs that are currently available in AuboScope.
- class [arcs::aubo_scope::ValueFactory](#)
[ValueFactory](#)
- class [arcs::aubo_scope::Variable](#)
[Variable](#) Represents a variable in AuboScope.
- class [arcs::aubo_scope::VariableFactory](#)
- class [arcs::aubo_scope::VariableModel](#)
[VariableModel](#) Provides methods that returns the variables that currently are registered in the system.

Functions

- IoModelPtr [arcs::aubo_scope::ApplicationApi::getIoModel \(\)](#)
[IoModel](#) Returns an interface for working with the inputs and outputs of the robot.
- FeatureModelPtr [arcs::aubo_scope::ApplicationApi::getFeatureModel \(\)](#)
[FeatureModel](#) Returns an interface for working with the features of the current installation.
- TcpModelPtr [arcs::aubo_scope::ApplicationApi::getTcpModel \(\)](#)
[TcpModel](#) Returns an interface for accessing the TCPs of the current installation.
- PayloadModelPtr [arcs::aubo_scope::ApplicationApi::getPayloadModel \(\)](#)
[PayloadModel](#) Gets an interface which provides access the payloads in the current AuboScope installation.
- VariableModelPtr [arcs::aubo_scope::ApplicationApi::getVariableModel \(\)](#)
[VariableModel](#) Returns an interface for working with variables.
- ValueFactoryPtr [arcs::aubo_scope::ApplicationApi::getValueFactory \(\)](#)
[ValueFactory](#) Returns an interface for accessing various factories capable of creating value objects.
- DeviceManagerPtr [arcs::aubo_scope::ApplicationApi::getDeviceManager \(\)](#)
[DeviceManager](#) This method can be used to get a specific [DeviceManager](#) instance.

8.1.1 Detailed Description

[ApplicationApi](#) Provides access to functionality and services which are relevant for various different applications and contexts.

This includes what is available from within AuboScope, as well as creating additional model elements to be used within AuboScope.

8.1.2 Function Documentation

8.1.2.1 getDeviceManager()

DeviceManagerPtr [arcs::aubo_scope::ApplicationApi::getDeviceManager \(\)](#)

[DeviceManager](#) This method can be used to get a specific [DeviceManager](#) instance.

Parameters

deviceManagerClass	The class for the device manager to return.
<T>	The generic type of class to return.

Returns

An interface for registering and working with a specific device.

8.1.2.2 getFeatureModel()

FeatureModelPtr arcs::aubo_scope::ApplicationApi::getFeatureModel ()

[FeatureModel](#) Returns an interface for working with the features of the current installation.

Returns

An interface for working with the features of the current installation.

8.1.2.3 getIoModel()

IoModelPtr arcs::aubo_scope::ApplicationApi::getIoModel ()

[IoModel](#) Returns an interface for working with the inputs and outputs of the robot.

Returns

An interface for working with the inputs and outputs of the robot.

8.1.2.4 getPayloadModel()

PayloadModelPtr arcs::aubo_scope::ApplicationApi::getPayloadModel ()

[PayloadModel](#) Gets an interface which provides access the payloads in the current AuboScope installation.

Returns

An interface for accessing the payloads in the current AuboScope installation.

8.1.2.5 getTcpModel()

TcpModelPtr arcs::aubo_scope::ApplicationApi::getTcpModel ()

[TcpModel](#) Returns an interface for accessing the TCPs of the current installation.

Returns

An interface for accessing the TCPs of the current installation.

8.1.2.6 getValueFactory()

ValueFactoryPtr arcs::aubo_scope::ApplicationApi::getValueFactory ()

[ValueFactory](#) Returns an interface for accessing various factories capable of creating value objects.

Returns

An interface for accessing various factories capable of creating value objects.

8.1.2.7 `getVariableModel()`

`VariableModelPtr arcs::aubo_scope::ApplicationApi::getVariableModel ()`

[VariableModel](#) Returns an interface for working with variables.

Returns

An interface for working with variables.

8.2 InstallationApi

[InstallationApi](#) Provides access to functionality and services related to the installation.

Classes

- class [arcs::aubo_scope::FeatureContributionModel](#)
[FeatureContributionModel](#) This interface provides functionality for adding, updating and removing features in AuboScope.
- class [arcs::aubo_scope::FunctionModel](#)
[FunctionModel](#) This interface provides methods to add and remove script functions in the [Expression Editor](#) accessible from Assignment and If program nodes among others.
- class [arcs::aubo_scope::Payload](#)
 API interface for payload parameters.
- class [arcs::aubo_scope::PayloadContributionModel](#)
 API interface for dynamic payload management.
- class [arcs::aubo_scope::TCP](#)
[TCP](#) This interface represents a [TCP](#) in AuboScope.
- class [arcs::aubo_scope::TcpContributionModel](#)
[TcpContributionModel](#) This interface provides functionality for adding, updating and removing TCPs in AuboScope.

Functions

- FunctionModelPtr [arcs::aubo_scope::InstallationApi::getFunctionModel \(\)](#)
 Returns an interface for adding script functions to AuboScope.
- TcpContributionModelPtr [arcs::aubo_scope::InstallationApi::getTcpContributionModel \(\)](#)
 Returns an interface for contributing TCPs to AuboScope.
- FeatureContributionModelPtr [arcs::aubo_scope::InstallationApi::getFeatureContributionModel \(\)](#)
 Returns an interface for contributing features to AuboScope.
- PayloadContributionModelPtr [arcs::aubo_scope::InstallationApi::getPayloadContributionModel \(\)](#)

8.2.1 Detailed Description

[InstallationApi](#) Provides access to functionality and services related to the installation.

This includes what is available from within AuboScope, as well as creating additional model elements to be used within AuboScope.

8.2.2 Function Documentation

8.2.2.1 getFeatureContributionModel()

FeatureContributionModelPtr arcs::aubo_scope::InstallationApi::getFeatureContributionModel ()

Returns an interface for contributing features to AuboScope.

Returns

An interface for contributing features to AuboScope.

8.2.2.2 getFunctionModel()

FunctionModelPtr arcs::aubo_scope::InstallationApi::getFunctionModel ()

Returns an interface for adding script functions to AuboScope.

Returns

An interface for adding script functions to AuboScope.

8.2.2.3 getPayloadContributionModel()

PayloadContributionModelPtr arcs::aubo_scope::InstallationApi::getPayloadContributionModel ()

Gets an interface which provides functionality for contributing payloads to the AuboScope installation.

Returns

An interface for contributing payloads to AuboScope.

8.2.2.4 getTcpContributionModel()

TcpContributionModelPtr arcs::aubo_scope::InstallationApi::getTcpContributionModel ()

Returns an interface for contributing TCPs to AuboScope.

Returns

An interface for contributing TCPs to AuboScope.

8.3 ProgramApi

[ProgramApi](#) Provides access to functionality and services related to programs.

Classes

- class [arcs::aubo_scope::ProgramModel](#)
[ProgramModel](#)
- class [arcs::aubo_scope::UndoRedoManager](#)
[UndoRedoManager](#) This interface is used to record changes triggered by an end user in the screen of a AuboCap program node contribution (e.g.
- class [arcs::aubo_scope::Waypoint](#)
[Waypoint](#) This interface composes parameters that define a position of the robot.
- class [arcs::aubo_scope::WaypointModel](#)
[WaypointModel](#) Provides methods that returns Waypoints from the current robot installation.

Functions

- ProgramModelPtr [arcs::aubo_scope::ProgramApi::getProgramModel](#) ()
[ProgramModel](#)
- WaypointModelPtr [arcs::aubo_scope::ProgramApi::getWaypointModel](#) ()
[WaypointModel](#)
- UndoRedoManagerPtr [arcs::aubo_scope::ProgramApi::getUndoRedoManager](#) ()
[UndoRedoManager](#)

8.3.1 Detailed Description

[ProgramApi](#) Provides access to functionality and services related to programs.

This includes what is available from within AuboScope, as well as creating additional model elements to be used within AuboScope.

8.3.2 Function Documentation

8.3.2.1 getProgramModel()

ProgramModelPtr [arcs::aubo_scope::ProgramApi::getProgramModel](#) ()

[ProgramModel](#)

Returns

An interface for working with the sub-trees of a program node

8.3.2.2 getUndoRedoManager()

UndoRedoManagerPtr [arcs::aubo_scope::ProgramApi::getUndoRedoManager](#) ()

[UndoRedoManager](#)

Returns

An interface for recording undoable changes for program node contributions

8.3.2.3 getWaypointModel()

WaypointModelPtr arcs::aubo_scope::ProgramApi::getWaypointModel ()

WaypointModel

Returns

An interface for working with waypoints

8.4 SystemApi

[SystemApi](#) Provides access to system information of the AuboScope software, including software version numbers, system locale, unit types, the robot serial number, as well as robot joint position limits, velocity, acceleration, and payload parameters.

Functions

- int [arcs::aubo_scope::SystemApi::getMajorVersion](#) ()
Get the version of Software.
- int [arcs::aubo_scope::SystemApi::getMinorVersion](#) ()
Get the minor version of Software.
- int [arcs::aubo_scope::SystemApi::getBugfixVersion](#) ()
Get the bug fix version of Software.
- int [arcs::aubo_scope::SystemApi::getBuildNumber](#) ()
Get the build number of Software.
- std::string [arcs::aubo_scope::SystemApi::getLocale](#) ()
Returns the Locale for the system.
- std::string [arcs::aubo_scope::SystemApi::getLocaleForProgrammingLanguage](#) ()
Returns the Locale for the programming language used in the system.
- [UnitType](#) [arcs::aubo_scope::SystemApi::getUnitType](#) ()
Returns the [UnitType](#) of the system as an enum.
- std::string [arcs::aubo_scope::SystemApi::getSerialNumber](#) ()
Returns the serial number of the robot.
- bool [arcs::aubo_scope::SystemApi::isRealRobot](#) ()
Returns true if controlling the real, physical robot and false if in simulation mode.
- std::vector< double > [arcs::aubo_scope::SystemApi::getJointMaxPositions](#) ()
Returns the max positions of robot joints.
- std::vector< double > [arcs::aubo_scope::SystemApi::getJointMinPositions](#) ()
Returns the min positions of robot joints.
- double [arcs::aubo_scope::SystemApi::getJointDefaultVelocity](#) ()
Returns the default velocity of robot joints.
- double [arcs::aubo_scope::SystemApi::getJointDefaultAcceleration](#) ()
Returns the default acceleration of robot joints.
- std::vector< double > [arcs::aubo_scope::SystemApi::getJointMaxVelocity](#) ()
Returns the max velocity of robot joints.
- std::vector< double > [arcs::aubo_scope::SystemApi::getJointMaxAcceleration](#) ()
Returns the max acceleration of robot joints.
- double [arcs::aubo_scope::SystemApi::getTcpDefaultVelocity](#) ()

- Returns the default velocity of TCP.

 - double `arcs::aubo_scope::SystemApi::getTcpDefaultAcceleration ()`

Returns the default acceleration of TCP.
- double `arcs::aubo_scope::SystemApi::getTcpMaxVelocity ()`

Returns the max velocity of TCP.
- double `arcs::aubo_scope::SystemApi::getTcpMaxAcceleration ()`

Returns the max acceleration of TCP.
- double `arcs::aubo_scope::SystemApi::getTcpMaxDistance ()`

Returns the max distance of TCP.
- double `arcs::aubo_scope::SystemApi::getMaxPayloadMass ()`

Returns the max payload mass.
- double `arcs::aubo_scope::SystemApi::getMaxPayloadDistance ()`

Returns the max payload distance.
- `std::vector< double >` `arcs::aubo_scope::SystemApi::getHomeDefaultPosition ()`

Returns the default home position.
- `std::vector< double >` `arcs::aubo_scope::SystemApi::getHomePackagePosition ()`

Returns the package home position.
- `std::unordered_map< std::string, std::vector< double > >` `arcs::aubo_scope::SystemApi::getRobotDhParams (bool is_real)`

Returns the robot DH parameters.
- `IoConfigure` `arcs::aubo_scope::SystemApi::getToolDigitalIOConfigure ()`

Returns the tool digital I/O configuration.
- `IoConfigure` `arcs::aubo_scope::SystemApi::getStandardDigitalInputConfigure ()`

Returns the standard digital input configuration.
- `IoConfigure` `arcs::aubo_scope::SystemApi::getConfigurableDigitalInputConfigure ()`

Returns the configurable digital input configuration.
- `IoConfigure` `arcs::aubo_scope::SystemApi::getStandardDigitalOutputConfigure ()`

Returns the standard digital output configuration.
- `IoConfigure` `arcs::aubo_scope::SystemApi::getConfigurableDigitalOutputConfigure ()`

Returns the configurable digital output configuration.
- `IoConfigure` `arcs::aubo_scope::SystemApi::getStandardAnalogInputConfigure ()`

Returns the standard analog input configuration.
- `IoConfigure` `arcs::aubo_scope::SystemApi::getToolAnalogInputConfigure ()`

Returns the tool analog input configuration.
- `IoConfigure` `arcs::aubo_scope::SystemApi::getStandardAnalogOutputConfigure ()`

Returns the standard analog output configuration.
- `IoConfigure` `arcs::aubo_scope::SystemApi::getToolAnalogOutputConfigure ()`

Returns the tool analog output configuration.
- `IoConfigure` `arcs::aubo_scope::SystemApi::getStaticLinkInputConfigure ()`

Returns the static link input configuration.
- `IoConfigure` `arcs::aubo_scope::SystemApi::getStaticLinkOutputConfigure ()`

Returns the static link output configuration.
- `IoConfigure` `arcs::aubo_scope::SystemApi::getENCConfigure ()`

Returns the encoder configuration.

8.4.1 Detailed Description

`SystemApi` Provides access to system information of the AuboScope software, including software version numbers, system locale, unit types, the robot serial number, as well as robot joint position limits, velocity, acceleration, and payload parameters.

8.4.2 Function Documentation

8.4.2.1 getBugfixVersion()

```
int arcs::aubo_scope::SystemApi::getBugfixVersion ()
```

Get the bug fix version of Software.

Returns

the bug fix version of Software.

8.4.2.2 getBuildNumber()

```
int arcs::aubo_scope::SystemApi::getBuildNumber ()
```

Get the build number of Software.

Returns

the build number of Software.

8.4.2.3 getConfigurableDigitalInputConfigure()

```
IoConfigure arcs::aubo_scope::SystemApi::getConfigurableDigitalInputConfigure ()
```

Returns the configurable digital input configuration.

Returns

the configurable digital input configuration.

8.4.2.4 getConfigurableDigitalOutputConfigure()

```
IoConfigure arcs::aubo_scope::SystemApi::getConfigurableDigitalOutputConfigure ()
```

Returns the configurable digital output configuration.

Returns

the configurable digital output configuration.

8.4.2.5 getENCConfigure()

```
IoConfigure arcs::aubo_scope::SystemApi::getENCConfigure ()
```

Returns the encoder configuration.

Returns

the encoder configuration.

8.4.2.6 getHomeDefaultPosition()

`std::vector< double > arcs::aubo_scope::SystemApi::getHomeDefaultPosition ()`

Returns the default home position.

Returns

the default home position.

8.4.2.7 getHomePackagePosition()

`std::vector< double > arcs::aubo_scope::SystemApi::getHomePackagePosition ()`

Returns the package home position.

Returns

the package home position.

8.4.2.8 getJointDefaultAcceleration()

`double arcs::aubo_scope::SystemApi::getJointDefaultAcceleration ()`

Returns the default acceleration of robot joints.

Returns

the default acceleration of robot joints.

8.4.2.9 getJointDefaultVelocity()

`double arcs::aubo_scope::SystemApi::getJointDefaultVelocity ()`

Returns the default velocity of robot joints.

Returns

the default velocity of robot joints.

8.4.2.10 getJointMaxAcceleration()

`std::vector< double > arcs::aubo_scope::SystemApi::getJointMaxAcceleration ()`

Returns the max acceleration of robot joints.

Returns

the max acceleration of robot joints.

8.4.2.11 getJointMaxPositions()

```
std::vector< double > arcs::aubo_scope::SystemApi::getJointMaxPositions ()
```

Returns the max positions of robot joints.

Returns

Max Positions of robot joints in radian.

8.4.2.12 getJointMaxVelocity()

```
std::vector< double > arcs::aubo_scope::SystemApi::getJointMaxVelocity ()
```

Returns the max velocity of robot joints.

Returns

the max velocity of robot joints.

8.4.2.13 getJointMinPositions()

```
std::vector< double > arcs::aubo_scope::SystemApi::getJointMinPositions ()
```

Returns the min positions of robot joints.

Returns

Min Positions of robot joints in radian.

8.4.2.14 getLocale()

```
std::string arcs::aubo_scope::SystemApi::getLocale ()
```

Returns the Locale for the system.

Returns

Locale.

8.4.2.15 getLocaleForProgrammingLanguage()

```
std::string arcs::aubo_scope::SystemApi::getLocaleForProgrammingLanguage ()
```

Returns the Locale for the programming language used in the system.

Returns

Locale.

8.4.2.16 getMajorVersion()

```
int arcs::aubo_scope::SystemApi::getMajorVersion ()
```

Get the version of Software.

Returns

the major version of Software.

8.4.2.17 getMaxPayloadDistance()

```
double arcs::aubo_scope::SystemApi::getMaxPayloadDistance ()
```

Returns the max payload distance.

Returns

the max payload distance.

8.4.2.18 getMaxPayloadMass()

```
double arcs::aubo_scope::SystemApi::getMaxPayloadMass ()
```

Returns the max payload mass.

Returns

the max payload mass.

8.4.2.19 getMinorVersion()

```
int arcs::aubo_scope::SystemApi::getMinorVersion ()
```

Get the minor version of Software.

Returns

the minor version of Software.

8.4.2.20 getRobotDhParams()

```
std::unordered_map< std::string, std::vector< double > > arcs::aubo_scope::SystemApi::getRobotDhParams (
    bool is_real)
```

Returns the robot DH parameters.

Parameters

is_real	whether to get the real parameters.

Returns

the DH parameters.

8.4.2.21 getSerialNumber()

`std::string arcs::aubo_scope::SystemApi::getSerialNumber ()`

Returns the serial number of the robot.

Returns

the serial number of the robot.

8.4.2.22 getStandardAnalogInputConfigure()

`IoConfigure arcs::aubo_scope::SystemApi::getStandardAnalogInputConfigure ()`

Returns the standard analog input configuration.

Returns

the standard analog input configuration.

8.4.2.23 getStandardAnalogOutputConfigure()

`IoConfigure arcs::aubo_scope::SystemApi::getStandardAnalogOutputConfigure ()`

Returns the standard analog output configuration.

Returns

the standard analog output configuration.

8.4.2.24 getStandardDigitalInputConfigure()

`IoConfigure arcs::aubo_scope::SystemApi::getStandardDigitalInputConfigure ()`

Returns the standard digital input configuration.

Returns

the standard digital input configuration.

8.4.2.25 getStandardDigitalOutputConfigure()

[IoConfigure](#) arcs::aubo_scope::SystemApi::getStandardDigitalOutputConfigure ()

Returns the standard digital output configuration.

Returns

the standard digital output configuration.

8.4.2.26 getStaticLinkInputConfigure()

[IoConfigure](#) arcs::aubo_scope::SystemApi::getStaticLinkInputConfigure ()

Returns the static link input configuration.

Returns

the static link input configuration.

8.4.2.27 getStaticLinkOutputConfigure()

[IoConfigure](#) arcs::aubo_scope::SystemApi::getStaticLinkOutputConfigure ()

Returns the static link output configuration.

Returns

the static link output configuration.

8.4.2.28 getTcpDefaultAcceleration()

double arcs::aubo_scope::SystemApi::getTcpDefaultAcceleration ()

Returns the default acceleration of [TCP](#).

Returns

the default acceleration of [TCP](#).

8.4.2.29 getTcpDefaultVelocity()

double arcs::aubo_scope::SystemApi::getTcpDefaultVelocity ()

Returns the default velocity of [TCP](#).

Returns

the default velocity of [TCP](#).

8.4.2.30 getTcpMaxAcceleration()

double arcs::aubo_scope::SystemApi::getTcpMaxAcceleration ()

Returns the max acceleration of [TCP](#).

Returns

the max acceleration of [TCP](#).

8.4.2.31 getTcpMaxDistance()

double arcs::aubo_scope::SystemApi::getTcpMaxDistance ()

Returns the max distance of [TCP](#).

Returns

the max distance of [TCP](#).

8.4.2.32 getTcpMaxVelocity()

double arcs::aubo_scope::SystemApi::getTcpMaxVelocity ()

Returns the max velocity of [TCP](#).

Returns

the max velocity of [TCP](#).

8.4.2.33 getToolAnalogInputConfigure()

[IoConfigure](#) arcs::aubo_scope::SystemApi::getToolAnalogInputConfigure ()

Returns the tool analog input configuration.

Returns

the tool analog input configuration.

8.4.2.34 getToolAnalogOutputConfigure()

[IoConfigure](#) arcs::aubo_scope::SystemApi::getToolAnalogOutputConfigure ()

Returns the tool analog output configuration.

Returns

the tool analog output configuration.

8.4.2.35 `getToolDigitalIOConfigure()`

`IoConfigure` `arcs::aubo_scope::SystemApi::getToolDigitalIOConfigure ()`

Returns the tool digital I/O configuration.

Returns

the tool digital I/O configuration.

8.4.2.36 `getUnitType()`

`UnitType` `arcs::aubo_scope::SystemApi::getUnitType ()`

Returns the `UnitType` of the system as an enum.

Returns

`UnitType`.

8.4.2.37 `isRealRobot()`

`bool` `arcs::aubo_scope::SystemApi::isRealRobot ()`

Returns true if controlling the real, physical robot and false if in simulation mode.

Returns

true if controlling the real, physical robot and false if in simulation mode.

8.5 `UserInterfaceApi`

`UserInterfaceApi` Provides access to functionality and services available from within AuboScope related to user interface and end user interaction.

Classes

- class `arcs::aubo_scope::RenderInterface`
`RenderInterface` Provides display option controls related to 3D rendering, including show/hide of ground, robot models, coordinate systems, trajectories and points.
- class `arcs::aubo_scope::UserInteraction`
`UserInteraction` This interface provides functionality for requesting input or actions from end users.

Functions

- `UserInteractionPtr` `arcs::aubo_scope::UserInterfaceApi::getUserInteraction ()`
`UserInteraction`

8.5.1 Detailed Description

[UserInterfaceApi](#) Provides access to functionality and services available from within AuboScope related to user interface and end user interaction.

8.5.2 Function Documentation

8.5.2.1 `getUserInteraction()`

`UserInteractionPtr arcs::aubo_scope::UserInterfaceApi::getUserInteraction ()`

[UserInteraction](#)

Returns

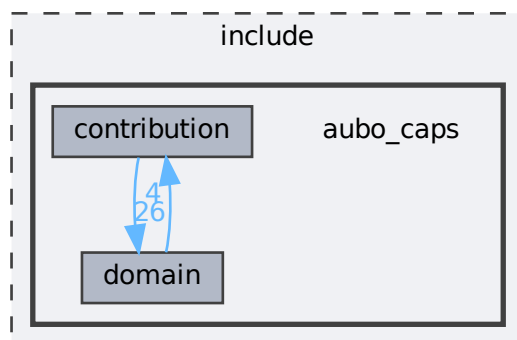
An interface which provides functionality for requesting input and actions from end users.

Chapter 9

Directory Documentation

9.1 include/aubo_caps Directory Reference

Directory dependency graph for aubo_caps:



Directories

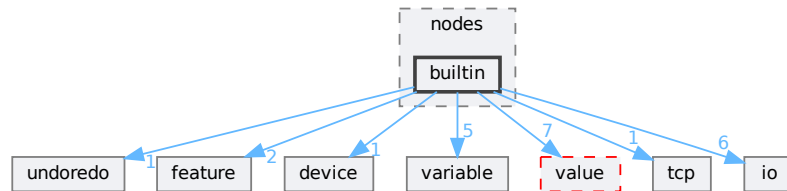
- directory [contribution](#)
- directory [domain](#)

Files

- file [class_forward.h](#)

9.2 include/aubo_caps/domain/program/nodes/builtin Directory Reference

Directory dependency graph for builtin:

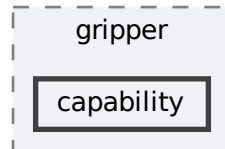


Files

- file [assembly_node.h](#)
- file [assignment_node.h](#)
- file [break_node.h](#)
- file [call_node.h](#)
- file [case_node.h](#)
- file [circle_move_node.h](#)
- file [comment_node.h](#)
- file [conveyor_tracking_node.h](#)
- file [default_case_node.h](#)
- file [direction_node.h](#)
- file [else_if_node.h](#)
- file [else_node.h](#)
- file [folder_node.h](#)
- file [force_node.h](#)
- file [force_option_node.h](#)
- file [gripper_node.h](#)
- file [halt_node.h](#)
- file [if_node.h](#)
- file [loop_node.h](#)
- file [move_node.h](#)
- file [offline_track_node.h](#)
- file [pallet_node.h](#)
- file [popup_node.h](#)
- file [program_home_node.h](#)
- file [record_track_node.h](#)
- file [screw_driving_node.h](#)
- file [script_node.h](#)
- file [seek_node.h](#)
- file [set_node.h](#)
- file [set_payload_node.h](#)
- file [sub_prog_node.h](#)
- file [switch_node.h](#)
- file [thread_node.h](#)
- file [timer_node.h](#)
- file [until_node.h](#)
- file [visual_localization_program_node.h](#)
- file [wait_node.h](#)
- file [waypoint_node.h](#)

9.3 include/aubo_caps/contribution/driver/gripper/capability Directory Reference

Directory dependency graph for capability:

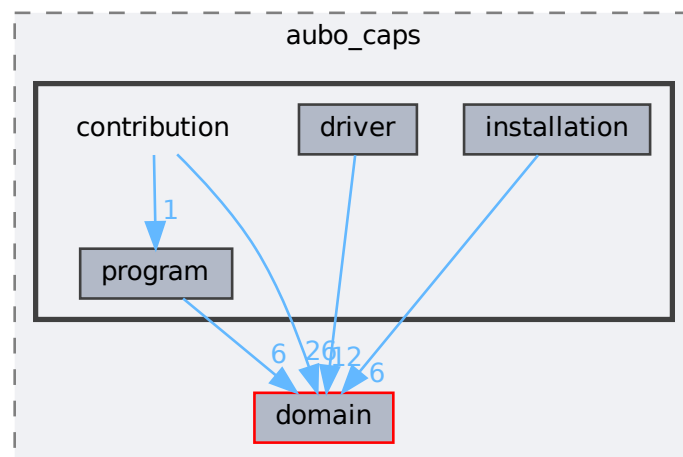


Files

- file [grip_force_capability.h](#)
- file [grip_vacuum_capability.h](#)
- file [registered_cpabilities.h](#)
- file [speed_capability.h](#)
- file [width_capability.h](#)

9.4 include/aubo_caps/contribution Directory Reference

Directory dependency graph for contribution:



Directories

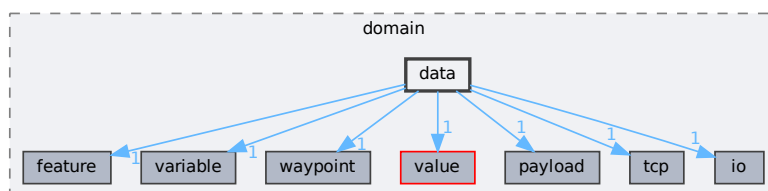
- directory [driver](#)
- directory [installation](#)
- directory [program](#)

Files

- file [contribution_provider.h](#)
- file [view_api_provider.h](#)

9.5 include/aubo_caps/domain/data Directory Reference

Directory dependency graph for data:

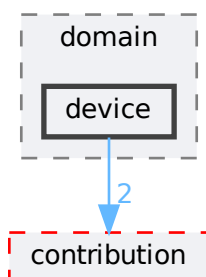


Files

- file [data_model.h](#)

9.6 include/aubo_caps/domain/device Directory Reference

Directory dependency graph for device:



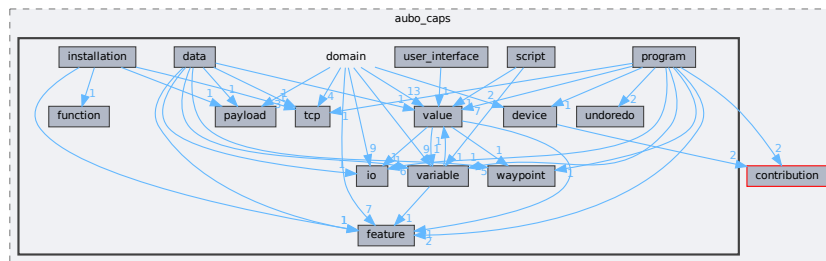
Files

- file [device.h](#)
- file [device_manager.h](#)
- file [device_registration_manager.h](#)
- file [gripper_device.h](#)

9.7 doc Directory Reference

9.8 include/aubo_caps/domain Directory Reference

Directory dependency graph for domain:



Directories

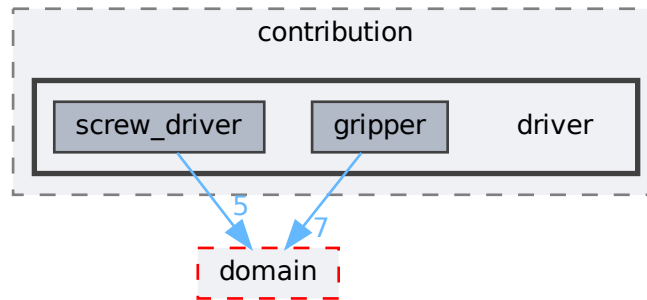
- directory [data](#)
- directory [device](#)
- directory [feature](#)
- directory [function](#)
- directory [installation](#)
- directory [io](#)
- directory [payload](#)
- directory [program](#)
- directory [script](#)
- directory [tcp](#)
- directory [undoredo](#)
- directory [user_interface](#)
- directory [value](#)
- directory [variable](#)
- directory [waypoint](#)

Files

- file [application_api.h](#)
- file [extension_info.h](#)
- file [system_api.h](#)

9.9 include/aubo_caps/contribution/driver Directory Reference

Directory dependency graph for driver:

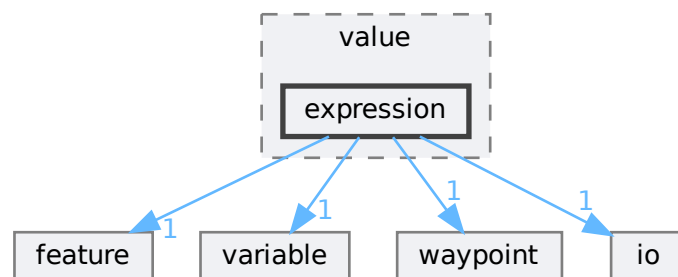


Directories

- directory [gripper](#)
- directory [screw_driver](#)

9.10 include/aubo_caps/domain/value/expression Directory Reference

Directory dependency graph for expression:

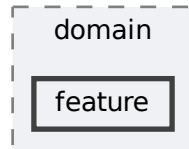


Files

- file [expression.h](#)

9.11 include/aubo_caps/domain/feature Directory Reference

Directory dependency graph for feature:

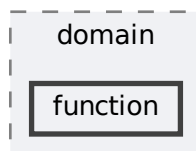


Files

- file [base_feature.h](#)
- file [feature.h](#)
- file [feature_contribution_model.h](#)
- file [feature_line.h](#)
- file [feature_model.h](#)
- file [feature_plane.h](#)
- file [feature_point.h](#)
- file [feature_tool.h](#)

9.12 include/aubo_caps/domain/function Directory Reference

Directory dependency graph for function:

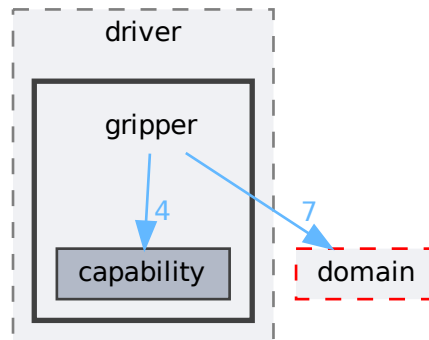


Files

- file [function.h](#)
- file [function_model.h](#)

9.13 include/aubo_caps/contribution/driver/gripper Directory Reference

Directory dependency graph for gripper:



Directories

- directory [capability](#)

Files

- file [grip_action_parameters.h](#)
- file [gripper_api_provider.h](#)
- file [gripper_configuration.h](#)
- file [gripper_contribution.h](#)
- file [release_action_parameters.h](#)

9.14 include Directory Reference

Directory dependency graph for include:

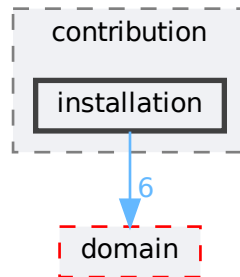


Directories

- directory [aubo_caps](#)

9.15 include/aubo_caps/contribution/installation Directory Reference

Directory dependency graph for installation:

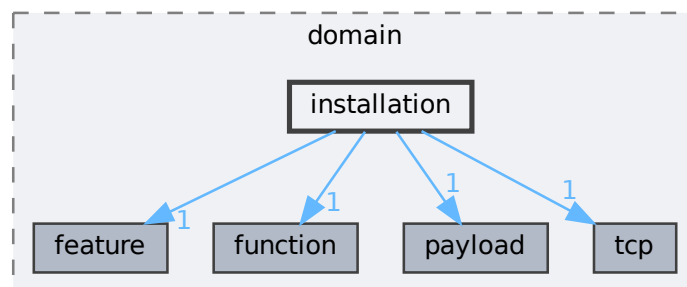


Files

- file [installation_api_provider.h](#)
- file [installation_contribution_configuration.h](#)
- file [installation_creation_context.h](#)
- file [installation_node_contribution.h](#)
- file [installation_node_service.h](#)
- file [installation_node_view.h](#)

9.16 include/aubo_caps/domain/installation Directory Reference

Directory dependency graph for installation:



Files

- file [installation_api.h](#)

9.17 include/aubo_caps/domain/io Directory Reference

Directory dependency graph for io:

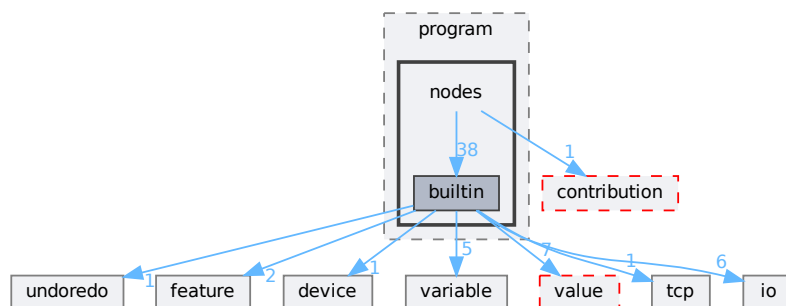


Files

- file [analog_io.h](#)
- file [boolean_register.h](#)
- file [digital_io.h](#)
- file [double_register.h](#)
- file [ethernet_ip.h](#)
- file [integer_register.h](#)
- file [io.h](#)
- file [io_model.h](#)
- file [modbus_io.h](#)
- file [profinet.h](#)
- file [register.h](#)

9.18 include/aubo_caps/domain/program/nodes Directory Reference

Directory dependency graph for nodes:



Directories

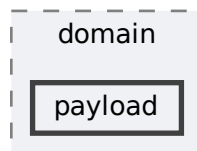
- directory [builtin](#)

Files

- file [program_node.h](#)
- file [program_node_factory.h](#)

9.19 include/aubo_caps/domain/payload Directory Reference

Directory dependency graph for payload:

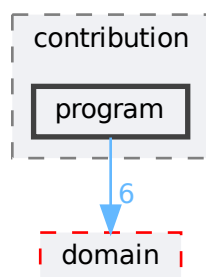


Files

- file [payload.h](#)
- file [payload_contribution_model.h](#)
- file [payload_model.h](#)

9.20 include/aubo_caps/contribution/program Directory Reference

Directory dependency graph for program:

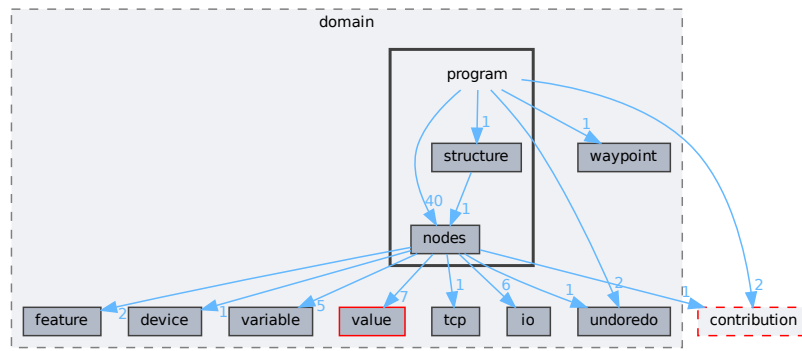


Files

- file [program_api_provider.h](#)
- file [program_contribution_configuration.h](#)
- file [program_creation_context.h](#)
- file [program_node_contribution.h](#)
- file [program_node_service.h](#)
- file [program_node_view.h](#)

9.21 include/aubo_caps/domain/program Directory Reference

Directory dependency graph for program:



Directories

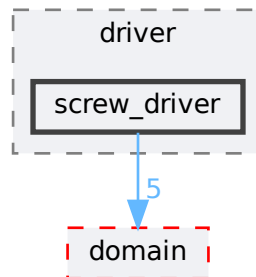
- directory [nodes](#)
- directory [structure](#)

Files

- file [program_api.h](#)
- file [program_model.h](#)

9.22 include/aubo_caps/contribution/driver/screw_driver Directory Reference

Directory dependency graph for screw_driver:

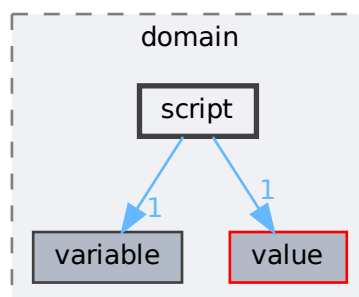


Files

- file [screw_driver_api_provider.h](#)
- file [screw_driver_configuration.h](#)
- file [screw_driver_contribution.h](#)
- file [screw_driver_parameters.h](#)

9.23 include/aubo_caps/domain/script Directory Reference

Directory dependency graph for script:

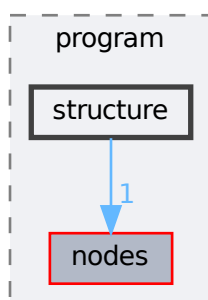


Files

- file [script_writer.h](#)

9.24 include/aubo__caps/domain/program/structure Directory Reference

Directory dependency graph for structure:

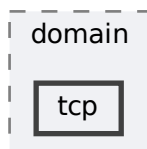


Files

- file [tree_node.h](#)

9.25 include/aubo__caps/domain/tcp Directory Reference

Directory dependency graph for tcp:

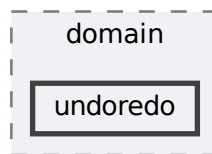


Files

- file [tcp.h](#)
- file [tcp_contribution_model.h](#)
- file [tcp_model.h](#)

9.26 include/aubo_caps/domain/undoredo Directory Reference

Directory dependency graph for undoredo:

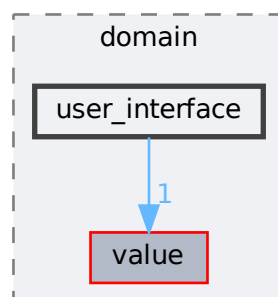


Files

- file [undo_redo_manager.h](#)

9.27 include/aubo_caps/domain/user_interface Directory Reference

Directory dependency graph for user_interface:

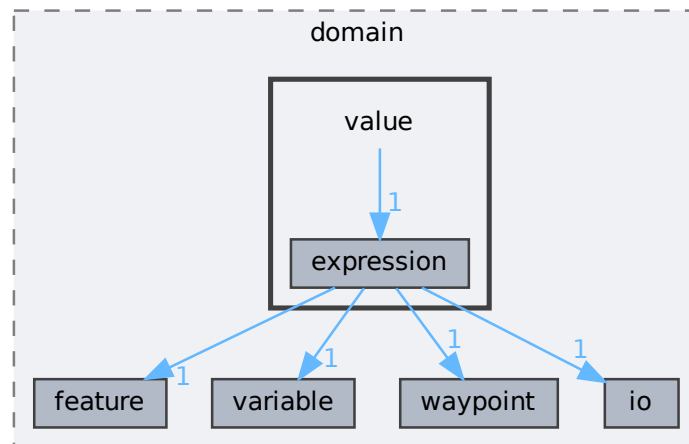


Files

- file [input_validator.h](#)
- file [keyboard_manager.h](#)
- file [popup_manager.h](#)
- file [render_interface.h](#)
- file [user_interaction.h](#)
- file [user_interface_api.h](#)

9.28 include/aubo_caps/domain/value Directory Reference

Directory dependency graph for value:



Directories

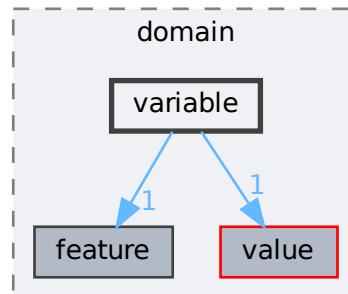
- directory [expression](#)

Files

- file [value_factory_provider.h](#)

9.29 include/aubo_caps/domain/variable Directory Reference

Directory dependency graph for variable:

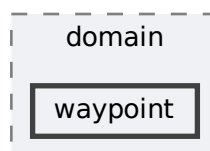


Files

- file [feature_variable.h](#)
- file [global_variable.h](#)
- file [persisted_variable.h](#)
- file [variable.h](#)
- file [variable_factory.h](#)
- file [variable_model.h](#)

9.30 include/aubo_caps/domain/waypoint Directory Reference

Directory dependency graph for waypoint:



Files

- file [waypoint.h](#)
- file [waypoint_model.h](#)

Chapter 10

Namespace Documentation

10.1 arcs Namespace Reference

Namespaces

- namespace [aubo_scope](#)
- namespace [aubo_sdk](#)

10.2 arcs::aubo_scope Namespace Reference

Classes

- class [ContributionProvider](#)
[ContributionProvider](#) This interface provides access to a contribution.
- class [GripForceCapability](#)
[GripForceCapability](#) This interface represents a registered force capability for a gripper which supports gripping using a user configurable force.
- class [GripVacuumCapability](#)
[GripVacuumCapability](#) This interface represents a registered vacuum capability for a gripper which supports gripping using a user configurable vacuum level.
- class [RegisteredCapabilities](#)
- class [SpeedCapability](#)
[SpeedCapability](#) This interface represents a registered speed capability for a gripper which supports operating using a user configurable speed.
- class [WidthCapability](#)
[WidthCapability](#) This interface represents a registered width/position capability for a gripper which supports moving to a user configurable position (open/close to a configurable width).
- class [GripActionParameters](#)
- class [GripperApiProvider](#)
[GripperApiProvider](#) This interface provides access to functionality and services available from within AuboScope relevant for gripper contributions.
- class [GripperConfiguration](#)
This interface provides access to register or setup various properties and capabilities of a gripper.
- class [GripperContribution](#)
[GripperContribution](#) This interface defines an API for adding functionality for gripping devices to AuboScope.

- class [ReleaseActionParameters](#)
- class [ScrewdriverApiProvider](#)

[ScrewdriverApiProvider](#) This interface provides access to functionality and services available from within AuboScope relevant for screwdriver contributions.

- class [ProgramSelectionParameters](#)
- class [PrepareToStartScrewdriverParameters](#)
- class [FeedScrewParameters](#)
- class [DriveScrewNotOKParameters](#)
- class [DriveScrewOKParameters](#)
- class [ScrewdriverReadyParameters](#)

This interface provides the parameters relevant for generating script for the Screwdriver Ready feedback operation capability (registered using `ScrewdriverFeedbackCapabilities#registerScrewdriverReadyCapability(ScriptCodeC`

- class [ScrewdriverConfiguration](#)

This interface provides access to register or setup various properties and capabilities of a screwdriver.

- class [ScrewdriverContribution](#)

[ScrewdriverContribution](#) This interface defines an API for adding functionality for screwdrivers to AuboScope.

- class [ScrewdriverProgram](#)

This abstract class represents a screwdriver program, typically defined on an external control box for the screwdriver.

- class [ScrewdriverParameters](#)

This interface represents parameters for a screwdriver operation defined/configured by the end user.

- class [InstallationApiProvider](#)

[InstallationApiProvider](#) Provides access to functionality and services relevant for installation nodes.

- class [InstallationContributionConfiguration](#)
- class [InstallationCreationContext](#)

[InstallationCreationContext](#) Provides information about the context in which an installation node is created.

- class [InstallationNodeContribution](#)

[InstallationNodeContribution](#) API for installation node contributions.

- class [InstallationNodeService](#)

[InstallationNodeService](#) Defines an API required for defining and adding to AuboScope an installation node and corresponding screen where the user interface is Qt-based.

- class [InstallationNodeView](#)

[InstallationNodeView](#) API for an installation node View which defines the UI for the corresponding [InstallationNodeContribution](#)

- class [ProgramApiProvider](#)

[ProgramApiProvider](#) Provides access to functionality and services relevant for program nodes.

- class [ProgramContributionConfiguration](#)

[ProgramContributionConfiguration](#) Provides access to configure the properties of an installation node contribution.

- class [ProgramCreationContext](#)

[ProgramCreationContext](#) Provides information about the context in which an installation node is created.

- class [ProgramNodeContribution](#)

[ProgramNodeContribution](#) Defines an API required for specifying a AuboCap Program Node within AuboScope.

- class [ProgramNodeService](#)

[ProgramNodeService](#) Defines an API required for adding a new type of program node to AuboScope.

- class [ProgramNodeView](#)

[ProgramNodeView](#) API for a program node View which defines the UI for the corresponding [ProgramNodeContribution](#).

- class [ViewApiProvider](#)

- [ViewApiProvider](#) Provides access to functionality relevant for the View (UI) of an installation or a program node contribution.
- class [ApplicationApi](#)
- class [DataModel](#)
 - [DataModel](#) This interface is used for storing and managing data that represents the current configuration of, e.g.
- class [Device](#)
 - [Device](#) This base interface represents a device supported by AuboScope, e.g.
- class [DeviceManager](#)
 - [DeviceManager](#) Base interface for device managers.
- class [DeviceRegistrationManager](#)
 - [DeviceRegistrationManager](#) Base interface for device registration managers.
- class [GripperDevice](#)
 - [GripperDevice](#) This interface represents a gripper device in AuboScope.
- class [AuboCapInfo](#)
 - This interface provides various metadata information about an installed AuboCap, such as the name of the AuboCap.
- class [BaseFeature](#)
 - This interface represents the Base feature in AuboScope.
- class [Feature](#)
 - [Feature](#) This base interface represents all types of features available in AuboScope.
- struct [FeatureDefinedFilter](#)
- struct [FeatureJoggableFilter](#)
- class [FeatureContributionModel](#)
 - [FeatureContributionModel](#) This interface provides functionality for adding, updating and removing features in AuboScope.
- class [FeatureLine](#)
- class [FeatureModel](#)
 - [FeatureModel](#) Provides methods that returns features from the current robot installation.
- class [FeaturePlane](#)
- class [FeaturePoint](#)
 - This interface represents a Point feature in AuboScope.
- class [FeatureTool](#)
 - This interface represents the Tool feature in AuboScope.
- class [Function](#)
 - [Function](#) Represents a function shown in the [Expression](#) Editor.
- class [FunctionModel](#)
 - [FunctionModel](#) This interface provides methods to add and remove script functions in the [Expression](#) Editor accessible from Assignment and If program nodes among others.
- class [InstallationApi](#)
- class [AnalogIo](#)
 - [AnalogIo](#) This interface provides support for analog I/Os.
- struct [AnalogIoFilter](#)
- class [BoolRegister](#)
 - This interface provides support for bool registers.
- class [DigitalIo](#)
 - [DigitalIo](#) This interface provides support for digital I/Os.
- struct [DigitalIoFilter](#)
- struct [InputIoFilter](#)
- struct [OutputIoFilter](#)
- class [DoubleRegister](#)
 - This interface provides support for float registers.

- class [EtherNetIP](#)
[EtherNetIP](#) This interface provides support for EtherNet/IP I/Os.
- struct [EtherNetIPFilter](#)
- class [IntegerRegister](#)
This interface provides support for integer registers.
- class [Io](#)
[Io](#) This is the base interface representing all types of I/Os available in AuboScope.
- class [IoModel](#)
[IoModel](#) Provides methods that returns I/Os from the current robot installation.
- class [ModbusIo](#)
[ModbusIo](#) This interface provides support for MODBUS I/Os.
- struct [ModbusIoFilter](#)
- class [Profinet](#)
[Profinet](#) This interface provides support for PROFINET I/Os.
- struct [ProfinetFilter](#)
- class [Register](#)
This interface provides support for registers.
- class [Payload](#)
API interface for payload parameters.
- class [PayloadContributionModel](#)
API interface for dynamic payload management.
- class [PayloadModel](#)
API interface for payload parameters.
- class [AssemblyNode](#)
- class [AssignmentNode](#)
- class [BreakNode](#)
- class [CallNode](#)
- class [CaseNode](#)
- class [CircleMoveNode](#)
- class [CommentNode](#)
- class [ConveyorTrackingNode](#)
- class [DefaultCaseNode](#)
- class [DirectionNode](#)
- class [ElseIfNode](#)
- class [ElseNode](#)
- class [FolderNode](#)
- class [ForceNode](#)
- class [ForceOptionNode](#)
- class [GripperNode](#)
- class [HaltNode](#)
- class [IfNode](#)
- class [LoopNode](#)
- class [MoveNode](#)
- class [OfflineTrackNode](#)
- class [PalletNode](#)
- class [PopupNode](#)
- class [ProgramHomeNode](#)
- class [RecordTrackNode](#)
- class [ScrewdrivingNode](#)
- class [ScriptNode](#)
- class [SeekNode](#)
- class [SetNode](#)
- class [SetPayloadNode](#)

- class [SubProgNode](#)
- class [SwitchNode](#)
- class [ThreadNode](#)
- class [TimerNode](#)
- class [UntilNode](#)
- class [VisualLocalizationProgramNode](#)
- class [WaitNode](#)
- class [WaypointNode](#)
- class [ProgramNode](#)
 - [ProgramNode](#) This interface represents AuboScope program nodes.
- class [ProgramNodeFactory](#)
 - [ProgramNodeFactory](#) This interfaces supplies methods to create different types of program nodes.
- class [ProgramApi](#)
- class [ProgramModel](#)
 - [ProgramModel](#)
- class [TreeNode](#)
 - [TreeNode](#) This interface represents a node in the program tree that can be used to construct a sub-tree rooted in a AuboCap program node.
- class [ScriptWriter](#)
 - [ScriptWriter](#) This interface provides support for generating Script code.
- struct [IoConfigure](#)
- class [SystemApi](#)
- class [TCP](#)
 - [TCP](#) This interface represents a [TCP](#) in AuboScope.
- class [TcpContributionModel](#)
 - [TcpContributionModel](#) This interface provides functionality for adding, updating and removing TCPs in AuboScope.
- class [TcpModel](#)
 - [TcpModel](#) This interface provides access to the TCPs that are currently available in AuboScope.
- class [UndoRedoManager](#)
 - [UndoRedoManager](#) This interface is used to record changes triggered by an end user in the screen of a AuboCap program node contribution (e.g.
- struct [InputValidator](#)
 - [InputValidator](#) Interface representing the input validators created by InputValidationFactory.
- struct [IntegerRangeValidator](#)
- struct [UIntegerRangeValidator](#)
- struct [DoubleRangeValidator](#)
- struct [StringLengthValidator](#)
- class [KeyboardManager](#)
- struct [DefaultSingletonTraits](#)
- class [PopupManager](#)
- class [RenderInterface](#)
 - [RenderInterface](#) Provides display option controls related to 3D rendering, including show/hide of ground, robot models, coordinate systems, trajectories and points.
- class [UserInteraction](#)
 - [UserInteraction](#) This interface provides functionality for requesting input or actions from end users.
- class [UserInterfaceApi](#)
- class [Expression](#)
 - [Expression](#) An expression can be used for, e.g.
- class [ValueFactory](#)
 - [ValueFactory](#)
- class [FeatureVariable](#)

[FeatureVariable](#) A [Feature](#) marked as a [Variable](#) in the installation will exist even when a robot is turned off and on.

- struct [FeatureVariableFilter](#)
- class [GlobalVariable](#)

Global variables are available everywhere in a program.

- struct [GlobalVariableFilter](#)
- class [PersistedVariable](#)

ingroup [ApplicationApi](#) This interface represents variables stored with the installation.

- struct [PersistedVariableFilter](#)
- class [Variable](#)

[Variable](#) Represents a variable in AuboScope.

- class [VariableFactory](#)
- class [VariableModel](#)

[VariableModel](#) Provides methods that returns the variables that currently are registered in the system.

- class [Waypoint](#)

[Waypoint](#) This interface composes parameters that define a position of the robot.

- class [WaypointModel](#)

[WaypointModel](#) Provides methods that returns Waypoints from the current robot installation.

Typedefs

- template<class T>
using [ContributionProviderPtr](#) = std::shared_ptr<[ContributionProvider](#)<T>>
- typedef long [WidgetHandle](#)
- using [RobotPositionCallback](#)
- using [RobotMovementCallback](#) = std::function<void(bool accepted, int error)>

Enumerations

- enum [ScrewdriverOperationType](#) { [TIGHTEN](#) , [LOOSEN](#) }
- This interface represents the type of screwing operation performed by the screwdriver.
- enum class [UnitType](#) { [METRIC](#) , [IMPERIAL](#) }

Functions

- [ARCS_CLASS_FORWARD](#) ([GripForceCapability](#))
- [ARCS_CLASS_FORWARD](#) ([GripVacuumCapability](#))
- [ARCS_CLASS_FORWARD](#) ([RegisteredCapabilities](#))
- [ARCS_CLASS_FORWARD](#) ([SpeedCapability](#))
- [ARCS_CLASS_FORWARD](#) ([WidthCapability](#))
- [ARCS_CLASS_FORWARD](#) ([GripActionParameters](#))
- [ARCS_CLASS_FORWARD](#) ([GripperApiProvider](#))
- [ARCS_CLASS_FORWARD](#) ([GripperConfiguration](#))
- [ARCS_CLASS_FORWARD](#) ([GripperContribution](#))
- [ARCS_CLASS_FORWARD](#) ([ReleaseActionParameters](#))
- [ARCS_CLASS_FORWARD](#) ([ScrewdriverApiProvider](#))
- [ARCS_CLASS_FORWARD](#) ([ProgramSelectionParameters](#))
- [ARCS_CLASS_FORWARD](#) ([PrepareToStartScrewdriverParameters](#))
- [ARCS_CLASS_FORWARD](#) ([FeedScrewParameters](#))
- [ARCS_CLASS_FORWARD](#) ([DriveScrewNotOKParameters](#))
- [ARCS_CLASS_FORWARD](#) ([DriveScrewOKParameters](#))

- [ARCS_CLASS_FORWARD \(ScrewdriverReadyParameters\)](#)
- [ARCS_CLASS_FORWARD \(ScrewdriverConfiguration\)](#)
- [ARCS_CLASS_FORWARD \(ScrewdriverContribution\)](#)
- [ARCS_CLASS_FORWARD \(ScrewdriverProgram\)](#)
- [ARCS_CLASS_FORWARD \(ScrewdriverParameters\)](#)
- [ARCS_CLASS_FORWARD \(InstallationApiProvider\)](#)
- [ARCS_CLASS_FORWARD \(InstallationContributionConfiguration\)](#)
- [ARCS_CLASS_FORWARD \(InstallationCreationContext\)](#)
- [ARCS_CLASS_FORWARD \(InstallationNodeContribution\)](#)
- [ARCS_CLASS_FORWARD \(InstallationNodeService\)](#)
- [ARCS_CLASS_FORWARD \(InstallationNodeView\)](#)
- [ARCS_CLASS_FORWARD \(ProgramApiProvider\)](#)
- [ARCS_CLASS_FORWARD \(ProgramContributionConfiguration\)](#)
- [ARCS_CLASS_FORWARD \(ProgramCreationContext\)](#)
- [ARCS_CLASS_FORWARD \(ProgramNodeContribution\)](#)
- [ARCS_CLASS_FORWARD \(ProgramNodeService\)](#)
- [ARCS_CLASS_FORWARD \(ProgramNodeView\)](#)
- [ARCS_CLASS_FORWARD \(ViewApiProvider\)](#)
- [ARCS_CLASS_FORWARD \(ApplicationApi\)](#)
- [ARCS_CLASS_FORWARD \(DataModel\)](#)
- [ARCS_CLASS_FORWARD \(Device\)](#)
- [ARCS_CLASS_FORWARD \(DeviceManager\)](#)
- [ARCS_CLASS_FORWARD \(DeviceRegistrationManager\)](#)
- [ARCS_CLASS_FORWARD \(GripperDevice\)](#)
- [ARCS_CLASS_FORWARD \(AuboCapInfo\)](#)
- [ARCS_CLASS_FORWARD \(BaseFeature\)](#)
- [ARCS_CLASS_FORWARD \(Feature\)](#)
- [ARCS_CLASS_FORWARD \(FeatureContributionModel\)](#)
- [ARCS_CLASS_FORWARD \(FeatureLine\)](#)
- [ARCS_CLASS_FORWARD \(FeatureModel\)](#)
- [ARCS_CLASS_FORWARD \(FeaturePlane\)](#)
- [ARCS_CLASS_FORWARD \(FeaturePoint\)](#)
- [ARCS_CLASS_FORWARD \(FeatureTool\)](#)
- [ARCS_CLASS_FORWARD \(Function\)](#)
- [ARCS_CLASS_FORWARD \(FunctionModel\)](#)
- [ARCS_CLASS_FORWARD \(InstallationApi\)](#)
- [ARCS_CLASS_FORWARD \(AnalogIo\)](#)
- [ARCS_CLASS_FORWARD \(BoolRegister\)](#)
- [ARCS_CLASS_FORWARD \(DigitalIo\)](#)
- [ARCS_CLASS_FORWARD \(DoubleRegister\)](#)
- [ARCS_CLASS_FORWARD \(EtherNetIP\)](#)
- [ARCS_CLASS_FORWARD \(IntegerRegister\)](#)
- [ARCS_CLASS_FORWARD \(Io\)](#)
- [ARCS_CLASS_FORWARD \(IoModel\)](#)
- [ARCS_CLASS_FORWARD \(ModbusIo\)](#)
- [ARCS_CLASS_FORWARD \(Profinet\)](#)
- [ARCS_CLASS_FORWARD \(Register\)](#)
- [ARCS_CLASS_FORWARD \(Payload\)](#)
- [ARCS_CLASS_FORWARD \(PayloadContributionModel\)](#)
- [ARCS_CLASS_FORWARD \(PayloadModel\)](#)
- [ARCS_CLASS_FORWARD \(AssemblyNode\)](#)
- [ARCS_CLASS_FORWARD \(AssignmentNode\)](#)
- [ARCS_CLASS_FORWARD \(BreakNode\)](#)
- [ARCS_CLASS_FORWARD \(CallNode\)](#)
- [ARCS_CLASS_FORWARD \(CaseNode\)](#)

- [ARCS_CLASS_FORWARD \(CircleMoveNode\)](#)
- [ARCS_CLASS_FORWARD \(CommentNode\)](#)
- [ARCS_CLASS_FORWARD \(ConveyorTrackingNode\)](#)
- [ARCS_CLASS_FORWARD \(DefaultCaseNode\)](#)
- [ARCS_CLASS_FORWARD \(DirectionNode\)](#)
- [ARCS_CLASS_FORWARD \(ElseIfNode\)](#)
- [ARCS_CLASS_FORWARD \(ElseNode\)](#)
- [ARCS_CLASS_FORWARD \(FolderNode\)](#)
- [ARCS_CLASS_FORWARD \(ForceNode\)](#)
- [ARCS_CLASS_FORWARD \(ForceOptionNode\)](#)
- [ARCS_CLASS_FORWARD \(GripperNode\)](#)
- [ARCS_CLASS_FORWARD \(HaltNode\)](#)
- [ARCS_CLASS_FORWARD \(Expression\)](#)
- [ARCS_CLASS_FORWARD \(IfNode\)](#)
- [ARCS_CLASS_FORWARD \(LoopNode\)](#)
- [ARCS_CLASS_FORWARD \(MoveNode\)](#)
- [ARCS_CLASS_FORWARD \(OfflineTrackNode\)](#)
- [ARCS_CLASS_FORWARD \(PalletNode\)](#)
- [ARCS_CLASS_FORWARD \(PopupNode\)](#)
- [ARCS_CLASS_FORWARD \(ProgramHomeNode\)](#)
- [ARCS_CLASS_FORWARD \(RecordTrackNode\)](#)
- [ARCS_CLASS_FORWARD \(ScrewdrivingNode\)](#)
- [ARCS_CLASS_FORWARD \(ScriptNode\)](#)
- [ARCS_CLASS_FORWARD \(SeekNode\)](#)
- [ARCS_CLASS_FORWARD \(SetNode\)](#)
- [ARCS_CLASS_FORWARD \(SetPayloadNode\)](#)
- [ARCS_CLASS_FORWARD \(SubProgNode\)](#)
- [ARCS_CLASS_FORWARD \(SwitchNode\)](#)
- [ARCS_CLASS_FORWARD \(ThreadNode\)](#)
- [ARCS_CLASS_FORWARD \(TimerNode\)](#)
- [ARCS_CLASS_FORWARD \(UntilNode\)](#)
- [ARCS_CLASS_FORWARD \(VisualLocalizationProgramNode\)](#)
- [ARCS_CLASS_FORWARD \(WaitNode\)](#)
- [ARCS_CLASS_FORWARD \(WaypointNode\)](#)
- [ARCS_CLASS_FORWARD \(ProgramNode\)](#)
- [ARCS_CLASS_FORWARD \(ProgramNodeFactory\)](#)
- [ARCS_CLASS_FORWARD \(ProgramApi\)](#)
- [ARCS_CLASS_FORWARD \(ProgramModel\)](#)
- [ARCS_CLASS_FORWARD \(TreeNode\)](#)
- [ARCS_CLASS_FORWARD \(ScriptWriter\)](#)
- [ARCS_CLASS_FORWARD \(SystemApi\)](#)
- [ARCS_CLASS_FORWARD \(TCP\)](#)
- [ARCS_CLASS_FORWARD \(TcpContributionModel\)](#)
- [ARCS_CLASS_FORWARD \(TcpModel\)](#)
- [ARCS_CLASS_FORWARD \(UndoRedoManager\)](#)
- [ARCS_CLASS_FORWARD \(KeyboardManager\)](#)
- [ARCS_CLASS_FORWARD \(PopupManager\)](#)
- [ARCS_CLASS_FORWARD \(RenderInterface\)](#)
- [ARCS_CLASS_FORWARD \(UserInteraction\)](#)
- [ARCS_CLASS_FORWARD \(UserInterfaceApi\)](#)
- [ARCS_CLASS_FORWARD \(ValueFactory\)](#)
- [ARCS_CLASS_FORWARD \(FeatureVariable\)](#)
- [ARCS_CLASS_FORWARD \(GlobalVariable\)](#)
- [ARCS_CLASS_FORWARD \(PersistedVariable\)](#)
- [ARCS_CLASS_FORWARD \(Variable\)](#)

- [ARCS_CLASS_FORWARD](#) ([VariableFactory](#))
- [ARCS_CLASS_FORWARD](#) ([VariableModel](#))
- [ARCS_CLASS_FORWARD](#) ([Waypoint](#))
- [ARCS_CLASS_FORWARD](#) ([WaypointModel](#))

10.2.1 Typedef Documentation

10.2.1.1 ContributionProviderPtr

```
template<class T>
using arcs::aubo_scope::ContributionProviderPtr = std::shared_ptr<ContributionProvider<T>>
```

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

10.2.1.2 RobotMovementCallback

```
using arcs::aubo_scope::RobotMovementCallback = std::function<void(bool accepted, int error)>
```

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

10.2.1.3 RobotPositionCallback

```
using arcs::aubo_scope::RobotPositionCallback
```

Initial value:

```
std::function<void(
    bool accepted, const std::vector<double> &pose,
    const std::vector<double> &q, const std::vector<double> &offset)>
```

Definition at line 18 of file [user_interaction.h](#).

10.2.1.4 WidgetHandle

```
typedef long arcs::aubo_scope::WidgetHandle
```

Definition at line 9 of file [program_node_view.h](#).

10.2.2 Enumeration Type Documentation

10.2.2.1 ScrewdriverOperationType

```
enum arcs::aubo_scope::ScrewdriverOperationType
```

This interface represents the type of screwing operation performed by the screwdriver.

Enumerator

TIGHTEN	The screwdriver operation will follow the screw in a tightening direction (in).
LOOSEN	The screwdriver operation will follow the screw in a loosening direction (out).

Definition at line 55 of file [screw_driver_parameters.h](#).

10.2.2.2 UnitType

enum class [arcs::aubo_scope::UnitType](#) [strong]

Enumerator

METRIC	mm Kg
IMPERIAL	

Definition at line 12 of file [system_api.h](#).

10.2.3 Function Documentation

10.2.3.1 ARCS_CLASS_FORWARD() [1/130]

[arcs::aubo_scope::ARCS_CLASS_FORWARD](#) (
[AnalogIo](#))

10.2.3.2 ARCS_CLASS_FORWARD() [2/130]

[arcs::aubo_scope::ARCS_CLASS_FORWARD](#) (
[ApplicationApi](#))

10.2.3.3 ARCS_CLASS_FORWARD() [3/130]

[arcs::aubo_scope::ARCS_CLASS_FORWARD](#) (
[AssemblyNode](#))

10.2.3.4 ARCS_CLASS_FORWARD() [4/130]

[arcs::aubo_scope::ARCS_CLASS_FORWARD](#) (
[AssignmentNode](#))

10.2.3.5 ARCS_CLASS_FORWARD() [5/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [AuboCapInfo](#))

10.2.3.6 ARCS_CLASS_FORWARD() [6/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [BaseFeature](#))

10.2.3.7 ARCS_CLASS_FORWARD() [7/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [BoolRegister](#))

10.2.3.8 ARCS_CLASS_FORWARD() [8/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [BreakNode](#))

10.2.3.9 ARCS_CLASS_FORWARD() [9/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [CallNode](#))

10.2.3.10 ARCS_CLASS_FORWARD() [10/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [CaseNode](#))

10.2.3.11 ARCS_CLASS_FORWARD() [11/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [CircleMoveNode](#))

10.2.3.12 ARCS_CLASS_FORWARD() [12/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [CommentNode](#))

10.2.3.13 ARCS_CLASS_FORWARD() [13/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [ConveyorTrackingNode](#))

10.2.3.14 ARCS_CLASS_FORWARD() [14/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [DataModel](#))

10.2.3.15 ARCS_CLASS_FORWARD() [15/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [DefaultCaseNode](#))

10.2.3.16 ARCS_CLASS_FORWARD() [16/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [Device](#))

10.2.3.17 ARCS_CLASS_FORWARD() [17/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [DeviceManager](#))

10.2.3.18 ARCS_CLASS_FORWARD() [18/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [DeviceRegistrationManager](#))

10.2.3.19 ARCS_CLASS_FORWARD() [19/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [DigitalIo](#))

10.2.3.20 ARCS_CLASS_FORWARD() [20/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [DirectionNode](#))

10.2.3.21 ARCS_CLASS_FORWARD() [21/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [DoubleRegister](#))

10.2.3.22 ARCS_CLASS_FORWARD() [22/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [DriveScrewNotOKParameters](#))

10.2.3.23 ARCS_CLASS_FORWARD() [23/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [DriveScrewOKParameters](#))

10.2.3.24 ARCS_CLASS_FORWARD() [24/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [ElseIfNode](#))

10.2.3.25 ARCS_CLASS_FORWARD() [25/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [ElseNode](#))

10.2.3.26 ARCS_CLASS_FORWARD() [26/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [EtherNetIP](#))

10.2.3.27 ARCS_CLASS_FORWARD() [27/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [Expression](#))

10.2.3.28 ARCS_CLASS_FORWARD() [28/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [Feature](#))

10.2.3.29 ARCS_CLASS_FORWARD() [29/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [FeatureContributionModel](#))

10.2.3.30 ARCS_CLASS_FORWARD() [30/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [FeatureLine](#))

10.2.3.31 ARCS_CLASS_FORWARD() [31/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [FeatureModel](#))

10.2.3.32 ARCS_CLASS_FORWARD() [32/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [FeaturePlane](#))

10.2.3.33 ARCS_CLASS_FORWARD() [33/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [FeaturePoint](#))

10.2.3.34 ARCS_CLASS_FORWARD() [34/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [FeatureTool](#))

10.2.3.35 ARCS_CLASS_FORWARD() [35/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [FeatureVariable](#))

10.2.3.36 ARCS_CLASS_FORWARD() [36/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [FeedScrewParameters](#))

10.2.3.37 ARCS_CLASS_FORWARD() [37/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [FolderNode](#))

10.2.3.38 ARCS_CLASS_FORWARD() [38/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [ForceNode](#))

10.2.3.39 ARCS_CLASS_FORWARD() [39/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [ForceOptionNode](#))

10.2.3.40 ARCS_CLASS_FORWARD() [40/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [Function](#))

10.2.3.41 ARCS_CLASS_FORWARD() [41/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [FunctionModel](#))

10.2.3.42 ARCS_CLASS_FORWARD() [42/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [GlobalVariable](#))

10.2.3.43 ARCS_CLASS_FORWARD() [43/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [GripActionParameters](#))

10.2.3.44 ARCS_CLASS_FORWARD() [44/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [GripForceCapability](#))

10.2.3.45 ARCS_CLASS_FORWARD() [45/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [GripperApiProvider](#))

10.2.3.46 ARCS_CLASS_FORWARD() [46/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [GripperConfiguration](#))

10.2.3.47 ARCS_CLASS_FORWARD() [47/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [GripperContribution](#))

10.2.3.48 ARCS_CLASS_FORWARD() [48/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [GripperDevice](#))

10.2.3.49 ARCS_CLASS_FORWARD() [49/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [GripperNode](#))

10.2.3.50 ARCS_CLASS_FORWARD() [50/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [GripVacuumCapability](#))

10.2.3.51 ARCS_CLASS_FORWARD() [51/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [HaltNode](#))

10.2.3.52 ARCS_CLASS_FORWARD() [52/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [IfNode](#))

10.2.3.53 ARCS_CLASS_FORWARD() [53/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [InstallationApi](#))

10.2.3.54 ARCS_CLASS_FORWARD() [54/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [InstallationApiProvider](#))

10.2.3.55 ARCS_CLASS_FORWARD() [55/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [InstallationContributionConfiguration](#))

10.2.3.56 ARCS_CLASS_FORWARD() [56/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [InstallationCreationContext](#))

10.2.3.57 ARCS_CLASS_FORWARD() [57/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [InstallationNodeContribution](#))

10.2.3.58 ARCS_CLASS_FORWARD() [58/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [InstallationNodeService](#))

10.2.3.59 ARCS_CLASS_FORWARD() [59/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [InstallationNodeView](#))

10.2.3.60 ARCS_CLASS_FORWARD() [60/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [IntegerRegister](#))

10.2.3.61 ARCS_CLASS_FORWARD() [61/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [Io](#))

10.2.3.62 ARCS_CLASS_FORWARD() [62/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [IoModel](#))

10.2.3.63 ARCS_CLASS_FORWARD() [63/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [KeyboardManager](#))

10.2.3.64 ARCS_CLASS_FORWARD() [64/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [LoopNode](#))

10.2.3.65 ARCS_CLASS_FORWARD() [65/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [ModbusIo](#))

10.2.3.66 ARCS_CLASS_FORWARD() [66/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [MoveNode](#))

10.2.3.67 ARCS_CLASS_FORWARD() [67/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [OfflineTrackNode](#))

10.2.3.68 ARCS_CLASS_FORWARD() [68/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [PalletNode](#))

10.2.3.69 ARCS_CLASS_FORWARD() [69/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [Payload](#))

10.2.3.70 ARCS_CLASS_FORWARD() [70/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [PayloadContributionModel](#))

10.2.3.71 ARCS_CLASS_FORWARD() [71/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [PayloadModel](#))

10.2.3.72 ARCS_CLASS_FORWARD() [72/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [PersistedVariable](#))

10.2.3.73 ARCS_CLASS_FORWARD() [73/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [PopupManager](#))

10.2.3.74 ARCS_CLASS_FORWARD() [74/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [PopupNode](#))

10.2.3.75 ARCS_CLASS_FORWARD() [75/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [PrepareToStartScrewdriverParameters](#))

10.2.3.76 ARCS_CLASS_FORWARD() [76/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [Profinet](#))

10.2.3.77 ARCS_CLASS_FORWARD() [77/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [ProgramApi](#))

10.2.3.78 ARCS_CLASS_FORWARD() [78/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [ProgramApiProvider](#))

10.2.3.79 ARCS_CLASS_FORWARD() [79/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [ProgramContributionConfiguration](#))

10.2.3.80 ARCS_CLASS_FORWARD() [80/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [ProgramCreationContext](#))

10.2.3.81 ARCS_CLASS_FORWARD() [81/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [ProgramHomeNode](#))

10.2.3.82 ARCS_CLASS_FORWARD() [82/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [ProgramModel](#))

10.2.3.83 ARCS_CLASS_FORWARD() [83/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [ProgramNode](#))

10.2.3.84 ARCS_CLASS_FORWARD() [84/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [ProgramNodeContribution](#))

10.2.3.85 ARCS_CLASS_FORWARD() [85/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [ProgramNodeFactory](#))

10.2.3.86 ARCS_CLASS_FORWARD() [86/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [ProgramNodeService](#))

10.2.3.87 ARCS_CLASS_FORWARD() [87/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [ProgramNodeView](#))

10.2.3.88 ARCS_CLASS_FORWARD() [88/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [ProgramSelectionParameters](#))

10.2.3.89 ARCS_CLASS_FORWARD() [89/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [RecordTrackNode](#))

10.2.3.90 ARCS_CLASS_FORWARD() [90/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [Register](#))

10.2.3.91 ARCS_CLASS_FORWARD() [91/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [RegisteredCapabilities](#))

10.2.3.92 ARCS_CLASS_FORWARD() [92/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [ReleaseActionParameters](#))

10.2.3.93 ARCS_CLASS_FORWARD() [93/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [RenderInterface](#))

10.2.3.94 ARCS_CLASS_FORWARD() [94/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [ScrewdriverApiProvider](#))

10.2.3.95 ARCS_CLASS_FORWARD() [95/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [ScrewdriverConfiguration](#))

10.2.3.96 ARCS_CLASS_FORWARD() [96/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [ScrewdriverContribution](#))

10.2.3.97 ARCS_CLASS_FORWARD() [97/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [ScrewdriverParameters](#))

10.2.3.98 ARCS_CLASS_FORWARD() [98/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [ScrewdriverProgram](#))

10.2.3.99 ARCS_CLASS_FORWARD() [99/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [ScrewdriverReadyParameters](#))

10.2.3.100 ARCS_CLASS_FORWARD() [100/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [ScrewdrivingNode](#))

10.2.3.101 ARCS_CLASS_FORWARD() [101/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [ScriptNode](#))

10.2.3.102 ARCS_CLASS_FORWARD() [102/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [ScriptWriter](#))

10.2.3.103 ARCS_CLASS_FORWARD() [103/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [SeekNode](#))

10.2.3.104 ARCS_CLASS_FORWARD() [104/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [SetNode](#))

10.2.3.105 ARCS_CLASS_FORWARD() [105/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [SetPayloadNode](#))

10.2.3.106 ARCS_CLASS_FORWARD() [106/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [SpeedCapability](#))

10.2.3.107 ARCS_CLASS_FORWARD() [107/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [SubProgNode](#))

10.2.3.108 ARCS_CLASS_FORWARD() [108/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [SwitchNode](#))

10.2.3.109 ARCS_CLASS_FORWARD() [109/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [SystemApi](#))

10.2.3.110 ARCS_CLASS_FORWARD() [110/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [TCP](#))

10.2.3.111 ARCS_CLASS_FORWARD() [111/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [TcpContributionModel](#))

10.2.3.112 ARCS_CLASS_FORWARD() [112/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [TcpModel](#))

10.2.3.113 ARCS_CLASS_FORWARD() [113/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [ThreadNode](#))

10.2.3.114 ARCS_CLASS_FORWARD() [114/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [TimerNode](#))

10.2.3.115 ARCS_CLASS_FORWARD() [115/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [TreeNode](#))

10.2.3.116 ARCS_CLASS_FORWARD() [116/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [UndoRedoManager](#))

10.2.3.117 ARCS_CLASS_FORWARD() [117/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [UntilNode](#))

10.2.3.118 ARCS_CLASS_FORWARD() [118/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [UserInteraction](#))

10.2.3.119 ARCS_CLASS_FORWARD() [119/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [UserInterfaceApi](#))

10.2.3.120 ARCS_CLASS_FORWARD() [120/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [ValueFactory](#))

10.2.3.121 ARCS_CLASS_FORWARD() [121/130]

arcs::aubo_scope::ARCS_CLASS_FORWARD (
 [Variable](#))

10.2.3.122 ARCS_CLASS_FORWARD() [122/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    VariableFactory )
```

10.2.3.123 ARCS_CLASS_FORWARD() [123/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    VariableModel )
```

10.2.3.124 ARCS_CLASS_FORWARD() [124/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    ViewApiProvider )
```

10.2.3.125 ARCS_CLASS_FORWARD() [125/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    VisualLocalizationProgramNode )
```

10.2.3.126 ARCS_CLASS_FORWARD() [126/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    WaitNode )
```

10.2.3.127 ARCS_CLASS_FORWARD() [127/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    Waypoint )
```

10.2.3.128 ARCS_CLASS_FORWARD() [128/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    WaypointModel )
```

10.2.3.129 ARCS_CLASS_FORWARD() [129/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    WaypointNode )
```

10.2.3.130 ARCS_CLASS_FORWARD() [130/130]

```
arcs::aubo_scope::ARCS_CLASS_FORWARD (
    WidthCapability )
```


10.3 arcs::aubo_sdk Namespace Reference

Functions

- [ARCS_CLASS_FORWARD](#) (RobotProxy)

10.3.1 Function Documentation

10.3.1.1 ARCS_CLASS_FORWARD()

arcs::aubo_sdk::ARCS_CLASS_FORWARD (
RobotProxy)

Chapter 11

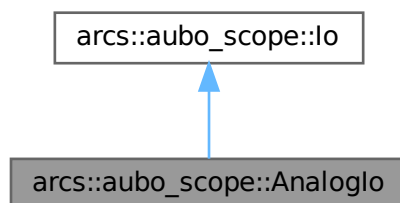
Class Documentation

11.1 arcs::aubo_scope::AnalogIo Class Reference

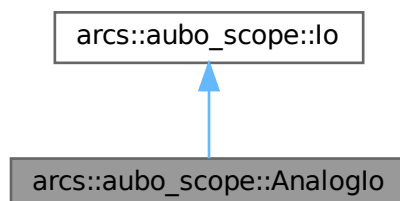
[AnalogIo](#) This interface provides support for analog I/Os.

```
#include <analog_io.h>
```

Inheritance diagram for arcs::aubo_scope::AnalogIo:



Collaboration diagram for arcs::aubo_scope::AnalogIo:



Public Member Functions

- [AnalogIo](#) (AnalogIo &f)
- [AnalogIo](#) (AnalogIo &&f)
- [~AnalogIo](#) ()
- double [getMinRangeValue](#) ()
- double [getMaxRangeValue](#) ()
- bool [isCurrent](#) ()
- bool [isVoltage](#) ()
- bool [setValue](#) (double value)
Set analog I/O output signal.
- double [getValue](#) ()
Returns the last reading of the I/O.

Public Member Functions inherited from [arcs::aubo_scope::Io](#)

- [Io](#) (Io &f)
- [Io](#) (Io &&f)
- virtual [~Io](#) ()
- std::string [getDisplayName](#) () const
Returns the user defined name of the IO in the currently selected installation.
- std::string [getDefaultName](#) () const
Returns the robot default name of the I/O.
- std::string [getValueStr](#) () const
Returns the current reading of the I/O as a string.
- [IoType](#) [getType](#) () const
Returns the I/O type.
- [InterfaceType](#) [getInterfaceType](#) () const
Returns the interface type this I/O belongs to.
- int [getAction](#) () const
Returns the action number of the I/O.
- bool [isInput](#) () const
Returns true if I/O supports reading of values and false otherwise.
- bool [isResolvable](#) () const
Some I/Os, such as MODBUS I/Os and analog Tool inputs, are in some situations not present in AuboScope or available for use.
- size_t [hashCode](#) ()

Private Member Functions

- [AnalogIo](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Public Types inherited from [arcs::aubo_scope::Io](#)

- enum [IoType](#) : int {
 [DIGITAL](#) , [ANALOG](#) , [INTEGER](#) , [BOOLEAN](#) ,
 [DOUBLE](#) }
- enum [InterfaceType](#) : int {
 [STANDARD](#) , [TOOL](#) , [CONFIGURABLE](#) , [LINKIO](#) ,
 [MODBUS](#) , [GENERAL_PURPOSE](#) , [PROFINET](#) , [ETHERNET_IP](#) }

Protected Member Functions inherited from [arcs::aubo_scope::Io](#)

- [Io](#) ()

11.1.1 Detailed Description

[AnalogIo](#) This interface provides support for analog I/Os.

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

11.1.2 Constructor & Destructor Documentation

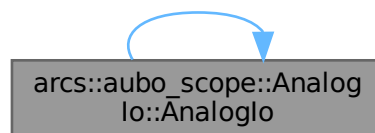
11.1.2.1 AnalogIo() [1/3]

[arcs::aubo_scope::AnalogIo::AnalogIo](#) (
 [AnalogIo](#) & f)

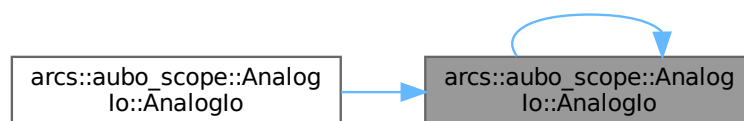
References [AnalogIo\(\)](#).

Referenced by [AnalogIo\(\)](#), and [AnalogIo\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

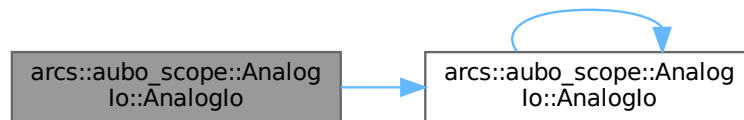


11.1.2.2 AnalogIo() [2/3]

```
arcs::aubo_scope::AnalogIo::AnalogIo (
    AnalogIo && f)
```

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

Here is the call graph for this function:



11.1.2.3 ~AnalogIo()

```
arcs::aubo_scope::AnalogIo::~~AnalogIo ()
```

11.1.2.4 AnalogIo() [3/3]

```
arcs::aubo_scope::AnalogIo::AnalogIo () [private]
```

11.1.3 Member Function Documentation

11.1.3.1 getMaxRangeValue()

```
double arcs::aubo_scope::AnalogIo::getMaxRangeValue ()
```

11.1.3.2 getMinRangeValue()

```
double arcs::aubo_scope::AnalogIo::getMinRangeValue ()
```

11.1.3.3 getValue()

```
double arcs::aubo_scope::AnalogIo::getValue ()
```

Returns the last reading of the I/O.

Returns

Returns the last reading of the I/O.

11.1.3.4 isCurrent()

```
bool arcs::aubo_scope::AnalogIo::isCurrent ()
```

11.1.3.5 isVoltage()

```
bool arcs::aubo_scope::AnalogIo::isVoltage ()
```

11.1.3.6 setValue()

```
bool arcs::aubo_scope::AnalogIo::setValue (
    double value)
```

Set analog I/O output signal.

Notice the I/O must be an output I/O.

Parameters

value	The analog value to be set.

Returns

true if the analog signal was set, false if the signal was not set, e.g. because the controller was not running.

11.1.4 Friends And Related Symbol Documentation

11.1.4.1 DataSwitch

```
friend class DataSwitch [friend]
```

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

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.1.5 Member Data Documentation

11.1.5.1 d_

```
void* arcs::aubo_scope::AnalogIo::d_ { nullptr } [private]
```

Definition at line 65 of file [analog_io.h](#).

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

- include/aubo_caps/domain/io/[analog_io.h](#)

11.2 arcs::aubo_scope::AnalogIoFilter Struct Reference

#include <analog_io.h>

Public Member Functions

- [AnalogIoFilter](#) ([Io::InterfaceType](#) interface, bool is_input)
- bool [operator\(\)](#) (IoPtr io)

Private Attributes

- [Io::InterfaceType](#) interface_
- bool is_input_

11.2.1 Detailed Description

Definition at line 68 of file [analog_io.h](#).

11.2.2 Constructor & Destructor Documentation

11.2.2.1 AnalogIoFilter()

```
arcs::aubo_scope::AnalogIoFilter::AnalogIoFilter (  
    Io::InterfaceType interface,  
    bool is_input) [inline]
```

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

References [interface_](#), and [is_input_](#).

11.2.3 Member Function Documentation

11.2.3.1 operator()()

```
bool arcs::aubo_scope::AnalogIoFilter::operator() (  
    IoPtr io) [inline]
```

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

References [interface_](#), and [is_input_](#).

11.2.4 Member Data Documentation

11.2.4.1 interface_

[Io::InterfaceType](#) arcs::aubo_scope::AnalogIoFilter::interface_ [private]

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

Referenced by [AnalogIoFilter\(\)](#), and [operator\(\)\(\)](#).

11.2.4.2 is_input__

bool arcs::aubo_scope::AnalogIoFilter::is_input__ [private]

Definition at line 83 of file [analog_io.h](#).

Referenced by [AnalogIoFilter\(\)](#), and [operator\(\)\(\)](#).

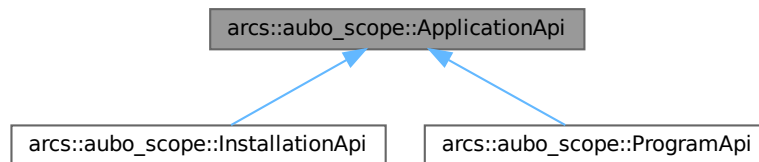
The documentation for this struct was generated from the following file:

- include/aubo_caps/domain/io/[analog_io.h](#)

11.3 arcs::aubo_scope::ApplicationApi Class Reference

```
#include <application_api.h>
```

Inheritance diagram for arcs::aubo_scope::ApplicationApi:



Public Member Functions

- [ApplicationApi](#) (ApplicationApi &f)
- [ApplicationApi](#) (ApplicationApi &&f)
- virtual [~ApplicationApi](#) ()
- IoModelPtr [getIoModel](#) ()
 - [IoModel](#) Returns an interface for working with the inputs and outputs of the robot.
- FeatureModelPtr [getFeatureModel](#) ()
 - [FeatureModel](#) Returns an interface for working with the features of the current installation.
- TcpModelPtr [getTcpModel](#) ()
 - [TcpModel](#) Returns an interface for accessing the TCPs of the current installation.
- PayloadModelPtr [getPayloadModel](#) ()
 - [PayloadModel](#) Gets an interface which provides access the payloads in the current AuboScope installation.
- VariableModelPtr [getVariableModel](#) ()
 - [VariableModel](#) Returns an interface for working with variables.
- ValueFactoryPtr [getValueFactory](#) ()
 - [ValueFactory](#) Returns an interface for accessing various factories capable of creating value objects.
- template<typename T>
std::shared_ptr< T > [getInstallationNode](#) ()
 - This method can be used to get a specific [InstallationNodeContribution](#) instance.
- InstallationNodeContributionPtr [getInstallationNode](#) (const char *typeidOfClass)
- DeviceManagerPtr [getDeviceManager](#) ()
 - [DeviceManager](#) This method can be used to get a specific [DeviceManager](#) instance.
- aubo_sdk::RobotProxyPtr [getRobotProxy](#) ()
 - Returns the robot proxy, providing an interaction interface with the underlying robot SDK.

Protected Member Functions

- [ApplicationApi](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

11.3.1 Detailed Description

Definition at line 41 of file [application_api.h](#).

11.3.2 Constructor & Destructor Documentation

11.3.2.1 ApplicationApi() [1/3]

```
arcs::aubo_scope::ApplicationApi::ApplicationApi (
    ApplicationApi & f)
```

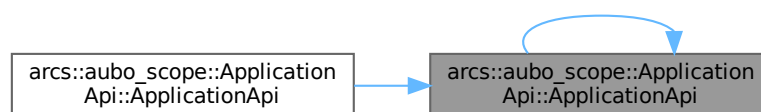
References [ApplicationApi\(\)](#).

Referenced by [ApplicationApi\(\)](#), and [ApplicationApi\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

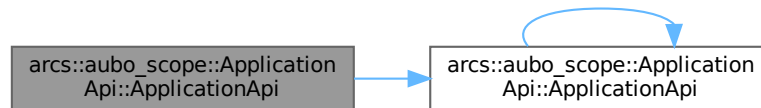


11.3.2.2 ApplicationApi() [2/3]

```
arcs::aubo_scope::ApplicationApi::ApplicationApi (
    ApplicationApi && f)
```

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

Here is the call graph for this function:



11.3.2.3 ~ApplicationApi()

```
virtual arcs::aubo_scope::ApplicationApi::~~ApplicationApi () [virtual]
```

11.3.2.4 ApplicationApi() [3/3]

```
arcs::aubo_scope::ApplicationApi::ApplicationApi () [protected]
```

11.3.3 Member Function Documentation

11.3.3.1 getInstallationNode() [1/2]

```
template<typename T>
std::shared_ptr< T > arcs::aubo_scope::ApplicationApi::getInstallationNode () [inline]
```

This method can be used to get a specific [InstallationNodeContribution](#) instance.

@return An interface with access to various queryable-only system

resources

Parameters

installationType	The class of the installation node contribution to return.
<T>	The generic for specifying the InstallationNodeContribution .

Returns

The installation node instance.

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

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

Referenced by [getInstallationNode\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



11.3.3.2 `getInstallationNode()` [2/2]

```
InstallationNodeContributionPtr arcs::aubo_scope::ApplicationApi::getInstallationNode (
    const char * typeIdOfClass)
```

11.3.3.3 `getRobotProxy()`

```
aubo_sdk::RobotProxyPtr arcs::aubo_scope::ApplicationApi::getRobotProxy ()
```

Returns the robot proxy, providing an interaction interface with the underlying robot SDK.

Returns

The robot proxy pointer.

11.3.4 Friends And Related Symbol Documentation

11.3.4.1 DataSwitch

friend class DataSwitch [friend]

Definition at line 213 of file [application_api.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.3.5 Member Data Documentation

11.3.5.1 d_

void* arcs::aubo_scope::ApplicationApi::d_ { nullptr } [private]

Definition at line 214 of file [application_api.h](#).

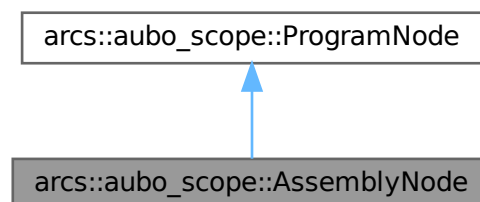
The documentation for this class was generated from the following file:

- include/aubo_caps/domain/[application_api.h](#)

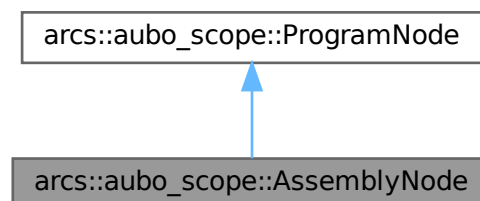
11.4 arcs::aubo_scope::AssemblyNode Class Reference

```
#include <assembly_node.h>
```

Inheritance diagram for arcs::aubo_scope::AssemblyNode:



Collaboration diagram for arcs::aubo_scope::AssemblyNode:



Public Types

- enum [ConfigType](#) { [NONE](#) , [HOLE_SEARCH](#) , [COMPLIANT_INSERTION](#) }

Public Member Functions

- [AssemblyNode](#) (AssemblyNode &f)
- [AssemblyNode](#) (AssemblyNode &&f)
- virtual [~AssemblyNode](#) ()
- void [addAssemblyUntilNode](#) ()

Public Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()
Get the program node contribution.

Private Member Functions

- [AssemblyNode](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Protected Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.4.1 Detailed Description

Definition at line 10 of file [assembly_node.h](#).

11.4.2 Member Enumeration Documentation

11.4.2.1 ConfigType

Enumerator

NONE	
HOLE_SEARCH	
COMPLIANT_INSERTION	

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

11.4.3 Constructor & Destructor Documentation

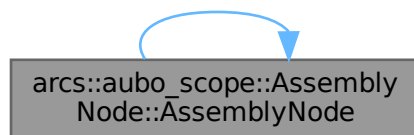
11.4.3.1 AssemblyNode() [1/3]

```
arcs::aubo_scope::AssemblyNode::AssemblyNode (  
    AssemblyNode & f)
```

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

Referenced by [AssemblyNode\(\)](#), and [AssemblyNode\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

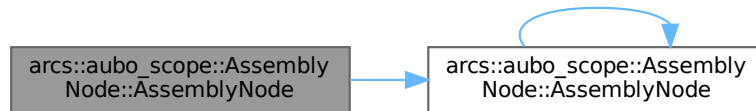


11.4.3.2 AssemblyNode() [2/3]

```
arcs::aubo_scope::AssemblyNode::AssemblyNode (
    AssemblyNode && f)
```

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

Here is the call graph for this function:



11.4.3.3 ~AssemblyNode()

```
virtual arcs::aubo_scope::AssemblyNode::~~AssemblyNode () [virtual]
```

11.4.3.4 AssemblyNode() [3/3]

```
arcs::aubo_scope::AssemblyNode::AssemblyNode () [private]
```

11.4.4 Member Function Documentation

11.4.4.1 addAssemblyUntilNode()

```
void arcs::aubo_scope::AssemblyNode::addAssemblyUntilNode ()
```

11.4.5 Friends And Related Symbol Documentation

11.4.5.1 DataSwitch

```
friend class DataSwitch [friend]
```

Definition at line 27 of file [assembly_node.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.4.6 Member Data Documentation

11.4.6.1 d_

```
void* arcs::aubo_scope::AssignmentNode::d_ { nullptr } [private]
```

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

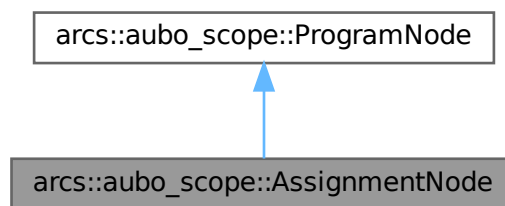
The documentation for this class was generated from the following file:

- include/aubo_caps/domain/program/nodes/builtin/[assembly_node.h](#)

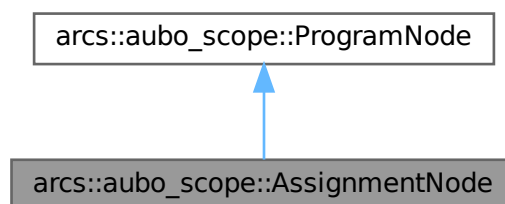
11.5 arcs::aubo_scope::AssignmentNode Class Reference

```
#include <assignment_node.h>
```

Inheritance diagram for arcs::aubo_scope::AssignmentNode:



Collaboration diagram for arcs::aubo_scope::AssignmentNode:



Public Types

- enum [ConfigType](#) { [EXPRESSION](#) , [OPERATOR](#) }
 - enum [InputMode](#) { [INPUT_BOOL](#) , [INPUT_INTEGER](#) , [INPUT_FLOAT](#) , [INPUT_STRING](#) }
- The configuration type used to determine which type of configuration this instance is.

Public Member Functions

- [AssignmentNode](#) (AssignmentNode &f)
- [AssignmentNode](#) (AssignmentNode &&f)
- virtual [~AssignmentNode](#) ()
- [ConfigType](#) [getConfigType](#) ()
- void [setConfigType](#) ([ConfigType](#) type)
- VariablePtr [getVariable](#) ()
- void [setVariable](#) (VariablePtr var)
- ExpressionPtr [getExpression](#) ()
- void [setExpression](#) (ExpressionPtr exp)

Public Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()
Get the program node contribution.

Private Member Functions

- [AssignmentNode](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Protected Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.5.1 Detailed Description

Definition at line 12 of file [assignment_node.h](#).

11.5.2 Member Enumeration Documentation

11.5.2.1 ConfigType

enum [arcs::aubo_scope::AssignmentNode::ConfigType](#)

Generated by Doxygen

The configuration type used to determine which type of configuration this instance is.

Enumerator

EXPRESSION	Expression assignment has been selected. The config instance can be cast to ExpressionAssignmentNodeConfig.
OPERATOR	Operator assignment has been selected. This type has no further information. Setting this type of config will be ignored and will have no effect on the node.

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

11.5.2.2 InputMode

enum [arcs::aubo_scope::AssignmentNode::InputMode](#)

Enumerator

INPUT_BOOL	
INPUT_INTEGER	
INPUT_FLOAT	
INPUT_STRING	

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

11.5.3 Constructor & Destructor Documentation

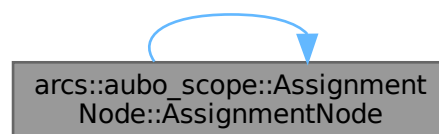
11.5.3.1 AssignmentNode() [1/3]

[arcs::aubo_scope::AssignmentNode::AssignmentNode](#) (
AssignmentNode & f)

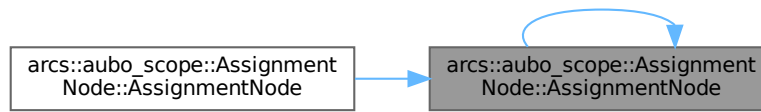
References [AssignmentNode\(\)](#).

Referenced by [AssignmentNode\(\)](#), and [AssignmentNode\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

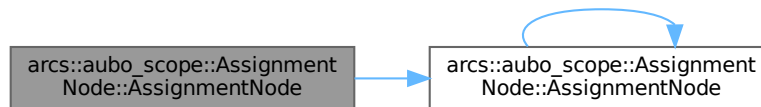


11.5.3.2 AssignmentNode() [2/3]

```
arcs::aubo_scope::AssignmentNode::AssignmentNode (
    AssignmentNode && f)
```

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

Here is the call graph for this function:



11.5.3.3 ~AssignmentNode()

```
virtual arcs::aubo_scope::AssignmentNode::~~AssignmentNode () [virtual]
```

11.5.3.4 AssignmentNode() [3/3]

```
arcs::aubo_scope::AssignmentNode::AssignmentNode () [private]
```

11.5.4 Member Function Documentation

11.5.4.1 getConfigType()

```
ConfigType arcs::aubo_scope::AssignmentNode::getConfigType ()
```

Returns

11.5.4.2 getExpression()

ExpressionPtr arcs::aubo_scope::AssignmentNode::getExpression ()

Returns

the expression whose evaluation is assigned to the variable.

11.5.4.3 getVariable()

VariablePtr arcs::aubo_scope::AssignmentNode::getVariable ()

Returns

the variable assigned to the expression evaluation.

11.5.4.4 setConfigType()

void arcs::aubo_scope::AssignmentNode::setConfigType (
 [ConfigType](#) type)

11.5.4.5 setExpression()

void arcs::aubo_scope::AssignmentNode::setExpression (
 ExpressionPtr exp)

11.5.4.6 setVariable()

void arcs::aubo_scope::AssignmentNode::setVariable (
 VariablePtr var)

11.5.5 Friends And Related Symbol Documentation

11.5.5.1 DataSwitch

friend class DataSwitch [friend]

Definition at line 75 of file [assignment_node.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.5.6 Member Data Documentation

11.5.6.1 d_

```
void* arcs::aubo_scope::AssignmentNode::d_ { nullptr } [private]
```

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

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

- [include/aubo_caps/domain/program/nodes/builtin/assignment_node.h](#)

11.6 arcs::aubo_scope::AuboCapInfo Class Reference

This interface provides various metadata information about an installed AuboCap, such as the name of the AuboCap.

```
#include <extension_info.h>
```

Public Member Functions

- virtual [~AuboCapInfo](#) ()=default
- virtual std::string [getSymbolicName](#) ()=0
@retAubon The symbolic name (id) of the AuboCap.
- virtual std::string [getName](#) ()=0
@retAubon The name of the AuboCap.
- virtual std::string [getVendor](#) ()=0
@retAubon The vendor of the AuboCap.

11.6.1 Detailed Description

This interface provides various metadata information about an installed AuboCap, such as the name of the AuboCap.

Since

1.1.0

Definition at line 17 of file [extension_info.h](#).

11.6.2 Constructor & Destructor Documentation

11.6.2.1 ~AuboCapInfo()

```
virtual arcs::aubo_scope::AuboCapInfo::~AuboCapInfo () [virtual], [default]
```

11.6.3 Member Function Documentation

11.6.3.1 getName()

virtual std::string arcs::aubo_scope::AuboCapInfo::getName () [pure virtual]

@retAubon The name of the AuboCap.

11.6.3.2 getSymbolicName()

virtual std::string arcs::aubo_scope::AuboCapInfo::getSymbolicName () [pure virtual]

@retAubon The symbolic name (id) of the AuboCap.

11.6.3.3 getVendor()

virtual std::string arcs::aubo_scope::AuboCapInfo::getVendor () [pure virtual]

@retAubon The vendor of the AuboCap.

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

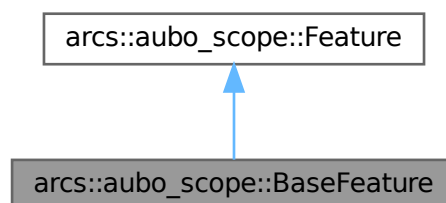
- include/aubo_caps/domain/[extension_info.h](#)

11.7 arcs::aubo_scope::BaseFeature Class Reference

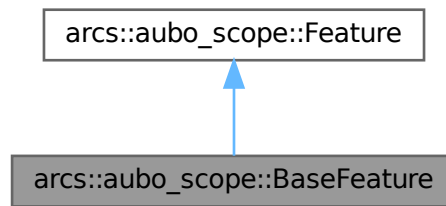
This interface represents the Base feature in AuboScope.

```
#include <base_feature.h>
```

Inheritance diagram for arcs::aubo_scope::BaseFeature:



Collaboration diagram for `arcs::aubo_scope::BaseFeature`:



Public Member Functions

- [BaseFeature](#) (BaseFeature &f)
- [BaseFeature](#) (BaseFeature &&f)
- virtual [~BaseFeature](#) ()

Public Member Functions inherited from [arcs::aubo_scope::Feature](#)

- [Feature](#) (Feature &f)
- [Feature](#) (Feature &&f)
- virtual [~Feature](#) ()
- std::string [getDisplayname](#) ()
 - Note: The feature can be renamed at any time.
- bool [isDefined](#) ()
 - Returns whether the position of the feature has been fully defined/specified.
- bool [isVariable](#) ()
 - Returns whether the feature can be used as a variable in the program.
- bool [isJoggable](#) ()
- bool [isResolvable](#) ()
 - A feature cannot be guaranteed to be present in AuboScope.
- std::vector< double > [getPose](#) ()
 - Returns the Cartesian location of the feature.
- size_t [hashCode](#) ()

Private Member Functions

- [BaseFeature](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Protected Member Functions inherited from [arcs::aubo_scope::Feature](#)

- [Feature](#) ()

11.7.1 Detailed Description

This interface represents the Base feature in AuboScope.

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

11.7.2 Constructor & Destructor Documentation

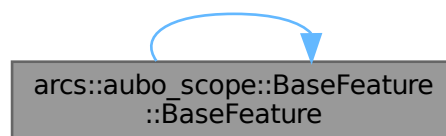
11.7.2.1 BaseFeature() [1/3]

```
arcs::aubo_scope::BaseFeature::BaseFeature (  
    BaseFeature & f)
```

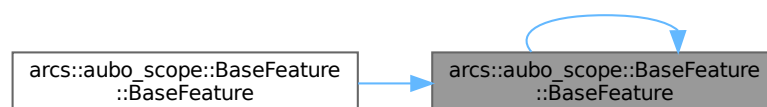
References [BaseFeature\(\)](#).

Referenced by [BaseFeature\(\)](#), and [BaseFeature\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

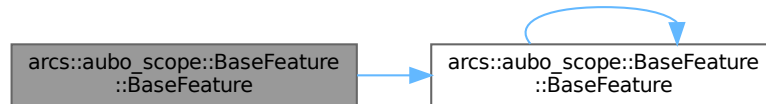


11.7.2.2 BaseFeature() [2/3]

```
arcs::aubo_scope::BaseFeature::BaseFeature (
    BaseFeature && f)
```

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

Here is the call graph for this function:



11.7.2.3 ~BaseFeature()

```
virtual arcs::aubo_scope::BaseFeature::~~BaseFeature () [virtual]
```

11.7.2.4 BaseFeature() [3/3]

```
arcs::aubo_scope::BaseFeature::BaseFeature () [private]
```

11.7.3 Friends And Related Symbol Documentation

11.7.3.1 DataSwitch

```
friend class DataSwitch [friend]
```

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

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.7.4 Member Data Documentation

11.7.4.1 d_

```
void* arcs::aubo_scope::BaseFeature::d_ { nullptr } [private]
```

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

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

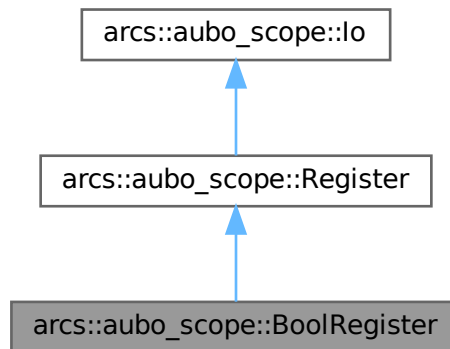
- [include/aubo_caps/domain/feature/base_feature.h](#)

11.8 arcs::aubo_scope::BoolRegister Class Reference

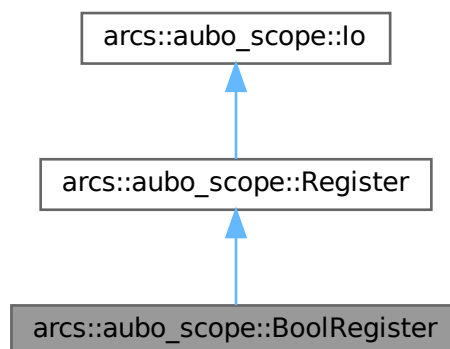
This interface provides support for bool registers.

```
#include <boolean_register.h>
```

Inheritance diagram for arcs::aubo_scope::BoolRegister:



Collaboration diagram for arcs::aubo_scope::BoolRegister:



Public Member Functions

- virtual [~BoolRegister](#) ()=default

Public Member Functions inherited from [arcs::aubo_scope::Register](#)

- virtual [~Register](#) ()=default

Public Member Functions inherited from [arcs::aubo_scope::Io](#)

- [Io](#) (Io &f)
- [Io](#) (Io &&f)
- virtual [~Io](#) ()
- std::string [getDisplayName](#) () const
Returns the user defined name of the IO in the currently selected installation.
- std::string [getDefaultName](#) () const
Returns the robot default name of the I/O.
- std::string [getValueStr](#) () const
Returns the current reading of the I/O as a string.
- [IoType](#) [getType](#) () const
Returns the I/O type.
- [InterfaceType](#) [getInterfaceType](#) () const
Returns the interface type this I/O belongs to.
- int [getAction](#) () const
Returns the action number of the I/O.
- bool [isInput](#) () const
Returns true if I/O supports reading of values and false otherwise.
- bool [isResolvable](#) () const
Some I/Os, such as MODBUS I/Os and analog Tool inputs, are in some situations not present in AuboScope or available for use.
- size_t [hashCode](#) ()

Additional Inherited Members

Public Types inherited from [arcs::aubo_scope::Io](#)

- enum [IoType](#) : int {
 [DIGITAL](#) , [ANALOG](#) , [INTEGER](#) , [BOOLEAN](#) ,
 [DOUBLE](#) }
- enum [InterfaceType](#) : int {
 [STANDARD](#) , [TOOL](#) , [CONFIGURABLE](#) , [LINKIO](#) ,
 [MODBUS](#) , [GENERAL_PURPOSE](#) , [PROFINET](#) , [ETHERNET_IP](#) }

Protected Member Functions inherited from [arcs::aubo_scope::Io](#)

- [Io](#) ()

11.8.1 Detailed Description

This interface provides support for bool registers.

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

11.8.2 Constructor & Destructor Documentation

11.8.2.1 ~BoolRegister()

virtual arcs::aubo_scope::BoolRegister::~~BoolRegister () [virtual], [default]

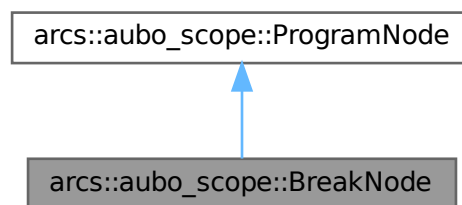
The documentation for this class was generated from the following file:

- [include/aubo_caps/domain/io/boolean_register.h](#)

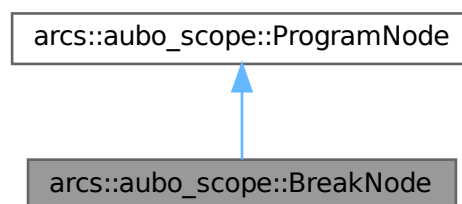
11.9 arcs::aubo_scope::BreakNode Class Reference

```
#include <break_node.h>
```

Inheritance diagram for arcs::aubo_scope::BreakNode:



Collaboration diagram for arcs::aubo_scope::BreakNode:



Public Member Functions

- [BreakNode](#) (BreakNode &f)
- [BreakNode](#) (BreakNode &&f)
- virtual [~BreakNode](#) ()

Public Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()
Get the program node contribution.

Private Member Functions

- [BreakNode](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Protected Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.9.1 Detailed Description

Definition at line 10 of file [break_node.h](#).

11.9.2 Constructor & Destructor Documentation

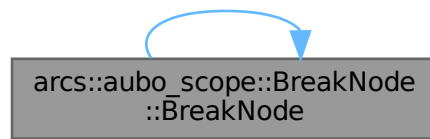
11.9.2.1 BreakNode() [1/3]

```
arcs::aubo_scope::BreakNode::BreakNode (  
    BreakNode & f)
```

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

Referenced by [BreakNode\(\)](#), and [BreakNode\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

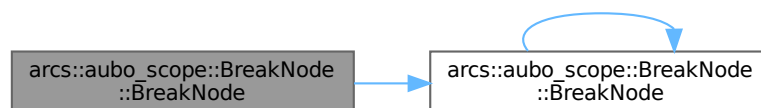


11.9.2.2 BreakNode() [2/3]

```
arcs::aubo_scope::BreakNode::BreakNode (  
    BreakNode && f)
```

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

Here is the call graph for this function:



11.9.2.3 ~BreakNode()

```
virtual arcs::aubo_scope::BreakNode::~~BreakNode () [virtual]
```

11.9.2.4 BreakNode() [3/3]

arcs::aubo_scope::BreakNode::BreakNode () [private]

11.9.3 Friends And Related Symbol Documentation

11.9.3.1 DataSwitch

friend class DataSwitch [friend]

Definition at line 18 of file [break_node.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.9.4 Member Data Documentation

11.9.4.1 d_

void* arcs::aubo_scope::BreakNode::d_ { nullptr } [private]

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

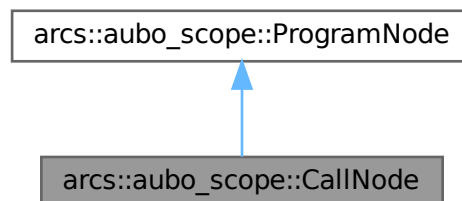
The documentation for this class was generated from the following file:

- [include/aubo_caps/domain/program/nodes/builtin/break_node.h](#)

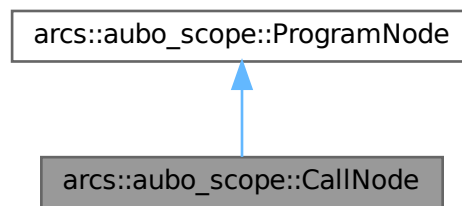
11.10 arcs::aubo_scope::CallNode Class Reference

```
#include <call_node.h>
```

Inheritance diagram for arcs::aubo_scope::CallNode:



Collaboration diagram for arcs::aubo_scope::CallNode:



Public Member Functions

- [CallNode](#) (CallNode &f)
- [CallNode](#) (CallNode &&f)
- virtual [~CallNode](#) ()

Public Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()
Get the program node contribution.

Private Member Functions

- [CallNode](#) ()

Private Attributes

- void * [d__](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Protected Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.10.1 Detailed Description

Definition at line 10 of file [call_node.h](#).

11.10.2 Constructor & Destructor Documentation

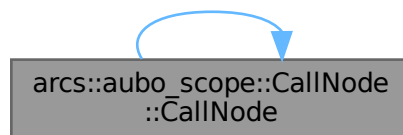
11.10.2.1 CallNode() [1/3]

```
arcs::aubo_scope::CallNode::CallNode (
    CallNode & f)
```

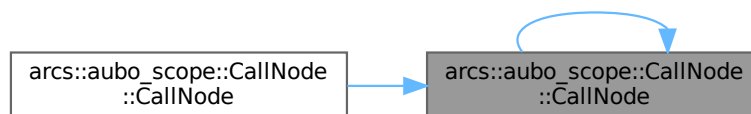
References [CallNode\(\)](#).

Referenced by [CallNode\(\)](#), and [CallNode\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

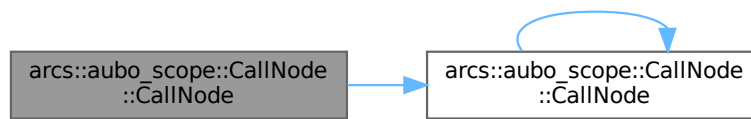


11.10.2.2 CallNode() [2/3]

```
arcs::aubo_scope::CallNode::CallNode (
    CallNode && f)
```

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

Here is the call graph for this function:



11.10.2.3 ~CallNode()

virtual arcs::aubo_scope::CallNode::~~CallNode () [virtual]

11.10.2.4 CallNode() [3/3]

arcs::aubo_scope::CallNode::CallNode () [private]

11.10.3 Friends And Related Symbol Documentation

11.10.3.1 DataSwitch

friend class DataSwitch [friend]

Definition at line 18 of file [call_node.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.10.4 Member Data Documentation

11.10.4.1 d__

void* arcs::aubo_scope::CallNode::d__ { nullptr } [private]

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

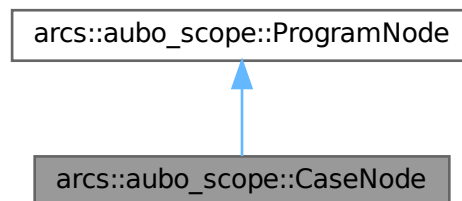
The documentation for this class was generated from the following file:

- [include/aubo_caps/domain/program/nodes/builtin/call_node.h](#)

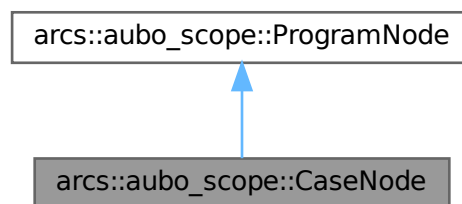
11.11 arcs::aubo_scope::CaseNode Class Reference

#include <case_node.h>

Inheritance diagram for arcs::aubo_scope::CaseNode:



Collaboration diagram for arcs::aubo_scope::CaseNode:



Public Member Functions

- [CaseNode](#) (CaseNode &f)
- [CaseNode](#) (CaseNode &&f)
- virtual [~CaseNode](#) ()

Public Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()
Get the program node contribution.

Private Member Functions

- [CaseNode](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Protected Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.11.1 Detailed Description

Definition at line 10 of file [case_node.h](#).

11.11.2 Constructor & Destructor Documentation

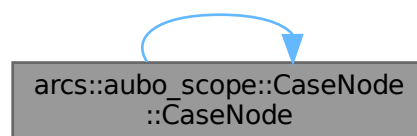
11.11.2.1 CaseNode() [1/3]

```
arcs::aubo_scope::CaseNode::CaseNode (
    CaseNode & f)
```

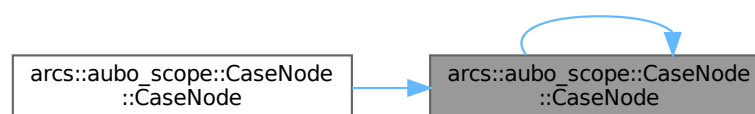
References [CaseNode\(\)](#).

Referenced by [CaseNode\(\)](#), and [CaseNode\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

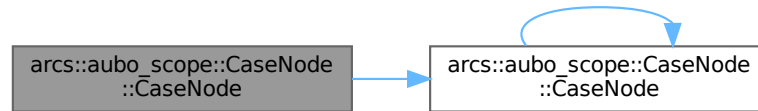


11.11.2.2 CaseNode() [2/3]

```
arcs::aubo_scope::CaseNode::CaseNode (
    CaseNode && f)
```

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

Here is the call graph for this function:



11.11.2.3 ~CaseNode()

```
virtual arcs::aubo_scope::CaseNode::~~CaseNode () [virtual]
```

11.11.2.4 CaseNode() [3/3]

```
arcs::aubo_scope::CaseNode::CaseNode () [private]
```

11.11.3 Friends And Related Symbol Documentation

11.11.3.1 DataSwitch

```
friend class DataSwitch [friend]
```

Definition at line 18 of file [case_node.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.11.4 Member Data Documentation

11.11.4.1 d__

```
void* arcs::aubo_scope::CaseNode::d__ { nullptr } [private]
```

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

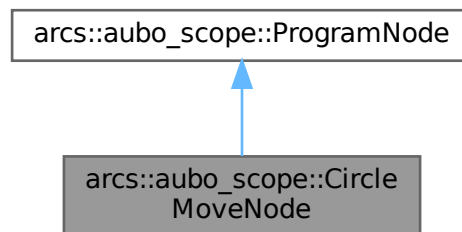
The documentation for this class was generated from the following file:

- [include/aubo_caps/domain/program/nodes/builtin/case_node.h](#)

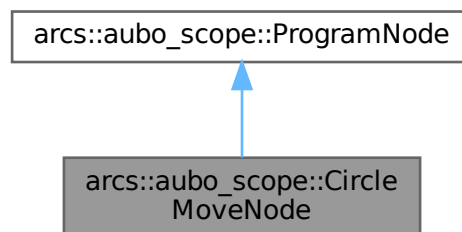
11.12 arcs::aubo_scope::CircleMoveNode Class Reference

#include <circle_move_node.h>

Inheritance diagram for arcs::aubo_scope::CircleMoveNode:



Collaboration diagram for arcs::aubo_scope::CircleMoveNode:



Public Member Functions

- [CircleMoveNode](#) (CircleMoveNode &f)
- [CircleMoveNode](#) (CircleMoveNode &&f)
- virtual [~CircleMoveNode](#) ()

Public Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()
Get the program node contribution.

Private Member Functions

- [CircleMoveNode](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Protected Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.12.1 Detailed Description

Definition at line 10 of file [circle_move_node.h](#).

11.12.2 Constructor & Destructor Documentation

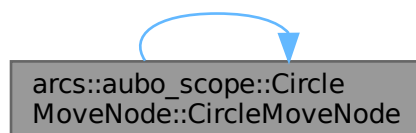
11.12.2.1 CircleMoveNode() [1/3]

```
arcs::aubo_scope::CircleMoveNode::CircleMoveNode (
    CircleMoveNode & f)
```

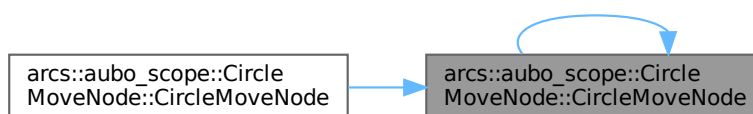
References [CircleMoveNode\(\)](#).

Referenced by [CircleMoveNode\(\)](#), and [CircleMoveNode\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

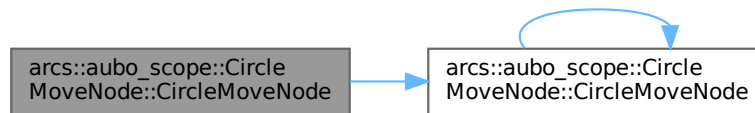


11.12.2.2 CircleMoveNode() [2/3]

```
arcs::aubo_scope::CircleMoveNode::CircleMoveNode (
    CircleMoveNode && f)
```

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

Here is the call graph for this function:



11.12.2.3 ~CircleMoveNode()

```
virtual arcs::aubo_scope::CircleMoveNode::~~CircleMoveNode () [virtual]
```

11.12.2.4 CircleMoveNode() [3/3]

```
arcs::aubo_scope::CircleMoveNode::CircleMoveNode () [private]
```

11.12.3 Friends And Related Symbol Documentation

11.12.3.1 DataSwitch

```
friend class DataSwitch [friend]
```

Definition at line 18 of file [circle_move_node.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.12.4 Member Data Documentation

11.12.4.1 d__

```
void* arcs::aubo_scope::CircleMoveNode::d__ { nullptr } [private]
```

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

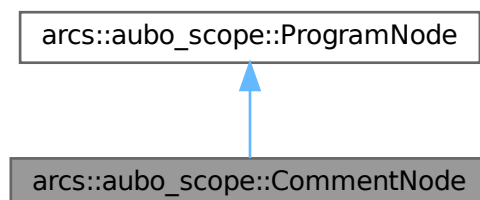
The documentation for this class was generated from the following file:

- `include/aubo_caps/domain/program/nodes/builtin/circle_move_node.h`

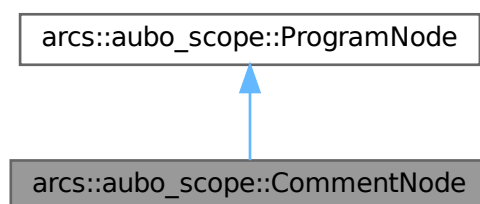
11.13 arcs::aubo_scope::CommentNode Class Reference

#include <comment_node.h>

Inheritance diagram for arcs::aubo_scope::CommentNode:



Collaboration diagram for arcs::aubo_scope::CommentNode:



Public Member Functions

- [CommentNode](#) (CommentNode &f)
- [CommentNode](#) (CommentNode &&f)
- virtual [~CommentNode](#) ()
- std::string [getComment](#) () const
- void [setComment](#) (const std::string &param_string)

Public Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()
Get the program node contribution.

Private Member Functions

- [CommentNode](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Protected Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.13.1 Detailed Description

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

11.13.2 Constructor & Destructor Documentation

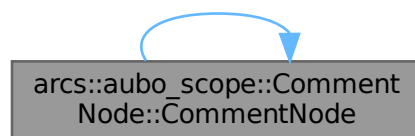
11.13.2.1 [CommentNode](#)() [1/3]

```
arcs::aubo_scope::CommentNode::CommentNode (
    CommentNode & f)
```

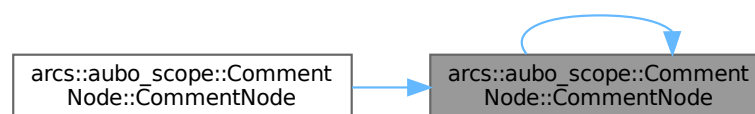
References [CommentNode\(\)](#).

Referenced by [CommentNode\(\)](#), and [CommentNode\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

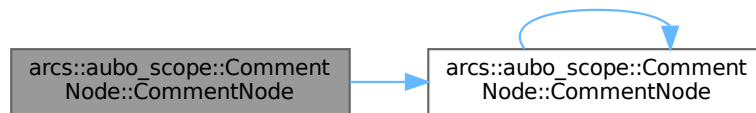


11.13.2.2 CommentNode() [2/3]

```
arcs::aubo_scope::CommentNode::CommentNode (
    CommentNode && f)
```

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

Here is the call graph for this function:



11.13.2.3 ~CommentNode()

```
virtual arcs::aubo_scope::CommentNode::~~CommentNode () [virtual]
```

11.13.2.4 CommentNode() [3/3]

```
arcs::aubo_scope::CommentNode::CommentNode () [private]
```

11.13.3 Member Function Documentation

11.13.3.1 getComment()

```
std::string arcs::aubo_scope::CommentNode::getComment () const
```

Returns

The text of the Comment node.

11.13.3.2 setComment()

```
void arcs::aubo_scope::CommentNode::setComment (
    const std::string & param_string)
```

- Set the text for the Comment node.

Parameters

comment	The Comment for the node.

Returns

This node.

11.13.4 Friends And Related Symbol Documentation

11.13.4.1 DataSwitch

friend class DataSwitch [friend]

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

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.13.5 Member Data Documentation

11.13.5.1 d__

void* arcs::aubo_scope::CommentNode::d__ { nullptr } [private]

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

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

- include/aubo_caps/domain/program/nodes/builtin/[comment_node.h](#)

11.14 arcs::aubo_scope::ContributionProvider< T > Class Template Reference

[ContributionProvider](#) This interfaces provides access to a contribution.

```
#include <contribution_provider.h>
```

Public Member Functions

- virtual [~ContributionProvider](#) ()=default
- virtual T * [get](#) ()=0
Gets a contribution.

```
template<class T>
```

```
class arcs::aubo_scope::ContributionProvider< T >
```

Template Parameters

<code><T></code>	the (generic) type parameter for the interface representing the type of contribution the interface provides access to

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

11.14.2 Constructor & Destructor Documentation

11.14.2.1 ~ContributionProvider()

```
template<class T>
virtual arcs::aubo\_scope::ContributionProvider< T >::~~ContributionProvider () [virtual], [default]
```

11.14.3 Member Function Documentation

11.14.3.1 get()

```
template<class T>
virtual T * arcs::aubo\_scope::ContributionProvider< T >::get () [pure virtual]
```

Gets a contribution.

Call this as the result of a user action to get the currently selected node.

Returns

the relevant contribution

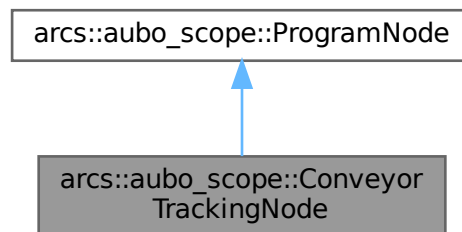
The documentation for this class was generated from the following file:

- [include/aubo_caps/contribution/contribution_provider.h](#)

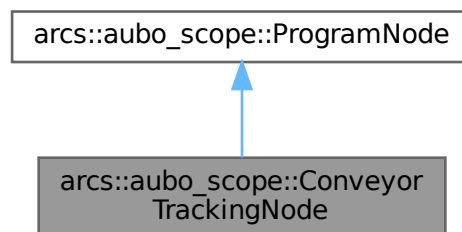
11.15 arcs::aubo_scope::ConveyorTrackingNode Class Reference

#include <conveyor_tracking_node.h>

Inheritance diagram for arcs::aubo_scope::ConveyorTrackingNode:



Collaboration diagram for arcs::aubo_scope::ConveyorTrackingNode:



Public Types

- enum `ConveyorTrackingTargetType` { `NEW_WORKPIECE` , `PREVIOUS_WORKPIECE` }
- enum `ConveyorStartActionQueueType` { `CLEAR_QUEUE` , `RESERVE_QUEUE` , `ASK` }
- enum `ConveyorStartActionType` { `CONTINUE_TRACKING` , `ABANDON_TRACKING` }

Public Member Functions

- `ConveyorTrackingNode` (`ConveyorTrackingNode &f`)
- `ConveyorTrackingNode` (`ConveyorTrackingNode &&f`)
- virtual `~ConveyorTrackingNode` ()

Public Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()
Get the program node contribution.

Private Member Functions

- [ConveyorTrackingNode](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Protected Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.15.1 Detailed Description

Definition at line 10 of file [conveyor_tracking_node.h](#).

11.15.2 Member Enumeration Documentation

11.15.2.1 ConveyorStartActionQueueType

enum [arcs::aubo_scope::ConveyorTrackingNode::ConveyorStartActionQueueType](#)

Enumerator

CLEAR_QUEUE	
RESERVE_QUEUE	
ASK	

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

11.15.2.2 ConveyorStartActionType

Enumerator

CONTINUE_TRACKING	
ABANDON_TRACKING	

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

11.15.2.3 ConveyorTrackingTargetType

enum [arcs::aubo_scope::ConveyorTrackingNode::ConveyorTrackingTargetType](#)

Enumerator

NEW_WORKPIECE	
PREVIOUS_WORKPIECE	

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

11.15.3 Constructor & Destructor Documentation

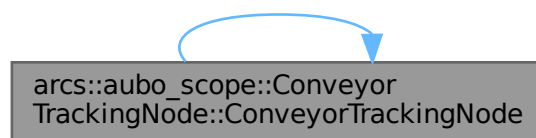
11.15.3.1 ConveyorTrackingNode() [1/3]

[arcs::aubo_scope::ConveyorTrackingNode::ConveyorTrackingNode](#) (
ConveyorTrackingNode & f)

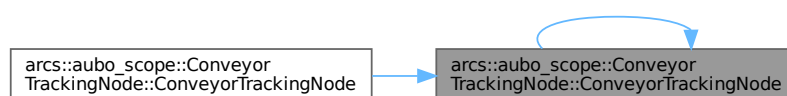
References [ConveyorTrackingNode\(\)](#).

Referenced by [ConveyorTrackingNode\(\)](#), and [ConveyorTrackingNode\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



11.15.3.2 ConveyorTrackingNode() [2/3]

```
arcs::aubo_scope::ConveyorTrackingNode::ConveyorTrackingNode (
    ConveyorTrackingNode && f)
```

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

Here is the call graph for this function:



11.15.3.3 ~ConveyorTrackingNode()

```
virtual arcs::aubo_scope::ConveyorTrackingNode::~~ConveyorTrackingNode () [virtual]
```

11.15.3.4 ConveyorTrackingNode() [3/3]

```
arcs::aubo_scope::ConveyorTrackingNode::ConveyorTrackingNode () [private]
```

11.15.4 Friends And Related Symbol Documentation

11.15.4.1 DataSwitch

```
friend class DataSwitch [friend]
```

Definition at line 37 of file [conveyor_tracking_node.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.15.5 Member Data Documentation

11.15.5.1 d__

```
void* arcs::aubo_scope::ConveyorTrackingNode::d__ { nullptr } [private]
```

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

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

- [include/aubo_caps/domain/program/nodes/builtin/conveyor_tracking_node.h](#)

11.16 arcs::aubo_scope::DataModel Class Reference

[DataModel](#) This interface is used for storing and managing data that represents the current configuration of, e.g.

```
#include <data_model.h>
```

Public Member Functions

- [DataModel](#) (DataModel &f)
- [DataModel](#) (DataModel &&f)
- [~DataModel](#) ()
- void [set](#) (const std::string &key, bool value)
Assign a bool value to the specified key.
- bool [get](#) (const std::string &key, bool default_value) const
Get the bool value assigned to the specified key.
- void [set](#) (const std::string &key, int value)
Assign an int value to the specified key.
- int [get](#) (const std::string &key, int default_value) const
Get the int value assigned to the specified key.
- void [set](#) (const std::string &key, long value)
Assign a long value to the specified key.
- void [set](#) (const std::string &key, uint64_t value)
- long [get](#) (const std::string &key, long default_value) const
Get the long value assigned to the specified key.
- uint64_t [get](#) (const std::string &key, uint64_t default_value) const
- void [set](#) (const std::string &key, float value)
Assign a float value to the specified key.
- float [get](#) (const std::string &key, float default_value) const
Get the float value assigned to the specified key.
- void [set](#) (const std::string &key, double value)
Assign a double value to the specified key.
- double [get](#) (const std::string &key, double default_value) const
Get the double value assigned to the specified key.
- void [set](#) (const std::string &key, const std::string &value)
Assign a std::string value to the specified key.
- std::string [get](#) (const std::string &key, const std::string &default_value) const
Get the std::string value assigned to the specified key.
- void [set](#) (const std::string &key, VariablePtr value)
Assign a [Variable](#) value to the specified key.
- void [set](#) (const std::string &key, ExpressionPtr value)
- void [set](#) (const std::string &key, PayloadPtr value)
- void [set](#) (const std::string &key, IoPtr value)
- void [set](#) (const std::string &key, WaypointPtr value)
- VariablePtr [get](#) (const std::string &key, VariablePtr default_value) const
Get the [Variable](#) value assigned to the specified key.
- ExpressionPtr [get](#) (const std::string &key, ExpressionPtr default_value) const
- PayloadPtr [get](#) (const std::string &key, PayloadPtr default_value) const
- IoPtr [get](#) (const std::string &key, IoPtr default_value) const
- WaypointPtr [get](#) (const std::string &key, WaypointPtr default_value) const
- void [set](#) (const std::string &key, const std::vector< bool > &value)

- Assign a `bool` as value to the specified key.
- `std::vector< bool > get (const std::string &key, const std::vector< bool > &default_value) const`
Get the `bool` as value assigned to the specified key.
- `void set (const std::string &key, const std::vector< int > &value)`
Assign a `int` as value to the specified key.
- `std::vector< int > get (const std::string &key, const std::vector< int > &default_value) const`
Get the `int` as value assigned to the specified key.
- `void set (const std::string &key, const std::vector< long > &value)`
Assign a `long` as value to the specified key.
- `void set (const std::string &key, const std::vector< uint64_t > &value)`
- `std::vector< long > get (const std::string &key, const std::vector< long > &default_value) const`
Get the `long` as value assigned to the specified key.
- `std::vector< uint64_t > get (const std::string &key, const std::vector< uint64_t > &default_value) const`
- `void set (const std::string &key, const std::vector< float > &value)`
Assign a `float` as value to the specified key.
- `std::vector< float > get (const std::string &key, const std::vector< float > &default_value) const`
Get the `float` as value assigned to the specified key.
- `void set (const std::string &key, const std::vector< double > &value)`
Assign a `double` as value to the specified key.
- `std::vector< double > get (const std::string &key, const std::vector< double > &default_value) const`
Get the `double` as value assigned to the specified key.
- `void set (const std::string &key, const std::vector< std::string > &value)`
Assign a `std::string` as value to the specified key.
- `std::vector< std::string > get (const std::string &key, const std::vector< std::string > &default_value) const`
Get the `std::string` as value assigned to the specified key.
- `void set (const std::string &key, TCPPtr value)`
Assign a `TCP` value to the specified key.
- `TCPPtr get (const std::string &key, TCPPtr default_value) const`
Get the `TCP` value assigned to the specified key.
- `void set (const std::string &key, FeaturePtr value)`
Assign a `Feature` value to the specified key.
- `FeaturePtr get (const std::string &key, FeaturePtr default_value) const`
Get the `Feature` value assigned to the specified key.
- `std::set< std::string > getKeys () const`
Get a set of all the keys in the data model.
- `bool isSet (const std::string &key) const`
Check if a key is present in the data model.
- `bool remove (const std::string &key)`
Remove a key-value pair from the data model.

Private Member Functions

- `DataModel ()`

Private Attributes

- `void * d_ { nullptr }`

Friends

- class [DataSwitch](#)

11.16.1 Detailed Description

[DataModel](#) This interface is used for storing and managing data that represents the current configuration of, e.g.

a [ProgramNodeContribution](#) or [InstallationNodeContribution](#). Methods for adding, removing, retrieving and changing values in a dictionary are provided.

Setting a new value for a key already in use, will overwrite the current value with the new value. This happens regardless of the type of value (e.g. storing the value true under the key myBool and afterwards storing an Angle object under the same key will overwrite the value true with the provided Angle object).

A auboCap installation screen has an underlying [DataModel](#) object. That object is saved and loaded along with each AuboScope installation.

Similarly, each contributed program node instance has an underlying [DataModel](#) object. That object is saved and loaded along with the program where the node occurs. Undo/redo actions are supported for all modifications to the [DataModel](#) object in HTML-based program node contributions. Qt-based aubo_studio plugins must use the [UndoRedoManager](#) to record the changes on the undo/redo stack.

When retrieving an object, both key and object type must match what was previously stored. This means that if a [TCP](#) object was stored using the key myAngle, then attempting to retrieve it using `get(std::string key, Angle default_value)` with the key myAngle will not return the stored value, since the types do not match. Instead the provided default_value will be returned.

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

11.16.2 Constructor & Destructor Documentation

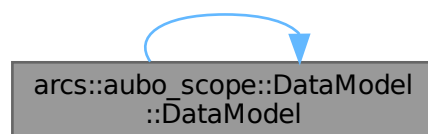
11.16.2.1 DataModel() [1/3]

```
arcs::aubo_scope::DataModel::DataModel (
    DataModel & f)
```

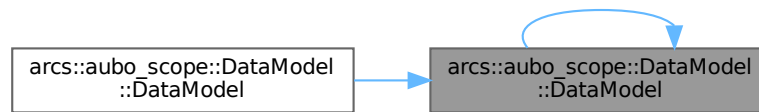
References [DataModel\(\)](#).

Referenced by [DataModel\(\)](#), and [DataModel\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



11.16.2.2 `DataModel()` [2/3]

```
arcs::aubo_scope::DataModel::DataModel (
    DataModel && f)
```

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

Here is the call graph for this function:



11.16.2.3 `~DataModel()`

```
arcs::aubo_scope::DataModel::~DataModel ()
```

11.16.2.4 `DataModel()` [3/3]

```
arcs::aubo_scope::DataModel::DataModel () [private]
```

11.16.3 Member Function Documentation

11.16.3.1 `get()` [1/21]

```
bool arcs::aubo_scope::DataModel::get (
    const std::string & key,
    bool default_value) const
```

Get the bool value assigned to the specified key.

Parameters

key	key in the data model.
default_value	value to be returned, if key does not exist.

Returns

the value assigned to the key. If not exist, default_value is returned.

11.16.3.2 get() [2/21]

```
std::string arcs::aubo_scope::DataModel::get (  
    const std::string & key,  
    const std::string & default_value) const
```

Get the std::string value assigned to the specified key.

Parameters

key	key in the data model.
default_value	value to be returned, if key does not exist.

Returns

the value assigned to the key. If not exist, default_value is returned.

11.16.3.3 get() [3/21]

```
std::vector< bool > arcs::aubo_scope::DataModel::get (  
    const std::string & key,  
    const std::vector< bool > & default_value) const
```

Get the bool[] as value assigned to the specified key.

Parameters

key	key in the data model.
default_value	value to be returned, if key does not exist.

Parameters

key	key in the data model.
default_value	value to be returned, if key does not exist.

Returns

the value assigned to the key. If not exist, default_value is returned.

11.16.3.5 get() [5/21]

```
std::vector< float > arcs::aubo_scope::DataModel::get (
    const std::string & key,
    const std::vector< float > & default_value) const
```

Get the float[] as value assigned to the specified key.

Parameters

key	key in the data model.
default_value	value to be returned, if key does not exist.

Returns

the value assigned to the key. If not exist, default_value is returned.

11.16.3.6 get() [6/21]

```
std::vector< int > arcs::aubo_scope::DataModel::get (
    const std::string & key,
    const std::vector< int > & default_value) const
```

Get the int[] as value assigned to the specified key.

Parameters

key	key in the data model.
default_value	value to be returned, if key does not exist.

Returns

the value assigned to the key. If not exist, default_value is returned.

Parameters

key	key in the data model.
default_value	value to be returned, if key does not exist.

Returns

the value assigned to the key. If not exist, default_value is returned.

11.16.3.8 get() [8/21]

```
std::vector< std::string > arcs::aubo_scope::DataModel::get (
    const std::string & key,
    const std::vector< std::string > & default_value) const
```

Get the std::string[] as value assigned to the specified key.

Parameters

key	key in the data model.
default_value	value to be returned, if key does not exist.

Returns

the value assigned to the key. If not exist, default_value is returned.

11.16.3.9 get() [9/21]

```
std::vector< uint64_t > arcs::aubo_scope::DataModel::get (
    const std::string & key,
    const std::vector< uint64_t > & default_value) const
```

11.16.3.10 get() [10/21]

```
double arcs::aubo_scope::DataModel::get (
    const std::string & key,
    double default_value) const
```

Get the double value assigned to the specified key.

Parameters

key	key in the data model.
default_value	value to be returned, if key does not exist.

Returns

the value assigned to the key. If not exist, default_value is returned.

11.16.3.11 get() [11/21]

```
ExpressionPtr arcs::aubo_scope::DataModel::get (
    const std::string & key,
    ExpressionPtr default_value) const
```

11.16.3.12 get() [12/21]

```
FeaturePtr arcs::aubo_scope::DataModel::get (
    const std::string & key,
    FeaturePtr default_value) const
```

Get the [Feature](#) value assigned to the specified key.

Parameters

key	key in the data model.
default_value	value to be returned, if key does not exist.

Returns

the value assigned to the key. If not exist, default_value is returned.

11.16.3.13 get() [13/21]

```
float arcs::aubo_scope::DataModel::get (
    const std::string & key,
    float default_value) const
```

Get the float value assigned to the specified key.

Parameters

key	key in the data model.
default_value	value to be returned, if key does not exist.

Returns

the value assigned to the key. If not exist, default_value is returned.

11.16.3.14 get() [14/21]

```
int arcs::aubo_scope::DataModel::get (  
    const std::string & key,  
    int default_value) const
```

Get the int value assigned to the specified key.

Parameters

key	key in the data model.
default_value	value to be returned, if key does not exist.

Returns

the value assigned to the key. If not exist, default_value is returned.

11.16.3.15 get() [15/21]

```
IoPtr arcs::aubo_scope::DataModel::get (  
    const std::string & key,  
    IoPtr default_value) const
```

11.16.3.16 get() [16/21]

```
long arcs::aubo_scope::DataModel::get (  
    const std::string & key,  
    long default_value) const
```

Get the long value assigned to the specified key.

Parameters

key	key in the data model.
default_value	value to be returned, if key does not exist.

Returns

the value assigned to the key. If not exist, default_value is returned.

11.16.3.17 get() [17/21]

```
PayloadPtr arcs::aubo_scope::DataModel::get (  
    const std::string & key,  
    PayloadPtr default_value) const
```

11.16.3.18 get() [18/21]

```
TCPPtr arcs::aubo_scope::DataModel::get (  
    const std::string & key,  
    TCPPtr default_value) const
```

Get the [TCP](#) value assigned to the specified key.

Parameters

key	key in the data model.
default_value	value to be returned, if key does not exist.

Returns

the value assigned to the key. If not exist, default_value is returned.

11.16.3.19 get() [19/21]

```
uint64_t arcs::aubo_scope::DataModel::get (  
    const std::string & key,  
    uint64_t default_value) const
```

11.16.3.20 get() [20/21]

```
VariablePtr arcs::aubo_scope::DataModel::get (  
    const std::string & key,  
    VariablePtr default_value) const
```

Get the [Variable](#) value assigned to the specified key.

Parameters

key	key in the data model.
default_value	value to be returned, if key does not exist.

Returns

the value assigned to the key. If not exist, default_value is returned.

11.16.3.21 get() [21/21]

```
WaypointPtr arcs::aubo_scope::DataModel::get (
    const std::string & key,
    WaypointPtr default_value) const
```

11.16.3.22 getKeys()

```
std::set< std::string > arcs::aubo_scope::DataModel::getKeys () const
```

Get a set of all the keys in the data model.

Returns

A Set of keys.

11.16.3.23 isSet()

```
bool arcs::aubo_scope::DataModel::isSet (
    const std::string & key) const
```

Check if a key is present in the data model.

Parameters

key	key in the data model (not null and not an empty std::string).

Returns

true, if key exist, otherwise false.

11.16.3.24 remove()

Generated by Doxygen

```
bool arcs::aubo_scope::DataModel::remove (
    const std::string & key)
```

Parameters

key	key in the data model (not null and not an empty std::string).

Returns

true, if succeed, otherwise false.

Exceptions

IllegalStateException	if called from a Qt-based AuboCap program node outside of an UndoableChanges scope (see also UndoRedoManager).

11.16.3.25 set() [1/21]

```
void arcs::aubo_scope::DataModel::set (
    const std::string & key,
    bool value)
```

Assign a bool value to the specified key.

Parameters

key	key in the data model (not null and not an empty std::string).
value	value assigned to key.

Exceptions

IllegalArgumentException	if the key is null or an empty std::string.
IllegalStateException	if called from a Qt-based AuboCap program node outside of an UndoableChanges scope (see also UndoRedoManager).

11.16.3.26 set() [2/21]

```
void arcs::aubo_scope::DataModel::set (
    const std::string & key,
    const std::string & value)
```

Assign a std::string value to the specified key.

Parameters

key	key in the data model (not null and not an empty std::string).
value	value assigned to key.

Exceptions

IllegalArgumentException	if the key is null or an empty std::string.
IllegalStateException	if called from a Qt-based AuboCap program node outside of an UndoableChanges scope (see also UndoRedoManager).

11.16.3.27 set() [3/21]

```
void arcs::aubo_scope::DataModel::set (
    const std::string & key,
    const std::vector< bool > & value)
```

Assign a bool[] as value to the specified key.

Parameters

key	key in the data model (not null) and not an empty std::string).
value	value assigned to key.

Exceptions

IllegalArgumentException	if the key is null or an empty std::string.
IllegalStateException	if called from a Qt-based AuboCap program node outside of an UndoableChanges scope (see also UndoRedoManager).

11.16.3.28 set() [4/21]

```
void arcs::aubo_scope::DataModel::set (
    const std::string & key,
    const std::vector< double > & value)
```

Assign a double[] as value to the specified key.

Parameters

key	key in the data model (not null and not an empty std::string).
value	value assigned to key.

Exceptions

IllegalArgumentException	if the key is null or an empty std::string.
IllegalStateException	if called from a Qt-based AuboCap program node outside of an UndoableChanges scope (see also UndoRedoManager).

11.16.3.29 set() [5/21]

```
void arcs::aubo_scope::DataModel::set (
    const std::string & key,
    const std::vector< float > & value)
```

Assign a float[] as value to the specified key.

Parameters

key	key in the data model (not null and not an empty std::string).
value	value assigned to key.

Exceptions

IllegalArgumentException	if the key is null or an empty std::string.
IllegalStateException	if called from a Qt-based AuboCap program node outside of an UndoableChanges scope (see also UndoRedoManager).

11.16.3.30 set() [6/21]

```
void arcs::aubo_scope::DataModel::set (
    const std::string & key,
    const std::vector< int > & value)
```

Assign a int[] as value to the specified key.

Parameters

key	key in the data model (not null and not an empty std::string).
value	value assigned to key.

Exceptions

IllegalArgumentException	if the key is null or an empty std::string.
IllegalStateException	if called from a Qt-based AuboCap program node outside of an UndoableChanges scope (see also UndoRedoManager).

11.16.3.31 set() [7/21]

```
void arcs::aubo_scope::DataModel::set (
    const std::string & key,
    const std::vector< long > & value)
```

Assign a long[] as value to the specified key.

Parameters

key	key in the data model (not null and not an empty std::string).
value	value assigned to key.

Exceptions

IllegalArgumentException	if the key is null} or an empty std::string.
IllegalStateException	if called from a Qt-based AuboCap program node outside of an UndoableChanges scope (see also UndoRedoManager).

11.16.3.32 set() [8/21]

```
void arcs::aubo_scope::DataModel::set (
    const std::string & key,
    const std::vector< std::string > & value)
```

Assign a std::string[] as value to the specified key.

Parameters

key	key in the data model (not null and not an empty std::string).
value	value assigned to key.

Exceptions

IllegalArgumentException	if the key is null or an empty std::string.
IllegalStateException	if called from a Qt-based AuboCap program node outside of an UndoableChanges scope (see also UndoRedoManager).

11.16.3.33 set() [9/21]

```
void arcs::aubo_scope::DataModel::set (
    const std::string & key,
    const std::vector< uint64_t > & value)
```

11.16.3.34 set() [10/21]

```
void arcs::aubo_scope::DataModel::set (
    const std::string & key,
    double value)
```

Assign a double value to the specified key.

Parameters

key	key in the data model (not null and not an empty std::string).
value	value assigned to key.

Exceptions

IllegalArgumentException	if the key is null or an empty std::string.
IllegalStateException	if called from a Qt-based AuboCap program node outside of an UndoableChanges scope (see also UndoRedoManager).

11.16.3.35 set() [11/21]

```
void arcs::aubo_scope::DataModel::set (
    const std::string & key,
    ExpressionPtr value)
```

11.16.3.36 set() [12/21]

```
void arcs::aubo_scope::DataModel::set (
    const std::string & key,
    FeaturePtr value)
```

Assign a [Feature](#) value to the specified key.

Parameters

key	key in the data model (not null and not an empty std::string).
value	value assigned to key.

Exceptions

IllegalArgumentException	if the key is null or an empty std::string.
IllegalStateException	if called from a Qt-based AuboCap program node outside of an UndoableChanges scope (see also UndoRedoManager).

11.16.3.37 set() [13/21]

```
void arcs::aubo_scope::DataModel::set (
    const std::string & key,
    float value)
```

Assign a float value to the specified key.

Parameters

key	key in the data model (not null and not an empty std::string).
value	value assigned to key.

Exceptions

IllegalArgumentException	if the key is null or an empty std::string.
IllegalStateException	if called from a Qt-based AboCap program node outside of an UndoableChanges scope (see also UndoRedoManager).

11.16.3.38 set() [14/21]

```
void arcs::aubo_scope::DataModel::set (
    const std::string & key,
    int value)
```

Assign an int value to the specified key.

Parameters

key	key in the data model (not null and not an empty std::string).
value	value assigned to key.

Exceptions

IllegalArgumentException	if the key is null or an empty std::string.
IllegalStateException	if called from a Qt-based AboCap program node outside of an UndoableChanges scope (see also UndoRedoManager).

11.16.3.39 set() [15/21]

```
void arcs::aubo_scope::DataModel::set (
    const std::string & key,
    IoPtr value)
```

11.16.3.40 set() [16/21]

```
void arcs::aubo_scope::DataModel::set (
    const std::string & key,
    long value)
```

Assign a long value to the specified key.

Parameters

key	key in the data model (not null and not an empty std::string).
value	value assigned to key.

Exceptions

IllegalArgumentException	if the key is null or an empty std::string.
IllegalStateException	if called from a Qt-based AuboCap program node outside of an UndoableChanges scope (see also UndoRedoManager).

11.16.3.41 set() [17/21]

```
void arcs::aubo_scope::DataModel::set (
    const std::string & key,
    PayloadPtr value)
```

11.16.3.42 set() [18/21]

```
void arcs::aubo_scope::DataModel::set (
    const std::string & key,
    TCPPtr value)
```

Assign a [TCP](#) value to the specified key.

Parameters

key	key in the data model (not null and not an empty std::string).
value	value assigned to key.

Exceptions

IllegalArgumentException	if the key is null or an empty std::string.
IllegalStateException	if called from a Qt-based AuboCap program node outside of an UndoableChanges scope (see also UndoRedoManager).

11.16.3.43 `set()` [19/21]

```
void arcs::aubo_scope::DataModel::set (
    const std::string & key,
    uint64_t value)
```

11.16.3.44 `set()` [20/21]

```
void arcs::aubo_scope::DataModel::set (
    const std::string & key,
    VariablePtr value)
```

Assign a [Variable](#) value to the specified key.

Parameters

key	key in the data model (not null and not an empty std::string).
value	value assigned to key.

Exceptions

IllegalArgumentException	if the key is null or an empty std::string.
IllegalStateException	if called from a Qt-based AuboCap program node outside of an UndoableChanges scope (see also UndoRedoManager).

11.16.3.45 `set()` [21/21]

```
void arcs::aubo_scope::DataModel::set (
    const std::string & key,
    WaypointPtr value)
```

11.16.4 Friends And Related Symbol Documentation

11.16.4.1 DataSwitch

friend class DataSwitch [friend]

Definition at line 708 of file [data_model.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.16.5 Member Data Documentation

11.16.5.1 d_

`void* arcs::aubo_scope::DataModel::d_ { nullptr } [private]`

Definition at line 710 of file [data_model.h](#).

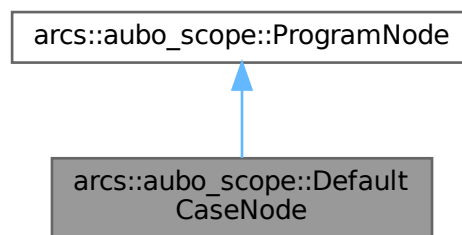
The documentation for this class was generated from the following file:

- `include/aubo_caps/domain/data/data_model.h`

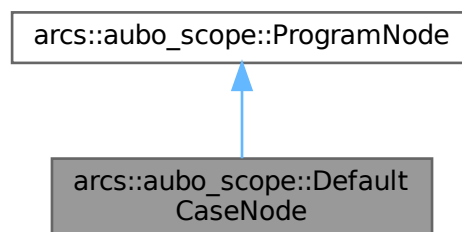
11.17 arcs::aubo_scope::DefaultCaseNode Class Reference

```
#include <default_case_node.h>
```

Inheritance diagram for arcs::aubo_scope::DefaultCaseNode:



Collaboration diagram for arcs::aubo_scope::DefaultCaseNode:



Public Member Functions

- [DefaultCaseNode](#) (DefaultCaseNode &f)
- [DefaultCaseNode](#) (DefaultCaseNode &&f)
- virtual [~DefaultCaseNode](#) ()

Public Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()
Get the program node contribution.

Private Member Functions

- [DefaultCaseNode](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Protected Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.17.1 Detailed Description

Definition at line 10 of file [default_case_node.h](#).

11.17.2 Constructor & Destructor Documentation

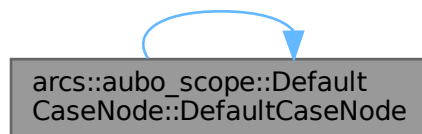
11.17.2.1 DefaultCaseNode() [1/3]

arcs::aubo_scope::DefaultCaseNode::DefaultCaseNode (
DefaultCaseNode & f)

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

Referenced by [DefaultCaseNode\(\)](#), and [DefaultCaseNode\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

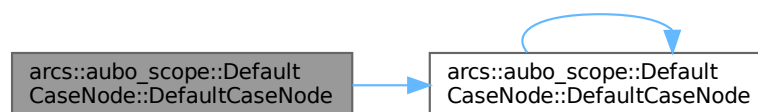


11.17.2.2 DefaultCaseNode() [2/3]

arcs::aubo_scope::DefaultCaseNode::DefaultCaseNode (
DefaultCaseNode && f)

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

Here is the call graph for this function:



11.17.2.3 ~DefaultCaseNode()

virtual arcs::aubo_scope::DefaultCaseNode::~~DefaultCaseNode () [virtual]

11.17.2.4 DefaultCaseNode() [3/3]

arcs::aubo_scope::DefaultCaseNode::DefaultCaseNode () [private]

11.17.3 Friends And Related Symbol Documentation

11.17.3.1 DataSwitch

friend class DataSwitch [friend]

Definition at line 18 of file [default_case_node.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.17.4 Member Data Documentation

11.17.4.1 d__

void* arcs::aubo_scope::DefaultCaseNode::d__ { nullptr } [private]

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

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

- [include/aubo_caps/domain/program/nodes/builtin/default_case_node.h](#)

11.18 arcs::aubo_scope::DefaultSingletonTraits< Type > Struct Template Reference

11.18.1 Detailed Description

```
template<typename Type>
struct arcs::aubo_scope::DefaultSingletonTraits< Type >
```

Definition at line 12 of file [popup_manager.h](#).

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

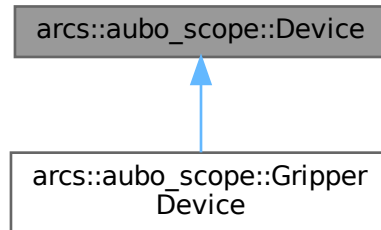
- [include/aubo_caps/domain/user_interface/popup_manager.h](#)

11.19 arcs::aubo_scope::Device Class Reference

Device This base interface represents a device supported by AuboScope, e.g.

```
#include <device.h>
```

Inheritance diagram for arcs::aubo_scope::Device:



Public Member Functions

- virtual `~Device ()=0`
- virtual `AuboCapInfoPtr getContributorInfo ()=0`
Get information about the AuboCap that added/contributed this gripper device to AuboScope.
- virtual `std::string getId ()=0`
- virtual `std::string getDisplayName ()=0`
- virtual `bool isResolvable ()=0`
A device cannot be guaranteed to be present in AuboScope.

11.19.1 Detailed Description

Device This base interface represents a device supported by AuboScope, e.g.

a gripping device.

Definition at line 18 of file `device.h`.

11.19.2 Constructor & Destructor Documentation

11.19.2.1 ~Device()

```
virtual arcs::aubo_scope::Device::~Device () [pure virtual]
```

11.19.3 Member Function Documentation

11.19.3.1 getContributorInfo()

virtual AuboCapInfoPtr arcs::aubo_scope::Device::getContributorInfo () [pure virtual]

Get information about the AuboCap that added/contributed this gripper device to AuboScope.

Returns

AuboCap information for this gripper device.

11.19.3.2 getDisplayName()

virtual std::string arcs::aubo_scope::Device::getDisplayName () [pure virtual]

The name of the gripper displayed in the AuboScope UI, e.g. in the Gripper toolbar and program node.

Note: This name must not be used to uniquely identify a specific individual gripper, since the name might be translated (depending on the selected language in AuboScope) or be changed between versions of the AuboCap that added the gripper.

Returns

The name of this gripper displayed in AuboScope.

11.19.3.3 getId()

virtual std::string arcs::aubo_scope::Device::getId () [pure virtual]

Get the unique identifier (id) for this gripper. This can be used to identify this specific gripper among the set of individual grippers available in a specific multi-gripper device.

Note that this id is not "globally" unique, but only unique for the individual grippers belonging to a specific multi-gripper device.

Returns

The unique identifier of this gripper.

11.19.3.4 isResolvable()

virtual bool arcs::aubo_scope::Device::isResolvable () [pure virtual]

A device cannot be guaranteed to be present in AuboScope.

A device will not be present if the AuboCap that added the device is not installed. This method can be used to determine, if the device is present.

Returns

true if this gripper device is present in AuboScope, otherwise false.

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

- include/aubo_caps/domain/device/[device.h](#)

11.20 arcs::aubo_scope::DeviceManager Class Reference

[DeviceManager](#) Base interface for device managers.

```
#include <device_manager.h>
```

Public Member Functions

- virtual [~DeviceManager](#) ()=default
- virtual std::vector< GripperDevicePtr > [getGrippers](#) ()=0
Returns the list of grippers (gripper devices) available in AuboScope.
- virtual std::vector< DevicePtr > [getScrewDrivers](#) ()=0
Returns the list of screwdrivers available in AuboScope.

11.20.1 Detailed Description

[DeviceManager](#) Base interface for device managers.

A device manager provides functionality for a specific type of device supported by AuboScope. The device manager can provide functionality for, e.g. accessing all registered devices (of that type) and using a specific device in a program.

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

11.20.2 Constructor & Destructor Documentation

11.20.2.1 ~DeviceManager()

```
virtual arcs::aubo_scope::DeviceManager::~~DeviceManager () [virtual], [default]
```

11.20.3 Member Function Documentation

11.20.3.1 getGrippers()

```
virtual std::vector< GripperDevicePtr > arcs::aubo_scope::DeviceManager::getGrippers () [pure virtual]
```

Returns the list of grippers (gripper devices) available in AuboScope.

Returns

The list of gripper devices available in AuboScope.

11.20.3.2 getScrewDrivers()

virtual std::vector< DevicePtr > arcs::aubo_scope::DeviceManager::getScrewDrivers () [pure virtual]

Returns the list of screwdrivers available in AuboScope.

Returns

The list of screwdriver devices available in AuboScope.

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

- include/aubo_caps/domain/device/[device_manager.h](#)

11.21 arcs::aubo_scope::DeviceRegistrationManager Class Reference

[DeviceRegistrationManager](#) Base interface for device registration managers.

```
#include <device_registration_manager.h>
```

Public Member Functions

- virtual [~DeviceRegistrationManager](#) ()=default
- virtual void [registerGripperContribution](#) (GripperContributionPtr contribution)=0
[Register](#) a gripper contribution.
- virtual void [registerScrewDriverContribution](#) (ScrewdriverContributionPtr contribution)=0
[Register](#) a screwdriver contribution.

11.21.1 Detailed Description

[DeviceRegistrationManager](#) Base interface for device registration managers.

A device registration manager is used for registering a program/installation node contribution/service as conformant with a specific device type supported by AuboScope.

Definition at line 27 of file [device_registration_manager.h](#).

11.21.2 Constructor & Destructor Documentation

11.21.2.1 ~DeviceRegistrationManager()

virtual arcs::aubo_scope::DeviceRegistrationManager::~~DeviceRegistrationManager () [virtual], [default]

11.21.3 Member Function Documentation

11.21.3.1 registerGripperContribution()

virtual void arcs::aubo_scope::DeviceRegistrationManager::registerGripperContribution (GripperContributionPtr contribution) [pure virtual]

Parameters

contribution	the gripper contribution to register.

11.21.3.2 registerScrewDriverContribution()

```
virtual void arcs::aubo_scope::DeviceRegistrationManager::registerScrewDriverContribution (
    ScrewdriverContributionPtr contribution) [pure virtual]
```

[Register](#) a screwdriver contribution.

Parameters

contribution	the screwdriver contribution to register.

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

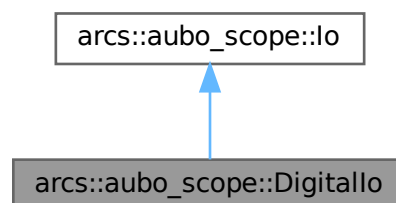
- include/aubo_caps/domain/device/[device_registration_manager.h](#)

11.22 arcs::aubo_scope::DigitalIo Class Reference

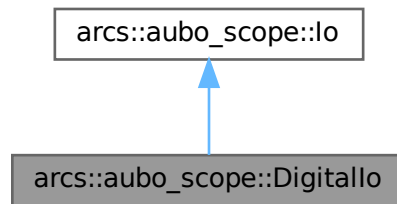
[DigitalIo](#) This interface provides support for digital I/Os.

```
#include <digital_io.h>
```

Inheritance diagram for arcs::aubo_scope::DigitalIo:



Collaboration diagram for `arcs::aubo_scope::DigitalIo`:



Public Member Functions

- [DigitalIo](#) (`DigitalIo &f`)
- [DigitalIo](#) (`DigitalIo &&f`)
- [~DigitalIo](#) (`()`)
- `bool` [setValue](#) (`bool high`)
Set digital I/O output signal.
- `bool` [getValue](#) (`()`)
Returns the last reading of the I/O.

Public Member Functions inherited from [arcs::aubo_scope::Io](#)

- [Io](#) (`Io &f`)
- [Io](#) (`Io &&f`)
- `virtual` [~Io](#) (`()`)
- `std::string` [getDisplayName](#) (`() const`)
Returns the user defined name of the IO in the currently selected installation.
- `std::string` [getDefaultName](#) (`() const`)
Returns the robot default name of the I/O.
- `std::string` [getValueStr](#) (`() const`)
Returns the current reading of the I/O as a string.
- [IoType](#) [getType](#) (`() const`)
Returns the I/O type.
- [InterfaceType](#) [getInterfaceType](#) (`() const`)
Returns the interface type this I/O belongs to.
- `int` [getAction](#) (`() const`)
Returns the action number of the I/O.
- `bool` [isInput](#) (`() const`)
Returns true if I/O supports reading of values and false otherwise.
- `bool` [isResolvable](#) (`() const`)
Some I/Os, such as MODBUS I/Os and analog Tool inputs, are in some situations not present in AuboScope or available for use.
- `size_t` [hashCode](#) (`()`)

Private Member Functions

- [DigitalIo](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Public Types inherited from [arcs::aubo_scope::Io](#)

- enum [IoType](#) : int {
 [DIGITAL](#) , [ANALOG](#) , [INTEGER](#) , [BOOLEAN](#) ,
 [DOUBLE](#) }
- enum [InterfaceType](#) : int {
 [STANDARD](#) , [TOOL](#) , [CONFIGURABLE](#) , [LINKIO](#) ,
 [MODBUS](#) , [GENERAL_PURPOSE](#) , [PROFINET](#) , [ETHERNET_IP](#) }

Protected Member Functions inherited from [arcs::aubo_scope::Io](#)

- [Io](#) ()

11.22.1 Detailed Description

[DigitalIo](#) This interface provides support for digital I/Os.

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

11.22.2 Constructor & Destructor Documentation

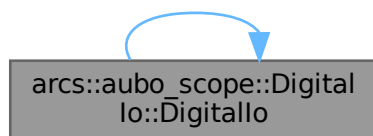
11.22.2.1 [DigitalIo](#)() [1/3]

```
arcs::aubo_scope::DigitalIo::DigitalIo (  
    DigitalIo & f)
```

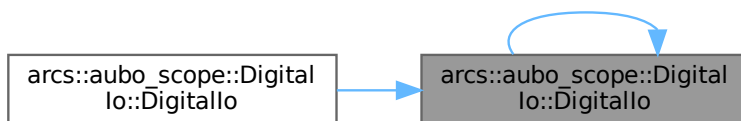
References [DigitalIo](#)().

Referenced by [DigitalIo](#)(), and [DigitalIo](#)().

Here is the call graph for this function:



Here is the caller graph for this function:

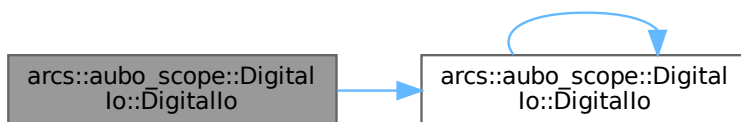


11.22.2.2 `DigitalIo()` [2/3]

```
arcs::aubo_scope::DigitalIo::DigitalIo (
    DigitalIo && f)
```

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

Here is the call graph for this function:



11.22.2.3 `~DigitalIo()`

```
arcs::aubo_scope::DigitalIo::~~DigitalIo ()
```

11.22.2.4 DigitalIo() [3/3]

arcs::aubo_scope::DigitalIo::DigitalIo () [private]

11.22.3 Member Function Documentation

11.22.3.1 getValue()

bool arcs::aubo_scope::DigitalIo::getValue ()

Returns the last reading of the I/O.

Returns

Returns the last reading of the I/O.

11.22.3.2 setValue()

bool arcs::aubo_scope::DigitalIo::setValue (
 bool high)

Set digital I/O output signal.

Note that the I/O must be an output I/O.

Parameters

high	the digital value to be set.

Returns

true if the digital signal was set, false if the signal was not set, e.g. because the controller was not running.

11.22.4 Friends And Related Symbol Documentation

11.22.4.1 DataSwitch

friend class DataSwitch [friend]

Definition at line 57 of file [digital_io.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.22.5 Member Data Documentation

11.22.5.1 d_

```
void* arcs::aubo_scope::DigitalIo::d_ { nullptr } [private]
```

Definition at line 59 of file [digital_io.h](#).

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

- [include/aubo_caps/domain/io/digital_io.h](#)

11.23 arcs::aubo_scope::DigitalIoFilter Struct Reference

```
#include <digital_io.h>
```

Public Member Functions

- [DigitalIoFilter](#) ([Io::InterfaceType](#) interface, bool is__input)
- bool [operator\(\)](#) (IoPtr io)

Private Attributes

- [Io::InterfaceType](#) [interface__](#)
- bool [is__input__](#)

11.23.1 Detailed Description

Definition at line 62 of file [digital_io.h](#).

11.23.2 Constructor & Destructor Documentation

11.23.2.1 DigitalIoFilter()

```
arcs::aubo_scope::DigitalIoFilter::DigitalIoFilter (  
    Io::InterfaceType interface,  
    bool is__input) [inline]
```

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

References [interface__](#), and [is__input__](#).

11.23.3 Member Function Documentation

11.23.3.1 operator()()

bool arcs::aubo_scope::DigitalIoFilter::operator() (
IoPtr io) [inline]

Definition at line 68 of file [digital_io.h](#).

References [interface__](#), and [is_input__](#).

11.23.4 Member Data Documentation

11.23.4.1 interface__

[Io::InterfaceType](#) arcs::aubo_scope::DigitalIoFilter::interface__ [private]

Definition at line 76 of file [digital_io.h](#).

Referenced by [DigitalIoFilter\(\)](#), and [operator\(\)\(\)](#).

11.23.4.2 is_input__

bool arcs::aubo_scope::DigitalIoFilter::is_input__ [private]

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

Referenced by [DigitalIoFilter\(\)](#), and [operator\(\)\(\)](#).

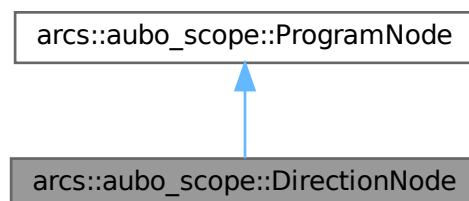
The documentation for this struct was generated from the following file:

- [include/aubo_caps/domain/io/digital_io.h](#)

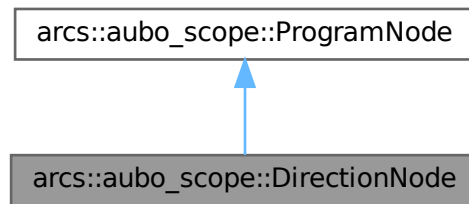
11.24 arcs::aubo_scope::DirectionNode Class Reference

```
#include <direction_node.h>
```

Inheritance diagram for arcs::aubo_scope::DirectionNode:



Collaboration diagram for arcs::aubo_scope::DirectionNode:



Public Types

- enum `ParameterType` : int { `SHARED_MOTION` , `TOOL_MOTION` }
- enum `Axis` : int { `X_PLUS` , `X_MINUS` , `Y_PLUS` , `Y_MINUS` , `Z_PLUS` , `Z_MINUS` }
- enum `DirectionSelectionType` : int { `EXPRESSION` , `AXIS` }
- enum `FeatureSelectionType` : int { `UNRESOLVED_FEATURE` , `SHARED` , `FEATURE` }

Public Member Functions

- `DirectionNode` (`DirectionNode &f`)
- `DirectionNode` (`DirectionNode &&f`)
- virtual `~DirectionNode` ()
- `ParameterType` `getType` ()
- void `setType` (`ParameterType` type)
- `Axis` `getAxis` ()
- void `setAxis` (`Axis` axis)
- `FeaturePtr` `getFeature` ()
- void `setFeature` (`FeaturePtr` feature)
- `DirectionSelectionType` `getDirectionSelectionType` ()
- void `setDirectionSelectionType` (`DirectionSelectionType` selection)
- `ExpressionPtr` `getExpression` ()
- void `setExpression` (`ExpressionPtr` expression)
- `FeatureSelectionType` `getFeatureSelectionType` ()
- void `setFeatureSelectionType` (`FeatureSelectionType` selection)
- double `getToolSpeed` ()
- void `setToolSpeed` (double speed)
- double `getToolAcceleration` ()
- void `setToolAcceleration` (double acc)

Public Member Functions inherited from arcs::aubo_scope::ProgramNode

- `ProgramNode` (`ProgramNode &f`)
- `ProgramNode` (`ProgramNode &&f`)
- virtual `~ProgramNode` ()
- `ProgramNodeContributionPtr` `getProgramNodeContribution` ()
Get the program node contribution.

Private Member Functions

- [DirectionNode](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Protected Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.24.1 Detailed Description

Definition at line 12 of file [direction_node.h](#).

11.24.2 Member Enumeration Documentation

11.24.2.1 Axis

enum [arcs::aubo_scope::DirectionNode::Axis](#) : int

Enumerator

X_PLUS	Move in the positive X direction relative to the feature.
X_MINUS	Move in the negative X direction relative to the feature.
Y_PLUS	Move in the positive Y direction relative to the feature.
Y_MINUS	Move in the negative Y direction relative to the feature.
Z_PLUS	Move in the positive Z direction relative to the feature.
Z_MINUS	Move in the negative Z direction relative to the feature.

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

11.24.2.2 DirectionSelectionType

enum [arcs::aubo_scope::DirectionNode::DirectionSelectionType](#) : int

Enumerator

EXPRESSION	A direction vector is specified using an Expression . The Selection instance can be cast to DirectionVectorExpression
AXIS	Direction is specified using an Axis . The Selection instance can be cast to DirectionAxis

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

11.24.2.3 FeatureSelectionType

```
enum arcs::aubo\_scope::DirectionNode::FeatureSelectionType : int
```

Enumerator

UNRESOLVED_FEATURE	The selected feature is no longer available (e.g. because the user has removed the feature in the Installation or the needed Installation is not currently loaded)
SHARED	Feature is inherited from the MoveL or MoveP parent.
FEATURE	Feature is selected. The Selection instance can be cast to DirectionNodeFeature

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

11.24.2.4 ParameterType

```
enum arcs::aubo\_scope::DirectionNode::ParameterType : int
```

Enumerator

SHARED_MOTION	The tool motion parameters are inherited from the MoveL or MoveP parent.
TOOL_MOTION	The tool motion parameters are selected. The Selection instance can be cast to ToolMotionParameters.

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

11.24.3 Constructor & Destructor Documentation

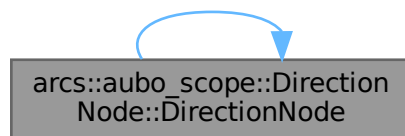
11.24.3.1 DirectionNode() [1/3]

arcs::aubo_scope::DirectionNode::DirectionNode (
DirectionNode & f)

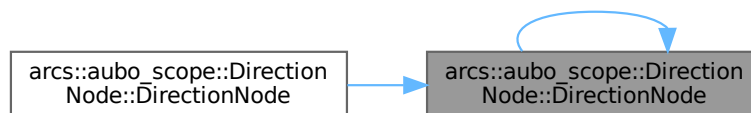
References [DirectionNode\(\)](#).

Referenced by [DirectionNode\(\)](#), and [DirectionNode\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

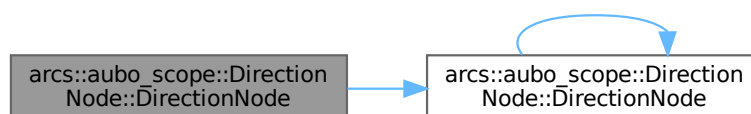


11.24.3.2 DirectionNode() [2/3]

arcs::aubo_scope::DirectionNode::DirectionNode (
DirectionNode && f)

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

Here is the call graph for this function:



11.24.3.3 ~DirectionNode()

virtual arcs::aubo_scope::DirectionNode::~~DirectionNode () [virtual]

11.24.3.4 DirectionNode() [3/3]

arcs::aubo_scope::DirectionNode::DirectionNode () [private]

11.24.4 Member Function Documentation

11.24.4.1 getAxis()

[Axis](#) arcs::aubo_scope::DirectionNode::getAxis ()

Returns

the [Axis](#) used to define the direction.

11.24.4.2 getDirectionSelectionType()

[DirectionSelectionType](#) arcs::aubo_scope::DirectionNode::getDirectionSelectionType ()

11.24.4.3 getExpression()

ExpressionPtr arcs::aubo_scope::DirectionNode::getExpression ()

Returns

the expression used to define the direction vector.

11.24.4.4 getFeature()

FeaturePtr arcs::aubo_scope::DirectionNode::getFeature ()

Returns

the feature selected for this Direction node.

11.24.4.5 getFeatureSelectionType()

[FeatureSelectionType](#) arcs::aubo_scope::DirectionNode::getFeatureSelectionType ()

11.24.4.6 getToolAcceleration()

double arcs::aubo_scope::DirectionNode::getToolAcceleration ()

Returns

the tool acceleration to use for the movement

11.24.4.7 getToolSpeed()

double arcs::aubo_scope::DirectionNode::getToolSpeed ()

Returns

the tool speed to be achieved.

11.24.4.8 getType()

[ParameterType](#) arcs::aubo_scope::DirectionNode::getType ()

11.24.4.9 setAxis()

void arcs::aubo_scope::DirectionNode::setAxis (
 [Axis](#) axis)

11.24.4.10 setDirectionSelectionType()

void arcs::aubo_scope::DirectionNode::setDirectionSelectionType (
 [DirectionSelectionType](#) selection)

11.24.4.11 setExpression()

void arcs::aubo_scope::DirectionNode::setExpression (
 ExpressionPtr expression)

11.24.4.12 setFeature()

void arcs::aubo_scope::DirectionNode::setFeature (
 FeaturePtr feature)

11.24.4.13 setFeatureSelectionType()

void arcs::aubo_scope::DirectionNode::setFeatureSelectionType (
 [FeatureSelectionType](#) selection)

11.24.4.14 setToolAcceleration()

```
void arcs::aubo_scope::DirectionNode::setToolAcceleration (  
    double acc)
```

11.24.4.15 setToolSpeed()

```
void arcs::aubo_scope::DirectionNode::setToolSpeed (  
    double speed)
```

11.24.4.16 setType()

```
void arcs::aubo_scope::DirectionNode::setType (  
    ParameterType type)
```

11.24.5 Friends And Related Symbol Documentation

11.24.5.1 DataSwitch

```
friend class DataSwitch [friend]
```

Definition at line [147](#) of file [direction_node.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.24.6 Member Data Documentation

11.24.6.1 d__

```
void* arcs::aubo_scope::DirectionNode::d__ { nullptr } [private]
```

Definition at line [149](#) of file [direction_node.h](#).

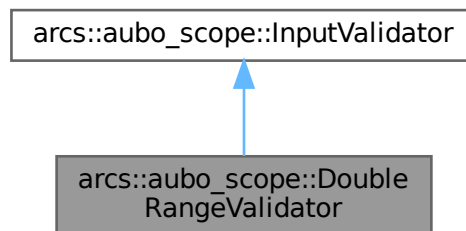
The documentation for this class was generated from the following file:

- [include/aubo_caps/domain/program/nodes/builtin/direction_node.h](#)

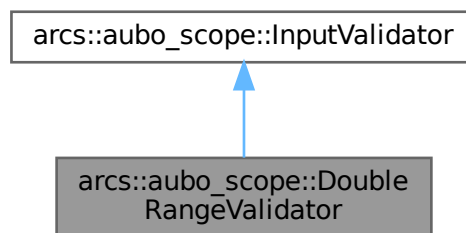
11.25 arcs::aubo_scope::DoubleRangeValidator Struct Reference

#include <input_validator.h>

Inheritance diagram for arcs::aubo_scope::DoubleRangeValidator:



Collaboration diagram for arcs::aubo_scope::DoubleRangeValidator:



Public Member Functions

- [DoubleRangeValidator](#) (double minValue, double maxValue, bool is_min_inclusive=true, bool is_max_inclusive=true)
- [DoubleRangeValidator](#) (std::function< std::pair< double, double >()> range_func)
- [DoubleRangeValidator](#) (std::function< std::tuple< double, double, bool, bool >()> range_func)
- bool [isValid](#) (std::string value) override
Validates whether the input value is valid.
- std::string [getMessage](#) (std::string value) override
Returns a meaningful message in case the value is not valid.

Private Member Functions

- void [updateValue](#) ()

Private Attributes

- double [min_value_](#) { DBL_MIN }
- double [max_value_](#) { DBL_MAX }
- std::function< std::pair< double, double >()> [range_func_](#) { nullptr }
- std::function< std::tuple< double, double, bool, bool >()> [range_func_with_flags_](#) { nullptr }
- bool [is_min_inclusive_](#) { true }
- bool [is_max_inclusive_](#) { true }

11.25.1 Detailed Description

Definition at line 118 of file [input_validator.h](#).

11.25.2 Constructor & Destructor Documentation

11.25.2.1 DoubleRangeValidator() [1/3]

```
arcs::aubo_scope::DoubleRangeValidator::DoubleRangeValidator (
    double minValue,
    double maxValue,
    bool is_min_inclusive = true,
    bool is_max_inclusive = true)
```

11.25.2.2 DoubleRangeValidator() [2/3]

```
arcs::aubo_scope::DoubleRangeValidator::DoubleRangeValidator (
    std::function< std::pair< double, double >()> range_func)
```

11.25.2.3 DoubleRangeValidator() [3/3]

```
arcs::aubo_scope::DoubleRangeValidator::DoubleRangeValidator (
    std::function< std::tuple< double, double, bool, bool >()> range_func)
```

11.25.3 Member Function Documentation

11.25.3.1 getMessage()

```
std::string arcs::aubo_scope::DoubleRangeValidator::getMessage (
    std::string value) [override], [virtual]
```

Returns a meaningful message in case the value is not valid.

Parameters

value	the invalid value. Can be included in the message if it makes sense.

Returns

message.

Implements [arcs::aubo_scope::InputValidator](#).

11.25.3.2 isValid()

```
bool arcs::aubo_scope::DoubleRangeValidator::isValid (
    std::string value)  [override], [virtual]
```

Validates whether the input value is valid.

Parameters

value	to be validated.

Returns

true if value is valid.

Implements [arcs::aubo_scope::InputValidator](#).

11.25.3.3 updateValue()

```
void arcs::aubo_scope::DoubleRangeValidator::updateValue ()  [private]
```

11.25.4 Member Data Documentation

11.25.4.1 is_max_inclusive__

```
bool arcs::aubo_scope::DoubleRangeValidator::is_max_inclusive__ { true }  [private]
```

Definition at line 143 of file [input_validator.h](#).

11.25.4.2 is_min_inclusive__

```
bool arcs::aubo_scope::DoubleRangeValidator::is_min_inclusive__ { true }  [private]
```

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

11.25.4.3 max_value__

double arcs::aubo_scope::DoubleRangeValidator::max_value__ { DBL_MAX } [private]

Definition at line 138 of file [input_validator.h](#).

11.25.4.4 min_value__

double arcs::aubo_scope::DoubleRangeValidator::min_value__ { DBL_MIN } [private]

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

11.25.4.5 range_func__

std::function<std::pair<double, double>()> arcs::aubo_scope::DoubleRangeValidator::range_func__ { nullptr } [private]

Definition at line 139 of file [input_validator.h](#).

11.25.4.6 range_func_with_flags__

std::function<std::tuple<double, double, bool, bool>()> arcs::aubo_scope::DoubleRangeValidator::range_func_with_flags__ { nullptr } [private]

Definition at line 141 of file [input_validator.h](#).

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

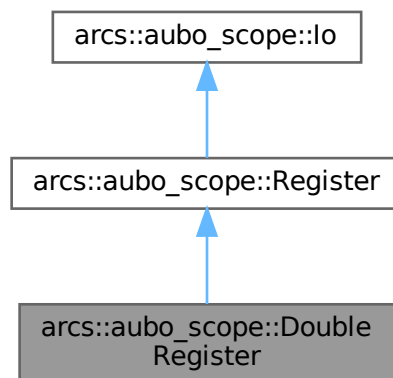
- [include/aubo_caps/domain/user_interface/input_validator.h](#)

11.26 arcs::aubo_scope::DoubleRegister Class Reference

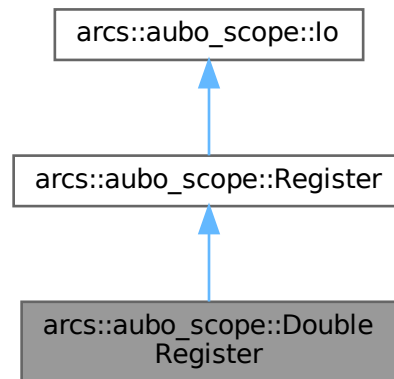
This interface provides support for float registers.

```
#include <double_register.h>
```

Inheritance diagram for arcs::aubo_scope::DoubleRegister:



Collaboration diagram for arcs::aubo_scope::DoubleRegister:



Public Member Functions

- virtual [~DoubleRegister](#) ()=default

Public Member Functions inherited from [arcs::aubo_scope::Register](#)

- virtual [~Register](#) ()=default

Public Member Functions inherited from [arcs::aubo_scope::Io](#)

- [Io](#) (Io &f)
- [Io](#) (Io &&f)
- virtual [~Io](#) ()
- std::string [getDisplayname](#) () const
Returns the user defined name of the IO in the currently selected installation.
- std::string [getDefaultName](#) () const
Returns the robot default name of the I/O.
- std::string [getValueStr](#) () const
Returns the current reading of the I/O as a string.
- [IoType](#) [getType](#) () const
Returns the I/O type.
- [InterfaceType](#) [getInterfaceType](#) () const
Returns the interface type this I/O belongs to.
- int [getAction](#) () const
Returns the action number of the I/O.
- bool [isInput](#) () const
Returns true if I/O supports reading of values and false otherwise.
- bool [isResolvable](#) () const
Some I/Os, such as MODBUS I/Os and analog Tool inputs, are in some situations not present in AuboScope or available for use.
- size_t [hashCode](#) ()

Additional Inherited Members

Public Types inherited from [arcs::aubo_scope::Io](#)

- enum [IoType](#) : int {
[DIGITAL](#) , [ANALOG](#) , [INTEGER](#) , [BOOLEAN](#) ,
[DOUBLE](#) }
- enum [InterfaceType](#) : int {
[STANDARD](#) , [TOOL](#) , [CONFIGURABLE](#) , [LINKIO](#) ,
[MODBUS](#) , [GENERAL_PURPOSE](#) , [PROFINET](#) , [ETHERNET_IP](#) }

Protected Member Functions inherited from [arcs::aubo_scope::Io](#)

- [Io](#) ()

11.26.1 Detailed Description

This interface provides support for float registers.

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

11.26.2 Constructor & Destructor Documentation

11.26.2.1 ~DoubleRegister()

virtual [arcs::aubo_scope::DoubleRegister::~~DoubleRegister](#) () [virtual], [default]

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

- [include/aubo_caps/domain/io/double_register.h](#)

11.27 arcs::aubo_scope::DriveScrewNotOKParameters Class Reference

```
#include <screw_driver_configuration.h>
```

Public Member Functions

- virtual [~DriveScrewNotOKParameters](#) ()=default

11.27.1 Detailed Description

This interface provides parameters relevant for generating script code for the Drive Screw Not OK feedback capability (registered using `ScrewdriverFeedbackCapabilities#registerDriveScrewNotOKCapability(ScriptCodeGenerator)`).

These parameters are passed when AuboScope calls the implementation of the `ScriptCodeGenerator#generateScript(ScriptCodeGenerator)` method responsible for the generating the script code.

The interface is intentionally empty, but could have parameters in future releases.

Definition at line 107 of file [screw_driver_configuration.h](#).

11.27.2 Constructor & Destructor Documentation

11.27.2.1 ~DriveScrewNotOKParameters()

virtual arcs::aubo_scope::DriveScrewNotOKParameters::~~DriveScrewNotOKParameters () [virtual], [default]

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

- include/aubo_caps/contribution/driver/screw_driver/[screw_driver_configuration.h](#)

11.28 arcs::aubo_scope::DriveScrewOKParameters Class Reference

```
#include <screw_driver_configuration.h>
```

Public Member Functions

- virtual [~DriveScrewOKParameters](#) ()=default

11.28.1 Detailed Description

This interface provides parameters relevant for generating script code for the Drive Screw OK feedback capability (registered using `ScrewdriverFeedbackCapabilities#registerDriveScrewOKCapability(ScriptCodeGenerator)`).

These parameters are passed when AuboScope calls the implementation of the `ScriptCodeGenerator#generateScript(ScriptCodeGenerator)` method responsible for the generating the script code.

The interface is intentionally empty, but could have parameters in future releases.

Definition at line 130 of file [screw_driver_configuration.h](#).

11.28.2 Constructor & Destructor Documentation

11.28.2.1 ~DriveScrewOKParameters()

virtual arcs::aubo_scope::DriveScrewOKParameters::~~DriveScrewOKParameters () [virtual], [default]

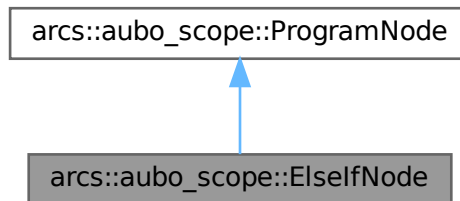
The documentation for this class was generated from the following file:

- include/aubo_caps/contribution/driver/screw_driver/[screw_driver_configuration.h](#)

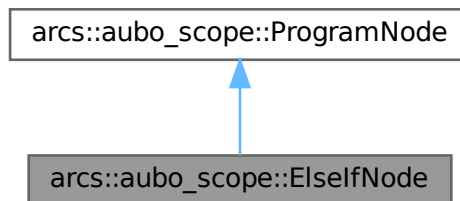
11.29 arcs::aubo_scope::ElseIfNode Class Reference

#include <else_if_node.h>

Inheritance diagram for arcs::aubo_scope::ElseIfNode:



Collaboration diagram for arcs::aubo_scope::ElseIfNode:



Public Member Functions

- [ElseIfNode](#) (ElseIfNode &f)
- [ElseIfNode](#) (ElseIfNode &&f)
- virtual [~ElseIfNode](#) ()
- void [setExpression](#) (ExpressionPtr expression)
- ExpressionPtr [getExpression](#) ()

Public Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()
Get the program node contribution.

Private Member Functions

- [ElseIfNode](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Protected Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.29.1 Detailed Description

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

11.29.2 Constructor & Destructor Documentation

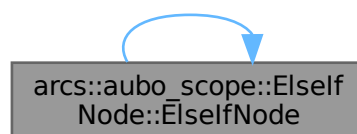
11.29.2.1 ElseIfNode() [1/3]

```
arcs::aubo_scope::ElseIfNode::ElseIfNode (  
    ElseIfNode & f)
```

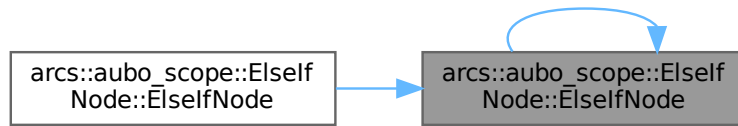
References [ElseIfNode\(\)](#).

Referenced by [ElseIfNode\(\)](#), and [ElseIfNode\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

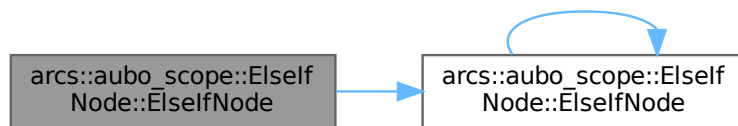


11.29.2.2 ElseIfNode() [2/3]

```
arcs::aubo_scope::ElseIfNode::ElseIfNode (
    ElseIfNode && f)
```

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

Here is the call graph for this function:



11.29.2.3 ~ElseIfNode()

```
virtual arcs::aubo_scope::ElseIfNode::~~ElseIfNode () [virtual]
```

11.29.2.4 ElseIfNode() [3/3]

```
arcs::aubo_scope::ElseIfNode::ElseIfNode () [private]
```

11.29.3 Member Function Documentation

11.29.3.1 getExpression()

```
ExpressionPtr arcs::aubo_scope::ElseIfNode::getExpression ()
```

Returns

the [Expression](#) to be evaluated in this ElseIf node.

Parameters

expression	The expression that will be evaluated in this ElseIfNode .

Returns

The same [ElseIfNode](#).

11.29.4 Friends And Related Symbol Documentation

11.29.4.1 DataSwitch

friend class DataSwitch [friend]

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

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.29.5 Member Data Documentation

11.29.5.1 d_

void* arcs::aubo_scope::ElseIfNode::d_ { nullptr } [private]

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

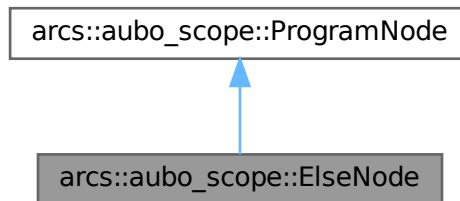
The documentation for this class was generated from the following file:

- [include/aubo_caps/domain/program/nodes/builtin/else_if_node.h](#)

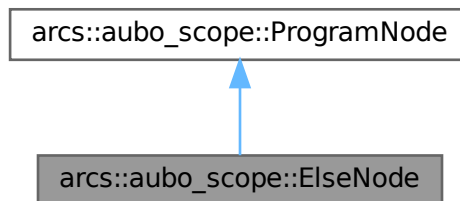
11.30 arcs::aubo_scope::ElseNode Class Reference

#include <else_node.h>

Inheritance diagram for arcs::aubo_scope::ElseNode:



Collaboration diagram for arcs::aubo_scope::ElseNode:



Public Member Functions

- [ElseNode](#) (ElseNode &f)
- [ElseNode](#) (ElseNode &&f)
- virtual [~ElseNode](#) ()

Public Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()
Get the program node contribution.

Private Member Functions

- [ElseNode](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Protected Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.30.1 Detailed Description

Definition at line 10 of file [else_node.h](#).

11.30.2 Constructor & Destructor Documentation

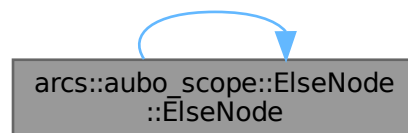
11.30.2.1 ElseNode() [1/3]

```
arcs::aubo_scope::ElseNode::ElseNode (  
    ElseNode & f)
```

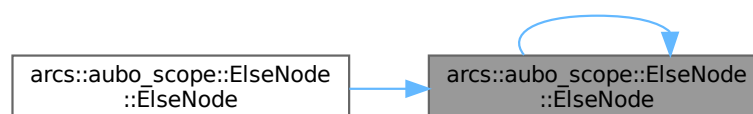
References [ElseNode\(\)](#).

Referenced by [ElseNode\(\)](#), and [ElseNode\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

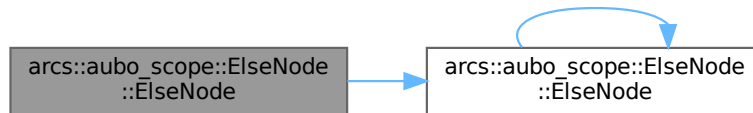


11.30.2.2 ElseNode() [2/3]

```
arcs::aubo_scope::ElseNode::ElseNode (
    ElseNode && f)
```

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

Here is the call graph for this function:



11.30.2.3 ~ElseNode()

```
virtual arcs::aubo_scope::ElseNode::~~ElseNode () [virtual]
```

11.30.2.4 ElseNode() [3/3]

```
arcs::aubo_scope::ElseNode::ElseNode () [private]
```

11.30.3 Friends And Related Symbol Documentation

11.30.3.1 DataSwitch

```
friend class DataSwitch [friend]
```

Definition at line 18 of file [else_node.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.30.4 Member Data Documentation

11.30.4.1 d__

```
void* arcs::aubo_scope::ElseNode::d__ { nullptr } [private]
```

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

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

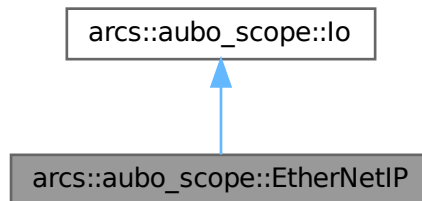
- [include/aubo_caps/domain/program/nodes/builtin/else_node.h](#)

11.31 arcs::aubo_scope::EtherNetIP Class Reference

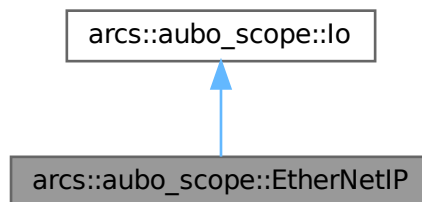
[EtherNetIP](#) This interface provides support for EtherNet/IP I/Os.

```
#include <ethernet_ip.h>
```

Inheritance diagram for arcs::aubo_scope::EtherNetIP:



Collaboration diagram for arcs::aubo_scope::EtherNetIP:



Public Member Functions

- [EtherNetIP](#) (EtherNetIP &f)
- [EtherNetIP](#) (EtherNetIP &&f)
- [~EtherNetIP](#) ()
- int [getByteOffset](#) ()
Returns the byte offset for this EtherNet/IP I/O.
- int [getBitOffset](#) ()
Returns the bit offset for this EtherNet/IP I/O.
- bool [isSlave](#) ()
Returns whether this EtherNet/IP I/O is a slave.
- bool [setValue](#) (int value)
Set EtherNet/IP signal to an integer value.
- int [getValue](#) ()
Returns the integer value of the EtherNet/IP signal.

Public Member Functions inherited from [arcs::aubo_scope::Io](#)

- [Io](#) (Io &f)
- [Io](#) (Io &&f)
- virtual [~Io](#) ()
- std::string [getDisplayName](#) () const
Returns the user defined name of the IO in the currently selected installation.
- std::string [getDefaultName](#) () const
Returns the robot default name of the I/O.
- std::string [getValueStr](#) () const
Returns the current reading of the I/O as a string.
- [IoType](#) [getType](#) () const
Returns the I/O type.
- [InterfaceType](#) [getInterfaceType](#) () const
Returns the interface type this I/O belongs to.
- int [getAction](#) () const
Returns the action number of the I/O.
- bool [isInput](#) () const
Returns true if I/O supports reading of values and false otherwise.
- bool [isResolvable](#) () const
Some I/Os, such as MODBUS I/Os and analog Tool inputs, are in some situations not present in AuboScope or available for use.
- size_t [hashCode](#) ()

Private Member Functions

- [EtherNetIP](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Public Types inherited from [arcs::aubo_scope::Io](#)

- enum [IoType](#) : int {
 [DIGITAL](#) , [ANALOG](#) , [INTEGER](#) , [BOOLEAN](#) ,
 [DOUBLE](#) }
- enum [InterfaceType](#) : int {
 [STANDARD](#) , [TOOL](#) , [CONFIGURABLE](#) , [LINKIO](#) ,
 [MODBUS](#) , [GENERAL_PURPOSE](#) , [PROFINET](#) , [ETHERNET_IP](#) }

Protected Member Functions inherited from [arcs::aubo_scope::Io](#)

- [Io](#) ()

11.31.1 Detailed Description

[EtherNetIP](#) This interface provides support for EtherNet/IP I/Os.

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

11.31.2 Constructor & Destructor Documentation

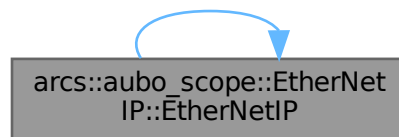
11.31.2.1 EtherNetIP() [1/3]

```
arcs::aubo_scope::EtherNetIP::EtherNetIP (  
    EtherNetIP & f)
```

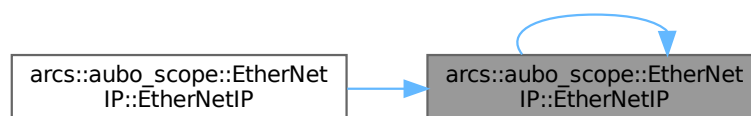
References [EtherNetIP\(\)](#).

Referenced by [EtherNetIP\(\)](#), and [EtherNetIP\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

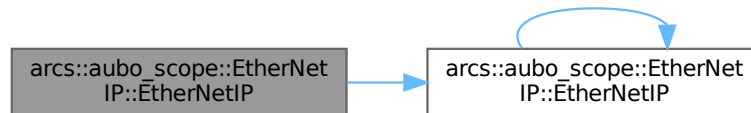


11.31.2.2 EtherNetIP() [2/3]

```
arcs::aubo_scope::EtherNetIP::EtherNetIP (
    EtherNetIP && f)
```

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

Here is the call graph for this function:



11.31.2.3 ~EtherNetIP()

```
arcs::aubo_scope::EtherNetIP::~~EtherNetIP ()
```

11.31.2.4 EtherNetIP() [3/3]

```
arcs::aubo_scope::EtherNetIP::EtherNetIP () [private]
```

11.31.3 Member Function Documentation

11.31.3.1 getBitOffset()

```
int arcs::aubo_scope::EtherNetIP::getBitOffset ()
```

Returns the bit offset for this EtherNet/IP I/O.

Returns

Get the bit offset for this EtherNet/IP I/O.

11.31.3.2 getByteOffset()

```
int arcs::aubo_scope::EtherNetIP::getByteOffset ()
```

Returns the byte offset for this EtherNet/IP I/O.

Returns

Get the byte offset for this EtherNet/IP I/O.

11.31.3.3 getValue()

int arcs::aubo_scope::EtherNetIP::getValue ()

Returns the integer value of the EtherNet/IP signal.

Returns

the integer value of the EtherNet/IP signal.

11.31.3.4 isSlave()

bool arcs::aubo_scope::EtherNetIP::isSlave ()

Returns whether this EtherNet/IP I/O is a slave.

Returns

Get the is slave for this EtherNet/IP I/O.

11.31.3.5 setValue()

bool arcs::aubo_scope::EtherNetIP::setValue (
int value)

Set EtherNet/IP signal to an integer value.

Parameters

value	Set EtherNet/IP signal to an integer value.

Returns

true if the EtherNet/IP signal was sent down to the controller, and false if the signal was not sent down.

11.31.4 Friends And Related Symbol Documentation

11.31.4.1 DataSwitch

friend class DataSwitch [friend]

Definition at line 91 of file [ethernet_ip.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.31.5 Member Data Documentation

11.31.5.1 d_

```
void* arcs::aubo_scope::EtherNetIP::d_ { nullptr } [private]
```

Definition at line 93 of file [ethernet_ip.h](#).

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

- [include/aubo_caps/domain/io/ethernet_ip.h](#)

11.32 arcs::aubo_scope::EtherNetIPFilter Struct Reference

```
#include <ethernet_ip.h>
```

Public Member Functions

- [EtherNetIPFilter](#) ([Io::IoType](#) type, bool is_input)
- bool [operator\(\)](#) (IoPtr io)

Private Attributes

- [Io::IoType](#) type_
- bool [is_input_](#)

11.32.1 Detailed Description

Definition at line 96 of file [ethernet_ip.h](#).

11.32.2 Constructor & Destructor Documentation

11.32.2.1 EtherNetIPFilter()

```
arcs::aubo_scope::EtherNetIPFilter::EtherNetIPFilter (  
    Io::IoType type,  
    bool is_input) [inline]
```

Definition at line 98 of file [ethernet_ip.h](#).

References [is_input_](#), and [type_](#).

11.32.3 Member Function Documentation

11.32.3.1 operator>()()

bool arcs::aubo_scope::EtherNetIPFilter::operator() (
IoPtr io) [inline]

Definition at line 102 of file [ethernet_ip.h](#).

References [is_input_](#), and [type_](#).

11.32.4 Member Data Documentation

11.32.4.1 is_input_

bool arcs::aubo_scope::EtherNetIPFilter::is_input_ [private]

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

Referenced by [EtherNetIPFilter\(\)](#), and [operator>\(\)\(\)](#).

11.32.4.2 type_

[Io::IoType](#) arcs::aubo_scope::EtherNetIPFilter::type_ [private]

Definition at line 109 of file [ethernet_ip.h](#).

Referenced by [EtherNetIPFilter\(\)](#), and [operator>\(\)\(\)](#).

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

- include/aubo_caps/domain/io/[ethernet_ip.h](#)

11.33 arcs::aubo_scope::Expression Class Reference

[Expression](#) An expression can be used for, e.g.

```
#include <expression.h>
```

Public Member Functions

- [Expression](#) (Expression &f)
- [Expression](#) (Expression &&f)
- virtual [~Expression](#) ()
- std::string [toString](#) ()
- void [append](#) (const std::string &expressionPart)
Append a string part of the expression.
- void [appendVariable](#) (VariablePtr variable)
Append a variable object to the expression.
- void [appendFeature](#) (FeaturePtr feature)
Append a [Feature](#) object to the expression.
- void [appendWaypoint](#) (WaypointPtr waypoint)
Append a waypoint to the expression.
- void [appendIo](#) (IoPtr io)
Append an I/O object to the expression.

Private Member Functions

- [Expression](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

11.33.1 Detailed Description

[Expression](#) An expression can be used for, e.g.

configuration of various built-in program nodes, such as If and Wait nodes. Expressions are typically conditional expressions (which evaluates to true or false).

An [Expression](#) is built using the ExpressionBuilder interface.

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

11.33.2 Constructor & Destructor Documentation

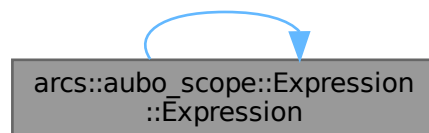
11.33.2.1 Expression() [1/3]

```
arcs::aubo_scope::Expression::Expression (
    Expression & f)
```

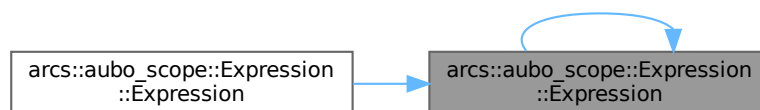
References [Expression\(\)](#).

Referenced by [Expression\(\)](#), and [Expression\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

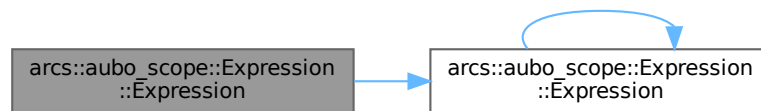


11.33.2.2 Expression() [2/3]

```
arcs::aubo_scope::Expression::Expression (
    Expression && f)
```

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

Here is the call graph for this function:



11.33.2.3 ~Expression()

```
virtual arcs::aubo_scope::Expression::~~Expression () [virtual]
```

11.33.2.4 Expression() [3/3]

```
arcs::aubo_scope::Expression::Expression () [private]
```

11.33.3 Member Function Documentation

11.33.3.1 append()

```
void arcs::aubo_scope::Expression::append (
    const std::string & expressionPart)
```

Append a string part of the expression.

Parameters

expressionPart	the expression part to be appended.

11.33.3.2 appendFeature()

```
void arcs::aubo_scope::Expression::appendFeature (
    FeaturePtr feature)
```

Generated by Doxygen
Append a [Feature](#) object to the expression.

Parameters

feature	the Feature .

11.33.3.3 `appendIo()`

```
void arcs::aubo_scope::Expression::appendIo (  
    IoPtr io)
```

Append an I/O object to the expression.

Parameters

io	the I/O.

11.33.3.4 `appendVariable()`

```
void arcs::aubo_scope::Expression::appendVariable (  
    VariablePtr variable)
```

Append a variable object to the expression.

Parameters

variable	the variable.

11.33.3.5 `appendWaypoint()`

```
void arcs::aubo_scope::Expression::appendWaypoint (  
    WaypointPtr waypoint)
```

Append a waypoint to the expression.

Parameters

waypoint	the waypoint.

11.33.3.6 toString()

```
std::string arcs::aubo_scope::Expression::toString ()
```

11.33.4 Friends And Related Symbol Documentation

11.33.4.1 DataSwitch

```
friend class DataSwitch [friend]
```

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

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.33.5 Member Data Documentation

11.33.5.1 d_

```
void* arcs::aubo_scope::Expression::d_ { nullptr } [private]
```

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

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

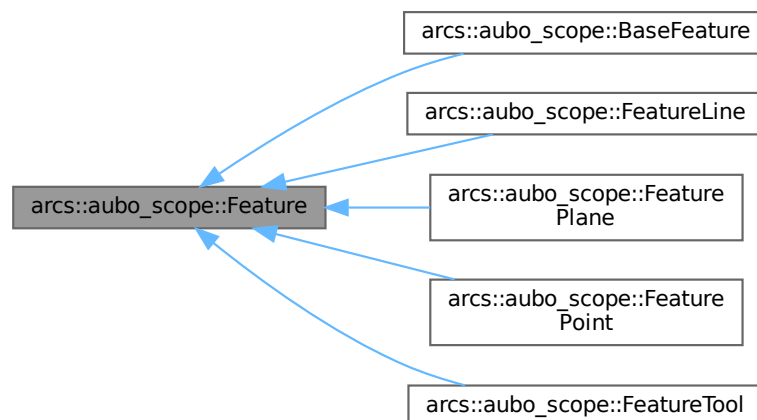
- [include/aubo_caps/domain/value/expression/expression.h](#)

11.34 arcs::aubo_scope::Feature Class Reference

Feature This base interface represents all types of features available in AuboScope.

```
#include <feature.h>
```

Inheritance diagram for arcs::aubo_scope::Feature:



Public Member Functions

- [Feature](#) (Feature &f)
- [Feature](#) (Feature &&f)
- virtual [~Feature](#) ()
- std::string [getDisplay_name](#) ()
Note: The feature can be renamed at any time.
- bool [isDefined](#) ()
Returns whether the position of the feature has been fully defined/specified.
- bool [isVariable](#) ()
Returns whether the feature can be used as a variable in the program.
- bool [isJoggable](#) ()
- bool [isResolvable](#) ()
A feature cannot be guaranteed to be present in AuboScope.
- std::vector< double > [getPose](#) ()
Returns the Cartesian location of the feature.
- size_t [hashCode](#) ()

Protected Member Functions

- [Feature](#) ()

Private Attributes

- void * [d__](#) { nullptr }

Friends

- class [DataSwitch](#)

11.34.1 Detailed Description

[Feature](#) This base interface represents all types of features available in AuboScope.

Definition at line 22 of file [feature.h](#).

11.34.2 Constructor & Destructor Documentation

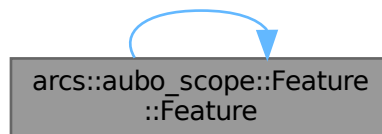
11.34.2.1 Feature() [1/3]

arcs::aubo_scope::Feature::Feature (
Feature & f)

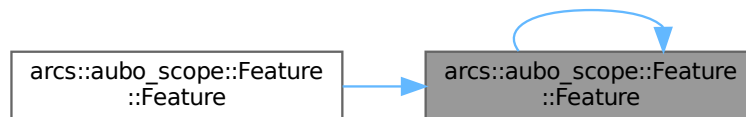
References [Feature\(\)](#).

Referenced by [Feature\(\)](#), and [Feature\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

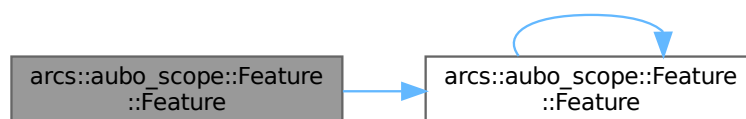


11.34.2.2 Feature() [2/3]

arcs::aubo_scope::Feature::Feature (
Feature && f)

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

Here is the call graph for this function:



11.34.2.3 ~Feature()

`virtual arcs::aubo_scope::Feature::~~Feature () [virtual]`

11.34.2.4 Feature() [3/3]

`arcs::aubo_scope::Feature::Feature () [protected]`

11.34.3 Member Function Documentation

11.34.3.1 getDisplayName()

`std::string arcs::aubo_scope::Feature::getDisplayName ()`

Note: The feature can be renamed at any time.

Returns

The name of the feature.

11.34.3.2 getPose()

`std::vector< double > arcs::aubo_scope::Feature::getPose ()`

Returns the Cartesian location of the feature.

Returns

A pose representing the Cartesian location of the feature.

11.34.3.3 hashCode()

`size_t arcs::aubo_scope::Feature::hashCode ()`

11.34.3.4 isDefined()

`bool arcs::aubo_scope::Feature::isDefined ()`

Returns whether the position of the feature has been fully defined/specified.

Returns

true if the feature is defined, false otherwise.

11.34.3.5 isJoggable()

bool arcs::aubo_scope::Feature::isJoggable ()

11.34.3.6 isResolvable()

bool arcs::aubo_scope::Feature::isResolvable ()

A feature cannot be guaranteed to be present in AuboScope.

The end user can load a different installation (not containing the feature) or the feature can be removed by the end user or the AuboCap that added the feature. This method can be used to determine, if the feature is present.

Returns

true if this feature is present in AuboScope, otherwise false.

11.34.3.7 isVariable()

bool arcs::aubo_scope::Feature::isVariable ()

Returns whether the feature can be used as a variable in the program.

Returns

true if the feature is variable, false otherwise.

11.34.4 Friends And Related Symbol Documentation

11.34.4.1 DataSwitch

friend class DataSwitch [friend]

Definition at line 108 of file [feature.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.34.5 Member Data Documentation

11.34.5.1 d_

void* arcs::aubo_scope::Feature::d_ { nullptr } [private]

Definition at line 109 of file [feature.h](#).

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

- include/aubo_caps/domain/feature/[feature.h](#)

11.35 arcs::aubo_scope::FeatureContributionModel Class Reference

[FeatureContributionModel](#) This interface provides functionality for adding, updating and removing features in AuboScope.

```
#include <feature_contribution_model.h>
```

Public Member Functions

- [FeatureContributionModel](#) (FeatureContributionModel &f)
- [FeatureContributionModel](#) (FeatureContributionModel &&f)
- [~FeatureContributionModel](#) ()
- FeaturePtr [addFeature](#) (const std::string &idKey, const std::string &suggestedName, const std::vector< double > &pose)
Add a feature to the current AuboScope installation.
- FeaturePtr [getFeature](#) (const std::string &idKey)
Returns the feature previously added by this AuboCap using the same idKey identifier.
- void [updateFeature](#) (const std::string &idKey, const std::vector< double > &newPose)
Update the pose of an existing feature added by this AuboCap.
- void [renameFeature](#) (const std::string &idKey, const std::string &newName)
- void [removeFeature](#) (const std::string &idKey)
Remove a feature added by this AuboCap from AuboScope.

Private Member Functions

- [FeatureContributionModel](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

11.35.1 Detailed Description

[FeatureContributionModel](#) This interface provides functionality for adding, updating and removing features in AuboScope.

Note: This interface must only be used in an installation contribution.

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

11.35.2 Constructor & Destructor Documentation

11.35.2.1 FeatureContributionModel() [1/3]

```
arcs::aubo_scope::FeatureContributionModel::FeatureContributionModel (
    FeatureContributionModel & f)
```

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

Referenced by [FeatureContributionModel\(\)](#), and [FeatureContributionModel\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



11.35.2.2 FeatureContributionModel() [2/3]

```
arcs::aubo_scope::FeatureContributionModel::FeatureContributionModel (
    FeatureContributionModel && f)
```

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

Here is the call graph for this function:



11.35.2.3 ~FeatureContributionModel()

```
arcs::aubo_scope::FeatureContributionModel::~~FeatureContributionModel ()
```

11.35.2.4 FeatureContributionModel() [3/3]

```
arcs::aubo_scope::FeatureContributionModel::FeatureContributionModel () [private]
```

11.35.3 Member Function Documentation

11.35.3.1 addFeature()

```
FeaturePtr arcs::aubo_scope::FeatureContributionModel::addFeature (
    const std::string & idKey,
    const std::string & suggestedName,
    const std::vector< double > & pose)
```

Add a feature to the current AuboScope installation.

This makes it selectable by the end user. The feature is not modifiable by the end user.

Parameters

idKey	The key to identify this feature by.
suggestedName	Suggested name for the feature.
pose	The pose of the feature with respect to the robot base.

Returns

The feature created and registered in AuboScope.

Exceptions

FeatureAlreadyAddedException	If a feature has previously been added the same idKey identifier.
IllegalFeatureNameException	If the suggested name does not match required regex.

11.35.3.2 getFeature()

```
FeaturePtr arcs::aubo_scope::FeatureContributionModel::getFeature (
    const std::string & idKey)
```

Returns the feature previously added by this AuboCap using the same idKey identifier.

Use this to verify if the feature is present in the current installation.

Parameters

idKey	The key to identify this feature by.

Returns

The feature previously added by this AuboCap. Returns null if no feature exists in the current installation.

11.35.3.3 removeFeature()

```
void arcs::aubo_scope::FeatureContributionModel::removeFeature (
    const std::string & idKey)
```

Remove a feature added by this AuboCap from AuboScope.

Program nodes using the feature will be become undefined because the feature is no longer resolvable.

Parameters

idKey	The key used to add the feature with.

Exceptions

FeatureNotFoundException	If no feature exists with the provided idKey.

11.35.3.4 renameFeature()

```
void arcs::aubo_scope::FeatureContributionModel::renameFeature (
    const std::string & idKey,
    const std::string & newName)
```

11.35.3.5 updateFeature()

```
void arcs::aubo_scope::FeatureContributionModel::updateFeature (
    const std::string & idKey,
    const std::vector< double > & newPose)
```

Update the pose of an existing feature added by this AuboCap.

Parameters

idKey	The key to identify the feature.
newPose	The new pose to set for the feature.

Exceptions

FeatureNotFoundException	If no feature exists with the provided idKey.

11.35.4 Friends And Related Symbol Documentation

11.35.4.1 DataSwitch

friend class DataSwitch [friend]

Definition at line 123 of file [feature_contribution_model.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.35.5 Member Data Documentation

11.35.5.1 d_

void* arcs::aubo_scope::FeatureContributionModel::d_ { nullptr } [private]

Definition at line 125 of file [feature_contribution_model.h](#).

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

- [include/aubo_caps/domain/feature/feature_contribution_model.h](#)

11.36 arcs::aubo_scope::FeatureDefinedFilter Struct Reference

```
#include <feature.h>
```

Public Member Functions

- bool [operator\(\)](#) (FeaturePtr feature)

11.36.1 Detailed Description

Definition at line 112 of file [feature.h](#).

11.36.2 Member Function Documentation

11.36.2.1 operator()()

```
bool arcs::aubo_scope::FeatureDefinedFilter::operator() (  
    FeaturePtr feature)    [inline]
```

Definition at line 114 of file [feature.h](#).

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

- [include/aubo_caps/domain/feature/feature.h](#)

11.37 arcs::aubo_scope::FeatureJoggableFilter Struct Reference

```
#include <feature.h>
```

Public Member Functions

- bool [operator\(\)](#) (FeaturePtr feature)

11.37.1 Detailed Description

Definition at line 120 of file [feature.h](#).

11.37.2 Member Function Documentation

11.37.2.1 operator()()

```
bool arcs::aubo_scope::FeatureJoggableFilter::operator() (  
    FeaturePtr feature)    [inline]
```

Definition at line 122 of file [feature.h](#).

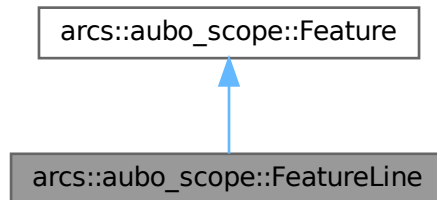
The documentation for this struct was generated from the following file:

- [include/aubo_caps/domain/feature/feature.h](#)

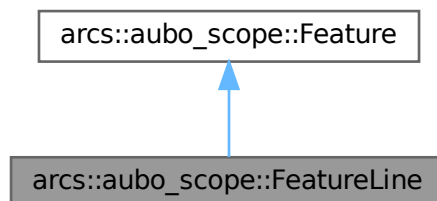
11.38 arcs::aubo_scope::FeatureLine Class Reference

```
#include <feature_line.h>
```

Inheritance diagram for arcs::aubo_scope::FeatureLine:



Collaboration diagram for arcs::aubo_scope::FeatureLine:



Public Member Functions

- [FeatureLine](#) (FeatureLine &f)
- [FeatureLine](#) (FeatureLine &&f)
- virtual [~FeatureLine](#) ()

Public Member Functions inherited from [arcs::aubo_scope::Feature](#)

- [Feature](#) (Feature &f)
- [Feature](#) (Feature &&f)
- virtual [~Feature](#) ()
- std::string [getDisplayname](#) ()

Note: The feature can be renamed at any time.

- bool [isDefined](#) ()

Returns whether the position of the feature has been fully defined/specified.

- bool [isVariable](#) ()

Returns whether the feature can be used as a variable in the program.

- bool [isJoggable](#) ()
- bool [isResolvable](#) ()

A feature cannot be guaranteed to be present in AuboScope.

- std::vector< double > [getPose](#) ()

Returns the Cartesian location of the feature.

- size_t [hashCode](#) ()

Private Member Functions

- [FeatureLine](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Protected Member Functions inherited from [arcs::aubo_scope::Feature](#)

- [Feature](#) ()

11.38.1 Detailed Description

Definition at line 12 of file [feature_line.h](#).

11.38.2 Constructor & Destructor Documentation

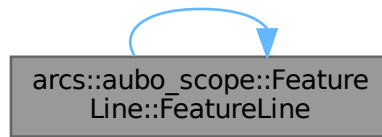
11.38.2.1 [FeatureLine](#)() [1/3]

```
arcs::aubo_scope::FeatureLine::FeatureLine (  
    FeatureLine & f)
```

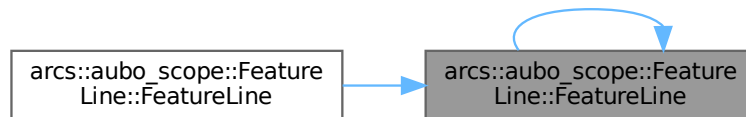
References [FeatureLine\(\)](#).

Referenced by [FeatureLine\(\)](#), and [FeatureLine\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

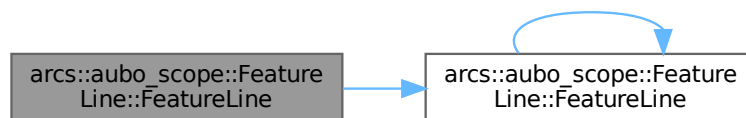


11.38.2.2 FeatureLine() [2/3]

```
arcs::aubo_scope::FeatureLine::FeatureLine (
    FeatureLine && f)
```

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

Here is the call graph for this function:



11.38.2.3 ~FeatureLine()

```
virtual arcs::aubo_scope::FeatureLine::~~FeatureLine () [virtual]
```

11.38.2.4 FeatureLine() [3/3]

arcs::aubo_scope::FeatureLine::FeatureLine () [private]

11.38.3 Friends And Related Symbol Documentation

11.38.3.1 DataSwitch

friend class DataSwitch [friend]

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

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.38.4 Member Data Documentation

11.38.4.1 d__

void* arcs::aubo_scope::FeatureLine::d__ { nullptr } [private]

Definition at line 22 of file [feature_line.h](#).

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

- include/aubo_caps/domain/feature/[feature_line.h](#)

11.39 arcs::aubo_scope::FeatureModel Class Reference

[FeatureModel](#) Provides methods that returns features from the current robot installation.

```
#include <feature_model.h>
```

Public Member Functions

- [FeatureModel](#) (FeatureModel &f)
- [FeatureModel](#) (FeatureModel &&f)
- [~FeatureModel](#) ()
- std::vector< FeaturePtr > [getAll](#) ()
- template<typename T>
std::vector< FeaturePtr > [getGeomFeture](#) ()
Returns the geometric features of the current installation.
- FeaturePtr [getByName](#) (const std::string &name)
Returns a feature by name.
- FeaturePtr [getBaseFeature](#) ()
Returns the base feature of the current installation.
- FeaturePtr [getToolFeature](#) ()
Returns the tool feature of the current installation.
- std::vector< FeaturePtr > [getFeatures](#) (std::function< bool(FeaturePtr)> filter)
Returns features matching the given filter.

Private Member Functions

- [FeatureModel](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

11.39.1 Detailed Description

[FeatureModel](#) Provides methods that returns features from the current robot installation.

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

11.39.2 Constructor & Destructor Documentation

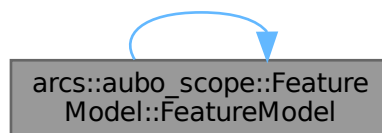
11.39.2.1 FeatureModel() [1/3]

```
arcs::aubo_scope::FeatureModel::FeatureModel (  
    FeatureModel & f)
```

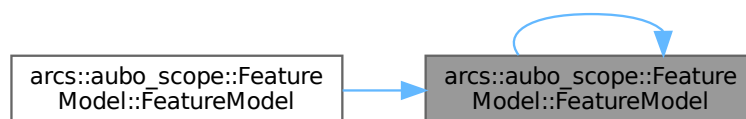
References [FeatureModel\(\)](#).

Referenced by [FeatureModel\(\)](#), and [FeatureModel\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

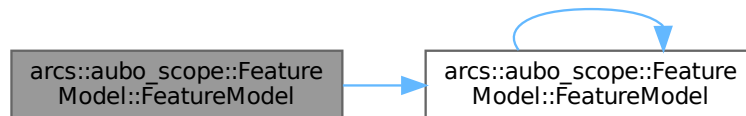


11.39.2.2 FeatureModel() [2/3]

```
arcs::aubo_scope::FeatureModel::FeatureModel (
    FeatureModel && f)
```

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

Here is the call graph for this function:



11.39.2.3 ~FeatureModel()

```
arcs::aubo_scope::FeatureModel::~~FeatureModel ()
```

11.39.2.4 FeatureModel() [3/3]

```
arcs::aubo_scope::FeatureModel::FeatureModel () [private]
```

11.39.3 Member Function Documentation

11.39.3.1 getAll()

```
std::vector< FeaturePtr > arcs::aubo_scope::FeatureModel::getAll ()
```

11.39.3.2 getBaseFeature()

```
FeaturePtr arcs::aubo_scope::FeatureModel::getBaseFeature ()
```

Returns the base feature of the current installation.

Returns

the base feature of the current installation.

11.39.3.3 getByName()

```
FeaturePtr arcs::aubo_scope::FeatureModel::getByName (
    Generated by Doxygen
    const std::string & name)
```

Returns a feature by name.

Parameters

name	the name of the feature.

Returns

the feature with the given name.

11.39.3.4 getFeatures()

```
std::vector< FeaturePtr > arcs::aubo_scope::FeatureModel::getFeatures (  
    std::function< bool(FeaturePtr)> filter)
```

Returns features matching the given filter.

Parameters

filter	the filter to apply.

Returns

a collection of features that match the filter.

11.39.3.5 getGeomFeture()

```
template<typename T>  
std::vector< FeaturePtr > arcs::aubo_scope::FeatureModel::getGeomFeture ()
```

Returns the geometric features of the current installation.

Returns

the geometric features of the current installation.

11.39.3.6 getToolFeature()

```
FeaturePtr arcs::aubo_scope::FeatureModel::getToolFeature ()
```

Returns the tool feature of the current installation.

Returns

the tool feature of the current installation.

11.39.4 Friends And Related Symbol Documentation

11.39.4.1 DataSwitch

friend class DataSwitch [friend]

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

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.39.5 Member Data Documentation

11.39.5.1 d_

void* arcs::aubo_scope::FeatureModel::d_ { nullptr } [private]

Definition at line 106 of file [feature_model.h](#).

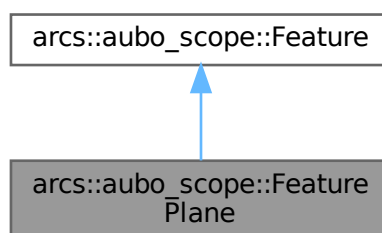
The documentation for this class was generated from the following file:

- include/aubo_caps/domain/feature/[feature_model.h](#)

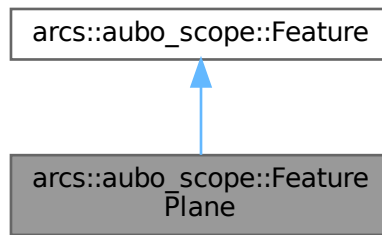
11.40 arcs::aubo_scope::FeaturePlane Class Reference

```
#include <feature_plane.h>
```

Inheritance diagram for arcs::aubo_scope::FeaturePlane:



Collaboration diagram for `arcs::aubo_scope::FeaturePlane`:



Public Member Functions

- [FeaturePlane](#) (FeaturePlane &f)
- [FeaturePlane](#) (FeaturePlane &&f)
- virtual [~FeaturePlane](#) ()

Public Member Functions inherited from [arcs::aubo_scope::Feature](#)

- [Feature](#) (Feature &f)
- [Feature](#) (Feature &&f)
- virtual [~Feature](#) ()
- std::string [getDisplayName](#) ()
 - Note: The feature can be renamed at any time.
- bool [isDefined](#) ()
 - Returns whether the position of the feature has been fully defined/specified.
- bool [isVariable](#) ()
 - Returns whether the feature can be used as a variable in the program.
- bool [isJoggable](#) ()
- bool [isResolvable](#) ()
 - A feature cannot be guaranteed to be present in AuboScope.
- std::vector< double > [getPose](#) ()
 - Returns the Cartesian location of the feature.
- size_t [hashCode](#) ()

Private Member Functions

- [FeaturePlane](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Protected Member Functions inherited from [arcs::aubo_scope::Feature](#)

- [Feature](#) ()

11.40.1 Detailed Description

Definition at line 10 of file [feature_plane.h](#).

11.40.2 Constructor & Destructor Documentation

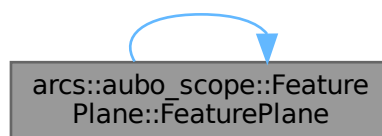
11.40.2.1 FeaturePlane() [1/3]

```
arcs::aubo_scope::FeaturePlane::FeaturePlane (  
    FeaturePlane & f)
```

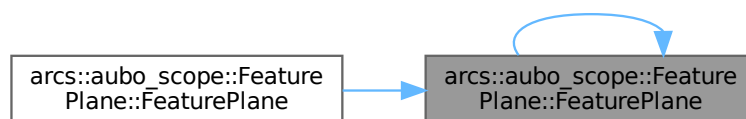
References [FeaturePlane\(\)](#).

Referenced by [FeaturePlane\(\)](#), and [FeaturePlane\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

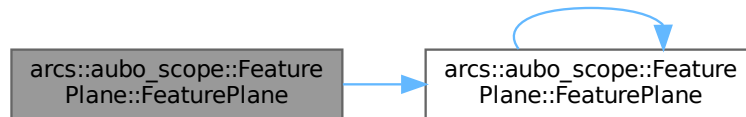


11.40.2.2 FeaturePlane() [2/3]

```
arcs::aubo_scope::FeaturePlane::FeaturePlane (
    FeaturePlane && f)
```

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

Here is the call graph for this function:



11.40.2.3 ~FeaturePlane()

```
virtual arcs::aubo_scope::FeaturePlane::~FeaturePlane () [virtual]
```

11.40.2.4 FeaturePlane() [3/3]

```
arcs::aubo_scope::FeaturePlane::FeaturePlane () [private]
```

11.40.3 Friends And Related Symbol Documentation

11.40.3.1 DataSwitch

```
friend class DataSwitch [friend]
```

Definition at line 18 of file [feature_plane.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.40.4 Member Data Documentation

11.40.4.1 d__

```
void* arcs::aubo_scope::FeaturePlane::d__ { nullptr } [private]
```

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

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

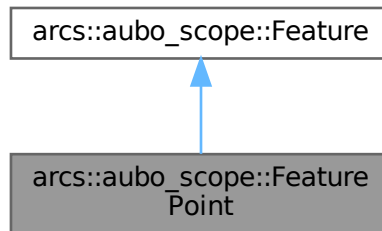
- `include/aubo_caps/domain/feature/feature_plane.h`

11.41 arcs::aubo_scope::FeaturePoint Class Reference

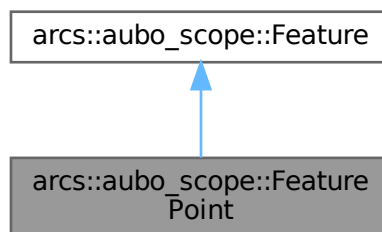
This interface represents a Point feature in AuboScope.

```
#include <feature_point.h>
```

Inheritance diagram for arcs::aubo_scope::FeaturePoint:



Collaboration diagram for arcs::aubo_scope::FeaturePoint:



Public Member Functions

- [FeaturePoint](#) (FeaturePoint &f)
- [FeaturePoint](#) (FeaturePoint &&f)
- virtual [~FeaturePoint](#) ()

Public Member Functions inherited from [arcs::aubo_scope::Feature](#)

- [Feature](#) (Feature &f)
- [Feature](#) (Feature &&f)
- virtual [~Feature](#) ()
- std::string [getDisplayname](#) ()

Note: The feature can be renamed at any time.

- bool [isDefined](#) ()
Returns whether the position of the feature has been fully defined/specified.
- bool [isVariable](#) ()
Returns whether the feature can be used as a variable in the program.
- bool [isJoggable](#) ()
- bool [isResolvable](#) ()
A feature cannot be guaranteed to be present in AuboScope.
- std::vector< double > [getPose](#) ()
Returns the Cartesian location of the feature.
- size_t [hashCode](#) ()

Private Member Functions

- [FeaturePoint](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Protected Member Functions inherited from [arcs::aubo_scope::Feature](#)

- [Feature](#) ()

11.41.1 Detailed Description

This interface represents a Point feature in AuboScope.

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

11.41.2 Constructor & Destructor Documentation

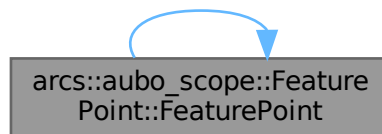
11.41.2.1 FeaturePoint() [1/3]

arcs::aubo_scope::FeaturePoint::FeaturePoint (
FeaturePoint & f)

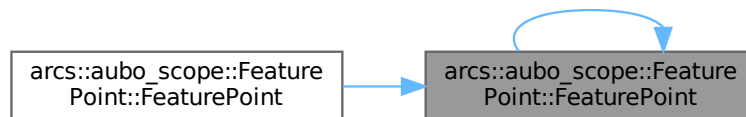
References [FeaturePoint\(\)](#).

Referenced by [FeaturePoint\(\)](#), and [FeaturePoint\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

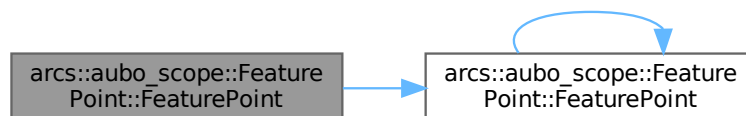


11.41.2.2 FeaturePoint() [2/3]

arcs::aubo_scope::FeaturePoint::FeaturePoint (
FeaturePoint && f)

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

Here is the call graph for this function:



11.41.2.3 ~FeaturePoint()

virtual arcs::aubo_scope::FeaturePoint::~~FeaturePoint () [virtual]

11.41.2.4 FeaturePoint() [3/3]

arcs::aubo_scope::FeaturePoint::FeaturePoint () [private]

11.41.3 Friends And Related Symbol Documentation

11.41.3.1 DataSwitch

friend class DataSwitch [friend]

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

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.41.4 Member Data Documentation

11.41.4.1 d_

void* arcs::aubo_scope::FeaturePoint::d_ { nullptr } [private]

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

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

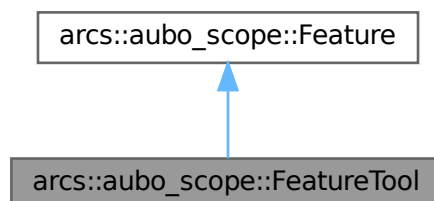
- [include/aubo_caps/domain/feature/feature_point.h](#)

11.42 arcs::aubo_scope::FeatureTool Class Reference

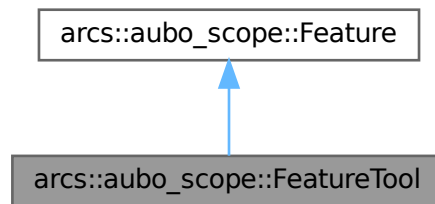
This interface represents the Tool feature in AuboScope.

`#include <feature_tool.h>`

Inheritance diagram for arcs::aubo_scope::FeatureTool:



Collaboration diagram for arcs::aubo_scope::FeatureTool:



Public Member Functions

- [FeatureTool](#) (FeatureTool &f)
- [FeatureTool](#) (FeatureTool &&f)
- virtual [~FeatureTool](#) ()

Public Member Functions inherited from [arcs::aubo_scope::Feature](#)

- [Feature](#) (Feature &f)
- [Feature](#) (Feature &&f)
- virtual [~Feature](#) ()
- std::string [getDisplayname](#) ()
 - Note: The feature can be renamed at any time.
- bool [isDefined](#) ()
 - Returns whether the position of the feature has been fully defined/specified.
- bool [isVariable](#) ()
 - Returns whether the feature can be used as a variable in the program.
- bool [isJoggable](#) ()
- bool [isResolvable](#) ()
 - A feature cannot be guaranteed to be present in AuboScope.
- std::vector< double > [getPose](#) ()
 - Returns the Cartesian location of the feature.
- size_t [hashCode](#) ()

Private Member Functions

- [FeatureTool](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Protected Member Functions inherited from [arcs::aubo_scope::Feature](#)

- [Feature](#) ()

11.42.1 Detailed Description

This interface represents the Tool feature in AuboScope.

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

11.42.2 Constructor & Destructor Documentation

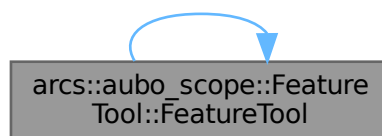
11.42.2.1 FeatureTool() [1/3]

```
arcs::aubo_scope::FeatureTool::FeatureTool (  
    FeatureTool & f)
```

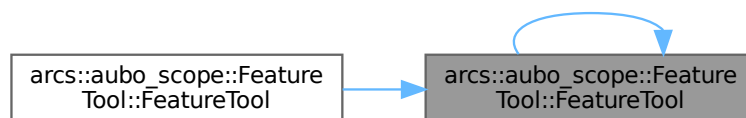
References [FeatureTool\(\)](#).

Referenced by [FeatureTool\(\)](#), and [FeatureTool\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

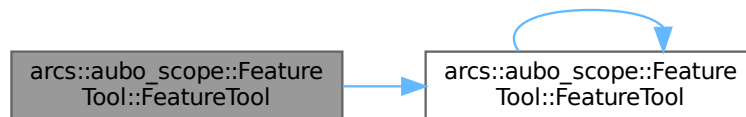


11.42.2.2 FeatureTool() [2/3]

```
arcs::aubo_scope::FeatureTool::FeatureTool (
    FeatureTool && f)
```

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

Here is the call graph for this function:



11.42.2.3 ~FeatureTool()

```
virtual arcs::aubo_scope::FeatureTool::~~FeatureTool () [virtual]
```

11.42.2.4 FeatureTool() [3/3]

```
arcs::aubo_scope::FeatureTool::FeatureTool () [private]
```

11.42.3 Friends And Related Symbol Documentation

11.42.3.1 DataSwitch

```
friend class DataSwitch [friend]
```

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

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.42.4 Member Data Documentation

11.42.4.1 d__

```
void* arcs::aubo_scope::FeatureTool::d__ { nullptr } [private]
```

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

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

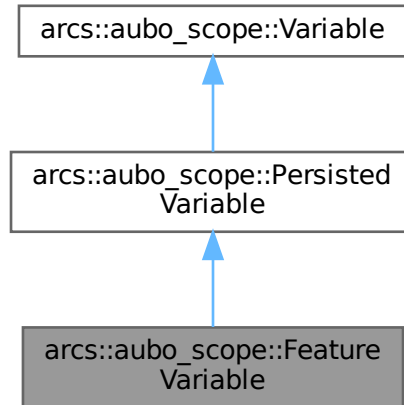
- `include/aubo_caps/domain/feature/feature_tool.h`

11.43 arcs::aubo_scope::FeatureVariable Class Reference

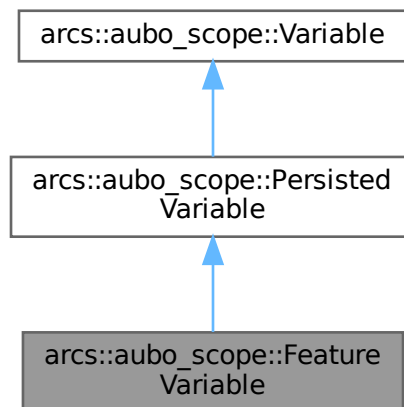
FeatureVariable A **Feature** marked as a **Variable** in the installation will exist even when a robot is turned off and on.

```
#include <feature_variable.h>
```

Inheritance diagram for arcs::aubo_scope::FeatureVariable:



Collaboration diagram for arcs::aubo_scope::FeatureVariable:



Public Member Functions

- **FeatureVariable** (FeatureVariable &f)
- **FeatureVariable** (FeatureVariable &&f)
- virtual **~FeatureVariable** ()

Public Member Functions inherited from arcs::aubo_scope::PersistedVariable

- [PersistedVariable](#) (PersistedVariable &f)
- [PersistedVariable](#) (PersistedVariable &&f)
- virtual [~PersistedVariable](#) ()
- bool [isResolvable](#) ()

A persisted variable can not be guaranteed to be present.

Public Member Functions inherited from arcs::aubo_scope::Variable

- [Variable](#) (Variable &f)
- [Variable](#) (Variable &&f)
- virtual [~Variable](#) ()
- [Type](#) [getType](#) ()
Returns the type of variable.
- std::string [getDisplayName](#) ()
Note: The variable can be renamed at any time.
- bool [equals](#) (VariablePtr object)
Checks if the object references the same internal [Variable](#).
- size_t [hashCode](#) ()
Returns the hash code for this object.
- std::string [toString](#) ()
Returns a string representation of the variable.

Private Member Functions

- [FeatureVariable](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Public Types inherited from arcs::aubo_scope::Variable

- enum [Type](#) : int { [GLOBAL](#) , [VALUE_PERSISTED](#) , [FEATURE](#) }
The variable type used to determine which type of variable this instance is.

Protected Member Functions inherited from arcs::aubo_scope::PersistedVariable

- [PersistedVariable](#) ()

Protected Member Functions inherited from [arcs::aubo_scope::Variable](#)

- [Variable](#) ()

11.43.1 Detailed Description

[FeatureVariable](#) A [Feature](#) marked as a [Variable](#) in the installation will exist even when a robot is turned off and on.

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

11.43.2 Constructor & Destructor Documentation

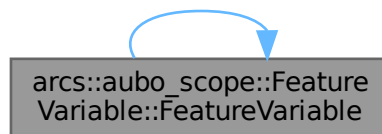
11.43.2.1 FeatureVariable() [1/3]

```
arcs::aubo_scope::FeatureVariable::FeatureVariable (
    FeatureVariable & f)
```

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

Referenced by [FeatureVariable\(\)](#), and [FeatureVariable\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

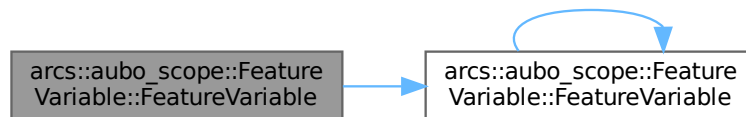


11.43.2.2 FeatureVariable() [2/3]

```
arcs::aubo_scope::FeatureVariable::FeatureVariable (
    FeatureVariable && f)
```

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

Here is the call graph for this function:



11.43.2.3 ~FeatureVariable()

```
virtual arcs::aubo_scope::FeatureVariable::~~FeatureVariable () [virtual]
```

11.43.2.4 FeatureVariable() [3/3]

```
arcs::aubo_scope::FeatureVariable::FeatureVariable () [private]
```

11.43.3 Friends And Related Symbol Documentation

11.43.3.1 DataSwitch

```
friend class DataSwitch [friend]
```

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

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.43.4 Member Data Documentation

11.43.4.1 d_

```
void* arcs::aubo_scope::FeatureVariable::d_ { nullptr } [private]
```

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

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

- `include/aubo_caps/domain/variable/feature_variable.h`

11.44 arcs::aubo_scope::FeatureVariableFilter Struct Reference

#include <feature_variable.h>

Public Member Functions

- bool [operator\(\)](#) (VariablePtr var)

11.44.1 Detailed Description

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

11.44.2 Member Function Documentation

11.44.2.1 operator()()

```
bool arcs::aubo_scope::FeatureVariableFilter::operator() (
    VariablePtr var)  [inline]
```

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

References [arcs::aubo_scope::Variable::FEATURE](#).

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

- include/aubo_caps/domain/variable/[feature_variable.h](#)

11.45 arcs::aubo_scope::FeedScrewParameters Class Reference

#include <screw_driver_configuration.h>

Public Member Functions

- virtual [~FeedScrewParameters](#) ()=default

11.45.1 Detailed Description

This interface provides parameters relevant for generating script code for the Feed Screw capability (registered using `ScrewdriverCapabilities#registerFeedScrewCapability(ScriptCodeGenerator)`).

These parameters are passed when AuboScope calls the implementation of the `ScriptCodeGenerator#generateScript(ScriptCodeGenerator)` method responsible for the generating the script code.

The interface is intentionally empty, but could have parameters in future releases.

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

11.45.2 Constructor & Destructor Documentation

11.45.2.1 ~FeedScrewParameters()

virtual arcs::aubo_scope::FeedScrewParameters::~~FeedScrewParameters () [virtual], [default]

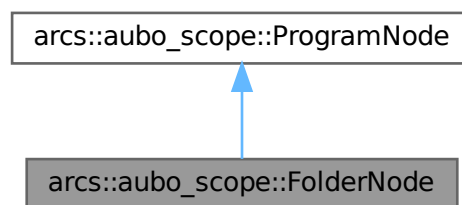
The documentation for this class was generated from the following file:

- include/aubo_caps/contribution/driver/screw_driver/[screw_driver_configuration.h](#)

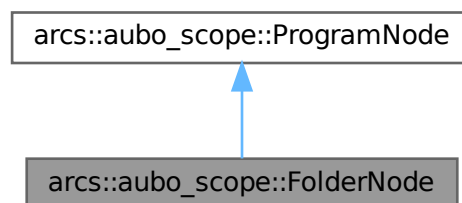
11.46 arcs::aubo_scope::FolderNode Class Reference

```
#include <folder_node.h>
```

Inheritance diagram for arcs::aubo_scope::FolderNode:



Collaboration diagram for arcs::aubo_scope::FolderNode:



Public Member Functions

- [FolderNode](#) (FolderNode &f)
- [FolderNode](#) (FolderNode &&f)
- virtual [~FolderNode](#) ()
- void [setFolderName](#) (const std::string &folder_name)

Public Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()
Get the program node contribution.

Private Member Functions

- [FolderNode](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Protected Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.46.1 Detailed Description

Definition at line 10 of file [folder_node.h](#).

11.46.2 Constructor & Destructor Documentation

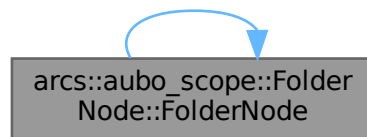
11.46.2.1 FolderNode() [1/3]

```
arcs::aubo_scope::FolderNode::FolderNode (  
    FolderNode & f)
```

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

Referenced by [FolderNode\(\)](#), and [FolderNode\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

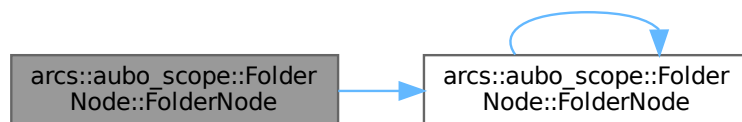


11.46.2.2 FolderNode() [2/3]

```
arcs::aubo_scope::FolderNode::FolderNode (
    FolderNode && f)
```

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

Here is the call graph for this function:



11.46.2.3 ~FolderNode()

```
virtual arcs::aubo_scope::FolderNode::~~FolderNode () [virtual]
```

11.46.2.4 FolderNode() [3/3]

```
arcs::aubo_scope::FolderNode::FolderNode () [private]
```

11.46.3 Member Function Documentation

11.46.3.1 setFolderName()

```
void arcs::aubo_scope::FolderNode::setFolderName (
    const std::string & folder_name)
```

- Set the name for the Folder node.

Parameters

set	The name for the node.

Returns

This node.

11.46.4 Friends And Related Symbol Documentation

11.46.4.1 DataSwitch

```
friend class DataSwitch [friend]
```

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

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.46.5 Member Data Documentation

11.46.5.1 d__

```
void* arcs::aubo_scope::FolderNode::d__ { nullptr } [private]
```

Definition at line 27 of file [folder_node.h](#).

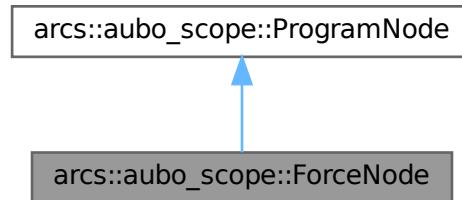
The documentation for this class was generated from the following file:

- [include/aubo_caps/domain/program/nodes/builtin/folder_node.h](#)

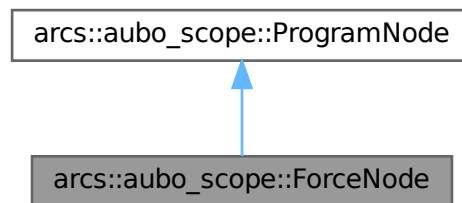
11.47 arcs::aubo_scope::ForceNode Class Reference

```
#include <force_node.h>
```

Inheritance diagram for arcs::aubo_scope::ForceNode:



Collaboration diagram for arcs::aubo_scope::ForceNode:



Public Types

- enum `ConfigType` { `SIMPLE` , `FRAME` , `POINT` , `MOTION` }

The configuration type used to determine which type of configuration this instance is.

Public Member Functions

- `ForceNode` (`ForceNode &f`)
- `ForceNode` (`ForceNode &&f`)
- virtual `~ForceNode` ()

Public Member Functions inherited from arcs::aubo_scope::ProgramNode

- `ProgramNode` (`ProgramNode &f`)
- `ProgramNode` (`ProgramNode &&f`)
- virtual `~ProgramNode` ()
- `ProgramNodeContributionPtr` `getProgramNodeContribution` ()
Get the program node contribution.

Private Member Functions

- [ForceNode](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Protected Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.47.1 Detailed Description

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

11.47.2 Member Enumeration Documentation

11.47.2.1 ConfigType

enum [arcs::aubo_scope::ForceNode::ConfigType](#)

The configuration type used to determine which type of configuration this instance is.

Enumerator

SIMPLE	Simple type is selected. This type has no further information. Setting this type of config will be ignored and will have no effect on the node.
FRAME	Frame type is selected. The config instance can be cast to FrameForceNodeConfig.
POINT	Point type is selected. This type has no further information. Setting this type of config will be ignored and will have no effect on the node.
MOTION	Motion type is selected. This type has no further information. Setting this type of config will be ignored and will have no effect on the node.

Definition at line 18 of file [force_node.h](#).

11.47.3 Constructor & Destructor Documentation

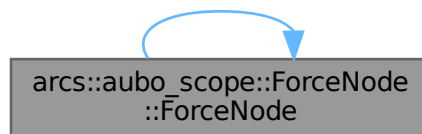
11.47.3.1 ForceNode() [1/3]

```
arcs::aubo_scope::ForceNode::ForceNode (  
    ForceNode & f)
```

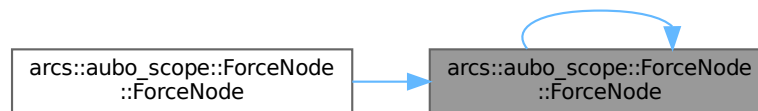
References [ForceNode\(\)](#).

Referenced by [ForceNode\(\)](#), and [ForceNode\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

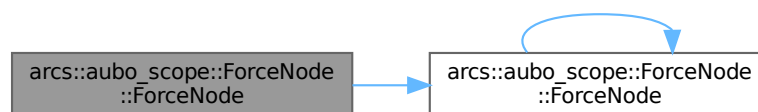


11.47.3.2 ForceNode() [2/3]

```
arcs::aubo_scope::ForceNode::ForceNode (  
    ForceNode && f)
```

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

Here is the call graph for this function:



11.47.3.3 ~ForceNode()

virtual arcs::aubo_scope::ForceNode::~~ForceNode () [virtual]

11.47.3.4 ForceNode() [3/3]

arcs::aubo_scope::ForceNode::ForceNode () [private]

11.47.4 Friends And Related Symbol Documentation

11.47.4.1 DataSwitch

friend class DataSwitch [friend]

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

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.47.5 Member Data Documentation

11.47.5.1 d_

void* arcs::aubo_scope::ForceNode::d_ { nullptr } [private]

Definition at line 68 of file [force_node.h](#).

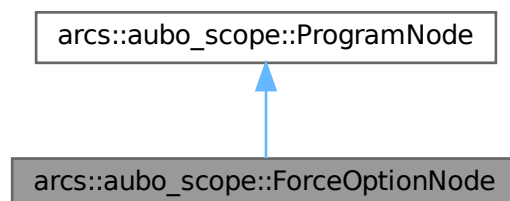
The documentation for this class was generated from the following file:

- [include/aubo_caps/domain/program/nodes/builtin/force_node.h](#)

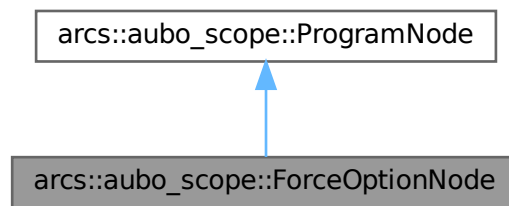
11.48 arcs::aubo_scope::ForceOptionNode Class Reference

#include <force_option_node.h>

Inheritance diagram for arcs::aubo_scope::ForceOptionNode:



Collaboration diagram for arcs::aubo_scope::ForceOptionNode:



Public Member Functions

- [ForceOptionNode](#) (ForceOptionNode &f)
- [ForceOptionNode](#) (ForceOptionNode &&f)
- virtual [~ForceOptionNode](#) ()

Public Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()
Get the program node contribution.

Private Member Functions

- [ForceOptionNode](#) ()

Private Attributes

- void * [d__](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Protected Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.48.1 Detailed Description

Definition at line 10 of file [force_option_node.h](#).

11.48.2 Constructor & Destructor Documentation

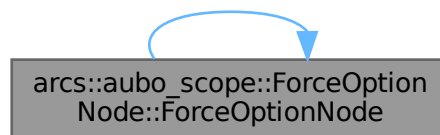
11.48.2.1 ForceOptionNode() [1/3]

```
arcs::aubo_scope::ForceOptionNode::ForceOptionNode (
    ForceOptionNode & f)
```

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

Referenced by [ForceOptionNode\(\)](#), and [ForceOptionNode\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

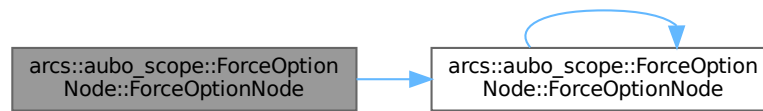


11.48.2.2 ForceOptionNode() [2/3]

```
arcs::aubo_scope::ForceOptionNode::ForceOptionNode (
    ForceOptionNode && f)
```

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

Here is the call graph for this function:



11.48.2.3 ~ForceOptionNode()

virtual arcs::aubo_scope::ForceOptionNode::~~ForceOptionNode () [virtual]

11.48.2.4 ForceOptionNode() [3/3]

arcs::aubo_scope::ForceOptionNode::ForceOptionNode () [private]

11.48.3 Friends And Related Symbol Documentation

11.48.3.1 DataSwitch

friend class DataSwitch [friend]

Definition at line 18 of file [force_option_node.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.48.4 Member Data Documentation

11.48.4.1 d__

void* arcs::aubo_scope::ForceOptionNode::d__ { nullptr } [private]

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

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

- include/aubo_caps/domain/program/nodes/builtin/[force_option_node.h](#)

11.49 arcs::aubo_scope::Function Class Reference

Function Represents a function shown in the [Expression](#) Editor.

```
#include <function.h>
```

Public Member Functions

- [Function](#) (Function &f)
- [Function](#) (Function &&f)
- [~Function](#) ()
- std::string [getName](#) ()
Returns the name of the function.
- std::vector< std::string > [getArguments](#) ()
Returns a list of the arguments of the function.
- AuboCapInfoPtr [getContributorInfo](#) ()
Get information about the AuboCap that added/contributed this function to AuboScope.

Private Member Functions

- [Function](#) ()

Private Attributes

- void * [d__](#) { nullptr }

Friends

- class [DataSwitch](#)

11.49.1 Detailed Description

[Function](#) Represents a function shown in the [Expression](#) Editor.

See also [FunctionModel](#).

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

11.49.2 Constructor & Destructor Documentation

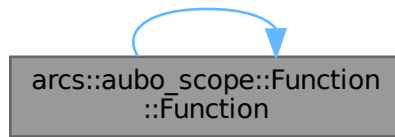
11.49.2.1 [Function](#)() [1/3]

```
arcs::aubo_scope::Function::Function (  
    Function & f)
```

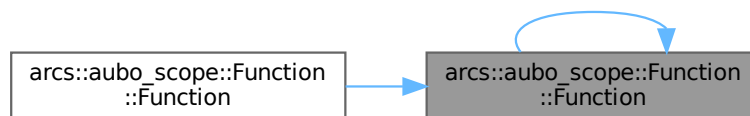
References [Function\(\)](#).

Referenced by [Function\(\)](#), and [Function\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

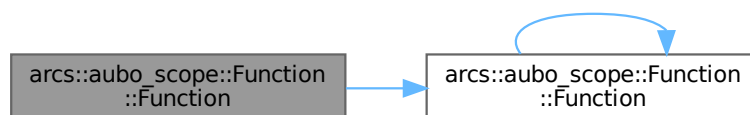


11.49.2.2 `Function()` [2/3]

```
arcs::aubo_scope::Function::Function (  
    Function && f)
```

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

Here is the call graph for this function:



11.49.2.3 `~Function()`

```
arcs::aubo_scope::Function::~~Function ()
```

11.49.2.4 Function() [3/3]

arcs::aubo_scope::Function::Function () [private]

11.49.3 Member Function Documentation

11.49.3.1 getArguments()

std::vector< std::string > arcs::aubo_scope::Function::getArguments ()

Returns a list of the arguments of the function.

Returns

A list of the arguments of the function.

11.49.3.2 getContributorInfo()

AuboCapInfoPtr arcs::aubo_scope::Function::getContributorInfo ()

Get information about the AuboCap that added/contributed this function to AuboScope.

Returns

AuboCap information for this function.

11.49.3.3 getName()

std::string arcs::aubo_scope::Function::getName ()

Returns the name of the function.

Returns

The name of the function.

11.49.4 Friends And Related Symbol Documentation

11.49.4.1 DataSwitch

friend class DataSwitch [friend]

Definition at line 71 of file [function.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.49.5 Member Data Documentation

11.49.5.1 d_

```
void* arcs::aubo_scope::Function::d_ { nullptr } [private]
```

Definition at line 73 of file [function.h](#).

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

- include/aubo_caps/domain/function/[function.h](#)

11.50 arcs::aubo_scope::FunctionModel Class Reference

[FunctionModel](#) This interface provides methods to add and remove script functions in the [Expression](#) Editor accessible from Assignment and If program nodes among others.

```
#include <function_model.h>
```

Public Member Functions

- [FunctionModel](#) (FunctionModel &f)
- [FunctionModel](#) (FunctionModel &&f)
- [~FunctionModel](#) ()
- FunctionPtr [addFunction](#) (const std::string &name, const std::vector< std::string > &argument←Names)
Add a script function to the [Expression](#) Editor.
- bool [removeFunction](#) (FunctionPtr func)
Remove a function.
- FunctionPtr [getFunction](#) (const std::string &name)
Get a function by name.
- std::vector< FunctionPtr > [getFunctions](#) ()
Get all added functions.

Private Member Functions

- [FunctionModel](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

11.50.1 Detailed Description

[FunctionModel](#) This interface provides methods to add and remove script functions in the [Expression Editor](#) accessible from Assignment and If program nodes among others.

Added functions merely serve as an easy way for the end customer to use the contributed functions. The implementation of a given script function must be added in the preamble of program through the use of `InstallationNodeContribution#generateScript(ScriptWriter)`.

The full set of functions which can be contributed must be added in the preamble at all times, regardless of whether they are added to the [Expression Editor](#) or not. Otherwise old programs that rely on a script function contribution might stop working.

See also [Function](#).

Definition at line 45 of file [function_model.h](#).

11.50.2 Constructor & Destructor Documentation

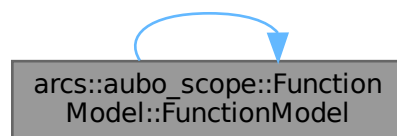
11.50.2.1 FunctionModel() [1/3]

```
arcs::aubo_scope::FunctionModel::FunctionModel (  
    FunctionModel & f)
```

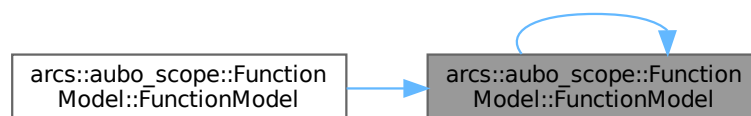
References [FunctionModel\(\)](#).

Referenced by [FunctionModel\(\)](#), and [FunctionModel\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

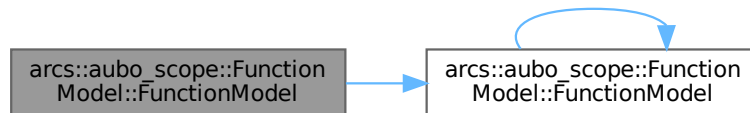


11.50.2.2 FunctionModel() [2/3]

```
arcs::aubo_scope::FunctionModel::FunctionModel (
    FunctionModel && f)
```

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

Here is the call graph for this function:



11.50.2.3 ~FunctionModel()

```
arcs::aubo_scope::FunctionModel::~~FunctionModel ()
```

11.50.2.4 FunctionModel() [3/3]

```
arcs::aubo_scope::FunctionModel::FunctionModel () [private]
```

11.50.3 Member Function Documentation

11.50.3.1 addFunction()

```
FunctionPtr arcs::aubo_scope::FunctionModel::addFunction (
    const std::string & name,
    const std::vector< std::string > & argumentNames)
```

Add a script function to the [Expression](#) Editor.

Parameters

name	Name of the function. Safe names must match regex <code>[a-zA-Z][a-zA-Z0-9_]{0,24}</code> for a total of 25 characters.
argumentNames	Is a list of hints for the function arguments.

Returns

If the script function was successfully added, an object representing the [Function](#) will be returned. Null otherwise.

Exceptions

FunctionException	if the name is illegal or more than five arguments are specified.

11.50.3.2 getFunction()

```
FunctionPtr arcs::aubo_scope::FunctionModel::getFunction (  
    const std::string & name)
```

Get a function by name.

Parameters

name	The name of the function.

Returns

The function if found, otherwise null.

11.50.3.3 getFunctions()

```
std::vector< FunctionPtr > arcs::aubo_scope::FunctionModel::getFunctions ()
```

Get all added functions.

Returns

A collection of added functions.

11.50.3.4 removeFunction()

```
bool arcs::aubo_scope::FunctionModel::removeFunction (  
    FunctionPtr func)
```

Remove a function.

Only functions added by this AuboCap can be removed.

Parameters

function	Function to be removed.

Returns

true, if the function was successfully removed.

11.50.4 Friends And Related Symbol Documentation

11.50.4.1 DataSwitch

friend class DataSwitch [friend]

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

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.50.5 Member Data Documentation

11.50.5.1 d_

void* arcs::aubo_scope::FunctionModel::d_ { nullptr } [private]

Definition at line 119 of file [function_model.h](#).

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

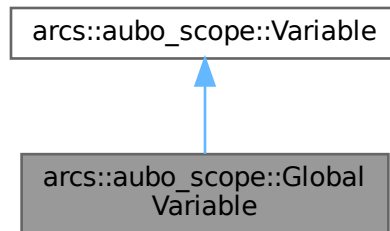
- include/aubo_caps/domain/function/[function_model.h](#)

11.51 arcs::aubo_scope::GlobalVariable Class Reference

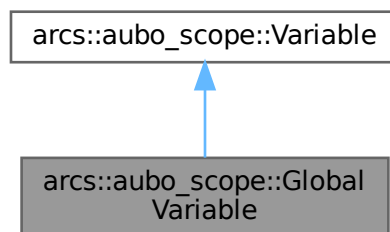
Global variables are available everywhere in a program.

```
#include <global_variable.h>
```

Inheritance diagram for arcs::aubo_scope::GlobalVariable:



Collaboration diagram for arcs::aubo_scope::GlobalVariable:



Public Member Functions

- [GlobalVariable](#) (GlobalVariable &f)
- [GlobalVariable](#) (GlobalVariable &&f)
- virtual [~GlobalVariable](#) ()

Public Member Functions inherited from [arcs::aubo_scope::Variable](#)

- [Variable](#) (Variable &f)
- [Variable](#) (Variable &&f)
- virtual [~Variable](#) ()
- [Type](#) [getType](#) ()

- Returns the type of variable.
- `std::string` [getDisplayName](#) ()
Note: The variable can be renamed at any time.
- `bool` [equals](#) (VariablePtr object)
Checks if the object references the same internal [Variable](#).
- `size_t` [hashCode](#) ()
Returns the hash code for this object.
- `std::string` [toString](#) ()
Returns a string representation of the variable.

Protected Member Functions

- [GlobalVariable](#) ()

Protected Member Functions inherited from [arcs::aubo_scope::Variable](#)

- [Variable](#) ()

Private Attributes

- `void *` [d_](#) { nullptr }

Friends

- `class` [DataSwitch](#)

Additional Inherited Members

Public Types inherited from [arcs::aubo_scope::Variable](#)

- `enum` [Type](#) : `int` { [GLOBAL](#) , [VALUE_PERSISTED](#) , [FEATURE](#) }
The variable type used to determine which type of variable this instance is.

11.51.1 Detailed Description

Global variables are available everywhere in a program.

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

11.51.2 Constructor & Destructor Documentation

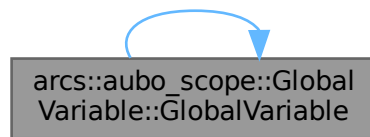
11.51.2.1 GlobalVariable() [1/3]

arcs::aubo_scope::GlobalVariable::GlobalVariable (
GlobalVariable & f)

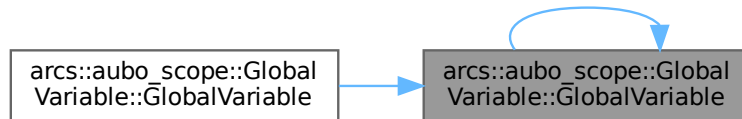
References [GlobalVariable\(\)](#).

Referenced by [GlobalVariable\(\)](#), and [GlobalVariable\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

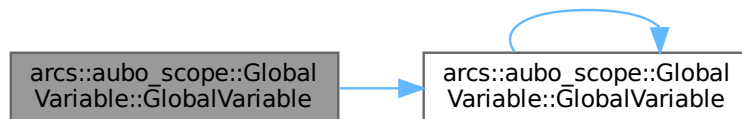


11.51.2.2 GlobalVariable() [2/3]

arcs::aubo_scope::GlobalVariable::GlobalVariable (
GlobalVariable && f)

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

Here is the call graph for this function:



11.51.2.3 ~GlobalVariable()

virtual arcs::aubo_scope::GlobalVariable::~~GlobalVariable () [virtual]

11.51.2.4 GlobalVariable() [3/3]

arcs::aubo_scope::GlobalVariable::GlobalVariable () [protected]

11.51.3 Friends And Related Symbol Documentation

11.51.3.1 DataSwitch

friend class DataSwitch [friend]

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

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.51.4 Member Data Documentation

11.51.4.1 d__

void* arcs::aubo_scope::GlobalVariable::d__ { nullptr } [private]

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

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

- include/aubo_caps/domain/variable/[global_variable.h](#)

11.52 arcs::aubo_scope::GlobalVariableFilter Struct Reference

#include <global_variable.h>

Public Member Functions

- bool [operator\(\)](#) (VariablePtr var)

11.52.1 Detailed Description

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

11.52.2 Member Function Documentation

11.52.2.1 operator()()

bool arcs::aubo_scope::GlobalVariableFilter::operator() (
 VariablePtr var) [inline]

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

References [arcs::aubo_scope::Variable::GLOBAL](#).

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

- [include/aubo_caps/domain/variable/global_variable.h](#)

11.53 arcs::aubo_scope::GripActionParameters Class Reference

#include <grip_action_parameters.h>

Public Member Functions

- virtual [~GripActionParameters](#) ()=default
- virtual double [getForce](#) ()=0
- virtual double [getWidth](#) ()=0
- virtual double [getVacuum](#) ()=0
- virtual double [getSpeed](#) ()=0
- virtual bool [isGripDetectionEnabled](#) ()=0

11.53.1 Detailed Description

This interface represents parameters for a gripping action defined/configured by the end user.

These parameters are used when generating script code for a gripping action (see [GripperContribution#generateGripAction](#)).

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

11.53.2 Constructor & Destructor Documentation

11.53.2.1 ~GripActionParameters()

virtual arcs::aubo_scope::GripActionParameters::~~GripActionParameters () [virtual], [default]

11.53.3 Member Function Documentation

11.53.3.1 getForce()

virtual double arcs::aubo_scope::GripActionParameters::getForce () [pure virtual]

Get the selected force to be used the gripping action.

Exceptions

UnsupportedOperationException	if the gripper has not registered the force capability

11.53.3.2 getSpeed()

virtual double arcs::aubo_scope::GripActionParameters::getSpeed () [pure virtual]

Get the selected speed to be used for the gripping action.

This is only valid, if the speed capability has been registered (using the method double, double, double, Speed.Unit)).

Returns

The speed to use for the gripping action

Exceptions

UnsupportedOperationException	if the gripper has not registered the speed capability

11.53.3.3 getVacuum()

virtual double arcs::aubo_scope::GripActionParameters::getVacuum () [pure virtual]

Get the selected vacuum level to be used for the gripping action.

This is only valid, if the vacuum capability has been registered (using the method double, double, Pressure.Unit)).

Returns

The vacuum level to use for the gripping action

Exceptions

UnsupportedOperationException	if the gripper has not registered the vacuum capability

11.53.3.4 getWidth()

virtual double arcs::aubo_scope::GripActionParameters::getWidth () [pure virtual]

Generated by Doxygen

Get the selected width to be used for the gripping action.

This is only valid, if the width capability has been registered (using the method double, double, double, Length.Unit)).

Exceptions

UnsupportedOperationException	if the gripper has not registered the width capability

11.53.3.5 isGripDetectionEnabled()

virtual bool arcs::aubo_scope::GripActionParameters::isGripDetectionEnabled () [pure virtual]

Use this method to determine whether the grip detection option in the Gripper program node is on or off.

Note that this method will report that grip detection is off, when script code is to be generated due to the user testing a grip configuration of a gripper program or operating the gripper using the toolbar.

This method should only be called, if the grip detected feedback capability has been registered (using GripperFeedbackCapabilities#registerGripDetectedCapability(ScriptCodeGenerator)).

The purpose of this method is to ensure that the appropriate script code for performing a grip action is generated when the end user has specified a new payload value. The user-defined payload value will be applied by AuboScope immediately after the generated grip action script code has finished executing. Depending on the enablement state of grip detection, the generated script should have the following properties:

- Grip detection off: The script code should not finish earlier than when it is appropriate to apply the new payload value (i.e. when the object has been gripped). This typically means that the script code should wait a time period corresponding to closing the gripper's "fingers" fully (if the gripper is vacuum operated then long enough to achieve some level of vacuum).
- Grip detection on: The payload will only be set if the grip detection was successful, i.e. it will be applied immediately after a grip was detected. This means that it is not necessary for the script code to wait (since the built-in detection timeout functionality will handle the waiting part).

Returns

true if the end user has enabled grip detection, and false if grip detection is disabled.

Exceptions

UnsupportedOperationException	if the gripper has not registered the grip detected feedback capability

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

- include/aubo_caps/contribution/driver/gripper/[grip_action_parameters.h](#)

11.54 arcs::aubo_scope::GripForceCapability Class Reference

[GripForceCapability](#) This interface represents a registered force capability for a gripper which supports gripping using a user configurable force.

```
#include <grip_force_capability.h>
```

Public Member Functions

- virtual [~GripForceCapability](#) ()=0
- virtual void [updateCapability](#) (double minForce, double maxForce, double defaultGripForce)=0

11.54.1 Detailed Description

[GripForceCapability](#) This interface represents a registered force capability for a gripper which supports gripping using a user configurable force.

Through this interface, it is possible to adjust the range and default value for the force after the capability has been registered. This is typically not applicable for most grippers, but can be relevant when the range depends on another custom gripper setting (which can be configured by the user).

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

11.54.2 Constructor & Destructor Documentation

11.54.2.1 ~GripForceCapability()

```
virtual arcs::aubo_scope::GripForceCapability::~~GripForceCapability () [pure virtual]
```

11.54.3 Member Function Documentation

11.54.3.1 updateCapability()

```
virtual void arcs::aubo_scope::GripForceCapability::updateCapability (  
    double minForce,  
    double maxForce,  
    double defaultGripForce) [pure virtual]
```

Update the range and default value for the force supported by the gripper.

Updating the range could result in an already entered value in the configuration of a gripper program node being outside the range. This will make the given program node undefined, and force the end user to adjust the value to be valid again.

Parameters

minForce	minimum force supported by the gripper
maxForce	maximum force supported by the gripper
defaultGripForce	default value for the force for a grip action. This value is used for, e.g. performing a "default" grip action using the toolbar
unit	the unit for all specified values, null

Exceptions

InvalidCapabilityRange	if minForce > maxForce} or defaultGripForce is outside the range defined by minForce and maxForce

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

- [include/aubo_caps/contribution/driver/gripper/capability/grip_force_capability.h](#)

11.55 arcs::aubo_scope::GripperApiProvider Class Reference

[GripperApiProvider](#) This interface provides access to functionality and services available from within AuboScope relevant for gripper contributions.

```
#include <gripper_api_provider.h>
```

Public Member Functions

- virtual [~GripperApiProvider](#) ()=default
- virtual SystemApiPtr [getSystemApi](#) ()=0
- virtual IoModelPtr [getIoModel](#) ()=0

11.55.1 Detailed Description

[GripperApiProvider](#) This interface provides access to functionality and services available from within AuboScope relevant for gripper contributions.

Definition at line 22 of file [gripper_api_provider.h](#).

11.55.2 Constructor & Destructor Documentation

11.55.2.1 ~GripperApiProvider()

```
virtual arcs::aubo_scope::GripperApiProvider::~~GripperApiProvider () [virtual], [default]
```

11.55.3 Member Function Documentation

11.55.3.1 getIoModel()

virtual IoModelPtr arcs::aubo_scope::GripperApiProvider::getIoModel () [pure virtual]

Returns

An I/O model interface (e.g. for getting I/Os to display in the custom user input for setting up the gripper)

11.55.3.2 getSystemApi()

virtual SystemApiPtr arcs::aubo_scope::GripperApiProvider::getSystemApi () [pure virtual]

Returns

An interface providing access to system related functionality.

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

- include/aubo_caps/contribution/driver/gripper/[gripper_api_provider.h](#)

11.56 arcs::aubo_scope::GripperConfiguration Class Reference

This interface provides access to register or setup various properties and capabilities of a gripper.

```
#include <gripper_configuration.h>
```

Public Member Functions

- virtual [~GripperConfiguration](#) ()=default
- virtual WidthCapabilityPtr [registerWidthCapability](#) (double minWidth, double maxWidth, double defaultGripWidth, double defaultReleaseWidth)=0
[Register](#) a width/position capability for a gripper that supports moving to a user configurable position (open/close to a configurable width).
- virtual GripForceCapabilityPtr [registerGrippingForceCapability](#) (double minForce, double maxForce, double defaultGripForce)=0
[Register](#) a force capability for a gripper that supports gripping using a user configurable force.
- virtual SpeedCapabilityPtr [registerSpeedCapability](#) (double minSpeed, double maxSpeed, double defaultGripSpeed, double defaultReleaseSpeed)=0
[Register](#) a speed capability for a gripper that supports operating using a user configurable speed.
- virtual GripVacuumCapabilityPtr [registerGrippingVacuumCapability](#) (double minVacuum, double maxVacuum, double defaultGripVacuum)=0
- virtual void [registerGripDetectedCapability](#) (std::function< void(ScriptWriterPtr scriptWriter)> scriptCodeGenerator)=0
- virtual void [registerReleaseDetectedCapability](#) (std::function< void(ScriptWriterPtr scriptWriter)> scriptCodeGenerator)=0

11.56.1 Detailed Description

This interface provides access to register or setup various properties and capabilities of a gripper.

It is optional to register these capabilities and properties.

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

11.56.2 Constructor & Destructor Documentation

11.56.2.1 ~GripperConfiguration()

```
virtual arcs::aubo_scope::GripperConfiguration::~~GripperConfiguration () [virtual], [default]
```

11.56.3 Member Function Documentation

11.56.3.1 registerGripDetectedCapability()

```
virtual void arcs::aubo_scope::GripperConfiguration::registerGripDetectedCapability (
    std::function< void(ScriptWriterPtr scriptWriter)> scriptCodeGenerator) [pure virtual]
```

[Register](#) this feedback capability if the gripper can inform AuboScope whether it has detected that an object has been gripped, after a grip action has been triggered. This will enable the end user to configure the actions to take when the gripper has detected that an object has been gripped and/or when the grip action timed out.

The provided implementation of the ScriptCodeGenerator interface must generate the script code for determining if a grip was detected. The return value of the script code must be a boolean, i.e. 'True' or 'False'.

Note: When this capability has been registered, the `GripperContribution#generateGripActionScript(ScriptWriter, GripActionParameters)` method should generate different grip action script code depending on whether the end user has enabled or disabled the grip detection option (in the Gripper program node). For more details, see Javadoc for [GripActionParameters#isGripDetectionEnabled\(\)](#).

Parameters

scriptCodeGenerator	script generator that generates the script code for determining if a grip was detected. When the script code needs to be generated, the method <code>ScriptCodeGenerator#generateScript(ScriptWriter, Object)</code> will be called (by AuboScope). The resulting script code will be embedded in a script function and must return a boolean, i.e. 'True' or 'False'.

Exceptions

CapabilityAlreadyRegistered	if this capability has already been registered
CalledOutsideGripperConfigurationPhase	if this method is called at the wrong time, i.e. outside the scope of the GripperContribution#configure↵ Gripper(GripperConfiguration, GripperAPIProvider) method.

11.56.3.2 registerGrippingForceCapability()

```
virtual GripForceCapabilityPtr arcs::aubo_scope::GripperConfiguration::registerGrippingForceCapability (
    double minForce,
    double maxForce,
    double defaultGripForce) [pure virtual]
```

[Register](#) a force capability for a gripper that supports gripping using a user configurable force.

Parameters

minForce	minimum force supported by the gripper
maxForce	maximum force supported by the gripper
defaultGripForce	default value for the force for a grip action. This value is used for, e.g. performing a "default" grip action using the toolbar
unit	the unit for all specified values, null

Returns

a force capability allowing for dynamic adjustments of the capability properties including the value range and default value

Exceptions

InvalidCapabilityRange	if minForce > maxForce or defaultGripForce is outside the range defined by minForce and maxForce.
CapabilityAlreadyRegistered	if this capability has already been registered
CalledOutsideGripperConfigurationPhase	if this method is called at the wrong time by a AuboCap implementing the GripperContribution interface, i.e. called outside the scope of the configuration of the gripper.

CalledOutsideMethodScope	if this method is called at the wrong time by a (regular) AuboCap which has registered its program node contribution/service as conformant with a AuboScope gripper (using GripperRegistrationManager#registerAsGripper(Class)), i.e. called outside the scope of the call to either ProgramNodeService#configureContribution(↵ ContributionConfiguration) (for a Qt-based AuboCap) or ProgramNodeServiceConfigurable#configure↵ Contribution(ProgramNodeConfiguration) (for a HTML-based AuboCap).

11.56.3.3 registerGrippingVacuumCapability()

```
virtual GripVacuumCapabilityPtr arcs::aubo_scope::GripperConfiguration::registerGrippingVacuumCapability (
    double minVacuum,
    double maxVacuum,
    double defaultGripVacuum) [pure virtual]
```

[Register](#) a vacuum capability for a gripper that supports gripping using a user configurable vacuum level.

If the gripper works with absolute pressure, specify a positive range for the supported vacuum level. If the gripper works with relative pressure (vacuum), specify a negative range for the supported vacuum level.

Parameters

minVacuum	minimum vacuum level supported by the gripper
maxVacuum	maximum vacuum level supported by the gripper
defaultGripVacuum	default value for the vacuum level for a grip action. This value is used for, e.g. performing a "default" grip action using the toolbar
unit	the unit for all specified values, not null

Returns

a vacuum capability allowing for dynamic adjustments of the capability properties including the value range and default value

Exceptions

InvalidCapabilityRange	if minVacuum > maxVacuum or defaultGripVacuum is outside the range defined by minVacuum and maxVacuum.
CapabilityAlreadyRegistered	if this capability has already been registered
CalledOutsideGripperConfigurationPhase	if this method is called at the wrong time by a AuboCap implementing the GripperContribution interface, i.e. called outside the scope of the configuration of the gripper.
CalledOutsideMethodScope	if this method is called at the wrong time by a (regular) AuboCap which has registered its program node contribution/service as conformant with a AuboScope gripper (using <code>GripperRegistrationManager#registerAsGripper(Class)</code>), i.e. called outside the scope of the call to either <code>ProgramNodeService#configureContribution(↔ContributionConfiguration)</code> (for a Qt-based AuboCap) or <code>ProgramNodeServiceConfigurable#configure↔Contribution(ProgramNodeConfiguration)</code> (for a HTML-based AuboCap).

11.56.3.4 registerReleaseDetectedCapability()

```
virtual void arcs::aubo_scope::GripperConfiguration::registerReleaseDetectedCapability (
    std::function< void(ScriptWriterPtr scriptWriter)> scriptCodeGenerator) [pure virtual]
```

[Register](#) this feedback capability if the gripper can inform AuboScope whether it has detected an object has been released, after a release action has been triggered. This will enable the end user to configure the actions to take when the gripper has detected that an object has been released and/or when the release action timed out.

The provided implementation of the ScriptCodeGenerator interface must generate the script code for determining if a release was detected. The return value of the script code must be a boolean, i.e. 'True' or 'False'.

Note: When this capability has been registered, the `GripperContribution#generateReleaseActionScript(ScriptWriter, ReleaseActionParameters)` method should generate different release action script code depending on whether the end user has enabled or disabled the release detection option (in the Gripper program node). For more details, see Javadoc for [GripActionParameters#isGripDetectionEnabled\(\)](#).

Parameters

scriptCodeGenerator	script generator that generates the script code for determining if a release was detected. When the script code needs to be generated, the method <code>ScriptCodeGenerator#generateScript(ScriptWriter, Object)</code> will be called (by AuboScope). The resulting script code will be embedded in a script function and must return a boolean, i.e. 'True' or 'False'.

Exceptions

CapabilityAlreadyRegistered	if this capability has already been registered
CalledOutsideGripperConfigurationPhase	if this method is called at the wrong time, i.e. outside the scope of the GripperContribution#configure↵ Gripper(GripperConfiguration, GripperAPIProvider) method.

11.56.3.5 registerSpeedCapability()

```
virtual SpeedCapabilityPtr arcs::aubo_scope::GripperConfiguration::registerSpeedCapability (
    double minSpeed,
    double maxSpeed,
    double defaultGripSpeed,
    double defaultReleaseSpeed) [pure virtual]
```

[Register](#) a speed capability for a gripper that supports operating using a user configurable speed.

Parameters

minSpeed	minimum speed supported by the gripper
maxSpeed	maximum speed supported by the gripper
defaultGripSpeed	default value for the speed for a grip action. This value is used for, e.g. performing a "default" grip action using the toolbar
defaultRelease↵Speed	default value for the speed for a grip action. This value is used for, e.g. performing a "default" release action using the toolbar
unit	the unit for all specified values, not null

Returns

a speed capability allowing for dynamic adjustments of the capability properties including the value range and default values

Exceptions

InvalidCapabilityRange	if minSpeed > maxSpeed
InvalidCapabilityRange	if defaultGripSpeed or defaultReleaseSpeed is outside the range defined by minSpeed and maxSpeed.
CapabilityAlreadyRegistered	if this capability has already been registered

CalledOutsideGripperConfigurationPhase	if this method is called at the wrong time by a AuboCap implementing the GripperContribution interface, i.e. called outside the scope of the configuration of the gripper.
CalledOutsideMethodScope	if this method is called at the wrong time by a (regular) AuboCap which has registered its program node contribution/service as conformant with a AuboScope gripper (using <code>GripperRegistrationManager#registerAsGripper(Class)</code>), i.e. called outside the scope of the call to either <code>ProgramNodeService#configureContribution(↔ContributionConfiguration)</code> (for a Qt-based AuboCap) or <code>ProgramNodeServiceConfigurable#configure↔Contribution(ProgramNodeConfiguration)</code> (for a HTML-based AuboCap).

11.56.3.6 registerWidthCapability()

```
virtual WidthCapabilityPtr arcs::aubo_scope::GripperConfiguration::registerWidthCapability (
    double minWidth,
    double maxWidth,
    double defaultGripWidth,
    double defaultReleaseWidth) [pure virtual]
```

[Register](#) a width/position capability for a gripper that supports moving to a user configurable position (open/close to a configurable width).

Parameters

minWidth	The minimum width supported by the gripper
maxWidth	The maximum width supported by the gripper
defaultGripWidth	default value for the width for a grip action. This value is used for, e.g. performing a "default" grip action using the toolbar
defaultRelease↔Width	default value for the width for a release action. This value is used for, e.g. performing a "default" release action using the toolbar
unit	the unit for all specified values, not null

Returns

a width capability allowing for dynamic adjustments of the capability properties including the value range and default values

Exceptions

InvalidCapabilityRange	if minWidth > maxWidth
InvalidCapabilityRange	if defaultGripWidth or defaultReleaseWidth is outside the range defined by minWidth and maxWidth.
CapabilityAlreadyRegistered	if this capability has already been registered
CalledOutsideGripperConfigurationPhase	if this method is called at the wrong time by a AboCap implementing the GripperContribution interface, i.e. called outside the scope of the configuration of the gripper.
CalledOutsideMethodScope	if this method is called at the wrong time by a (regular) AboCap which has registered its program node contribution/service as conformant with a AboScope gripper (using <code>GripperRegistrationManager#registerAsGripper(Class)</code>), i.e. called outside the scope of the call to either <code>ProgramNodeService#configureContribution(ContributionConfiguration)</code> (for a Qt-based AboCap) or <code>ProgramNodeServiceConfigurable#configureContribution(ProgramNodeConfiguration)</code> (for a HTML-based AboCap).

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

- include/aubo_caps/contribution/driver/gripper/[gripper_configuration.h](#)

11.57 arcs::aubo_scope::GripperContribution Class Reference

[GripperContribution](#) This interface defines an API for adding functionality for gripping devices to AboScope.

```
#include <gripper_contribution.h>
```

Public Member Functions

- virtual [~GripperContribution](#) ()=default
- virtual std::string [getTitle](#) ()=0
This method must return the title of the gripper contribution.
- virtual void [configureGripper](#) (GripperConfigurationPtr gripperConfiguration, GripperApiProviderPtr apiProvider)=0
- virtual void [generateGripActionScript](#) (ScriptWriterPtr scriptWriter, GripActionParametersPtr parameters)=0
- virtual void [generateReleaseActionScript](#) (ScriptWriterPtr scriptWriter, ReleaseActionParametersPtr parameters)=0

11.57.1 Detailed Description

GripperContribution This interface defines an API for adding functionality for gripping devices to AuboScope.

Implementing this interface will create a program node contribution, an installation contribution and a toolbar contribution exposing gripper functionality to the end user. Note that the toolbar contribution is not supported on CB3 robots.

The program node contribution allows the end user to program grip and release actions. The configuration options available to the end user are based on the properties and capabilities supported by the gripper. As a minimum, all grippers must support "default" grip and release actions. All other capabilities are optional to support and can be registered when the `GripperContribution#configureGripper(GripperConfiguration, GripperAPIProvider)` method is called.

The installation node contribution can be used, if any custom input from the end user is required to setup the gripper, e.g. specifying an IP address. The corresponding generated UI will be accessible in the installation node. This functionality is optional, since such configuration properties are specific for each gripper. Access to create a custom configuration based on user inputs is provided when the method `GripperContribution#configureInstallation(CustomUserInputConfiguration, SystemConfiguration, TCPConfiguration, GripperAPIProvider)` is called. If no custom user inputs, texts, etc. are defined, the installation node will not be visible in AuboScope. Preamble script code (generated by `generatePreambleScript(ScriptWriter)`) will still be added to the robot program.

The toolbar contribution provides the end user quick access to perform grip and release actions using "default" gripper action parameters.

Definition at line 76 of file [gripper_contribution.h](#).

11.57.2 Constructor & Destructor Documentation

11.57.2.1 ~GripperContribution()

```
virtual arcs::aubo_scope::GripperContribution::~GripperContribution () [virtual], [default]
```

11.57.3 Member Function Documentation

11.57.3.1 configureGripper()

```
virtual void arcs::aubo_scope::GripperContribution::configureGripper (
    GripperConfigurationPtr gripperConfiguration,
    GripperApiProviderPtr apiProvider) [pure virtual]
```

When this method is called, use the configuration parameter to register or setup optional properties and capabilities of the gripper.

The method is called in the Gripper Configuration phase after this contribution has been registered when a new installation is loaded or created.

Note: If the gripper only supports basic "default" grip and release actions, leave the implementation of this method empty.

Parameters

gripperConfiguration	A configuration instance that can be used for registering or setting up the properties and capabilities of the gripper.
apiProvider	Provides access to functionality and services available from within AuboScope which can be relevant for setting up the gripper capabilities (e.g. the SystemAPI interface with functionality for querying information about the robot).

11.57.3.2 generateGripActionScript()

```
virtual void arcs::aubo_scope::GripperContribution::generateGripActionScript (
    ScriptWriterPtr scriptWriter,
    GripActionParametersPtr parameters) [pure virtual]
```

When this method is called, the script code for performing a grip action with the gripper must be generated.

...

Parameters

scriptWriter	Use this script writer instance to generate the script code for gripping
parameters	The parameters for the gripper action defined/configured by the end user

11.57.3.3 generateReleaseActionScript()

```
virtual void arcs::aubo_scope::GripperContribution::generateReleaseActionScript (
    ScriptWriterPtr scriptWriter,
    ReleaseActionParametersPtr parameters) [pure virtual]
```

When this method is called, the script code for performing a release action with the gripper must be generated.

...

Parameters

scriptWriter	Use this script writer instance to generate the script code for releasing the gripper
parameters	The parameters for the gripper action defined/configured by the end user

11.57.3.4 getTitle()

virtual std::string arcs::aubo_scope::GripperContribution::getTitle () [pure virtual]

This method must return the title of the gripper contribution.

The title is displayed in:

- The Structure section of the Program Tab for this type of gripper program node (from which the Gripper program node can be inserted)
- The program node screen for this gripper contribution.
- The Program Tree as title for the program node for this gripper contribution
- The left-hand side navigation of the Installation Tab for this gripper contribution.
- The installation node screen for this gripper contribution.
- The toolbar contribution for this gripper. Note: The toolbar contribution is not available on CB3 robots.

The return value of this method cannot be null nor an empty string.

This method is called once upon startup.

Parameters

locale	The current locale of AuboScope. Can be used for supporting titles in several languages.

Returns

The title of this gripper contribution, not null nor an empty string.

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

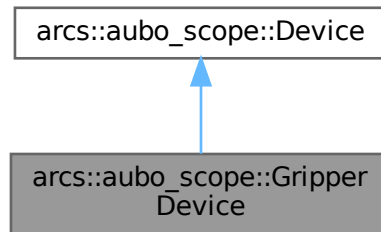
- include/aubo_caps/contribution/driver/gripper/[gripper_contribution.h](#)

11.58 arcs::aubo_scope::GripperDevice Class Reference

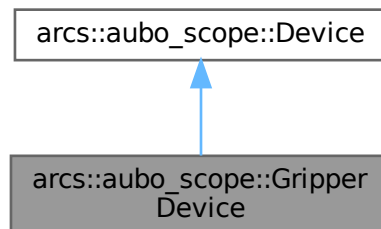
[GripperDevice](#) This interface represents a gripper device in AuboScope.

```
#include <gripper_device.h>
```

Inheritance diagram for arcs::aubo_scope::GripperDevice:



Collaboration diagram for arcs::aubo_scope::GripperDevice:



Public Member Functions

- virtual [~GripperDevice](#) ()=default

Public Member Functions inherited from [arcs::aubo_scope::Device](#)

- virtual [~Device](#) ()=0
- virtual AuboCapInfoPtr [getContributorInfo](#) ()=0
 - Get information about the AuboCap that added/contributed this gripper device to AuboScope.
- virtual std::string [getId](#) ()=0
- virtual std::string [getDisplayName](#) ()=0
- virtual bool [isResolvable](#) ()=0
 - A device cannot be guaranteed to be present in AuboScope.

11.58.1 Detailed Description

[GripperDevice](#) This interface represents a gripper device in AuboScope.

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

11.58.2 Constructor & Destructor Documentation

11.58.2.1 ~GripperDevice()

virtual arcs::aubo_scope::GripperDevice::~~GripperDevice () [virtual], [default]

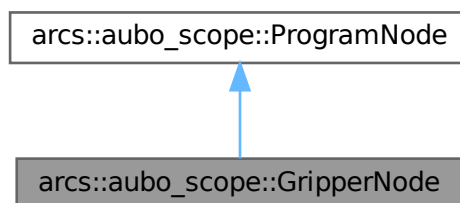
The documentation for this class was generated from the following file:

- include/aubo_caps/domain/device/[gripper_device.h](#)

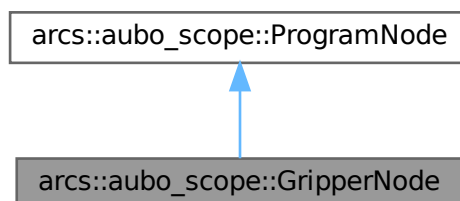
11.59 arcs::aubo_scope::GripperNode Class Reference

```
#include <gripper_node.h>
```

Inheritance diagram for arcs::aubo_scope::GripperNode:



Collaboration diagram for arcs::aubo_scope::GripperNode:



Public Types

- enum [ConfigType](#) { [UNDEFINED](#) , [GRIP_ACTION](#) , [RELEASE_ACTION](#) }
The configuration type used to determine which type of configuration this instance is.
- enum [SettingType](#) { [UNDEFINED0](#) , [PAYLOAD_SELECTION](#) , [CUSTOM_PARAMETERS](#) }
The setting type used to determine which type of setting this instance is.

Public Member Functions

- [GripperNode](#) (GripperNode &f)
- [GripperNode](#) (GripperNode &&f)
- virtual [~GripperNode](#) ()
- [ConfigType](#) [getConfigType](#) ()
This method returns the type of configuration.
- void [setConfigType](#) ([ConfigType](#) type)
- GripperDevicePtr [getGripperDevice](#) ()
- [SettingType](#) [getSettingType](#) ()
This method returns the type of setting.

Public Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()
Get the program node contribution.

Private Member Functions

- [GripperNode](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Protected Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.59.1 Detailed Description

This interface represents a program node which can be used for programming grip and release actions with a selected gripper device.

An instance of this interface can be built using the GripperNodeFactory interface and is returned as a result when GripperNodeFactory#createGripperNode(GripperDevice) is called.

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

11.59.2 Member Enumeration Documentation

11.59.2.1 ConfigType

enum [arcs::aubo_scope::GripperNode::ConfigType](#)

The configuration type used to determine which type of configuration this instance is.

Enumerator

UNDEFINED	A Gripper action (i.e. grip or release) has not been selected The config instance can be cast to UndefinedActionConfig. This is only relevant if the Gripper node uses a multi-gripper device, i.e. a gripper device supporting the multi-gripper capability (see the MultiGripperSupport interface). The instance will provide access to the individual gripper selection.
GRIP_ACTION	The gripper is configured for a grip action The config instance can be cast to GripActionConfig.
RELEASE_ACTION	The gripper is configured for a release action The config instance can be cast to ReleaseActionConfig.

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

11.59.2.2 SettingType

enum [arcs::aubo_scope::GripperNode::SettingType](#)

The setting type used to determine which type of setting this instance is.

Enumerator

UNDEFINED0	No selection has been made.
PAYLOAD_SELECTION	A payload (from the installation) is selected. The setting instance can be cast to PayloadSelection.

CUSTOM_PARAMETERS	Custom payload parameters are specified. The setting instance can be cast to CustomPayloadParameters.

Definition at line 101 of file [gripper_node.h](#).

11.59.3 Constructor & Destructor Documentation

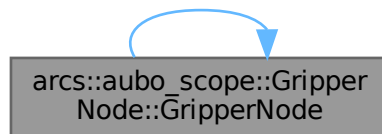
11.59.3.1 GripperNode() [1/3]

```
arcs::aubo_scope::GripperNode::GripperNode (
    GripperNode & f)
```

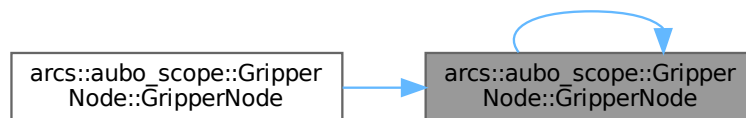
References [GripperNode\(\)](#).

Referenced by [GripperNode\(\)](#), and [GripperNode\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

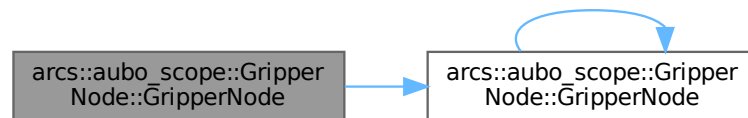


11.59.3.2 GripperNode() [2/3]

```
arcs::aubo_scope::GripperNode::GripperNode (
    GripperNode && f)
```

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

Here is the call graph for this function:



11.59.3.3 ~GripperNode()

```
virtual arcs::aubo_scope::GripperNode::~~GripperNode () [virtual]
```

11.59.3.4 GripperNode() [3/3]

```
arcs::aubo_scope::GripperNode::GripperNode () [private]
```

11.59.4 Member Function Documentation

11.59.4.1 getConfigType()

```
ConfigType arcs::aubo_scope::GripperNode::getConfigType ()
```

This method returns the type of configuration.

Cast this instance appropriately to have access to specific getters.

Returns

The type of this config.

11.59.4.2 getGripperDevice()

```
GripperDevicePtr arcs::aubo_scope::GripperNode::getGripperDevice ()
```

Returns

The gripper device used by this Gripper node

11.59.4.3 `getSettingType()`

[SettingType](#) arcs::aubo_scope::GripperNode::getSettingType ()

This method returns the type of setting.

Cast this instance appropriately to have access to specific getters.

Returns

The type of this payload setting.

11.59.4.4 `setConfigType()`

void arcs::aubo_scope::GripperNode::setConfigType (
[ConfigType](#) type)

11.59.5 Friends And Related Symbol Documentation

11.59.5.1 `DataSwitch`

friend class `DataSwitch` [friend]

Definition at line 144 of file [gripper_node.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.59.6 Member Data Documentation

11.59.6.1 `d__`

void* arcs::aubo_scope::GripperNode::d__ { nullptr } [private]

Definition at line 146 of file [gripper_node.h](#).

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

- include/aubo_caps/domain/program/nodes/builtin/[gripper_node.h](#)

11.60 arcs::aubo_scope::GripVacuumCapability Class Reference

[GripVacuumCapability](#) This interface represents a registered vacuum capability for a gripper which supports gripping using a user configurable vacuum level.

```
#include <grip_vacuum_capability.h>
```

Public Member Functions

- virtual [~GripVacuumCapability](#) ()=0
- virtual void [updateCapability](#) (double minVacuum, double maxVacuum, double defaultGripVacuum)=0

11.60.1 Detailed Description

[GripVacuumCapability](#) This interface represents a registered vacuum capability for a gripper which supports gripping using a user configurable vacuum level.

Through this interface, it is possible to adjust the range and default value for the vacuum level after the capability has been registered. This is typically not applicable for most grippers, but can be relevant when the range depends on another custom gripper setting (which can be configured by the user).

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

11.60.2 Constructor & Destructor Documentation

11.60.2.1 ~GripVacuumCapability()

```
virtual arcs::aubo_scope::GripVacuumCapability::~~GripVacuumCapability () [pure virtual]
```

11.60.3 Member Function Documentation

11.60.3.1 updateCapability()

```
virtual void arcs::aubo_scope::GripVacuumCapability::updateCapability (
    double minVacuum,
    double maxVacuum,
    double defaultGripVacuum) [pure virtual]
```

Update the range and default value for the vacuum level supported by the gripper.

Updating the range could result in an already entered value in the configuration of a gripper program node being outside the range. This will make the given program node undefined, and force the end user to adjust the value to be valid again.

If the gripper works with absolute pressure, specify a positive range for the supported vacuum level. If the gripper works with relative pressure (vacuum), specify a negative range for the supported vacuum level.

Parameters

minVacuum	minimum vacuum level supported by the gripper
maxVacuum	maximum vacuum level supported by the gripper

defaultGripVacuum	default value for the vacuum level for a grip action. This value is used for, e.g. performing a "default" grip action using the toolbar
unit	the unit for all specified values, not null

Exceptions

InvalidCapabilityRange	if minVacuum > maxVacuum or defaultGripVacuum is outside the range defined by minVacuum and maxVacuum.

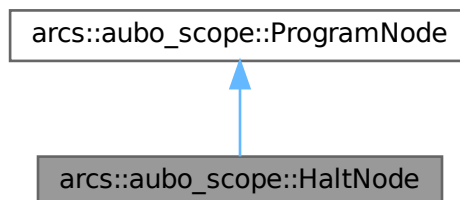
The documentation for this class was generated from the following file:

- include/aubo_caps/contribution/driver/gripper/capability/[grip_vacuum_capability.h](#)

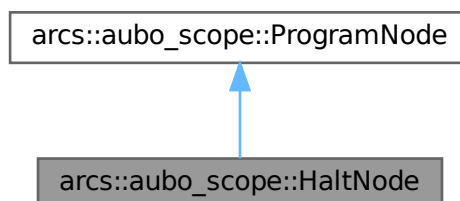
11.61 arcs::aubo_scope::HaltNode Class Reference

#include <halt_node.h>

Inheritance diagram for arcs::aubo_scope::HaltNode:



Collaboration diagram for arcs::aubo_scope::HaltNode:



Public Member Functions

- [HaltNode](#) (HaltNode &f)
- [HaltNode](#) (HaltNode &&f)
- virtual [~HaltNode](#) ()

Public Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()
Get the program node contribution.

Private Member Functions

- [HaltNode](#) ()

Private Attributes

- void * [d__](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Protected Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.61.1 Detailed Description

Definition at line 10 of file [halt_node.h](#).

11.61.2 Constructor & Destructor Documentation

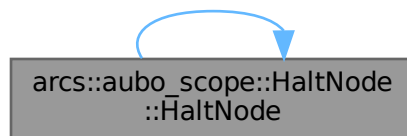
11.61.2.1 HaltNode() [1/3]

arcs::aubo_scope::HaltNode::HaltNode (
HaltNode & f)

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

Referenced by [HaltNode\(\)](#), and [HaltNode\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

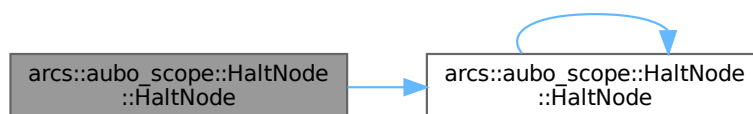


11.61.2.2 HaltNode() [2/3]

arcs::aubo_scope::HaltNode::HaltNode (
HaltNode && f)

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

Here is the call graph for this function:



11.61.2.3 ~HaltNode()

virtual arcs::aubo_scope::HaltNode::~HaltNode () [virtual]

11.61.2.4 HaltNode() [3/3]

arcs::aubo_scope::HaltNode::HaltNode () [private]

11.61.3 Friends And Related Symbol Documentation

11.61.3.1 DataSwitch

friend class DataSwitch [friend]

Definition at line 18 of file [halt_node.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.61.4 Member Data Documentation

11.61.4.1 d__

void* arcs::aubo_scope::HaltNode::d__ { nullptr } [private]

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

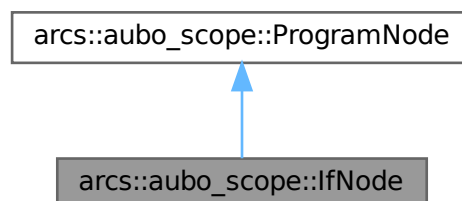
The documentation for this class was generated from the following file:

- [include/aubo_caps/domain/program/nodes/builtin/halt_node.h](#)

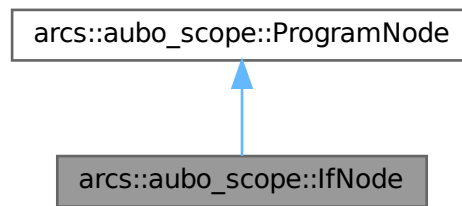
11.62 arcs::aubo_scope::IfNode Class Reference

#include <if_node.h>

Inheritance diagram for arcs::aubo_scope::IfNode:



Collaboration diagram for `arcs::aubo_scope::IfNode`:



Public Member Functions

- [IfNode](#) (IfNode &f)
- [IfNode](#) (IfNode &&f)
- virtual [~IfNode](#) ()
- void [setExpression](#) (ExpressionPtr expression)
Sets the expression used by the node.
- ExpressionPtr [getExpression](#) ()

Public Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()
Get the program node contribution.

Private Member Functions

- [IfNode](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Protected Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.62.1 Detailed Description

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

11.62.2 Constructor & Destructor Documentation

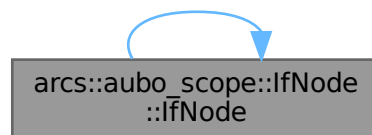
11.62.2.1 IfNode() [1/3]

```
arcs::aubo_scope::IfNode::IfNode (  
    IfNode & f)
```

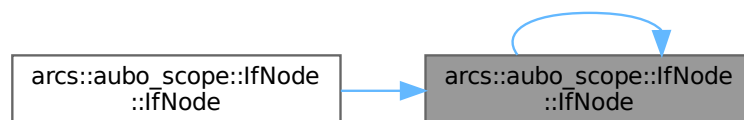
References [IfNode\(\)](#).

Referenced by [IfNode\(\)](#), and [IfNode\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

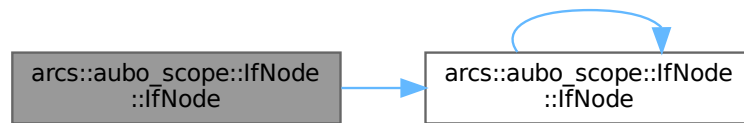


11.62.2.2 IfNode() [2/3]

```
arcs::aubo_scope::IfNode::IfNode (  
    IfNode && f)
```

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

Here is the call graph for this function:



11.62.2.3 ~IfNode()

virtual arcs::aubo_scope::IfNode::~IfNode () [virtual]

11.62.2.4 IfNode() [3/3]

arcs::aubo_scope::IfNode::IfNode () [private]

11.62.3 Member Function Documentation

11.62.3.1 getExpression()

ExpressionPtr arcs::aubo_scope::IfNode::getExpression ()

Returns

the [Expression](#) to be evaluated in this If node.

11.62.3.2 setExpression()

void arcs::aubo_scope::IfNode::setExpression (
ExpressionPtr expression)

Sets the expression used by the node.

Parameters

expression	The expression that will be evaluated in this If Node.

Exceptions

IllegalStateException	if called from a Qt-based AuboCap outside of an UndoableChanges scope.

11.62.4 Friends And Related Symbol Documentation

11.62.4.1 DataSwitch

friend class DataSwitch [friend]

Definition at line 44 of file [if_node.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.62.5 Member Data Documentation

11.62.5.1 d__

void* arcs::aubo_scope::IfNode::d__ { nullptr } [private]

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

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

- [include/aubo_caps/domain/program/nodes/builtin/if_node.h](#)

11.63 arcs::aubo_scope::InputIoFilter Struct Reference

```
#include <digital_io.h>
```

Public Member Functions

- bool [operator\(\)](#) (IoPtr io)

11.63.1 Detailed Description

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

11.63.2 Member Function Documentation

11.63.2.1 operator()()

```
bool arcs::aubo_scope::InputIoFilter::operator() (
    IoPtr io) [inline]
```

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

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

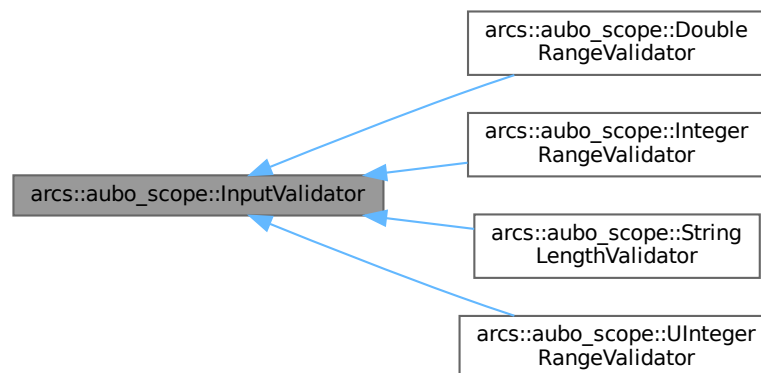
- [include/aubo_caps/domain/io/digital_io.h](#)

11.64 arcs::aubo_scope::InputValidator Struct Reference

[InputValidator](#) Interface representing the input validators created by InputValidationFactory.

```
#include <input_validator.h>
```

Inheritance diagram for arcs::aubo_scope::InputValidator:



Public Member Functions

- virtual bool [isValid](#) (std::string value)=0
Validates whether the input value is valid.
- virtual std::string [getMessage](#) (std::string value)=0
Returns a meaningful message in case the value is not valid.

11.64.1 Detailed Description

[InputValidator](#) Interface representing the input validators created by InputValidationFactory.

This factory provides a set of standard input validators that can be used for validating the input entered by the user using the virtual keyboard/keypad.

If the standard validators available in the InputValidationFactory does not satisfy your needs, you are free to implement your own validators.

Parameters

<T>	the (generic) type parameter for the interface representing the type of input data being validated (e.g. Integer or Double).

Definition at line 41 of file [input_validator.h](#).

11.64.2 Member Function Documentation

11.64.2.1 getMessage()

```
virtual std::string arcs::aubo_scope::InputValidator::getMessage (  
    std::string value) [pure virtual]
```

Returns a meaningful message in case the value is not valid.

Parameters

value	the invalid value. Can be included in the message if it makes sense.

Returns

message.

Implemented in [arcs::aubo_scope::DoubleRangeValidator](#), [arcs::aubo_scope::IntegerRangeValidator](#), [arcs::aubo_scope::StringLengthValidator](#), and [arcs::aubo_scope::UIntegerRangeValidator](#).

11.64.2.2 isValid()

```
virtual bool arcs::aubo_scope::InputValidator::isValid (  
    std::string value) [pure virtual]
```

Validates whether the input value is valid.

Parameters

value	to be validated.

Returns

true if value is valid.

Implemented in [arcs::aubo_scope::DoubleRangeValidator](#), [arcs::aubo_scope::IntegerRangeValidator](#), [arcs::aubo_scope::StringLengthValidator](#), and [arcs::aubo_scope::UIntegerRangeValidator](#).

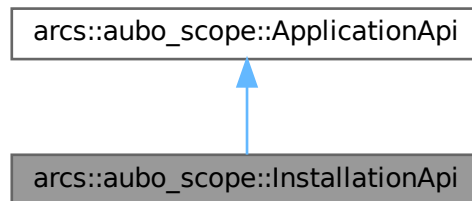
The documentation for this struct was generated from the following file:

- [include/aubo_caps/domain/user_interface/input_validator.h](#)

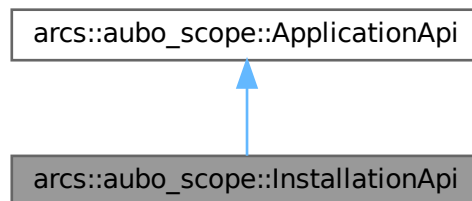
11.65 arcs::aubo_scope::InstallationApi Class Reference

#include <installation_api.h>

Inheritance diagram for arcs::aubo_scope::InstallationApi:



Collaboration diagram for arcs::aubo_scope::InstallationApi:



Public Member Functions

- [InstallationApi](#) (InstallationApi &f)
- [InstallationApi](#) (InstallationApi &&f)
- virtual [~InstallationApi](#) ()
- FunctionModelPtr [getFunctionModel](#) ()
 - Returns an interface for adding script functions to AuboScope.
- TcpContributionModelPtr [getTcpContributionModel](#) ()
 - Returns an interface for contributing TCPs to AuboScope.
- FeatureContributionModelPtr [getFeatureContributionModel](#) ()
 - Returns an interface for contributing features to AuboScope.
- PayloadContributionModelPtr [getPayloadContributionModel](#) ()

Public Member Functions inherited from [arcs::aubo_scope::ApplicationApi](#)

- [ApplicationApi](#) (ApplicationApi &f)
- [ApplicationApi](#) (ApplicationApi &&f)
- virtual [~ApplicationApi](#) ()
- IoModelPtr [getIoModel](#) ()
[IoModel](#) Returns an interface for working with the inputs and outputs of the robot.
- FeatureModelPtr [getFeatureModel](#) ()
[FeatureModel](#) Returns an interface for working with the features of the current installation.
- TcpModelPtr [getTcpModel](#) ()
[TcpModel](#) Returns an interface for accessing the TCPs of the current installation.
- PayloadModelPtr [getPayloadModel](#) ()
[PayloadModel](#) Gets an interface which provides access the payloads in the current AuboScope installation.
- VariableModelPtr [getVariableModel](#) ()
[VariableModel](#) Returns an interface for working with variables.
- ValueFactoryPtr [getValueFactory](#) ()
[ValueFactory](#) Returns an interface for accessing various factories capable of creating value objects.
- template<typename T>
std::shared_ptr< T > [getInstallationNode](#) ()
This method can be used to get a specific [InstallationNodeContribution](#) instance.
- InstallationNodeContributionPtr [getInstallationNode](#) (const char *typeidOfClass)
- DeviceManagerPtr [getDeviceManager](#) ()
[DeviceManager](#) This method can be used to get a specific [DeviceManager](#) instance.
- aubo_sdk::RobotProxyPtr [getRobotProxy](#) ()
Returns the robot proxy, providing an interaction interface with the underlying robot SDK.

Private Member Functions

- [InstallationApi](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Protected Member Functions inherited from [arcs::aubo_scope::ApplicationApi](#)

- [ApplicationApi](#) ()

11.65.1 Detailed Description

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

11.65.2 Constructor & Destructor Documentation

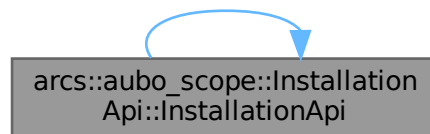
11.65.2.1 InstallationApi() [1/3]

arcs::aubo_scope::InstallationApi::InstallationApi (
InstallationApi & f)

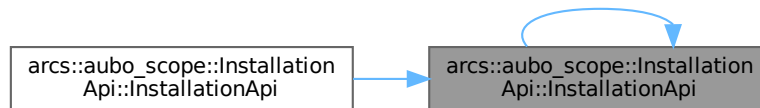
References [InstallationApi\(\)](#).

Referenced by [InstallationApi\(\)](#), and [InstallationApi\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

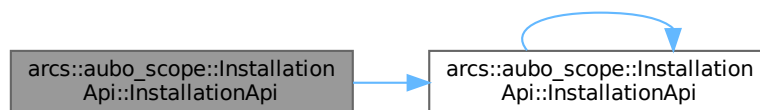


11.65.2.2 InstallationApi() [2/3]

arcs::aubo_scope::InstallationApi::InstallationApi (
InstallationApi && f)

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

Here is the call graph for this function:



11.65.2.3 ~InstallationApi()

virtual arcs::aubo_scope::InstallationApi::~~InstallationApi () [virtual]

11.65.2.4 InstallationApi() [3/3]

arcs::aubo_scope::InstallationApi::InstallationApi () [private]

11.65.3 Friends And Related Symbol Documentation

11.65.3.1 DataSwitch

friend class DataSwitch [friend]

Definition at line 98 of file [installation_api.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.65.4 Member Data Documentation

11.65.4.1 d__

void* arcs::aubo_scope::InstallationApi::d__ { nullptr } [private]

Definition at line 100 of file [installation_api.h](#).

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

- include/aubo_caps/domain/installation/[installation_api.h](#)

11.66 arcs::aubo_scope::InstallationApiProvider Class Reference

[InstallationApiProvider](#) Provides access to functionality and services relevant for installation nodes.

```
#include <installation_api_provider.h>
```

Public Member Functions

- [InstallationApiProvider](#) (InstallationApiProvider &f)
- [InstallationApiProvider](#) (InstallationApiProvider &&f)
- virtual [~InstallationApiProvider](#) ()
- SystemApiPtr [getSystemApi](#) ()
Provides access to system related functionality
- UserInterfaceApiPtr [getUserInterfaceApi](#) ()
Provides access to functionality related to user interface and end user interaction
- InstallationApiPtr [getInstallationApi](#) ()
Provides access to functionality relevant related to the installation

Private Member Functions

- [InstallationApiProvider](#) ()

Private Attributes

- void * [d__](#) { nullptr }

Friends

- class [DataSwitch](#)

11.66.1 Detailed Description

[InstallationApiProvider](#) Provides access to functionality and services relevant for installation nodes.

This includes what is available from within AuboScope, as well as creating additional model elements to be used within AuboScope.

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

11.66.2 Constructor & Destructor Documentation

11.66.2.1 InstallationApiProvider() [1/3]

```
arcs::aubo_scope::InstallationApiProvider::InstallationApiProvider (
    InstallationApiProvider & f)
```

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

Referenced by [InstallationApiProvider\(\)](#), and [InstallationApiProvider\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

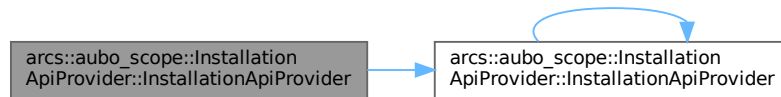


11.66.2.2 InstallationApiProvider() [2/3]

```
arcs::aubo_scope::InstallationApiProvider::InstallationApiProvider (
    InstallationApiProvider && f)
```

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

Here is the call graph for this function:



11.66.2.3 ~InstallationApiProvider()

```
virtual arcs::aubo_scope::InstallationApiProvider::~InstallationApiProvider () [virtual]
```

11.66.2.4 InstallationApiProvider() [3/3]

```
arcs::aubo_scope::InstallationApiProvider::InstallationApiProvider () [private]
```

11.66.3 Member Function Documentation

11.66.3.1 getInstallationApi()

```
InstallationApiPtr arcs::aubo_scope::InstallationApiProvider::getInstallationApi ()
```

Provides access to functionality relevant related to the installation

Returns

an instance of InstallationAPI

11.66.3.2 getSystemApi()

```
SystemApiPtr arcs::aubo_scope::InstallationApiProvider::getSystemApi ()
```

Provides access to system related functionality

Returns

an instance of SystemAPI

11.66.3.3 getUserInterfaceApi()

UIInterfaceApiPtr arcs::aubo_scope::InstallationApiProvider::getUserInterfaceApi ()

Provides access to functionality related to user interface and end user interaction

Returns

an instance of UIInterfaceAPI

11.66.4 Friends And Related Symbol Documentation

11.66.4.1 DataSwitch

friend class DataSwitch [friend]

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

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.66.5 Member Data Documentation

11.66.5.1 d_

void* arcs::aubo_scope::InstallationApiProvider::d_ { nullptr } [private]

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

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

- [include/aubo_caps/contribution/installation/installation_api_provider.h](#)

11.67 arcs::aubo_scope::InstallationContributionConfiguration Class Reference

```
#include <installation_contribution_configuration.h>
```

Public Member Functions

- virtual [~InstallationContributionConfiguration](#) ()=default

11.67.1 Detailed Description

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

11.67.2 Constructor & Destructor Documentation

11.67.2.1 ~InstallationContributionConfiguration()

virtual arcs::aubo_scope::InstallationContributionConfiguration::~~InstallationContributionConfiguration () [virtual], [default]

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

- include/aubo_caps/contribution/installation/[installation_contribution_configuration.h](#)

11.68 arcs::aubo_scope::InstallationCreationContext Class Reference

[InstallationCreationContext](#) Provides information about the context in which an installation node is created.

```
#include <installation_creation_context.h>
```

Public Types

- enum [NodeCreationType](#) { [NEW](#) , [LOAD](#) }

Public Member Functions

- [InstallationCreationContext](#) (InstallationCreationContext &f)
- [InstallationCreationContext](#) (InstallationCreationContext &&f)
- [~InstallationCreationContext](#) ()
- [NodeCreationType](#) [getNodeCreationType](#) ()
- std::string [getCreatedVersion](#) ()
- std::string [getLastSavedVersion](#) ()

Private Member Functions

- [InstallationCreationContext](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

11.68.1 Detailed Description

[InstallationCreationContext](#) Provides information about the context in which an installation node is created.
Generated by Doxygen

Definition at line 22 of file [installation_creation_context.h](#).

Enumerator

NEW	
LOAD	

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

11.68.3 Constructor & Destructor Documentation

11.68.3.1 InstallationCreationContext() [1/3]

```
arcs::aubo_scope::InstallationCreationContext::InstallationCreationContext (
    InstallationCreationContext & f)
```

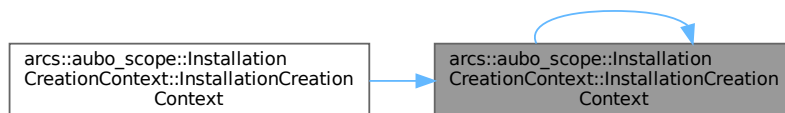
References [InstallationCreationContext\(\)](#).

Referenced by [InstallationCreationContext\(\)](#), and [InstallationCreationContext\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

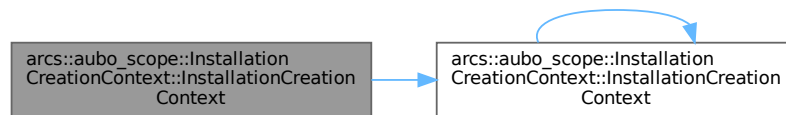


11.68.3.2 InstallationCreationContext() [2/3]

```
arcs::aubo_scope::InstallationCreationContext::InstallationCreationContext (
    InstallationCreationContext && f)
```

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

Here is the call graph for this function:



11.68.3.3 ~InstallationCreationContext()

```
arcs::aubo_scope::InstallationCreationContext::~~InstallationCreationContext ()
```

11.68.3.4 InstallationCreationContext() [3/3]

```
arcs::aubo_scope::InstallationCreationContext::InstallationCreationContext () [private]
```

11.68.4 Member Function Documentation

11.68.4.1 getCreatedVersion()

```
std::string arcs::aubo_scope::InstallationCreationContext::getCreatedVersion ()
```

11.68.4.2 getLastSavedVersion()

```
std::string arcs::aubo_scope::InstallationCreationContext::getLastSavedVersion ()
```

11.68.4.3 getNodeCreationType()

```
NodeCreationType arcs::aubo_scope::InstallationCreationContext::getNodeCreationType ()
```

11.68.5 Friends And Related Symbol Documentation

11.68.5.1 DataSwitch

```
friend class DataSwitch [friend]
```

Definition at line 41 of file [installation_creation_context.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.68.6 Member Data Documentation

11.68.6.1 d_

void* arcs::aubo_scope::InstallationCreationContext::d_ { nullptr } [private]

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

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

- include/aubo_caps/contribution/installation/[installation_creation_context.h](#)

11.69 arcs::aubo_scope::InstallationNodeContribution Class Reference

[InstallationNodeContribution](#) API for installation node contributions.

```
#include <installation_node_contribution.h>
```

Public Member Functions

- virtual [~InstallationNodeContribution](#) ()=default
- virtual void [openView](#) ()=0
Called each time the end user opens this AuboCap contribution in the Installation Tab.
- virtual void [closeView](#) ()=0
Called each time the end user exits this AuboCap contribution in the Installation Tab.
- virtual void [generateScript](#) (ScriptWriterPtr script_writer)=0
Defines script code that is added to the beginning of the script code executed when a program is launched.

11.69.1 Detailed Description

[InstallationNodeContribution](#) API for installation node contributions.

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

11.69.2 Constructor & Destructor Documentation

11.69.2.1 ~InstallationNodeContribution()

virtual arcs::aubo_scope::InstallationNodeContribution::~~InstallationNodeContribution () [virtual], [default]

11.69.3 Member Function Documentation

11.69.3.1 closeView()

virtual void arcs::aubo_scope::InstallationNodeContribution::closeView () [pure virtual]

Generated by Doxygen

Called each time the end user exits this AuboCap contribution in the Installation Tab.

Parameters

script_writer	serves to add script commands to program script code preamble.

11.69.3.3 openView()

virtual void arcs::aubo_scope::InstallationNodeContribution::openView () [pure virtual]

Called each time the end user opens this AuboCap contribution in the Installation Tab.

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

- include/aubo_caps/contribution/installation/[installation_node_contribution.h](#)

11.70 arcs::aubo_scope::InstallationNodeService Class Reference

[InstallationNodeService](#) Defines an API required for defining and adding to AuboScope an installation node and corresponding screen where the user interface is Qt-based.

```
#include <installation_node_service.h>
```

Public Member Functions

- virtual [~InstallationNodeService](#) ()=default
- virtual std::string [getTitle](#) ()=0
- virtual std::string [getIcon](#) ()=0
- virtual void [configureContribution](#) (InstallationContributionConfigurationPtr configuration)=0
This method is called once after this service is registered.
- virtual InstallationNodeViewPtr [createView](#) (ViewApiProviderPtr api_provider)=0
Creates a new View instance which implements the UI for your installation node screen.
- virtual InstallationNodeContributionPtr [createInstallationNode](#) (InstallationApiProviderPtr api_provider, InstallationNodeViewPtr view, DataModelPtr model, InstallationCreationContextPtr context)=0

11.70.1 Detailed Description

[InstallationNodeService](#) Defines an API required for defining and adding to AuboScope an installation node and corresponding screen where the user interface is Qt-based.

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

11.70.2 Constructor & Destructor Documentation

11.70.2.1 ~InstallationNodeService()

virtual arcs::aubo_scope::InstallationNodeService::~InstallationNodeService () [virtual], [default]

11.70.3 Member Function Documentation

11.70.3.1 configureContribution()

```
virtual void arcs::aubo_scope::InstallationNodeService::configureContribution (
    InstallationContributionConfigurationPtr configuration) [pure virtual]
```

This method is called once after this service is registered.

Modify the configuration parameter to configure your contribution. The configuration object will already have default values for its properties matching most use cases.

The values of the ContributionConfiguration object will be read once immediately after this method call. Changing values at a later stage will have no effect, so do not store a reference to the configuration object.

If the default values are appropriate, leave this method empty.

Parameters

configuration	a modifiable ContributionConfiguration with default values

11.70.3.2 createInstallationNode()

```
virtual InstallationNodeContributionPtr arcs::aubo_scope::InstallationNodeService::createInstallationNode (
    InstallationApiProviderPtr api_provider,
    InstallationNodeViewPtr view,
    DataModelPtr model,
    InstallationContextPtr context) [pure virtual]
```

Creates a new installation node instance.

The returned node must use the supplied data model object to retrieve and store the data contained in it. All modifications to the supplied data model from the installation node constructor are ignored when an existing installation is loaded. The data model object is shared between all installation nodes contributed by the same AuboCap.

Parameters

api_provider	Provides access to functionality and services available from within AuboScope relevant for the installation node
view	the View created by createView(ViewAPIProvider)
model	object where all configuration data of the new installation node instance is to be stored in and retrieved from
context	the context in which this node is created

Returns

the newly constructed installation node contribution instance

11.70.3.3 createView()

```
virtual InstallationNodeViewPtr arcs::aubo_scope::InstallationNodeService::createView (  
    ViewApiProviderPtr api_provider) [pure virtual]
```

Creates a new View instance which implements the UI for your installation node screen.

Called once when a new installation is loaded or created.

Parameters

api_provider	Provides access to functionality and services available from within AboScope related to user interface and end user interaction

Returns

the view

11.70.3.4 getIcon()

```
virtual std::string arcs::aubo_scope::InstallationNodeService::getIcon () [pure virtual]
```

Returns

The icon displayed for this installation contribution. Called once at start up.

11.70.3.5 getTitle()

```
virtual std::string arcs::aubo_scope::InstallationNodeService::getTitle () [pure virtual]
```

Parameters

locale	the current locale of AboScope. Can be used for supporting titles in several languages.

Returns

The text displayed for this installation contribution in the left-hand side navigation of the Installation Tab as well as the title of the corresponding installation node screen. Called once at start up.

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

- include/aubo_caps/contribution/installation/[installation_node_service.h](#)

11.71 arcs::aubo_scope::InstallationNodeView Class Reference

[InstallationNodeView](#) API for an installation node View which defines the UI for the corresponding [InstallationNodeContribution](#)

```
#include <installation_node_view.h>
```

Public Member Functions

- virtual [~InstallationNodeView](#) ()=default
- virtual void [buildUi](#) ([WidgetHandle](#) panel, [InstallationNodeContributionPtr](#) contribution)=0
Build the UI for the corresponding [InstallationNodeContribution](#) on the provided panel.

11.71.1 Detailed Description

[InstallationNodeView](#) API for an installation node View which defines the UI for the corresponding [InstallationNodeContribution](#)

Parameters

<C>	the (generic) type parameter for the interface representing the type of the InstallationNodeContribution for which this view implements the UI

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

11.71.2 Constructor & Destructor Documentation

11.71.2.1 ~InstallationNodeView()

```
virtual arcs::aubo_scope::InstallationNodeView::~~InstallationNodeView () [virtual], [default]
```

11.71.3 Member Function Documentation

11.71.3.1 buildUi()

```
virtual void arcs::aubo_scope::InstallationNodeView::buildUi (  
    WidgetHandle panel,  
    InstallationNodeContributionPtr contribution) [pure virtual]
```

Build the UI for the corresponding [InstallationNodeContribution](#) on the provided panel.

Use the contribution for your logic and data model manipulations.

Parameters

panel	the panel to build the UI on
contribution	the corresponding installation node contribution

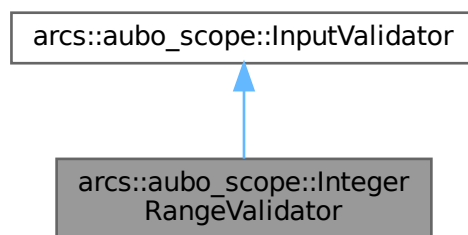
The documentation for this class was generated from the following file:

- [include/aubo_caps/contribution/installation/installation_node_view.h](#)

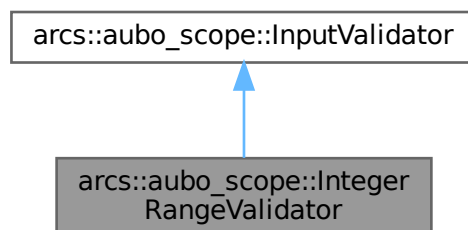
11.72 arcs::aubo_scope::IntegerRangeValidator Struct Reference

```
#include <input_validator.h>
```

Inheritance diagram for arcs::aubo_scope::IntegerRangeValidator:



Collaboration diagram for arcs::aubo_scope::IntegerRangeValidator:



Public Member Functions

- [IntegerRangeValidator](#) (int min_value, int max_value)
- [IntegerRangeValidator](#) (std::function< std::pair< int, int >()> range_func)
- bool [isValid](#) (std::string value) override
Validates whether the input value is valid.
- std::string [getMessage](#) (std::string value) override
Returns a meaningful message in case the value is not valid.

Private Member Functions

- void [updateValue](#) ()

Private Attributes

- int [min_value_](#) { INT_MIN }
- int [max_value_](#) { INT_MAX }
- std::function< std::pair< int, int >()> [range_func_](#) { nullptr }

11.72.1 Detailed Description

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

11.72.2 Constructor & Destructor Documentation

11.72.2.1 IntegerRangeValidator() [1/2]

```
arcs::aubo_scope::IntegerRangeValidator::IntegerRangeValidator (  
    int min_value,  
    int max_value)
```

11.72.2.2 IntegerRangeValidator() [2/2]

```
arcs::aubo_scope::IntegerRangeValidator::IntegerRangeValidator (  
    std::function< std::pair< int, int >()> range_func)
```

11.72.3 Member Function Documentation

11.72.3.1 getMessage()

```
std::string arcs::aubo_scope::IntegerRangeValidator::getMessage (  
    std::string value) [override], [virtual]
```

Returns a meaningful message in case the value is not valid.

Parameters

value	the invalid value. Can be included in the message if it makes sense.

Returns

message.

Implements [arcs::aubo_scope::InputValidator](#).

11.72.3.2 isValid()

```
bool arcs::aubo_scope::IntegerRangeValidator::isValid (
    std::string value)  [override], [virtual]
```

Validates whether the input value is valid.

Parameters

value	to be validated.

Returns

true if value is valid.

Implements [arcs::aubo_scope::InputValidator](#).

11.72.3.3 updateValue()

```
void arcs::aubo_scope::IntegerRangeValidator::updateValue ()  [private]
```

11.72.4 Member Data Documentation

11.72.4.1 max_value__

```
int arcs::aubo_scope::IntegerRangeValidator::max_value__ { INT_MAX }  [private]
```

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

11.72.4.2 min_value__

```
int arcs::aubo_scope::IntegerRangeValidator::min_value__ { INT_MIN }  [private]
```

Definition at line 91 of file [input_validator.h](#).

11.72.4.3 range_func_

`std::function<std::pair<int, int>()> arcs::aubo_scope::IntegerRangeValidator::range_func_ { nullptr } [private]`

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

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

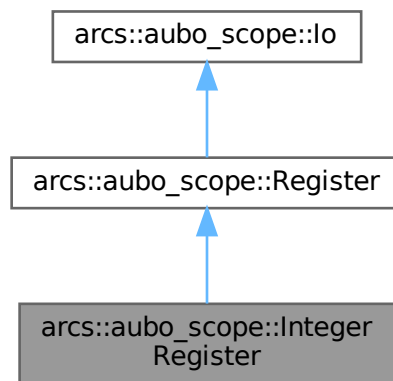
- `include/aubo_caps/domain/user_interface/input_validator.h`

11.73 arcs::aubo_scope::IntegerRegister Class Reference

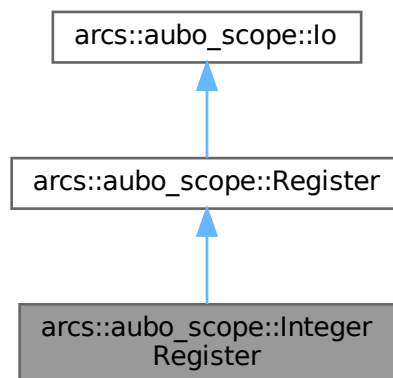
This interface provides support for integer registers.

`#include <integer_register.h>`

Inheritance diagram for `arcs::aubo_scope::IntegerRegister`:



Collaboration diagram for `arcs::aubo_scope::IntegerRegister`:



Public Member Functions

- virtual [~IntegerRegister](#) ()=default

Public Member Functions inherited from [arcs::aubo_scope::Register](#)

- virtual [~Register](#) ()=default

Public Member Functions inherited from [arcs::aubo_scope::Io](#)

- [Io](#) (Io &f)
- [Io](#) (Io &&f)
- virtual [~Io](#) ()
- std::string [getDisplay_name](#) () const
Returns the user defined name of the IO in the currently selected installation.
- std::string [getDefaultName](#) () const
Returns the robot default name of the I/O.
- std::string [getValueStr](#) () const
Returns the current reading of the I/O as a string.
- [IoType](#) [getType](#) () const
Returns the I/O type.
- [InterfaceType](#) [getInterfaceType](#) () const
Returns the interface type this I/O belongs to.
- int [getAction](#) () const
Returns the action number of the I/O.
- bool [isInput](#) () const
Returns true if I/O supports reading of values and false otherwise.
- bool [isResolvable](#) () const
Some I/Os, such as MODBUS I/Os and analog Tool inputs, are in some situations not present in AuboScope or available for use.
- size_t [hashCode](#) ()

Additional Inherited Members

Public Types inherited from [arcs::aubo_scope::Io](#)

- enum [IoType](#) : int {
 [DIGITAL](#) , [ANALOG](#) , [INTEGER](#) , [BOOLEAN](#) ,
 [DOUBLE](#) }
- enum [InterfaceType](#) : int {
 [STANDARD](#) , [TOOL](#) , [CONFIGURABLE](#) , [LINKIO](#) ,
 [MODBUS](#) , [GENERAL_PURPOSE](#) , [PROFINET](#) , [ETHERNET_IP](#) }

Protected Member Functions inherited from [arcs::aubo_scope::Io](#)

- [Io](#) ()

11.73.1 Detailed Description

This interface provides support for integer registers.

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

11.73.2 Constructor & Destructor Documentation

11.73.2.1 ~IntegerRegister()

virtual arcs::aubo_scope::IntegerRegister::~IntegerRegister () [virtual], [default]

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

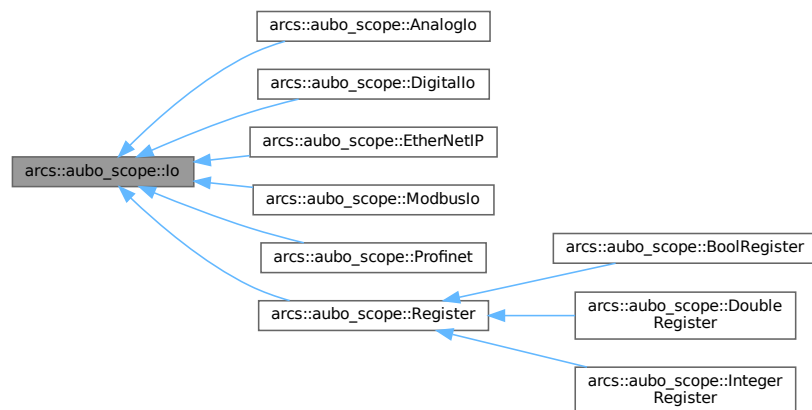
- [include/aubo_caps/domain/io/integer_register.h](#)

11.74 arcs::aubo_scope::Io Class Reference

[Io](#) This is the base interface representing all types of I/Os available in AuboScope.

```
#include <io.h>
```

Inheritance diagram for arcs::aubo_scope::Io:



Public Types

- enum [IoType](#) : int {
DIGITAL , ANALOG , INTEGER , BOOLEAN ,
DOUBLE }
- enum [InterfaceType](#) : int {
STANDARD , TOOL , CONFIGURABLE , LINKIO ,
MODBUS , GENERAL_PURPOSE , PROFINET , ETHERNET_IP }

Public Member Functions

- [Io](#) (Io &f)
- [Io](#) (Io &&f)
- virtual [~Io](#) ()
- std::string [getDisplay_name](#) () const
Returns the user defined name of the IO in the currently selected installation.
- std::string [getDefaultName](#) () const
Returns the robot default name of the I/O.
- std::string [getValueStr](#) () const
Returns the current reading of the I/O as a string.
- [IoType](#) [getType](#) () const
Returns the I/O type.
- [InterfaceType](#) [getInterfaceType](#) () const
Returns the interface type this I/O belongs to.
- int [getAction](#) () const
Returns the action number of the I/O.
- bool [isInput](#) () const
Returns true if I/O supports reading of values and false otherwise.
- bool [isResolvable](#) () const
Some I/Os, such as MODBUS I/Os and analog Tool inputs, are in some situations not present in AuboScope or available for use.
- size_t [hashCode](#) ()

Protected Member Functions

- [Io](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

11.74.1 Detailed Description

[Io](#) This is the base interface representing all types of I/Os available in AuboScope.

Definition at line 22 of file [io.h](#).

11.74.2 Member Enumeration Documentation

11.74.2.1 InterfaceType

```
enum arcs::aubo_scope::Io::InterfaceType : int
```

Enumerator

STANDARD	
TOOL	
CONFIGURABLE	
LINKIO	
MODBUS	
GENERAL_PURPOSE	
PROFINET	
ETHERNET_IP	

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

11.74.2.2 IoType

```
enum arcs::aubo\_scope::Io::IoType : int
```

Enumerator

DIGITAL	
ANALOG	
INTEGER	
BOOLEAN	
DOUBLE	

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

11.74.3 Constructor & Destructor Documentation

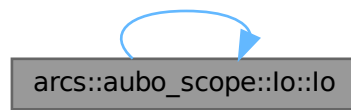
11.74.3.1 Io() [1/3]

```
arcs::aubo\_scope::Io::Io (  
    Io & f)
```

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

Referenced by [Io\(\)](#), and [Io\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



11.74.3.2 Io() [2/3]

```
arcs::aubo_scope::Io::Io (
    Io && f)
```

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

Here is the call graph for this function:



11.74.3.3 ~Io()

```
virtual arcs::aubo_scope::Io::~Io () [virtual]
```

11.74.3.4 Io() [3/3]

arcs::aubo_scope::Io::Io () [protected]

11.74.4 Member Function Documentation

11.74.4.1 getAction()

int arcs::aubo_scope::Io::getAction () const

Returns the action number of the I/O.

Returns

the action number of the I/O.

11.74.4.2 getDefaultName()

std::string arcs::aubo_scope::Io::getDefaultName () const

Returns the robot default name of the I/O.

Returns

the robot default name of the I/O.

11.74.4.3 getDisplayName()

std::string arcs::aubo_scope::Io::getDisplayName () const

Returns the user defined name of the IO in the currently selected installation.

Returns

the user defined name of the IO.

11.74.4.4 getInterfaceType()

[InterfaceType](#) arcs::aubo_scope::Io::getInterfaceType () const

Returns the interface type this I/O belongs to.

Returns

the interface type this I/O belongs to.

11.74.4.5 getType()

`IoType arcs::aubo_scope::Io::getType () const`

Returns the I/O type.

Returns

the I/O type.

11.74.4.6 getValueStr()

`std::string arcs::aubo_scope::Io::getValueStr () const`

Returns the current reading of the I/O as a string.

Returns

get current reading of the I/O as a string.

11.74.4.7 hashCode()

`size_t arcs::aubo_scope::Io::hashCode ()`

11.74.4.8 isInput()

`bool arcs::aubo_scope::Io::isInput () const`

Returns true if I/O supports reading of values and false otherwise.

Returns

true if I/O supports reading of values.

11.74.4.9 isResolvable()

`bool arcs::aubo_scope::Io::isResolvable () const`

Some I/Os, such as MODBUS I/Os and analog Tool inputs, are in some situations not present in Aubo↔ Scope or available for use.

This method can be used to determine if the I/O is available.

Returns

true, if I/O can be resolved. false otherwise.

11.74.5 Friends And Related Symbol Documentation

11.74.5.1 DataSwitch

friend class DataSwitch [friend]

Definition at line 158 of file [io.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.74.6 Member Data Documentation

11.74.6.1 d__

void* arcs::aubo_scope::Io::d__ { nullptr } [private]

Definition at line 159 of file [io.h](#).

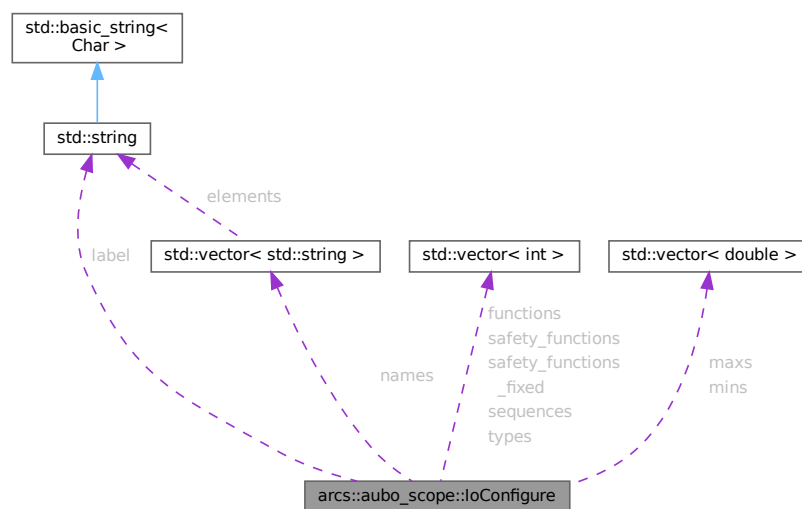
The documentation for this class was generated from the following file:

- [include/aubo_caps/domain/io/io.h](#)

11.75 arcs::aubo_scope::IoConfigure Struct Reference

```
#include <system_api.h>
```

Collaboration diagram for arcs::aubo_scope::IoConfigure:



Public Attributes

- std::string [label](#)
- int [num](#)
- std::vector< std::string > [names](#)
- std::vector< int > [sequences](#)
- std::vector< int > [functions](#)
- std::vector< int > [safety_functions](#)
- std::vector< int > [safety_functions_fixed](#)
- bool [function_lock](#) { false }
- std::vector< int > [types](#)
- std::vector< double > [mins](#)
- std::vector< double > [maxs](#)

11.75.1 Detailed Description

Definition at line [18](#) of file [system_api.h](#).

11.75.2 Member Data Documentation

11.75.2.1 function_lock

```
bool arcs::aubo_scope::IoConfigure::function_lock { false }
```

Definition at line [27](#) of file [system_api.h](#).

11.75.2.2 functions

```
std::vector<int> arcs::aubo_scope::IoConfigure::functions
```

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

11.75.2.3 label

```
std::string arcs::aubo_scope::IoConfigure::label
```

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

11.75.2.4 maxs

```
std::vector<double> arcs::aubo_scope::IoConfigure::maxs
```

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

11.75.2.5 mins

```
std::vector<double> arcs::aubo_scope::IoConfigure::mins
```

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

11.75.2.6 names

`std::vector<std::string> arcs::aubo_scope::IoConfigure::names`

Definition at line 22 of file [system_api.h](#).

11.75.2.7 num

`int arcs::aubo_scope::IoConfigure::num`

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

11.75.2.8 safety_functions

`std::vector<int> arcs::aubo_scope::IoConfigure::safety_functions`

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

11.75.2.9 safety_functions_fixed

`std::vector<int> arcs::aubo_scope::IoConfigure::safety_functions_fixed`

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

11.75.2.10 sequences

`std::vector<int> arcs::aubo_scope::IoConfigure::sequences`

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

11.75.2.11 types

`std::vector<int> arcs::aubo_scope::IoConfigure::types`

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

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

- [include/aubo_caps/domain/system_api.h](#)

11.76 arcs::aubo_scope::IoModel Class Reference

[IoModel](#) Provides methods that returns I/Os from the current robot installation.

```
#include <io_model.h>
```

Public Member Functions

- [IoModel](#) (IoModel &f)
- [IoModel](#) (IoModel &&f)
- [~IoModel](#) ()
- `std::vector< IoPtr > getAll ()`
Returns the collection of available I/Os, including Modbus, EuroMap67 and named General Purpose Registers.
- `IoPtr getIo (const std::string &name)`
[Io](#) Get an I/O by name.
- `std::vector< IoPtr > getIos (std::function< bool(IoPtr)> filter)`
Get a subset of I/Os using a filter.

Private Member Functions

- [IoModel](#) ()

Private Attributes

- `void * d\_ { nullptr }`

Friends

- class [DataSwitch](#)

11.76.1 Detailed Description

[IoModel](#) Provides methods that returns I/Os from the current robot installation.

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

11.76.2 Constructor & Destructor Documentation

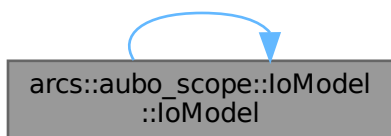
11.76.2.1 [IoModel\(\)](#) [1/3]

```
arcs::aubo_scope::IoModel::IoModel (
    IoModel & f)
```

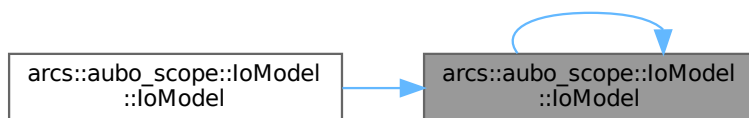
References [IoModel\(\)](#).

Referenced by [IoModel\(\)](#), and [IoModel\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

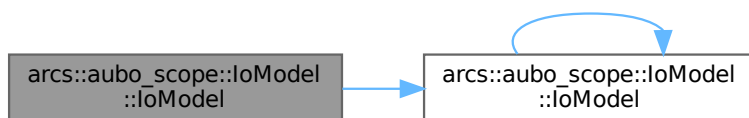


11.76.2.2 IoModel() [2/3]

```
arcs::aubo_scope::IoModel::IoModel (
    IoModel && f)
```

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

Here is the call graph for this function:



11.76.2.3 ~IoModel()

```
arcs::aubo_scope::IoModel::~IoModel ()
```

11.76.2.4 IoModel() [3/3]

arcs::aubo_scope::IoModel::IoModel () [private]

11.76.3 Member Function Documentation

11.76.3.1 getAll()

std::vector< IoPtr > arcs::aubo_scope::IoModel::getAll ()

Returns the collection of available I/Os, including Modbus, EuroMap67 and named General Purpose Registers.

Returns

the collection of available I/Os.

11.76.3.2 getIo()

IoPtr arcs::aubo_scope::IoModel::getIo (
const std::string & name)

[Io](#) Get an I/O by name.

Parameters

name	the name of the I/O.

Returns

the I/O with the given name.

11.76.3.3 getIos()

std::vector< IoPtr > arcs::aubo_scope::IoModel::getIos (
std::function< bool(IoPtr)> filter)

Get a subset of I/Os using a filter.

Parameters

filter	see IOFilterFactory .

Returns

the collection of IOs that are accepted by the filter.

11.76.4 Friends And Related Symbol Documentation

11.76.4.1 DataSwitch

friend class DataSwitch [friend]

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

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.76.5 Member Data Documentation

11.76.5.1 d_

void* arcs::aubo_scope::IoModel::d_ { nullptr } [private]

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

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

- [include/aubo_caps/domain/io/io_model.h](#)

11.77 arcs::aubo_scope::KeyboardManager Class Reference

```
#include <keyboard_manager.h>
```


Public Member Functions

- [KeyboardManager](#) (KeyboardManager &f)
- [KeyboardManager](#) (KeyboardManager &&f)
- virtual [~KeyboardManager](#) ()
- ARCS_DEPRECATED void [registerStandardKeyboard](#) (QWidget *wd, std::function< std::string(void)> init_cb=nullptr, const std::shared_ptr< [InputValidator](#) > &validator=nullptr, std::function< void(std::string)> result_cb=nullptr)
 - [registerStandardKeyboard](#)
- void [registerStandardKeyboard](#) (QWidget *wd, std::function< ExpressionPtr(void)> init_cb, std::function< void(ExpressionPtr)> result_cb, const std::shared_ptr< [InputValidator](#) > &validator=nullptr)
 - [registerStandardKeyboard](#)
- ARCS_DEPRECATED void [registerStringKeyboard](#) (QWidget *wd, std::function< std::string(void)> init_cb=nullptr, const std::shared_ptr< [InputValidator](#) > &validator=nullptr, std::function< void(std::string)> result_cb=nullptr)
- void [registerStringKeyboard](#) (QWidget *wd, std::function< ExpressionPtr(void)> init_cb, std::function< void(ExpressionPtr)> result_cb, const std::shared_ptr< [InputValidator](#) > &validator=nullptr)
- ARCS_DEPRECATED void [registerPasswordKeyboard](#) (QWidget *wd, std::function< std::string(void)> init_cb=nullptr, const std::shared_ptr< [InputValidator](#) > &validator=nullptr, std::function< void(std::string)> result_cb=nullptr)
- void [registerPasswordKeyboard](#) (QWidget *wd, std::function< ExpressionPtr(void)> init_cb, std::function< void(ExpressionPtr)> result_cb, const std::shared_ptr< [InputValidator](#) > &validator=nullptr)
- ARCS_DEPRECATED void [registerCodeEditorKeyboard](#) (QWidget *wd, std::function< std::string(void)> init_cb=nullptr, const std::shared_ptr< [InputValidator](#) > &validator=nullptr, std::function< void(std::string)> result_cb=nullptr)
- void [registerCodeEditorKeyboard](#) (QWidget *wd, std::function< ExpressionPtr(void)> init_cb, std::function< void(ExpressionPtr)> result_cb, const std::shared_ptr< [InputValidator](#) > &validator=nullptr)
- void [registerNumberKeyboard](#) (QWidget *wd, std::function< std::string(void)> init_cb=nullptr, const std::shared_ptr< [InputValidator](#) > &validator=nullptr, std::function< void(std::string)> result_cb=nullptr)
- void [registerDoubleKeyboard](#) (QWidget *wd, std::function< std::string(void)> init_cb=nullptr, const std::shared_ptr< [InputValidator](#) > &validator=nullptr, std::function< void(std::string)> result_cb=nullptr)
- void [registerPositiveDoubleKeyboard](#) (QWidget *wd, std::function< std::string(void)> init_cb=nullptr, const std::shared_ptr< [InputValidator](#) > &validator=nullptr, std::function< void(std::string)> result_cb=nullptr)
- void [registerIntegerKeyboard](#) (QWidget *wd, std::function< std::string(void)> init_cb=nullptr, const std::shared_ptr< [InputValidator](#) > &validator=nullptr, std::function< void(std::string)> result_cb=nullptr)
- void [registerPositiveIntegerKeyboard](#) (QWidget *wd, std::function< std::string(void)> init_cb=nullptr, const std::shared_ptr< [InputValidator](#) > &validator=nullptr, std::function< void(std::string)> result_cb=nullptr)
- void [registerIPAddressKeyboard](#) (QWidget *wd, std::function< std::string(void)> init_cb=nullptr, std::function< void(std::string)> result_cb=nullptr)
- void [registerExpressionKeyboard](#) (QWidget *wd, std::function< ExpressionPtr(void)> init_cb=nullptr, std::function< void(ExpressionPtr result_value)> result_cb=nullptr, std::shared_ptr< [InputValidator](#) > validator=nullptr)

Private Member Functions

- [KeyboardManager](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

11.77.1 Detailed Description

Definition at line 18 of file [keyboard_manager.h](#).

11.77.2 Constructor & Destructor Documentation

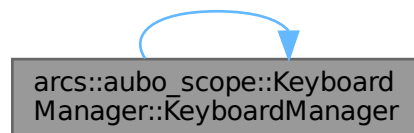
11.77.2.1 KeyboardManager() [1/3]

```
arcs::aubo_scope::KeyboardManager::KeyboardManager (
    KeyboardManager & f)
```

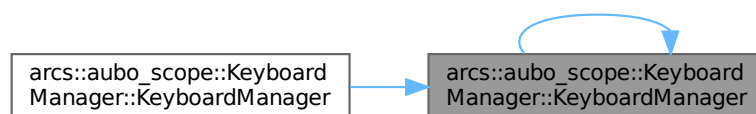
References [KeyboardManager\(\)](#).

Referenced by [KeyboardManager\(\)](#), and [KeyboardManager\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

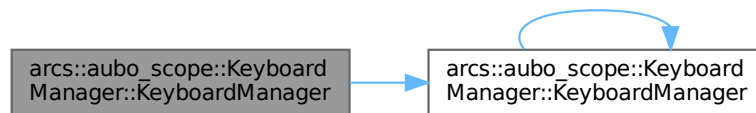


11.77.2.2 KeyboardManager() [2/3]

```
arcs::aubo_scope::KeyboardManager::KeyboardManager (
    KeyboardManager && f)
```

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

Here is the call graph for this function:



11.77.2.3 ~KeyboardManager()

```
virtual arcs::aubo_scope::KeyboardManager::~~KeyboardManager () [virtual]
```

11.77.2.4 KeyboardManager() [3/3]

```
arcs::aubo_scope::KeyboardManager::KeyboardManager () [private]
```

11.77.3 Member Function Documentation

11.77.3.1 registerCodeEditorKeyboard() [1/2]

```
void arcs::aubo_scope::KeyboardManager::registerCodeEditorKeyboard (
    QWidget * wd,
    std::function< ExpressionPtr(void)> init_cb,
    std::function< void(ExpressionPtr)> result_cb,
    const std::shared_ptr< InputValidator > & validator = nullptr)
```

11.77.3.2 registerCodeEditorKeyboard() [2/2]

```
ARCS_DEPRECATED void arcs::aubo_scope::KeyboardManager::registerCodeEditorKeyboard (
    QWidget * wd,
    std::function< std::string(void)> init_cb = nullptr,
    const std::shared_ptr< InputValidator > & validator = nullptr,
    std::function< void(std::string)> result_cb = nullptr)
```

11.77.3.3 registerDoubleKeyboard()

```
void arcs::aubo_scope::KeyboardManager::registerDoubleKeyboard (
    QWidget * wd,
    std::function< std::string(void)> init_cb = nullptr,
    const std::shared_ptr< InputValidator > & validator = nullptr,
    std::function< void(std::string)> result_cb = nullptr)
```

11.77.3.4 registerExpressionKeyboard()

```
void arcs::aubo_scope::KeyboardManager::registerExpressionKeyboard (
    QWidget * wd,
    std::function< ExpressionPtr(void)> init_cb = nullptr,
    std::function< void(ExpressionPtr result_value)> result_cb = nullptr,
    std::shared_ptr< InputValidator > validator = nullptr)
```

11.77.3.5 registerIntegerKeyboard()

```
void arcs::aubo_scope::KeyboardManager::registerIntegerKeyboard (
    QWidget * wd,
    std::function< std::string(void)> init_cb = nullptr,
    const std::shared_ptr< InputValidator > & validator = nullptr,
    std::function< void(std::string)> result_cb = nullptr)
```

11.77.3.6 registerIPAddressKeyboard()

```
void arcs::aubo_scope::KeyboardManager::registerIPAddressKeyboard (
    QWidget * wd,
    std::function< std::string(void)> init_cb = nullptr,
    std::function< void(std::string)> result_cb = nullptr)
```

11.77.3.7 registerNumberKeyboard()

```
void arcs::aubo_scope::KeyboardManager::registerNumberKeyboard (
    QWidget * wd,
    std::function< std::string(void)> init_cb = nullptr,
    const std::shared_ptr< InputValidator > & validator = nullptr,
    std::function< void(std::string)> result_cb = nullptr)
```

11.77.3.8 registerPasswordKeyboard() [1/2]

```
void arcs::aubo_scope::KeyboardManager::registerPasswordKeyboard (
    QWidget * wd,
    std::function< ExpressionPtr(void)> init_cb,
    std::function< void(ExpressionPtr)> result_cb,
    const std::shared_ptr< InputValidator > & validator = nullptr)
```

11.77.3.9 registerPasswordKeyboard() [2/2]

```
ARCS_DEPRECATED void arcs::aubo_scope::KeyboardManager::registerPasswordKeyboard (
    QWidget * wd,
    std::function< std::string(void)> init_cb = nullptr,
    const std::shared_ptr< InputValidator > & validator = nullptr,
    std::function< void(std::string)> result_cb = nullptr)
```

11.77.3.10 registerPositiveDoubleKeyboard()

```
void arcs::aubo_scope::KeyboardManager::registerPositiveDoubleKeyboard (
    QWidget * wd,
    std::function< std::string(void)> init_cb = nullptr,
    const std::shared_ptr< InputValidator > & validator = nullptr,
    std::function< void(std::string)> result_cb = nullptr)
```

11.77.3.11 registerPositiveIntegerKeyboard()

```
void arcs::aubo_scope::KeyboardManager::registerPositiveIntegerKeyboard (
    QWidget * wd,
    std::function< std::string(void)> init_cb = nullptr,
    const std::shared_ptr< InputValidator > & validator = nullptr,
    std::function< void(std::string)> result_cb = nullptr)
```

11.77.3.12 registerStandardKeyboard() [1/2]

```
void arcs::aubo_scope::KeyboardManager::registerStandardKeyboard (
    QWidget * wd,
    std::function< ExpressionPtr(void)> init_cb,
    std::function< void(ExpressionPtr)> result_cb,
    const std::shared_ptr< InputValidator > & validator = nullptr)
```

registerStandardKeyboard

Parameters

wd	ui->lineedit
init_cb	
result_cb	Expression string
validator	

11.77.3.13 registerStandardKeyboard() [2/2]

```
ARCS_DEPRECATED void arcs::aubo_scope::KeyboardManager::registerStandardKeyboard (
```

```
    QWidget * wd,
    std::function< std::string(void)> init_cb = nullptr,
    const std::shared_ptr< InputValidator > & validator = nullptr,
    std::function< void(std::string)> result_cb = nullptr)
```

Parameters

wd	ui->lineedit
init_cb	edit
validator	
result_cb	edit

11.77.3.14 registerStringKeyboard() [1/2]

```
void arcs::aubo_scope::KeyboardManager::registerStringKeyboard (
    QWidget * wd,
    std::function< ExpressionPtr(void)> init_cb,
    std::function< void(ExpressionPtr)> result_cb,
    const std::shared_ptr< InputValidator > & validator = nullptr)
```

11.77.3.15 registerStringKeyboard() [2/2]

```
ARCS_DEPRECATED void arcs::aubo_scope::KeyboardManager::registerStringKeyboard (
    QWidget * wd,
    std::function< std::string(void)> init_cb = nullptr,
    const std::shared_ptr< InputValidator > & validator = nullptr,
    std::function< void(std::string)> result_cb = nullptr)
```

11.77.4 Friends And Related Symbol Documentation

11.77.4.1 DataSwitch

friend class DataSwitch [friend]

Definition at line 109 of file [keyboard_manager.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.77.5 Member Data Documentation

11.77.5.1 d_

```
void* arcs::aubo_scope::KeyboardManager::d_ { nullptr } [private]
```

Definition at line 111 of file [keyboard_manager.h](#).

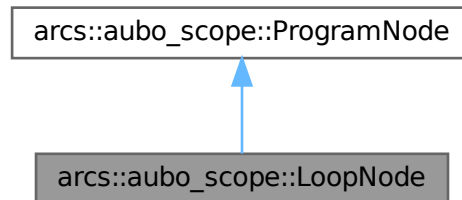
The documentation for this class was generated from the following file:

- [include/aubo_caps/domain/user_interface/keyboard_manager.h](#)

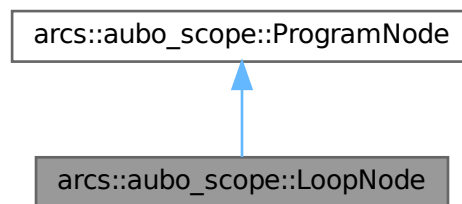
11.78 arcs::aubo_scope::LoopNode Class Reference

#include <loop_node.h>

Inheritance diagram for arcs::aubo_scope::LoopNode:



Collaboration diagram for arcs::aubo_scope::LoopNode:



Public Types

- enum [ConfigType](#) { [ALWAYS](#) , [COUNTER](#) , [EXPRESSION](#) }

The configuration type used to determine which type of configuration this instance is.

Public Member Functions

- [LoopNode](#) (LoopNode &f)
- [LoopNode](#) (LoopNode &&f)
- virtual [~LoopNode](#) ()
- [ConfigType](#) [getConfigType](#) ()
This method returns the type of configuration.
- void [setConfigType](#) ([ConfigType](#) type)
- void [setLoopCount](#) (int loop_count)
Creates a loop counter configuration.
- int [getLoopCount](#) ()

- void [setExpression](#) (ExpressionPtr expression)
Create a loop expression configuration.
- ExpressionPtr [getExpression](#) ()
- bool [isEvaluateContinously](#) ()
- void [setEvaluateContinously](#) (bool cont)
- void [setLoopCountVariable](#) (VariablePtr var)
- VariablePtr [getLoopCountVariable](#) ()

Public Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()
Get the program node contribution.

Private Member Functions

- [LoopNode](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Protected Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.78.1 Detailed Description

Definition at line 12 of file [loop_node.h](#).

11.78.2 Member Enumeration Documentation

11.78.2.1 ConfigType

enum [arcs::aubo_scope::LoopNode::ConfigType](#)

The configuration type used to determine which type of configuration this instance is.

Enumerator

ALWAYS	Looping infinitely has been selected. The config instance can be cast to AlwaysLoopNodeConfig
COUNTER	Looping a certain number of times has been selected. The config instance can be cast to CounterLoopNodeConfig
EXPRESSION	Looping as long as the evaluation of an expression is true has been selected. The config instance can be cast to ExpressionLoopNodeConfig

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

11.78.3 Constructor & Destructor Documentation

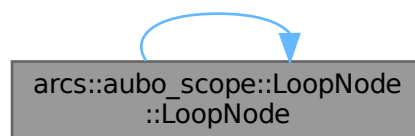
11.78.3.1 LoopNode() [1/3]

```
arcs::aubo_scope::LoopNode::LoopNode (
    LoopNode & f)
```

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

Referenced by [LoopNode\(\)](#), and [LoopNode\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

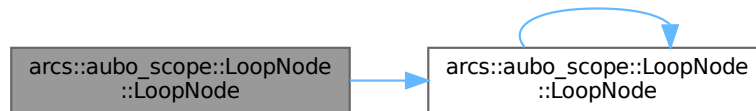


11.78.3.2 LoopNode() [2/3]

```
arcs::aubo_scope::LoopNode::LoopNode (
    LoopNode && f)
```

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

Here is the call graph for this function:



11.78.3.3 ~LoopNode()

```
virtual arcs::aubo_scope::LoopNode::~LoopNode () [virtual]
```

11.78.3.4 LoopNode() [3/3]

```
arcs::aubo_scope::LoopNode::LoopNode () [private]
```

11.78.4 Member Function Documentation

11.78.4.1 getConfigType()

```
ConfigType arcs::aubo_scope::LoopNode::getConfigType ()
```

This method returns the type of configuration.

Cast this instance appropriately to have access to specific getters.

Returns

the type of this config.

11.78.4.2 getExpression()

```
ExpressionPtr arcs::aubo_scope::LoopNode::getExpression ()
```

11.78.4.3 getLoopCount()

```
int arcs::aubo_scope::LoopNode::getLoopCount ()
```

11.78.4.4 getLoopCountVariable()

VariablePtr arcs::aubo_scope::LoopNode::getLoopCountVariable ()

11.78.4.5 isEvaluateContinuously()

bool arcs::aubo_scope::LoopNode::isEvaluateContinuously ()

Returns

true if the expression will be evaluated continuously during the looping (i.e. the loop can be interrupted anytime during its execution), otherwise false and the expression will be evaluated once per loop iteration.

11.78.4.6 setConfigType()

void arcs::aubo_scope::LoopNode::setConfigType (
 ConfigType type)

11.78.4.7 setEvaluateContinuously()

void arcs::aubo_scope::LoopNode::setEvaluateContinuously (
 bool cont)

11.78.4.8 setExpression()

void arcs::aubo_scope::LoopNode::setExpression (
 ExpressionPtr expression)

Create a loop expression configuration.

Parameters

expression	the expression whose evaluation must be true to continue looping, not null.
evaluateContinuously	if true the expression will be evaluated continuously during the looping (i.e. the loop can be interrupted anytime during its execution), otherwise false and the expression will be evaluated once per loop iteration.

Returns

the configuration.

11.78.4.9 setLoopCount()

Generated by Doxygen

void arcs::aubo_scope::LoopNode::setLoopCount (
 int loop_count)

Parameters

loopCount	the number of iterations through the loop node.
errorHandler	error handler for handling validation. If using ErrorHandler#AUTO_CORRECT this will clamp the value to the nearest valid loop count value.

Returns

the configuration.

11.78.4.10 setLoopCountVariable()

```
void arcs::aubo_scope::LoopNode::setLoopCountVariable (
    VariablePtr var)
```

Returns

the variable for storing the loop count

11.78.5 Friends And Related Symbol Documentation

11.78.5.1 DataSwitch

```
friend class DataSwitch [friend]
```

Definition at line 108 of file [loop_node.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.78.6 Member Data Documentation

11.78.6.1 d__

```
void* arcs::aubo_scope::LoopNode::d__ { nullptr } [private]
```

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

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

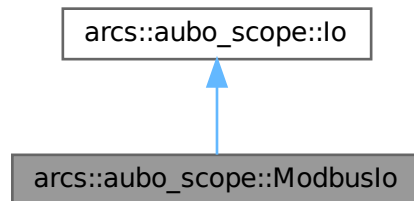
- [include/aubo_caps/domain/program/nodes/builtin/loop_node.h](#)

11.79 arcs::aubo_scope::ModbusIo Class Reference

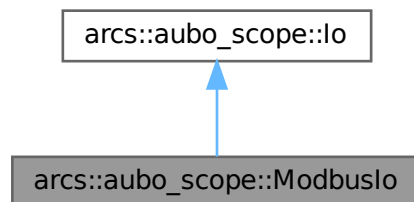
[ModbusIo](#) This interface provides support for MODBUS I/Os.

```
#include <modbus_io.h>
```

Inheritance diagram for arcs::aubo_scope::ModbusIo:



Collaboration diagram for arcs::aubo_scope::ModbusIo:



Public Member Functions

- [ModbusIo](#) (ModbusIo &f)
- [ModbusIo](#) (ModbusIo &&f)
- [~ModbusIo](#) ()
- std::string [getIpAddress](#) ()
Returns the IP address for this MODBUS I/O.
- std::string [getSignalAddress](#) ()
Returns the signal address for this MODBUS I/O.
- bool [isSlave](#) ()
Returns whether this MODBUS I/O is a slave.
- bool [setValue](#) (int value)
Set MODBUS signal to an integer value.
- int [getValue](#) ()
Returns the integer value of the MODBUS signal.

Public Member Functions inherited from [arcs::aubo_scope::Io](#)

- [Io](#) (Io &f)
- [Io](#) (Io &&f)
- virtual [~Io](#) ()
- std::string [getDisplayName](#) () const
Returns the user defined name of the IO in the currently selected installation.
- std::string [getDefaultName](#) () const
Returns the robot default name of the I/O.
- std::string [getValueStr](#) () const
Returns the current reading of the I/O as a string.
- [IoType](#) [getType](#) () const
Returns the I/O type.
- [InterfaceType](#) [getInterfaceType](#) () const
Returns the interface type this I/O belongs to.
- int [getAction](#) () const
Returns the action number of the I/O.
- bool [isInput](#) () const
Returns true if I/O supports reading of values and false otherwise.
- bool [isResolvable](#) () const
Some I/Os, such as MODBUS I/Os and analog Tool inputs, are in some situations not present in AuboScope or available for use.
- size_t [hashCode](#) ()

Private Member Functions

- [ModbusIo](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Public Types inherited from [arcs::aubo_scope::Io](#)

- enum [IoType](#) : int {
 [DIGITAL](#) , [ANALOG](#) , [INTEGER](#) , [BOOLEAN](#) ,
 [DOUBLE](#) }
- enum [InterfaceType](#) : int {
 [STANDARD](#) , [TOOL](#) , [CONFIGURABLE](#) , [LINKIO](#) ,
 [MODBUS](#) , [GENERAL_PURPOSE](#) , [PROFINET](#) , [ETHERNET_IP](#) }

Protected Member Functions inherited from [arcs::aubo_scope::Io](#)

- [Io](#) ()

11.79.1 Detailed Description

[ModbusIo](#) This interface provides support for MODBUS I/Os.

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

11.79.2 Constructor & Destructor Documentation

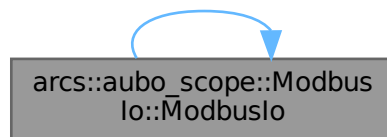
11.79.2.1 ModbusIo() [1/3]

```
arcs::aubo_scope::ModbusIo::ModbusIo (  
    ModbusIo & f)
```

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

Referenced by [ModbusIo\(\)](#), and [ModbusIo\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

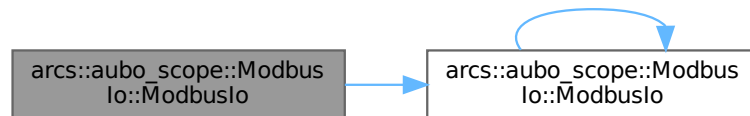


11.79.2.2 ModbusIo() [2/3]

```
arcs::aubo_scope::ModbusIo::ModbusIo (
    ModbusIo && f)
```

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

Here is the call graph for this function:



11.79.2.3 ~ModbusIo()

```
arcs::aubo_scope::ModbusIo::~~ModbusIo ()
```

11.79.2.4 ModbusIo() [3/3]

```
arcs::aubo_scope::ModbusIo::ModbusIo () [private]
```

11.79.3 Member Function Documentation

11.79.3.1 getIpAddress()

```
std::string arcs::aubo_scope::ModbusIo::getIpAddress ()
```

Returns the IP address for this MODBUS I/O.

Returns

Get the IP address for this MODBUS I/O.

11.79.3.2 getSignalAddress()

```
std::string arcs::aubo_scope::ModbusIo::getSignalAddress ()
```

Returns the signal address for this MODBUS I/O.

Returns

the signal address for this MODBUS I/O.

11.79.3.3 getValue()

int arcs::aubo_scope::ModbusIo::getValue ()

Returns the integer value of the MODBUS signal.

Returns

the integer value of the MODBUS signal.

11.79.3.4 isSlave()

bool arcs::aubo_scope::ModbusIo::isSlave ()

Returns whether this MODBUS I/O is a slave.

Returns

Get the is slave for this MODBUS I/O.

11.79.3.5 setValue()

bool arcs::aubo_scope::ModbusIo::setValue (
int value)

Set MODBUS signal to an integer value.

Parameters

value	Set MODBUS signal to an integer value.

Returns

true if the MODBUS signal was sent down to the controller, and false if the signal was not sent down.

11.79.4 Friends And Related Symbol Documentation

11.79.4.1 DataSwitch

friend class DataSwitch [friend]

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

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.79.5 Member Data Documentation

11.79.5.1 d_

void* arcs::aubo_scope::ModbusIo::d_ { nullptr } [private]

Definition at line 99 of file [modbus_io.h](#).

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

- [include/aubo_caps/domain/io/modbus_io.h](#)

11.80 arcs::aubo_scope::ModbusIoFilter Struct Reference

`#include <modbus_io.h>`

Public Member Functions

- [ModbusIoFilter](#) ([Io::IoType](#) type, bool is__input)
- bool [operator\(\)](#) (IoPtr io)

Private Attributes

- [Io::IoType](#) type__
- bool [is__input__](#)

11.80.1 Detailed Description

Definition at line 102 of file [modbus_io.h](#).

11.80.2 Constructor & Destructor Documentation

11.80.2.1 ModbusIoFilter()

```
arcs::aubo_scope::ModbusIoFilter::ModbusIoFilter (  
    Io::IoType type,  
    bool is__input) [inline]
```

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

References [is__input__](#), and [type__](#).

11.80.3 Member Function Documentation

11.80.3.1 operator()()

bool arcs::aubo_scope::ModbusIoFilter::operator() (
IoPtr io) [inline]

Definition at line 108 of file [modbus_io.h](#).

References [is_input__](#), and [type__](#).

11.80.4 Member Data Documentation

11.80.4.1 is_input__

bool arcs::aubo_scope::ModbusIoFilter::is_input__ [private]

Definition at line 116 of file [modbus_io.h](#).

Referenced by [ModbusIoFilter\(\)](#), and [operator\(\)\(\)](#).

11.80.4.2 type__

[Io::IoType](#) arcs::aubo_scope::ModbusIoFilter::type__ [private]

Definition at line 115 of file [modbus_io.h](#).

Referenced by [ModbusIoFilter\(\)](#), and [operator\(\)\(\)](#).

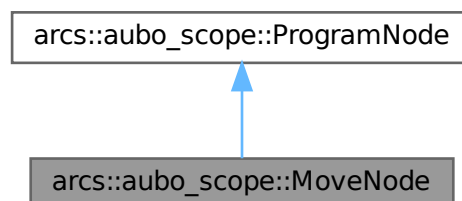
The documentation for this struct was generated from the following file:

- [include/aubo_caps/domain/io/modbus_io.h](#)

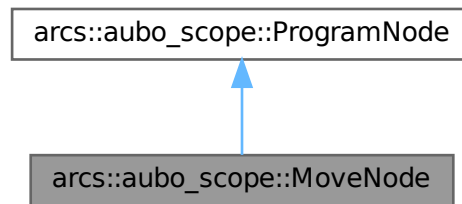
11.81 arcs::aubo_scope::MoveNode Class Reference

```
#include <move_node.h>
```

Inheritance diagram for arcs::aubo_scope::MoveNode:



Collaboration diagram for arcs::aubo_scope::MoveNode:



Public Types

- enum `ConfigType` : int { `MOVE_J` = 0 , `MOVE_L` , `MOVE_S` }
- enum `TcpSelectionType` : int { `USE_ACTIVE_TCP` = 0 , `IGNORE_ACTIVE_TCP` , `MOVE_TCP` }
- enum `FeatureSelectionType` : int { `UNRESOLVED_FEATURE` , `FEATURE` }

Public Member Functions

- `MoveNode` (MoveNode &f)
- `MoveNode` (MoveNode &&f)
- virtual `~MoveNode` ()
- `ConfigType` `getConfigType` ()
This method returns the type of configuration.
- void `setConfigType` (`ConfigType` type)
- bool `getUseJoints` ()
- void `setUseJoints` (bool use)
- void `setJointSpeed` (double joint_speed)
- double `getJointSpeed` ()
- void `setJointAcceleration` (double joint_acceleration)
- double `getJointAcceleration` ()
- void `setTCPSelection` (TCPPtr tcp_selection)
- TCPPtr `getTCPSelection` ()
- void `setToolSpeed` (double tool_speed)
- double `getToolSpeed` ()
- void `setToolAcceleration` (double tool_acceleration)
- double `getToolAcceleration` ()
- void `setFeature` (FeaturePtr feature)
- void `setFeature` (FeatureVariablePtr feature_variable)
- FeaturePtr `getFeature` ()
- void `setBlend` (double blend)
- double `getBlend` ()
- void `setTcpSelectionType` (`TcpSelectionType` type)
- `TcpSelectionType` `getTcpSelectionType` ()

Public Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()
Get the program node contribution.

Private Member Functions

- [MoveNode](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Protected Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.81.1 Detailed Description

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

11.81.2 Member Enumeration Documentation

11.81.2.1 ConfigType

enum [arcs::aubo_scope::MoveNode::ConfigType](#) : int

Enumerator

MOVE_J	Movement type is MoveJ. The config instance can be cast to MoveJMoveNodeConfig.
MOVE_L	Movement type is MoveL. The config instance can be cast to MoveLMoveNodeConfig.
MOVE_S	

Enumerator

UNRESOLVED_FEATURE	The selected feature is no longer available (e.g. because the user has removed the feature in the Installation or the needed Installation is not currently loaded)
FEATURE	Feature is selected. The Selection instance can be cast to MoveNodeFeature

Definition at line 62 of file [move_node.h](#).

11.81.2.3 TcpSelectionType

```
enum arcs::aubo_scope::MoveNode::TcpSelectionType : int
```

Enumerator

USE_ACTIVE_TCP	Motions are adjusted to the currently active TCP . The TCP is determined during runtime of the program.
IGNORE_ACTIVE_TCP	No TCP is used and the motion under this Move command will be with respect to the tool output flange.
MOVE_TCP	A specific TCP is selected. The TCPSelection instance can be cast to MoveNodeTCP.

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

11.81.3 Constructor & Destructor Documentation

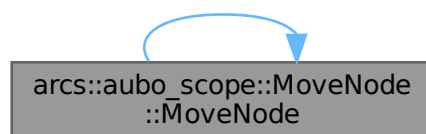
11.81.3.1 MoveNode() [1/3]

```
arcs::aubo_scope::MoveNode::MoveNode (  
    MoveNode & f)
```

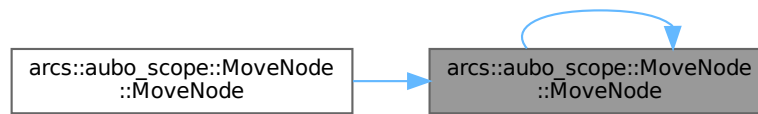
References [MoveNode\(\)](#).

Referenced by [MoveNode\(\)](#), and [MoveNode\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

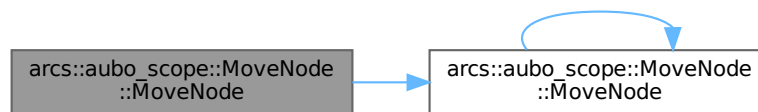


11.81.3.2 MoveNode() [2/3]

```
arcs::aubo_scope::MoveNode::MoveNode (
    MoveNode && f)
```

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

Here is the call graph for this function:



11.81.3.3 ~MoveNode()

```
virtual arcs::aubo_scope::MoveNode::~MoveNode () [virtual]
```

11.81.3.4 MoveNode() [3/3]

```
arcs::aubo_scope::MoveNode::MoveNode () [private]
```

11.81.4 Member Function Documentation

11.81.4.1 getBlend()

```
double arcs::aubo_scope::MoveNode::getBlend ()
```

11.81.4.2 getConfigType()

[ConfigType](#) arcs::aubo_scope::MoveNode::getConfigType ()

This method returns the type of configuration.

Cast this instance appropriately to have access to specific getters.

Returns

the type of this config.

11.81.4.3 getFeature()

FeaturePtr arcs::aubo_scope::MoveNode::getFeature ()

11.81.4.4 getJointAcceleration()

double arcs::aubo_scope::MoveNode::getJointAcceleration ()

11.81.4.5 getJointSpeed()

double arcs::aubo_scope::MoveNode::getJointSpeed ()

11.81.4.6 getTCPSelection()

TCPPtr arcs::aubo_scope::MoveNode::getTCPSelection ()

11.81.4.7 getTcpSelectionType()

[TcpSelectionType](#) arcs::aubo_scope::MoveNode::getTcpSelectionType ()

11.81.4.8 getToolAcceleration()

double arcs::aubo_scope::MoveNode::getToolAcceleration ()

11.81.4.9 getToolSpeed()

double arcs::aubo_scope::MoveNode::getToolSpeed ()

11.81.4.10 getUseJoints()

bool arcs::aubo_scope::MoveNode::getUseJoints ()

11.81.4.11 setBlend()

Parameters

blend	The shared blend used in the motion of this Move node.

11.81.4.12 setConfigType()

```
void arcs::aubo_scope::MoveNode::setConfigType (
    ConfigType type)
```

11.81.4.13 setFeature() [1/2]

```
void arcs::aubo_scope::MoveNode::setFeature (
    FeaturePtr feature)
```

Set feature for the MoveL configuration.

The feature will be used for the movement of the robot arm, from the previous position through the waypoints under the Move node.

Parameters

feature	The shared feature for this movement.

11.81.4.14 setFeature() [2/2]

```
void arcs::aubo_scope::MoveNode::setFeature (
    FeatureVariablePtr feature_variable)
```

11.81.4.15 setJointAcceleration()

```
void arcs::aubo_scope::MoveNode::setJointAcceleration (
    double joint_acceleration)
```

Set joint acceleration parameter for the MoveJ configuration.

The parameter applies to the movement of the robot arm, from the previous position through the waypoints under the Move node.

Parameters

joint_acceleration	The shared joint acceleration to be used.

11.81.4.16 setJointSpeed()

```
void arcs::aubo_scope::MoveNode::setJointSpeed (
    double joint_speed)
```

Set joint speed parameter for the MoveJ configuration.

The parameter applies to the movement of the robot arm, from the previous position through the way-points under the Move node.

Parameters

joint_speed	The shared joint speed to be achieved.

11.81.4.17 setTCPSelection()

```
void arcs::aubo_scope::MoveNode::setTCPSelection (
    TCPPtr tcp_selection)
```

Set the [TCP](#) selection for the MoveJ configuration.

The [TCP](#) will be set as the active [TCP](#) before the movement of the robot arm is performed.

Parameters

tcp_selection	The TCP setting to be applied.

11.81.4.18 setTcpSelectionType()

```
void arcs::aubo_scope::MoveNode::setTcpSelectionType (
    TcpSelectionType type)
```

11.81.4.19 setToolAcceleration()

```
void arcs::aubo_scope::MoveNode::setToolAcceleration (
    double tool_acceleration)
```

Set tool acceleration parameter for the MoveL configuration.

The parameter applies to the movement of the robot arm, from the previous position through the way-

Parameters

tool_acceleration	The shared tool acceleration to be used.

11.81.4.20 setToolSpeed()

```
void arcs::aubo_scope::MoveNode::setToolSpeed (
    double tool_speed)
```

Set tool speed parameter for the MoveL configuration.

The parameter applies to the movement of the robot arm, from the previous position through the way-points under the Move node.

Parameters

tool_speed	The shared tool speed to be achieved.

11.81.4.21 setUseJoints()

```
void arcs::aubo_scope::MoveNode::setUseJoints (
    bool use)
```

11.81.5 Friends And Related Symbol Documentation

11.81.5.1 DataSwitch

```
friend class DataSwitch [friend]
```

Definition at line 218 of file [move_node.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.81.6 Member Data Documentation

11.81.6.1 d__

```
void* arcs::aubo_scope::MoveNode::d_ { nullptr } [private]
```

Definition at line 220 of file [move_node.h](#).

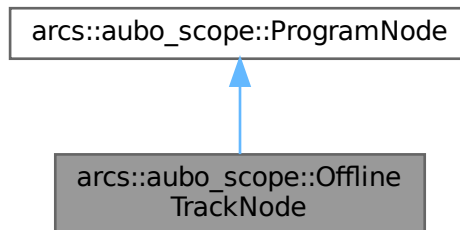
The documentation for this class was generated from the following file:

- [include/aubo_caps/domain/program/nodes/builtin/move_node.h](#)

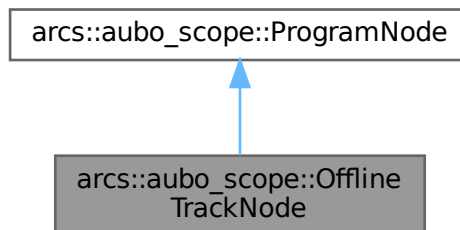
11.82 arcs::aubo_scope::OfflineTrackNode Class Reference

```
#include <offline_track_node.h>
```

Inheritance diagram for arcs::aubo_scope::OfflineTrackNode:



Collaboration diagram for arcs::aubo_scope::OfflineTrackNode:



Public Member Functions

- [OfflineTrackNode](#) (`OfflineTrackNode &f`)
- [OfflineTrackNode](#) (`OfflineTrackNode &&f`)
- virtual [~OfflineTrackNode](#) ()

Public Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (`ProgramNode &f`)
- [ProgramNode](#) (`ProgramNode &&f`)
- virtual [~ProgramNode](#) ()
- `ProgramNodeContributionPtr` [getProgramNodeContribution](#) ()
Get the program node contribution.

Private Member Functions

- [OfflineTrackNode](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Protected Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.82.1 Detailed Description

Definition at line 10 of file [offline_track_node.h](#).

11.82.2 Constructor & Destructor Documentation

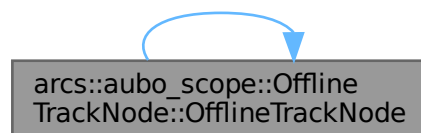
11.82.2.1 OfflineTrackNode() [1/3]

`arcs::aubo_scope::OfflineTrackNode::OfflineTrackNode (`
 `OfflineTrackNode & f)`

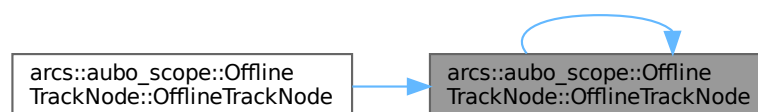
References [OfflineTrackNode\(\)](#).

Referenced by [OfflineTrackNode\(\)](#), and [OfflineTrackNode\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



11.82.2.2 OfflineTrackNode() [2/3]

```
arcs::aubo_scope::OfflineTrackNode::OfflineTrackNode (
    OfflineTrackNode && f)
```

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

Here is the call graph for this function:



11.82.2.3 ~OfflineTrackNode()

```
virtual arcs::aubo_scope::OfflineTrackNode::~~OfflineTrackNode () [virtual]
```

11.82.2.4 OfflineTrackNode() [3/3]

```
arcs::aubo_scope::OfflineTrackNode::OfflineTrackNode () [private]
```

11.82.3 Friends And Related Symbol Documentation

11.82.3.1 DataSwitch

```
friend class DataSwitch [friend]
```

Definition at line 18 of file [offline_track_node.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.82.4 Member Data Documentation

11.82.4.1 d__

```
void* arcs::aubo_scope::OfflineTrackNode::d__ { nullptr } [private]
```

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

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

- [include/aubo_caps/domain/program/nodes/builtin/offline_track_node.h](#)

11.83 arcs::aubo_scope::OutputIoFilter Struct Reference

#include <digital_io.h>

Public Member Functions

- bool [operator\(\)](#) (IoPtr io)

11.83.1 Detailed Description

Definition at line 85 of file [digital_io.h](#).

11.83.2 Member Function Documentation

11.83.2.1 operator>()()

bool arcs::aubo_scope::OutputIoFilter::operator() (
IoPtr io) [inline]

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

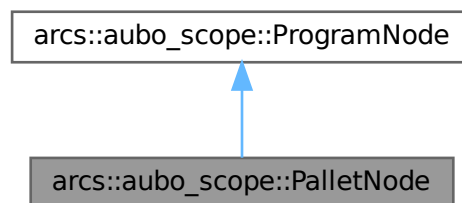
The documentation for this struct was generated from the following file:

- include/aubo_caps/domain/io/[digital_io.h](#)

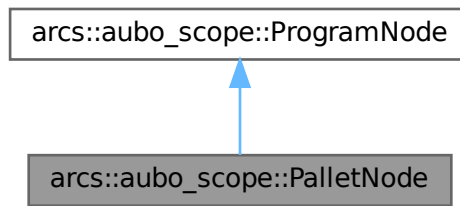
11.84 arcs::aubo_scope::PalletNode Class Reference

#include <pallet_node.h>

Inheritance diagram for arcs::aubo_scope::PalletNode:



Collaboration diagram for `arcs::aubo_scope::PalletNode`:



Public Member Functions

- [PalletNode](#) (PalletNode &f)
- [PalletNode](#) (PalletNode &&f)
- virtual [~PalletNode](#) ()

Public Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()
Get the program node contribution.

Private Member Functions

- [PalletNode](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Protected Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.84.1 Detailed Description

Definition at line 10 of file [pallet_node.h](#).

11.84.2 Constructor & Destructor Documentation

11.84.2.1 PalletNode() [1/3]

```
arcs::aubo_scope::PalletNode::PalletNode (  
    PalletNode & f)
```

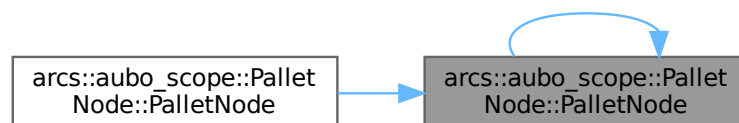
References [PalletNode\(\)](#).

Referenced by [PalletNode\(\)](#), and [PalletNode\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

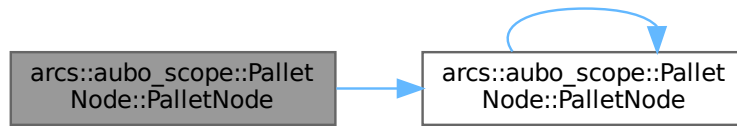


11.84.2.2 PalletNode() [2/3]

```
arcs::aubo_scope::PalletNode::PalletNode (  
    PalletNode && f)
```

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

Here is the call graph for this function:



11.84.2.3 ~PalletNode()

virtual arcs::aubo_scope::PalletNode::~~PalletNode () [virtual]

11.84.2.4 PalletNode() [3/3]

arcs::aubo_scope::PalletNode::PalletNode () [private]

11.84.3 Friends And Related Symbol Documentation

11.84.3.1 DataSwitch

friend class DataSwitch [friend]

Definition at line 18 of file [pallet_node.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.84.4 Member Data Documentation

11.84.4.1 d_

void* arcs::aubo_scope::PalletNode::d_ { nullptr } [private]

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

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

- [include/aubo_caps/domain/program/nodes/builtin/pallet_node.h](#)

11.85 arcs::aubo_scope::Payload Class Reference

API interface for payload parameters.

```
#include <payload.h>
```

Public Types

- enum [ParametersWarning](#) { [NO_WARNING](#) , [MASS_0KG](#) , [RATED_MASS_EXCEEDED](#) , [MASS_AND_CENTER_OF_GRAVITY_EXCEEDED](#) }

Public Member Functions

- [Payload](#) (Payload &f)
- [Payload](#) (Payload &&f)
- [~Payload](#) ()
- double [getMass](#) () const
- std::vector< double > [getCenterOfGravity](#) () const
Gets the payload's center of gravity (CoG), also referred to as center of mass.
- std::vector< double > [getInertiaMatrix](#) () const
- std::string [getDisplayname](#) () const
- bool [isResolvable](#) ()
- bool [isUsable](#) ()
- [ParametersWarning](#) [getParametersWarning](#) ()
- size_t [hashCode](#) ()

Private Member Functions

- [Payload](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

11.85.1 Detailed Description

API interface for payload parameters.

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

11.85.2 Member Enumeration Documentation

11.85.2.1 ParametersWarning

Generated by Doxygen

```
enum arcs::aubo_scope::Payload::ParametersWarning
```

This enum represents the different types of warnings that the parameters of a payload can have.

Enumerator

NO_WARNING	None of the parameters of the payload or the combination of payload parameters has any warnings.
MASS_0KG	The payload has a mass of 0 kg which is most likely not intended/valid. Note that the value for the payload mass should be the total mass of the payload attached to the tool output flange of the robot.
RATED_MASS_EXCEEDED	The mass of the payload exceeds the rated maximum supported payload mass of the robot. Using the payload could affect performance of the robot.
MASS_AND_CENTER_OF_GRAVITY_EXCEEDED	The combination of the mass and the center of gravity of could affect the performance of the robot, if the payload is used.

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

11.85.3 Constructor & Destructor Documentation

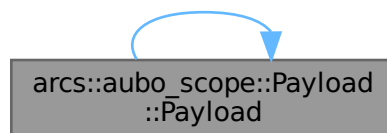
11.85.3.1 Payload() [1/3]

```
arcs::aubo_scope::Payload::Payload (
    Payload & f)
```

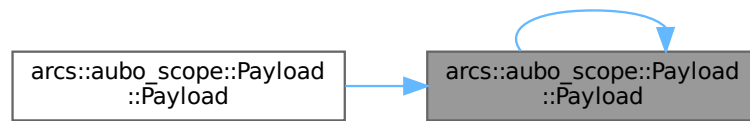
References [Payload\(\)](#).

Referenced by [Payload\(\)](#), and [Payload\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

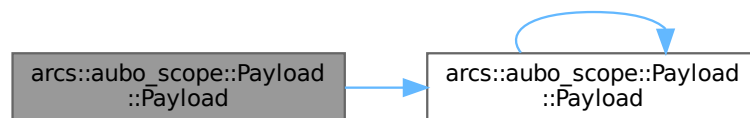


11.85.3.2 Payload() [2/3]

```
arcs::aubo_scope::Payload::Payload (
    Payload && f)
```

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

Here is the call graph for this function:



11.85.3.3 ~Payload()

```
arcs::aubo_scope::Payload::~~Payload ()
```

11.85.3.4 Payload() [3/3]

```
arcs::aubo_scope::Payload::Payload () [private]
```

11.85.4 Member Function Documentation

11.85.4.1 getCenterOfGravity()

```
std::vector< double > arcs::aubo_scope::Payload::getCenterOfGravity () const
```

Gets the payload's center of gravity (CoG), also referred to as center of mass.

It is defined as the offset between the center of the tool output flange and the center of gravity of the attached payload.

Returns

The center of gravity (CoG) of the payload relative to the center of the tool output flange

11.85.4.2 `getDisplayName()`

```
std::string arcs::aubo_scope::Payload::getDisplayName () const
```

Gets the name of the payload displayed in the AuboScope UI.

Note: The payload can change name at any time, and thus the name must not be used to uniquely identify a specific payload. The payload name can change in the following situations:

- The payload is renamed by the end user
- The name gets translated (depending on the selected language in AuboScope)
- The name is changed between versions of the AuboCap that added the payload

Returns

The name of the payload displayed in AuboScope

11.85.4.3 `getInertiaMatrix()`

```
std::vector< double > arcs::aubo_scope::Payload::getInertiaMatrix () const
```

Gets the inertia matrix of the payload. The inertia matrix is defined in a coordinate system with center at the payload's center of gravity (CoG) and the axes aligned with the tool output flange coordinate system (axes).

Note: The inertia matrix specified by the end user can be invalid. The [isUsable](#) method can be used determine, if the payload can be used (i.e., has a valid inertia matrix).

Returns

The inertia matrix of the payload

11.85.4.4 `getMass()`

```
double arcs::aubo_scope::Payload::getMass () const
```

Returns

The total mass of the payload (attached to the tool output flange of the robot)

11.85.4.5 `getParametersWarning()`

```
ParametersWarning arcs::aubo_scope::Payload::getParametersWarning ()
```

Use this method to check if the parameters of this payload have any warnings.

A payload can be usable, but may have a value for a payload parameter or may have a combination of (some of the) payload parameters, that could affect performance of the robot, if the payload is used (i.e., if the parameters are applied).

An unresolvable payload (see [isResolvable\(\)](#)) will have no warning for its parameters (i.e. [ParametersWarning#NO_WARNING](#) is returned).

Returns

The warning (if any) for the parameters of this payload.

11.85.4.6 hashCode()

size_t arcs::aubo_scope::Payload::hashCode ()

11.85.4.7 isResolvable()

bool arcs::aubo_scope::Payload::isResolvable ()

A payload cannot be guaranteed to be present in AuboScope. This method can be used to determine, if the payload is present.

The payload will not be present, if the end user loads a different installation which does not contain the payload, or if the payload is removed by the end user or the AuboCap that added the payload.

Note: Storing an undefined payload in a program node contribution's (a [ProgramNodeContribution](#) instance) data model ([DataModel](#) interface) will automatically make the program node (and the program) undefined.

Returns

true if this payload is present in AuboScope, otherwise false.

11.85.4.8 isUsable()

bool arcs::aubo_scope::Payload::isUsable ()

Use this method to determine if the payload can be used. An unresolvable payload (see the [isResolvable](#) method) will always be unusable.

A payload can have an invalid inertia matrix, where all the individual values of the matrix are within the valid range, but the combination of values is invalid.

Note: Storing an unusable payload in a program node contribution's (a [ProgramNodeContribution](#) instance) data model ([DataModel](#) interface) will automatically make the program node (and the program) undefined.

Returns

true, if the payload is usable, otherwise false. false will always be returned when the payload is unresolvable (see the [isResolvable](#) method).

11.85.5 Friends And Related Symbol Documentation

11.85.5.1 DataSwitch

friend class DataSwitch [friend]

Definition at line 189 of file [payload.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.85.6 Member Data Documentation

11.85.6.1 d_

void* arcs::aubo_scope::Payload::d_ { nullptr } [private]

Definition at line 191 of file [payload.h](#).

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

- [include/aubo_caps/domain/payload/payload.h](#)

11.86 arcs::aubo_scope::PayloadContributionModel Class Reference

API interface for dynamic payload management.

```
#include <payload_contribution_model.h>
```

Public Member Functions

- [PayloadContributionModel](#) ([PayloadContributionModel](#) &f)
- [PayloadContributionModel](#) ([PayloadContributionModel](#) &&f)
- [~PayloadContributionModel](#) ()
- PayloadPtr [addPayload](#) (const std::string &id_key, const std::string &suggested_name, double mass, const std::vector< double > &center_of_gravity)
 - [Payload](#)
- PayloadPtr [addPayload](#) (const std::string &id_key, const std::string &suggested_name, double mass, const std::vector< double > &center_of_gravity, const std::vector< double > &inertia↵Matrix)
 - [Payload](#)
- PayloadPtr [getPayload](#) (const std::string &id_key)
 - [Payload](#) Returns the payload previously added by this AuboCap using the same idKey identifier.
- void [updatePayload](#) (const std::string &id_key, double mass, const std::vector< double > &center↵_of_gravity)
 - [Payload](#)
- void [updatePayload](#) (const std::string id_key, double mass, const std::vector< double > &center↵_of_gravity, const std::vector< double > &inertia_matrix)
 - Update the mass, center of gravity (CoG) and inertia matrix of an existing payload added by this AuboCap.
- void [removePayload](#) (const std::string &id_key)
 - Remove a payload added by this AuboCap from AuboScope.

Private Member Functions

- [PayloadContributionModel](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

11.86.1 Detailed Description

API interface for dynamic payload management.

Definition at line 17 of file [payload_contribution_model.h](#).

11.86.2 Constructor & Destructor Documentation

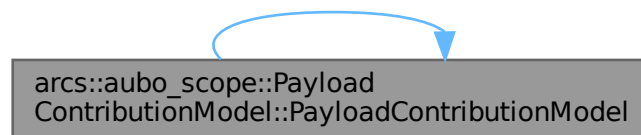
11.86.2.1 PayloadContributionModel() [1/3]

```
arcs::aubo_scope::PayloadContributionModel::PayloadContributionModel (  
    PayloadContributionModel & f)
```

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

Referenced by [PayloadContributionModel\(\)](#), and [PayloadContributionModel\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



11.86.2.2 PayloadContributionModel() [2/3]

```
arcs::aubo_scope::PayloadContributionModel::PayloadContributionModel (
    PayloadContributionModel && f)
```

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

Here is the call graph for this function:



11.86.2.3 ~PayloadContributionModel()

```
arcs::aubo_scope::PayloadContributionModel::~~PayloadContributionModel ()
```

11.86.2.4 PayloadContributionModel() [3/3]

```
arcs::aubo_scope::PayloadContributionModel::PayloadContributionModel () [private]
```

11.86.3 Member Function Documentation

11.86.3.1 addPayload() [1/2]

```
PayloadPtr arcs::aubo_scope::PayloadContributionModel::addPayload (
    const std::string & id_key,
    const std::string & suggested_name,
    double mass,
    const std::vector< double > & center_of_gravity)
```

Payload

Add a payload to the current installation in AuboScope.

This makes the payload selectable by the end user as well as available to other AuboCap contributions (through the [PayloadModel](#) interface). The payload is not modifiable by the end user or other aubo_↔ studio plugins.

A default guess for the inertia matrix for the payload will be computed based on the specified payload mass and center of gravity. If data for the payload's inertia matrix is available, the method `addPayload(String, String, Mass, Position, InertiaMatrix)` can be used instead (this might give better performance of the robot arm).

Note: The specified payload mass must be the total mass of the payload attached to the tool output flange of the robot.

The valid range for the payload mass and the coordinates of the center of gravity (CoG) are available through the `RobotLimits` interface which can be accessed with `RobotModel#getRobotLimits()` (an instance of the `RobotModel` interface can be retrieved through `SystemAPI#getRobotModel()`).

Parameters

idKey	The key to identify this payload by, not null nor an empty string. The key is for this AuboCap only, i.e. it only has to be unique for this AuboCap and not "globally" for other aubo_studio plugins.
suggestedName	Suggested name for the payload, not null nor an empty string. Valid names must match regex <code>[a-zA-Z][a-zA-Z0-9_]{0,14}</code> for a total of 15 characters. The final name can be retrieved from the returned payload instance.
mass	The total mass of the payload attached to the tool output flange of the robot, not null. Create it using <code>SimpleValueFactory#createMass(double, Mass.Unit)</code> .
centerOfGravity	The center of gravity (CoG), also referred to as center of mass, for the payload. It is defined as the offset between the center of the tool output flange and the center of gravity of the attached payload. Cannot be null. Create it using the <code>PositionFactory</code> interface.

Returns

The payload created and registered in AuboScope.

Exceptions

PayloadAlreadyAddedException	If a payload has previously been added using the same idKey identifier. Use <code>getPayload(String)</code> to check if the payload has already been added to the current installation. Use <code>updatePayload(String, Mass, Position)</code> or <code>updatePayload(String, Mass, Position, InertiaMatrix)</code> to update the payload if the payload parameters have changed.
IllegalPayloadNameException	If the suggested name does not match required regex.
IllegalMassException	If the specified mass is not inside the valid range as defined by AuboScope.
IllegalCenterOfGravityException	If any of the values of the specified center of gravity are not inside the valid range as defined by AuboScope.

11.86.3.2 addPayload() [2/2]

```
PayloadPtr arcs::aubo_scope::PayloadContributionModel::addPayload (
    const std::string & id_key,
    const std::string & suggested_name,
    double mass,
    const std::vector< double > & center_of_gravity,
    const std::vector< double > & inertiaMatrix)
```

[Payload](#)

Add a payload to the current installation in AuboScope.

This makes the payload selectable by the end user as well as available to other AuboCap contributions (through the [PayloadModel](#) interface). The payload is not modifiable by the end user or other aubo_↔ studio plugins.

Use the `addPayload(String, String, Mass, Position, InertiaMatrix)` method instead if data for the payload's inertia matrix is not available.

Note: The specified payload mass must be the total mass of the payload attached to the tool output flange of the robot.

The valid range for the payload mass and the coordinates of the center of gravity (CoG) are available through the `RobotLimits` interface which can be accessed with `RobotModel#getRobotLimits()` (an instance of the `RobotModel` interface can be retrieved through `SystemAPI#getRobotModel()`).

Parameters

idKey	The key to identify this payload by, not null nor an empty string. The key is for this AuboCap only, i.e. it only has to be unique for this AuboCap and not "globally" for other aubo_studio plugins.
suggestedName	Suggested name for the payload, not null nor an empty string. Valid names must match regex <code>[a-zA-Z][a-zA-Z0-9_]{0,14}</code> for a total of 15 characters. The final name can be retrieved from the returned payload instance.
mass	The total mass of the payload attached to the tool output flange of the robot, not null. Create it using <code>SimpleValueFactory#createMass(double, Mass.Unit)</code> .
centerOfGravity	The center of gravity (CoG), also referred to as center of mass, for the payload. It is defined as the offset between the center of the tool output flange and the center of gravity of the attached payload. Cannot be null. Create it using the <code>PositionFactory</code> interface.
inertiaMatrix	The inertia matrix of the payload, defined in a coordinate system with center at the payload's center of gravity and the axes aligned with the tool output flange coordinate system (axes). Cannot be null. Create it using <code>double, double, double, double, double, MomentOfInertia.Unit</code> .

Returns

The payload created and registered in AuboScope.

Exceptions

<code>PayloadAlreadyAddedException</code>	If a payload has previously been added using the same idKey identifier. Use <code>getPayload(String)</code> to check if the payload has already been added to the current installation. Use <code>updatePayload(String, Mass, Position)</code> or <code>updatePayload(String, Mass, Position, InertiaMatrix)</code> to update the payload if the payload parameters have changed.
<code>IllegalPayloadNameException</code>	If the suggested name does not match required regex.

IllegalMassException	If the specified mass is not inside the valid range as defined by AboScope.
IllegalCenterOfGravityException	If any of the values of the specified center of gravity are not inside the valid range as defined by AboScope.
IllegalInertiaMatrixException	If any of the values of the specified inertia matrix are not inside the valid range as defined by AboScope.

11.86.3.3 getPayload()

```
PayloadPtr arcs::aubo_scope::PayloadContributionModel::getPayload (
    const std::string & id_key)
```

Payload Returns the payload previously added by this AboCap using the same idKey identifier.

Use this to verify if the payload is present in the current installation.

Parameters

idKey	The key to identify this payload by, not null nor an empty string.

Returns

The payload previously added by this AboCap. Returns null if no payload exists in current installation (with the specified idKey identifier).

11.86.3.4 removePayload()

```
void arcs::aubo_scope::PayloadContributionModel::removePayload (
    const std::string & id_key)
```

Remove a payload added by this AboCap from AboScope.

Program nodes using the payload will be become undefined because the payload is no longer resolvable.

Parameters

idKey	The identifier key used to add the payload with, not null nor an empty string.

Exceptions

PayloadNotFoundException	If no payload exists with the provided idKey identifier in the current installation.

11.86.3.5 updatePayload() [1/2]

```
void arcs::aubo_scope::PayloadContributionModel::updatePayload (
    const std::string & id_key,
    double mass,
    const std::vector< double > & center_of_gravity)
```

[Payload](#)

Update the mass and center of gravity (CoG) of an existing payload added by this AuboCap.

Note: If an inertia matrix has been specified for the payload, calling this method will replace (overwrite) it with a computed default guess based the new specified payload mass and center of gravity. To avoid overwriting an existing inertia matrix (or to update an existing inertia matrix), instead use the method updatePayload(String, Mass, Position, InertiaMatrix).

Parameters

idKey	The key to identify the payload, not null nor an empty string. A payload must have been added using the same key prior to calling this method.
mass	The total mass of the payload attached to the tool output flange of the robot, not null. Create it using SimpleValueFactory#createMass(double, Mass.Unit).
centerOfGravity	The center of gravity of the payload, not null. Create it using the PositionFactory interface.

Exceptions

PayloadNotFoundException	If no payload exists with the provided idKey identifier in the current installation.
IllegalMassException	If the specified mass is not inside the valid range as defined by AuboScope.
IllegalCenterOfGravityException	If any of the values of the specified center of gravity are not inside the valid range as defined by AuboScope.

11.86.3.6 updatePayload() [2/2]

```
void arcs::aubo_scope::PayloadContributionModel::updatePayload (
    const std::string id_key,
    double mass,
    const std::vector< double > & center_of_gravity,
    const std::vector< double > & inertia_matrix)
```

Parameters

idKey	The key to identify the payload, not null nor an empty string. A payload must have been added using the same key prior to calling this method.
mass	The total mass of the payload attached to the tool output flange of the robot, not null. Create it using SimpleValueFactory#createMass(double, Mass.Unit).
centerOfGravity	The center of gravity of the payload, not null. Create it using the PositionFactory interface.
inertiaMatrix	The inertia matrix of the payload, not null. Create it using double, double, double, double, double, MomentOfInertia.Unit)

Exceptions

PayloadNotFoundException	If no payload exists with the provided idKey identifier in the current installation.
IllegalMassException	If the specified mass is not inside the valid range as defined by AuboScope.
IllegalCenterOfGravityException	If any of the values of the specified center of gravity are not inside the valid range as defined by AuboScope.
IllegalInertiaMatrixException	If any of the values of the specified inertia matrix are not inside the valid range as defined by AuboScope.

11.86.4 Friends And Related Symbol Documentation

11.86.4.1 DataSwitch

friend class DataSwitch [friend]

Definition at line 412 of file [payload_contribution_model.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.86.5 Member Data Documentation

11.86.5.1 d_

void* arcs::aubo_scope::PayloadContributionModel::d_ { nullptr } [private]

Definition at line 414 of file [payload_contribution_model.h](#).

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

- include/aubo_caps/domain/payload/[payload_contribution_model.h](#)

11.87 arcs::aubo_scope::PayloadModel Class Reference

API interface for payload parameters.

```
#include <payload_model.h>
```

Public Member Functions

- [PayloadModel](#) (PayloadModel &f)
- [PayloadModel](#) (PayloadModel &&f)
- virtual [~PayloadModel](#) ()
- std::vector< PayloadPtr > [getPayloads](#) () const
getPayloads
- PayloadPtr [getPayload](#) (const std::string &name) const
Payload
- PayloadPtr [getDefaultPayload](#) () const
Payload
- PayloadPtr [getCurrentPayload](#) () const
Payload

Private Member Functions

- [PayloadModel](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

11.87.1 Detailed Description

API interface for payload parameters.

Definition at line 18 of file [payload_model.h](#).

11.87.2 Constructor & Destructor Documentation

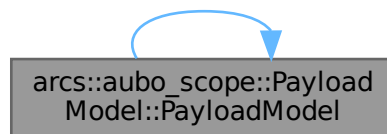
11.87.2.1 PayloadModel() [1/3]

arcs::aubo_scope::PayloadModel::PayloadModel (
PayloadModel & f)

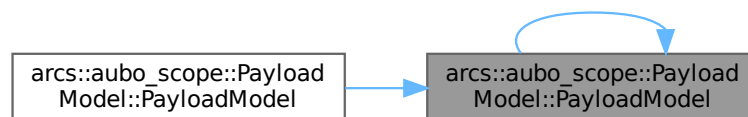
References [PayloadModel\(\)](#).

Referenced by [PayloadModel\(\)](#), and [PayloadModel\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

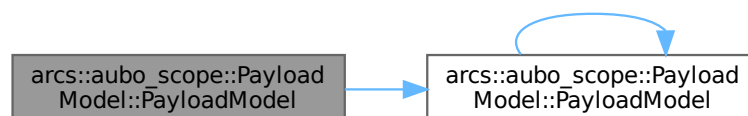


11.87.2.2 PayloadModel() [2/3]

arcs::aubo_scope::PayloadModel::PayloadModel (
PayloadModel && f)

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

Here is the call graph for this function:



11.87.2.3 ~PayloadModel()

virtual arcs::aubo_scope::PayloadModel::~~PayloadModel () [virtual]

11.87.2.4 PayloadModel() [3/3]

arcs::aubo_scope::PayloadModel::PayloadModel () [private]

11.87.3 Member Function Documentation

11.87.3.1 getCurrentPayload()

PayloadPtr arcs::aubo_scope::PayloadModel::getCurrentPayload () const

[Payload](#)

getCurrentPayload

Returns

11.87.3.2 getDefaultPayload()

PayloadPtr arcs::aubo_scope::PayloadModel::getDefaultPayload () const

[Payload](#)

getDefaultPayload

Returns

11.87.3.3 getPayload()

PayloadPtr arcs::aubo_scope::PayloadModel::getPayload (
const std::string & name) const

[Payload](#)

getPayload

Parameters

name	

Returns

11.87.3.4 getPayloads()

`std::vector< PayloadPtr > arcs::aubo_scope::PayloadModel::getPayloads () const`

getPayloads

Returns

A collection of all the payloads from the current installation in AuboScope, including user-defined payloads and payloads added by aubo_studio plugins.

11.87.4 Friends And Related Symbol Documentation

11.87.4.1 DataSwitch

`friend class DataSwitch [friend]`

Definition at line 60 of file [payload_model.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.87.5 Member Data Documentation

11.87.5.1 d__

`void* arcs::aubo_scope::PayloadModel::d__ { nullptr } [private]`

Definition at line 62 of file [payload_model.h](#).

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

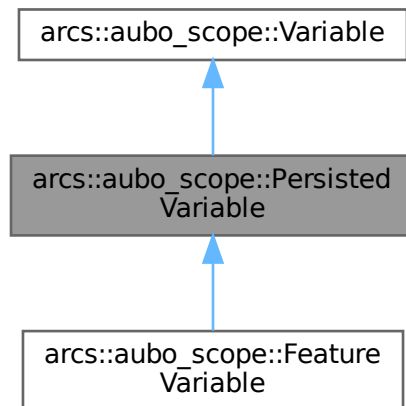
- [include/aubo_caps/domain/payload/payload_model.h](#)

11.88 arcs::aubo_scope::PersistedVariable Class Reference

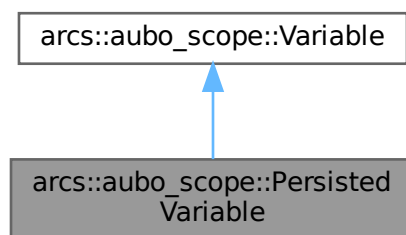
ingroup [ApplicationApi](#) This interface represents variables stored with the installation.

```
#include <persisted_variable.h>
```

Inheritance diagram for arcs::aubo_scope::PersistedVariable:



Collaboration diagram for arcs::aubo_scope::PersistedVariable:



Public Member Functions

- [PersistedVariable](#) (PersistedVariable &f)
- [PersistedVariable](#) (PersistedVariable &&f)
- virtual [~PersistedVariable](#) ()
- bool [isResolvable](#) ()

A persisted variable can not be guaranteed to be present.

Public Member Functions inherited from arcs::aubo_scope::Variable

- [Variable](#) (Variable &f)
- [Variable](#) (Variable &&f)
- virtual [~Variable](#) ()
- [Type](#) [getType](#) ()
Returns the type of variable.
- std::string [getDisplayNames](#) ()
Note: The variable can be renamed at any time.
- bool [equals](#) (VariablePtr object)
Checks if the object references the same internal [Variable](#).
- size_t [hashCode](#) ()
Returns the hash code for this object.
- std::string [toString](#) ()
Returns a string representation of the variable.

Protected Member Functions

- [PersistedVariable](#) ()

Protected Member Functions inherited from arcs::aubo_scope::Variable

- [Variable](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Public Types inherited from arcs::aubo_scope::Variable

- enum [Type](#) : int { [GLOBAL](#) , [VALUE_PERSISTED](#) , [FEATURE](#) }
The variable type used to determine which type of variable this instance is.

11.88.1 Detailed Description

ingroup [ApplicationApi](#) This interface represents variables stored with the installation.

These variables are defined in the installation and will exist even when a robot is turned off and on.

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

11.88.2 Constructor & Destructor Documentation

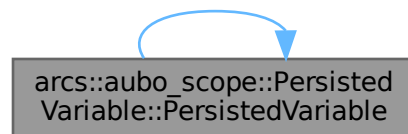
11.88.2.1 PersistedVariable() [1/3]

arcs::aubo_scope::PersistedVariable::PersistedVariable (
PersistedVariable & f)

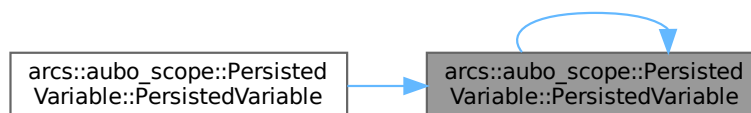
References [PersistedVariable\(\)](#).

Referenced by [PersistedVariable\(\)](#), and [PersistedVariable\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

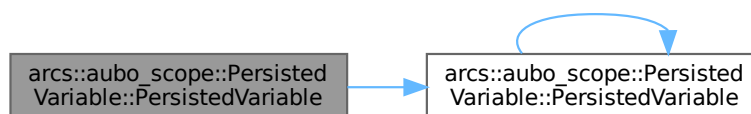


11.88.2.2 PersistedVariable() [2/3]

arcs::aubo_scope::PersistedVariable::PersistedVariable (
PersistedVariable && f)

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

Here is the call graph for this function:



11.88.2.3 ~PersistedVariable()

virtual arcs::aubo_scope::PersistedVariable::~~PersistedVariable () [virtual]

11.88.2.4 PersistedVariable() [3/3]

arcs::aubo_scope::PersistedVariable::PersistedVariable () [protected]

11.88.3 Member Function Documentation

11.88.3.1 isResolvable()

bool arcs::aubo_scope::PersistedVariable::isResolvable ()

A persisted variable can not be guaranteed to be present.

This method can be used to determine, if the variable is present.

The persisted variable will not be present, if the end user loads a different installation which does not contain the variable, or if the variable is deleted by the end user.

A program node storing a [PersistedVariable](#) in its Data Model will be undefined if the variable cannot be resolved (i.e. the variable is not present).

Returns

true, if this persisted variable is resolvable (i.e. present in the current installation).

11.88.4 Friends And Related Symbol Documentation

11.88.4.1 DataSwitch

friend class DataSwitch [friend]

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

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.88.5 Member Data Documentation

11.88.5.1 d_

void* arcs::aubo_scope::PersistedVariable::d_ { nullptr } [private]

Definition at line 54 of file [persisted_variable.h](#).

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

- include/aubo_caps/domain/variable/[persisted_variable.h](#)

11.89 arcs::aubo_scope::PersistedVariableFilter Struct Reference

#include <persisted_variable.h>

Public Member Functions

- bool [operator\(\)](#) (VariablePtr var)

11.89.1 Detailed Description

Definition at line 57 of file [persisted_variable.h](#).

11.89.2 Member Function Documentation

11.89.2.1 operator>()()

bool arcs::aubo_scope::PersistedVariableFilter::operator() (
VariablePtr var) [inline]

Definition at line 59 of file [persisted_variable.h](#).

References [arcs::aubo_scope::Variable::VALUE_PERSISTED](#).

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

- include/aubo_caps/domain/variable/[persisted_variable.h](#)

11.90 arcs::aubo_scope::PopupManager Class Reference

#include <popup_manager.h>

Public Member Functions

- [PopupManager](#) (PopupManager &f)
- [PopupManager](#) (PopupManager &&f)
- virtual [~PopupManager](#) ()
- void [notifyPopupShown](#) ()
- void [notifyPopupHidden](#) ()
- int [popupCount](#) () const
- bool [isPopupCountZero](#) () const
- void [setPopupCountChangedCallback](#) (std::function< void(bool)> callback)

Private Member Functions

- [PopupManager](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- struct [DefaultSingletonTraits< PopupManager >](#)

11.90.1 Detailed Description

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

11.90.2 Constructor & Destructor Documentation

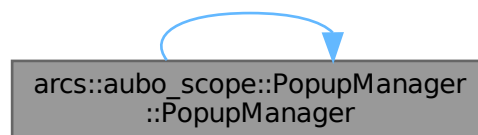
11.90.2.1 PopupManager() [1/3]

arcs::aubo_scope::PopupManager::PopupManager (
 PopupManager & f)

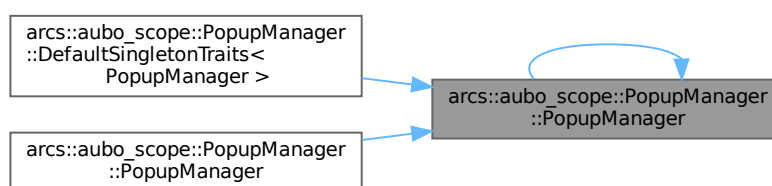
References [PopupManager\(\)](#).

Referenced by [DefaultSingletonTraits< PopupManager >](#), [PopupManager\(\)](#), and [PopupManager\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



11.90.2.2 PopupManager() [2/3]

```
arcs::aubo_scope::PopupManager::PopupManager (
    PopupManager && f)
```

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

Here is the call graph for this function:



11.90.2.3 ~PopupManager()

```
virtual arcs::aubo_scope::PopupManager::~~PopupManager () [virtual]
```

11.90.2.4 PopupManager() [3/3]

```
arcs::aubo_scope::PopupManager::PopupManager () [private]
```

11.90.3 Member Function Documentation

11.90.3.1 isPopupCountZero()

```
bool arcs::aubo_scope::PopupManager::isPopupCountZero () const
```

11.90.3.2 notifyPopupHidden()

```
void arcs::aubo_scope::PopupManager::notifyPopupHidden ()
```

11.90.3.3 notifyPopupShown()

```
void arcs::aubo_scope::PopupManager::notifyPopupShown ()
```

11.90.3.4 popupCount()

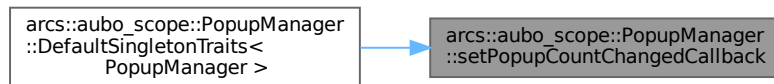
```
int arcs::aubo_scope::PopupManager::popupCount () const
```

11.90.3.5 setPopupCountChangedCallback()

```
void arcs::aubo_scope::PopupManager::setPopupCountChangedCallback (
    std::function< void(bool)> callback)
```

Referenced by [DefaultSingletonTraits< PopupManager >](#).

Here is the caller graph for this function:



11.90.4 Friends And Related Symbol Documentation

11.90.4.1 DefaultSingletonTraits< PopupManager >

friend struct [DefaultSingletonTraits< PopupManager >](#) [friend]

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

References [PopupManager\(\)](#), and [setPopupCountChangedCallback\(\)](#).

11.90.5 Member Data Documentation

11.90.5.1 d__

```
void* arcs::aubo_scope::PopupManager::d__ { nullptr } [private]
```

Definition at line 41 of file [popup_manager.h](#).

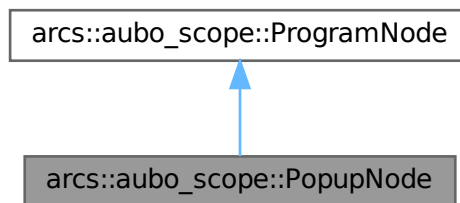
The documentation for this class was generated from the following file:

- [include/aubo_caps/domain/user_interface/popup_manager.h](#)

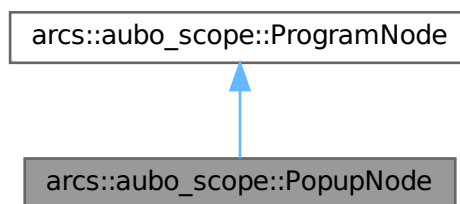
11.91 arcs::aubo_scope::PopupNode Class Reference

#include <popup_node.h>

Inheritance diagram for arcs::aubo_scope::PopupNode:



Collaboration diagram for arcs::aubo_scope::PopupNode:



Public Types

- enum `Type` { `MESSAGE` , `WARNING` , `ERROR` }

Public Member Functions

- `PopupNode` (`PopupNode` &f)
- `PopupNode` (`PopupNode` &&f)
- virtual `~PopupNode` ()
- void `setMessage` (`std::string` message)
- `std::string` `getMessage` ()
- void `setType` (`Type` type)
- `Type` `getType` () const

Public Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()
Get the program node contribution.

Private Member Functions

- [PopupNode](#) ()

Private Attributes

- void * [d__](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Protected Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.91.1 Detailed Description

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

11.91.2 Member Enumeration Documentation

11.91.2.1 Type

enum [arcs::aubo_scope::PopupNode::Type](#)

Enumerator

MESSAGE	
WARNING	
ERROR	

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

11.91.3 Constructor & Destructor Documentation

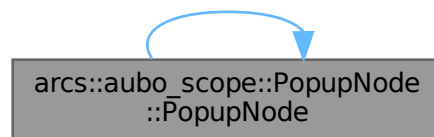
11.91.3.1 PopupNode() [1/3]

arcs::aubo_scope::PopupNode::PopupNode (
 PopupNode & f)

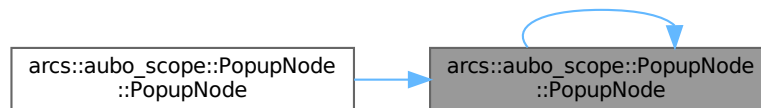
References [PopupNode\(\)](#).

Referenced by [PopupNode\(\)](#), and [PopupNode\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

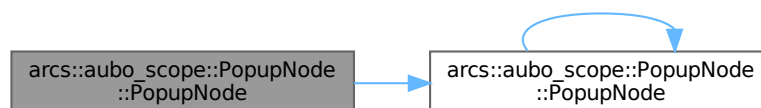


11.91.3.2 PopupNode() [2/3]

arcs::aubo_scope::PopupNode::PopupNode (
 PopupNode && f)

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

Here is the call graph for this function:



11.91.3.3 ~PopupNode()

virtual arcs::aubo_scope::PopupNode::~~PopupNode () [virtual]

11.91.3.4 PopupNode() [3/3]

arcs::aubo_scope::PopupNode::PopupNode () [private]

11.91.4 Member Function Documentation

11.91.4.1 getMessage()

std::string arcs::aubo_scope::PopupNode::getMessage ()

11.91.4.2 getType()

[Type](#) arcs::aubo_scope::PopupNode::getType () const

11.91.4.3 setMessage()

void arcs::aubo_scope::PopupNode::setMessage (
std::string message)

Parameters

message	the message in the pop up for the end user

Returns

this node

Exceptions

IllegalStateException	if called from a Qt-based AuboCap outside of an UndoableChanges scope (see also UndoRedoManager)

11.91.4.4 setType()

void arcs::aubo_scope::PopupNode::setType (
[Type](#) type)

Parameters

type	the type of pop up

Returns

this node

Exceptions

IllegalStateException	if called from a Qt-based AuboCap outside of an UndoableChanges scope (see also UndoRedoManager)

11.91.5 Friends And Related Symbol Documentation

11.91.5.1 DataSwitch

friend class DataSwitch [friend]

Definition at line 48 of file [popup_node.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.91.6 Member Data Documentation

11.91.6.1 d__

void* arcs::aubo_scope::PopupNode::d__ { nullptr } [private]

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

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

- [include/aubo_caps/domain/program/nodes/builtin/popup_node.h](#)

11.92 arcs::aubo_scope::PrepareToStartScrewdriverParameters Class Reference

```
#include <screw_driver_configuration.h>
```


Public Member Functions

- virtual [~PrepareToStartScrewdriverParameters](#) ()=default

11.92.1 Detailed Description

This interface provides parameters relevant for generating script code for the Prepare To Start Screwdriver capability (registered using `ScrewdriverCapabilities#registerPrepareToStartScrewdriverCapability(ScriptCodeGenerator)`)

These parameters are passed when AuboScope calls the implementation of the `ScriptCodeGenerator#generateScript(ScriptCodeGenerator)` method responsible for the generating the script code.

The interface is intentionally empty, but could have parameters in future releases.

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

11.92.2 Constructor & Destructor Documentation

11.92.2.1 ~PrepareToStartScrewdriverParameters()

```
virtual arcs::aubo_scope::PrepareToStartScrewdriverParameters::~~PrepareToStartScrewdriverParameters () [virtual],
[default]
```

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

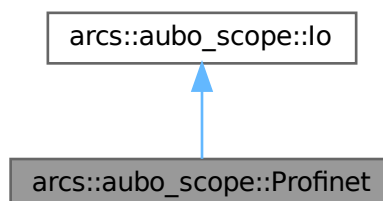
- include/aubo_caps/contribution/driver/screw_driver/[screw_driver_configuration.h](#)

11.93 arcs::aubo_scope::Profinet Class Reference

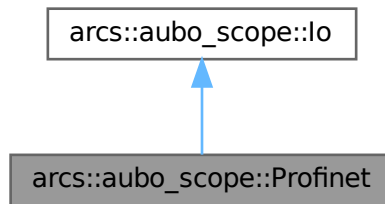
[Profinet](#) This interface provides support for PROFINET I/Os.

```
#include <profinet.h>
```

Inheritance diagram for arcs::aubo_scope::Profinet:



Collaboration diagram for `arcs::aubo_scope::Profinet`:



Public Member Functions

- [Profinet](#) (Profinet &f)
- [Profinet](#) (Profinet &&f)
- [~Profinet](#) ()
- [int getByteOffset](#) ()
Returns the byte offset for this [Profinet](#) I/O.
- [int getBitOffset](#) ()
Returns the bit offset for this [Profinet](#) I/O.
- [bool isSlave](#) ()
Returns whether this [Profinet](#) I/O is a slave.
- [bool setValue](#) (int value)
Set PROFINET signal to an integer value.
- [int getValue](#) ()
Returns the integer value of the PROFINET signal.

Public Member Functions inherited from [arcs::aubo_scope::Io](#)

- [Io](#) (Io &f)
- [Io](#) (Io &&f)
- [virtual ~Io](#) ()
- [std::string getDisplayName](#) () const
Returns the user defined name of the IO in the currently selected installation.
- [std::string getDefaultName](#) () const
Returns the robot default name of the I/O.
- [std::string getValueStr](#) () const
Returns the current reading of the I/O as a string.
- [IoType getType](#) () const
Returns the I/O type.
- [InterfaceType getInterfaceType](#) () const
Returns the interface type this I/O belongs to.
- [int getAction](#) () const
Returns the action number of the I/O.
- [bool isInput](#) () const
Returns true if I/O supports reading of values and false otherwise.
- [bool isResolvable](#) () const
Some I/Os, such as MODBUS I/Os and analog Tool inputs, are in some situations not present in AuboScope or available for use.
- [size_t hashCode](#) ()

Private Member Functions

- [Profinet](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Public Types inherited from [arcs::aubo_scope::Io](#)

- enum [IoType](#) : int {
 [DIGITAL](#) , [ANALOG](#) , [INTEGER](#) , [BOOLEAN](#) ,
 [DOUBLE](#) }
- enum [InterfaceType](#) : int {
 [STANDARD](#) , [TOOL](#) , [CONFIGURABLE](#) , [LINKIO](#) ,
 [MODBUS](#) , [GENERAL_PURPOSE](#) , [PROFINET](#) , [ETHERNET_IP](#) }

Protected Member Functions inherited from [arcs::aubo_scope::Io](#)

- [Io](#) ()

11.93.1 Detailed Description

[Profinet](#) This interface provides support for PROFINET I/Os.

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

11.93.2 Constructor & Destructor Documentation

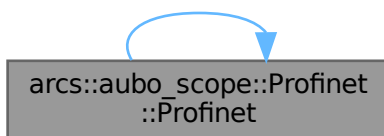
11.93.2.1 [Profinet](#)() [1/3]

```
arcs::aubo_scope::Profinet::Profinet (  
    Profinet & f)
```

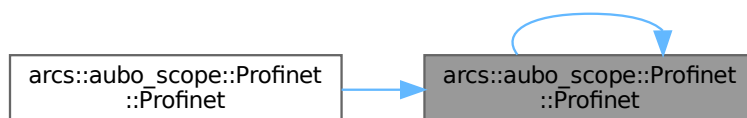
References [Profinet\(\)](#).

Referenced by [Profinet\(\)](#), and [Profinet\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

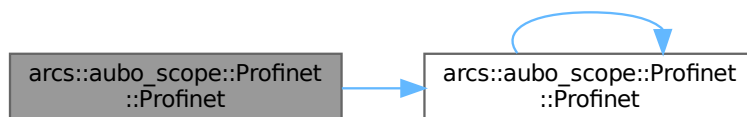


11.93.2.2 Profinet() [2/3]

```
arcs::aubo_scope::Profinet::Profinet (
    Profinet && f)
```

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

Here is the call graph for this function:



11.93.2.3 ~Profinet()

```
arcs::aubo_scope::Profinet::~~Profinet ()
```

11.93.2.4 Profinet() [3/3]

arcs::aubo_scope::Profinet::Profinet () [private]

11.93.3 Member Function Documentation

11.93.3.1 getBitOffset()

int arcs::aubo_scope::Profinet::getBitOffset ()

Returns the bit offset for this [Profinet](#) I/O.

Returns

Get the bit offset for this [Profinet](#) I/O.

11.93.3.2 getByteOffset()

int arcs::aubo_scope::Profinet::getByteOffset ()

Returns the byte offset for this [Profinet](#) I/O.

Returns

Get the byte offset for this [Profinet](#) I/O.

11.93.3.3 getValue()

int arcs::aubo_scope::Profinet::getValue ()

Returns the integer value of the PROFINET signal.

If the PROFINET signal is digital then false is represented by 0 and true is represented by the value 1.

Returns

the integer value of the PROFINET signal.

11.93.3.4 isSlave()

bool arcs::aubo_scope::Profinet::isSlave ()

Returns whether this [Profinet](#) I/O is a slave.

Returns

true if this is a slave.

Generated by Doxygen

11.93.3.5 setValue()

Parameters

value	the integer value to set.

Returns

true if the PROFINET signal was sent down to the controller, and false if the signal was not sent down, e.g. because the controller was not running. Notice that the PROFINET signal must support writes, consult the Aubo PROFINET protocol.

11.93.4 Friends And Related Symbol Documentation

11.93.4.1 DataSwitch

friend class DataSwitch [friend]

Definition at line 109 of file [profinet.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.93.5 Member Data Documentation

11.93.5.1 d_

void* arcs::aubo_scope::Profinet::d_ { nullptr } [private]

Definition at line 111 of file [profinet.h](#).

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

- [include/aubo_caps/domain/io/profinet.h](#)

11.94 arcs::aubo_scope::ProfinetFilter Struct Reference

```
#include <profinet.h>
```

Public Member Functions

- [ProfinetFilter](#) ([Io::IoType](#) type, bool is__input)
- bool [operator\(\)](#) ([IoPtr](#) io)

Private Attributes

- [Io::IoType](#) `type__`
- `bool` `is_input__`

11.94.1 Detailed Description

Definition at line 114 of file [profinet.h](#).

11.94.2 Constructor & Destructor Documentation

11.94.2.1 ProfinetFilter()

```
arcs::aubo_scope::ProfinetFilter::ProfinetFilter (  
    Io::IoType type,  
    bool is_input) [inline]
```

Definition at line 116 of file [profinet.h](#).

References [is_input__](#), and [type__](#).

11.94.3 Member Function Documentation

11.94.3.1 operator()()

```
bool arcs::aubo_scope::ProfinetFilter::operator() (  
    IoPtr io) [inline]
```

Definition at line 120 of file [profinet.h](#).

References [is_input__](#), and [type__](#).

11.94.4 Member Data Documentation

11.94.4.1 is_input__

```
bool arcs::aubo_scope::ProfinetFilter::is_input__ [private]
```

Definition at line 128 of file [profinet.h](#).

Referenced by [operator\(\)\(\)](#), and [ProfinetFilter\(\)](#).

11.94.4.2 type__

```
Io::IoType arcs::aubo_scope::ProfinetFilter::type__ [private]
```

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

Referenced by [operator\(\)\(\)](#), and [ProfinetFilter\(\)](#).

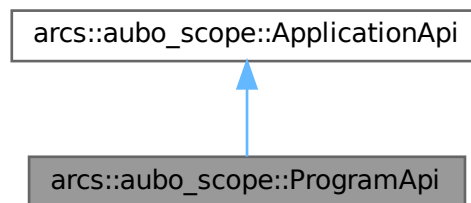
The documentation for this struct was generated from the following file:

- `include/aubo_caps/domain/io/profinet.h`

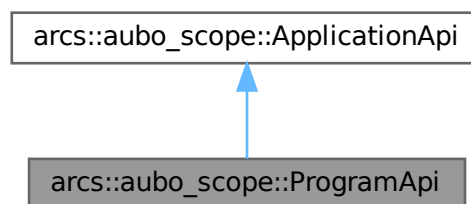
11.95 arcs::aubo_scope::ProgramApi Class Reference

#include <program_api.h>

Inheritance diagram for arcs::aubo_scope::ProgramApi:



Collaboration diagram for arcs::aubo_scope::ProgramApi:



Public Member Functions

- [ProgramApi](#) (ProgramApi &f)
- [ProgramApi](#) (ProgramApi &&f)
- virtual [~ProgramApi](#) ()
- ProgramModelPtr [getProgramModel](#) ()
 [ProgramModel](#)
- WaypointModelPtr [getWaypointModel](#) ()
 [WaypointModel](#)
- UndoRedoManagerPtr [getUndoRedoManager](#) ()
 [UndoRedoManager](#)

Public Member Functions inherited from [arcs::aubo_scope::ApplicationApi](#)

- [ApplicationApi](#) (ApplicationApi &f)
- [ApplicationApi](#) (ApplicationApi &&f)
- virtual [~ApplicationApi](#) ()
- IoModelPtr [getIoModel](#) ()
[IoModel](#) Returns an interface for working with the inputs and outputs of the robot.
- FeatureModelPtr [getFeatureModel](#) ()
[FeatureModel](#) Returns an interface for working with the features of the current installation.
- TcpModelPtr [getTcpModel](#) ()
[TcpModel](#) Returns an interface for accessing the TCPs of the current installation.
- PayloadModelPtr [getPayloadModel](#) ()
[PayloadModel](#) Gets an interface which provides access the payloads in the current AuboScope installation.
- VariableModelPtr [getVariableModel](#) ()
[VariableModel](#) Returns an interface for working with variables.
- ValueFactoryPtr [getValueFactory](#) ()
[ValueFactory](#) Returns an interface for accessing various factories capable of creating value objects.
- template<typename T>
std::shared_ptr< T > [getInstallationNode](#) ()
This method can be used to get a specific [InstallationNodeContribution](#) instance.
- InstallationNodeContributionPtr [getInstallationNode](#) (const char *typeidOfClass)
- DeviceManagerPtr [getDeviceManager](#) ()
[DeviceManager](#) This method can be used to get a specific [DeviceManager](#) instance.
- aubo_sdk::RobotProxyPtr [getRobotProxy](#) ()
Returns the robot proxy, providing an interaction interface with the underlying robot SDK.

Private Member Functions

- [ProgramApi](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Protected Member Functions inherited from [arcs::aubo_scope::ApplicationApi](#)

- [ApplicationApi](#) ()

11.95.1 Detailed Description

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

11.95.2 Constructor & Destructor Documentation

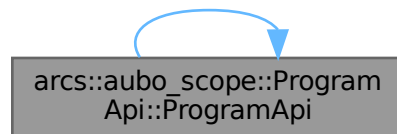
11.95.2.1 ProgramApi() [1/3]

arcs::aubo_scope::ProgramApi::ProgramApi (
ProgramApi & f)

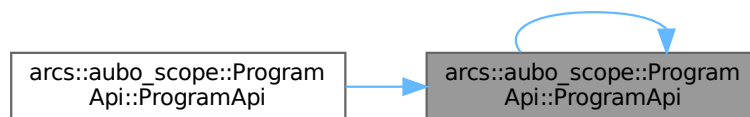
References [ProgramApi\(\)](#).

Referenced by [ProgramApi\(\)](#), and [ProgramApi\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

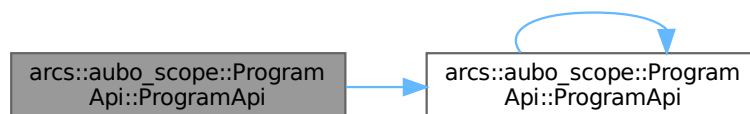


11.95.2.2 ProgramApi() [2/3]

arcs::aubo_scope::ProgramApi::ProgramApi (
ProgramApi && f)

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

Here is the call graph for this function:



11.95.2.3 ~ProgramApi()

virtual arcs::aubo_scope::ProgramApi::~~ProgramApi () [virtual]

11.95.2.4 ProgramApi() [3/3]

arcs::aubo_scope::ProgramApi::ProgramApi () [private]

11.95.3 Friends And Related Symbol Documentation

11.95.3.1 DataSwitch

friend class DataSwitch [friend]

Definition at line 75 of file [program_api.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.95.4 Member Data Documentation

11.95.4.1 d__

void* arcs::aubo_scope::ProgramApi::d__ { nullptr } [private]

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

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

- include/aubo_caps/domain/program/[program_api.h](#)

11.96 arcs::aubo_scope::ProgramApiProvider Class Reference

[ProgramApiProvider](#) Provides access to functionality and services relevant for program nodes.

```
#include <program_api_provider.h>
```

Public Member Functions

- [ProgramApiProvider](#) (ProgramApiProvider &f)
- [ProgramApiProvider](#) (ProgramApiProvider &&f)
- virtual [~ProgramApiProvider](#) ()
- SystemApiPtr [getSystemApi](#) ()
Provides access to system related functionality
- UserInterfaceApiPtr [getUserInterfaceApi](#) ()
Provides access to functionality related to user interface and end user interaction
- ProgramApiPtr [getProgramApi](#) ()
Provides access to functionality relevant related to programs

Private Member Functions

- [ProgramApiProvider](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

11.96.1 Detailed Description

[ProgramApiProvider](#) Provides access to functionality and services relevant for program nodes.

This includes what is available from within AuboScope, as well as creating additional model elements to be used within AuboScope.

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

11.96.2 Constructor & Destructor Documentation

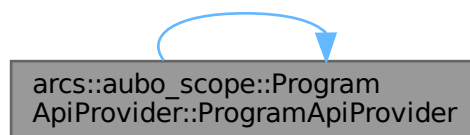
11.96.2.1 ProgramApiProvider() [1/3]

```
arcs::aubo_scope::ProgramApiProvider::ProgramApiProvider (
    ProgramApiProvider & f)
```

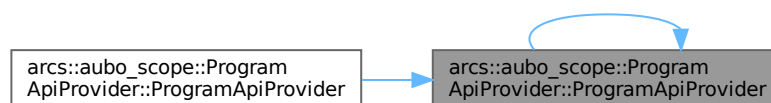
References [ProgramApiProvider\(\)](#).

Referenced by [ProgramApiProvider\(\)](#), and [ProgramApiProvider\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

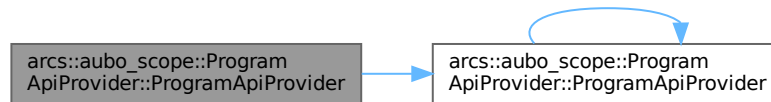


11.96.2.2 ProgramApiProvider() [2/3]

```
arcs::aubo_scope::ProgramApiProvider::ProgramApiProvider (
    ProgramApiProvider && f)
```

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

Here is the call graph for this function:



11.96.2.3 ~ProgramApiProvider()

```
virtual arcs::aubo_scope::ProgramApiProvider::~~ProgramApiProvider () [virtual]
```

11.96.2.4 ProgramApiProvider() [3/3]

```
arcs::aubo_scope::ProgramApiProvider::ProgramApiProvider () [private]
```

11.96.3 Member Function Documentation

11.96.3.1 getProgramApi()

```
ProgramApiPtr arcs::aubo_scope::ProgramApiProvider::getProgramApi ()
```

Provides access to functionality relevant related to programs

Returns

an instance of ProgramAPI

11.96.3.2 getSystemApi()

```
SystemApiPtr arcs::aubo_scope::ProgramApiProvider::getSystemApi ()
```

Provides access to system related functionality

Returns

an instance of SystemAPI

11.96.3.3 getUserInterfaceApi()

UserInterfaceApiPtr arcs::aubo_scope::ProgramApiProvider::getUserInterfaceApi ()

Provides access to functionality related to user interface and end user interaction

Returns

an instance of UserInterfaceAPI

11.96.4 Friends And Related Symbol Documentation

11.96.4.1 DataSwitch

friend class DataSwitch [friend]

Definition at line 78 of file [program_api_provider.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.96.5 Member Data Documentation

11.96.5.1 d__

void* arcs::aubo_scope::ProgramApiProvider::d__ { nullptr } [private]

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

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

- [include/aubo_caps/contribution/program/program_api_provider.h](#)

11.97 arcs::aubo_scope::ProgramContributionConfiguration Class Reference

[ProgramContributionConfiguration](#) Provides access to configure the properties of an installation node contribution.

```
#include <program_contribution_configuration.h>
```

Public Member Functions

- [ProgramContributionConfiguration](#) ([ProgramContributionConfiguration](#) &f)
- [ProgramContributionConfiguration](#) ([ProgramContributionConfiguration](#) &&f)
- virtual [~ProgramContributionConfiguration](#) ()
- void [setChildrenAllowed](#) (bool childrenAllowed)

If set to true it is possible for the program node to have child nodes.
- void [setChildrenAllowedOnly](#) (const std::vector< std::string > &childrenList)
- void [setUserInsertable](#) (bool userInsertable)

If set to false this program node will not appear in the Program Tab in AuboScope and can not be inserted in the program tree by the end user.
- void [setDeprecated](#) (bool deprecated)

If set to true it is not possible for the end user to create new program nodes of this type.
- void [setDisplayOrderId](#) (double displayOrderId)

Set the display order id for this type of program node contribution.
- void [setAllowBreakpointOnNode](#) (bool isBreakpointAllowed)
- void [setAllowBreakpointOnChildNodesInSubtree](#) (bool isBreakpointOnChildNodesAllowed)
- void [setAllowStartFromNode](#) (bool isStartFromNodeAllowed)
- void [setAllowStartFromChildNodesInSubtree](#) (bool isStartFromChildNodesAllowed)
- void [setAllowInsertOnlyIntoNodes](#) (const std::vector< std::string > &parentNodeNames)

Configure parent node of this node.

Private Member Functions

- [ProgramContributionConfiguration](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

11.97.1 Detailed Description

[ProgramContributionConfiguration](#) Provides access to configure the properties of an installation node contribution.

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

11.97.2 Constructor & Destructor Documentation

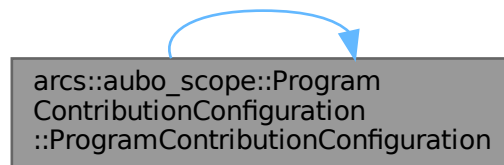
11.97.2.1 ProgramContributionConfiguration() [1/3]

arcs::aubo_scope::ProgramContributionConfiguration::ProgramContributionConfiguration (
ProgramContributionConfiguration & f)

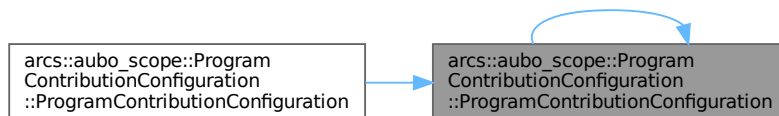
References [ProgramContributionConfiguration\(\)](#).

Referenced by [ProgramContributionConfiguration\(\)](#), and [ProgramContributionConfiguration\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

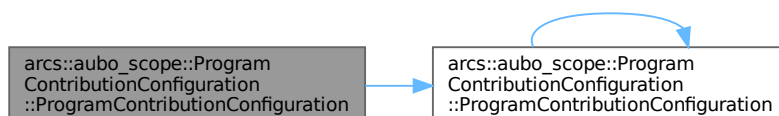


11.97.2.2 ProgramContributionConfiguration() [2/3]

arcs::aubo_scope::ProgramContributionConfiguration::ProgramContributionConfiguration (
ProgramContributionConfiguration && f)

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

Here is the call graph for this function:



11.97.2.3 ~ProgramContributionConfiguration()

```
virtual arcs::aubo_scope::ProgramContributionConfiguration::~~ProgramContributionConfiguration () [virtual]
```

11.97.2.4 ProgramContributionConfiguration() [3/3]

```
arcs::aubo_scope::ProgramContributionConfiguration::ProgramContributionConfiguration () [private]
```

11.97.3 Member Function Documentation

11.97.3.1 setAllowBreakpointOnChildNodesInSubtree()

```
void arcs::aubo_scope::ProgramContributionConfiguration::setAllowBreakpointOnChildNodesInSubtree (
    bool isBreakpointOnChildNodesAllowed)
```

Configure whether or not the end user is allowed to set a breakpoint on or single step any of the child nodes in the sub-tree under the AuboCap program node. Note that it will only be possible to set a breakpoint on a child node if the child node itself supports breakpoints.

The default value is false.

Parameters

isBreakpointOnChildNodesAllowed	true if it should be allowed to set a breakpoint on or single step any of the child nodes in the sub-tree under this AuboCap program node. If it should be disallowed, the value must be false.

11.97.3.2 setAllowBreakpointOnNode()

```
void arcs::aubo_scope::ProgramContributionConfiguration::setAllowBreakpointOnNode (
    bool isBreakpointAllowed)
```

Configure whether or not the end user is allowed to set a breakpoint on or single step the AuboCap program node.

The default value is true.

Parameters

isBreakpointAllowed	true if it should be allowed to set a breakpoint on or single step this AuboCap program node. If it should be disallowed, the value must be false.

Parameters

parentNodeName	

11.97.3.4 setAllowStartFromChildNodesInSubtree()

```
void arcs::aubo_scope::ProgramContributionConfiguration::setAllowStartFromChildNodesInSubtree (
    bool isStartFromChildNodesAllowed)
```

Configure whether or not the end user can start the program directly from any selected child node in the sub-tree under the AuboCap program node. Note that it will only be possible to start from a selected child node if the child node itself supports starting from the child node.

The default value is false.

Parameters

isStartFromChildNodesAllowed	true if it should be allowed to start the program directly from any child program node. If it should be disallowed, the value must be false..

11.97.3.5 setAllowStartFromNode()

```
void arcs::aubo_scope::ProgramContributionConfiguration::setAllowStartFromNode (
    bool isStartFromNodeAllowed)
```

Configure whether or not the end user can start the program from the AuboCap program node when it is selected in the program tree.

The default value is true.

Parameters

isStartFromNodeAllowed	true if it should be allowed to start the program directly from this AuboCap program node. If it should be disallowed, the value must be false.

11.97.3.6 setChildrenAllowed()

```
void arcs::aubo_scope::ProgramContributionConfiguration::setChildrenAllowed (
    bool childrenAllowed)
```

If set to true it is possible for the program node to have child nodes.

The default value is false.

Parameters

childrenAllowed	determines whether this node can have child nodes

11.97.3.7 setChildrenAllowedOnly()

```
void arcs::aubo_scope::ProgramContributionConfiguration::setChildrenAllowedOnly (
    const std::vector< std::string > & childrenList)
```

11.97.3.8 setDeprecated()

```
void arcs::aubo_scope::ProgramContributionConfiguration::setDeprecated (
    bool deprecated)
```

If set to true it is not possible for the end user to create new program nodes of this type.

Loading of existing programs will however still be supported for this program node.

The default value is false.

Parameters

deprecated	determines whether this node is deprecated

11.97.3.9 setDisplayOrderId()

```
void arcs::aubo_scope::ProgramContributionConfiguration::setDisplayOrderId (
    double displayOrderId)
```

Set the display order id for this type of program node contribution.

This makes it possible to specify how all the program node contributions from a AuboCap should be ordered/sorted in AuboScope.

All program node contributions from a AuboCap will be grouped together and sorted (in ascending order) according to their display order id.

The default value is 0.0. If the display order id for all program node contributions in a AuboCap have the default value, the standard built-in sorting in AuboScope will be used.

Parameters

displayOrderId	Determines the display order of the AuboCap program node type

11.97.3.10 setUserInsertable()

```
void arcs::aubo_scope::ProgramContributionConfiguration::setUserInsertable (
    bool userInsertable)
```

If set to false this program node will not appear in the Program Tab in AuboScope and can not be inserted in the program tree by the end user.

It can only be inserted programmatically using the [ProgramNodeFactory](#). This also means, it can not be inserted using copy/paste of that node directly. It can however be inserted by copy/paste of an ancestor of such a node.

The default value is true.

Parameters

userInsertable	determines whether the end user can insert this node

11.97.4 Friends And Related Symbol Documentation

11.97.4.1 DataSwitch

```
friend class DataSwitch [friend]
```

Definition at line 181 of file [program_contribution_configuration.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.97.5 Member Data Documentation

11.97.5.1 d__

```
void* arcs::aubo_scope::ProgramContributionConfiguration::d__ { nullptr } [private]
```

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

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

- include/aubo_caps/contribution/program/[program_contribution_configuration.h](#)

11.98 arcs::aubo_scope::ProgramCreationContext Class Reference

[ProgramCreationContext](#) Provides information about the context in which an installation node is created.

```
#include <program_creation_context.h>
```

Public Types

- enum [NodeCreationType](#) { [NEW](#) , [LOAD](#) }

Public Member Functions

- [ProgramCreationContext](#) (ProgramCreationContext &f)
- [ProgramCreationContext](#) (ProgramCreationContext &&f)
- [~ProgramCreationContext](#) ()
- [NodeCreationType](#) [getNodeCreationType](#) ()
- std::string [getCreatedVersion](#) ()
- std::string [getLastSavedVersion](#) ()

Private Member Functions

- [ProgramCreationContext](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

11.98.1 Detailed Description

[ProgramCreationContext](#) Provides information about the context in which an installation node is created.

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

11.98.2 Member Enumeration Documentation

11.98.2.1 NodeCreationType

enum [arcs::aubo_scope::ProgramCreationContext::NodeCreationType](#)

Enumerator

NEW	
LOAD	

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

11.98.3 Constructor & Destructor Documentation

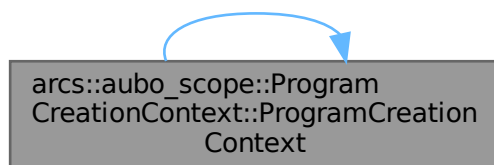
11.98.3.1 ProgramCreationContext() [1/3]

```
arcs::aubo_scope::ProgramCreationContext::ProgramCreationContext (
    ProgramCreationContext & f)
```

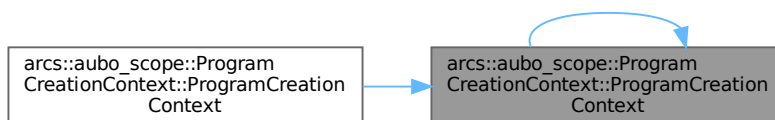
References [ProgramCreationContext\(\)](#).

Referenced by [ProgramCreationContext\(\)](#), and [ProgramCreationContext\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

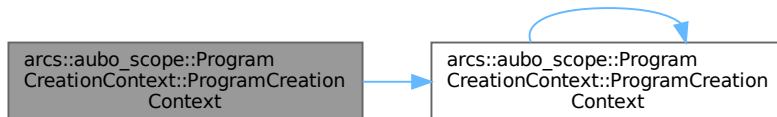


11.98.3.2 ProgramCreationContext() [2/3]

```
arcs::aubo_scope::ProgramCreationContext::ProgramCreationContext (
    ProgramCreationContext && f)
```

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

Here is the call graph for this function:



11.98.3.3 ~ProgramCreationContext()

```
arcs::aubo_scope::ProgramCreationContext::~~ProgramCreationContext ()
```

11.98.3.4 ProgramCreationContext() [3/3]

```
arcs::aubo_scope::ProgramCreationContext::ProgramCreationContext () [private]
```

11.98.4 Member Function Documentation

11.98.4.1 getCreatedVersion()

```
std::string arcs::aubo_scope::ProgramCreationContext::getCreatedVersion ()
```

11.98.4.2 getLastSavedVersion()

```
std::string arcs::aubo_scope::ProgramCreationContext::getLastSavedVersion ()
```

11.98.4.3 getNodeCreationType()

```
NodeCreationType arcs::aubo_scope::ProgramCreationContext::getNodeCreationType ()
```

11.98.5 Friends And Related Symbol Documentation

11.98.5.1 DataSwitch

friend class DataSwitch [friend]

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

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.98.6 Member Data Documentation

11.98.6.1 d__

void* arcs::aubo_scope::ProgramCreationContext::d__ { nullptr } [private]

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

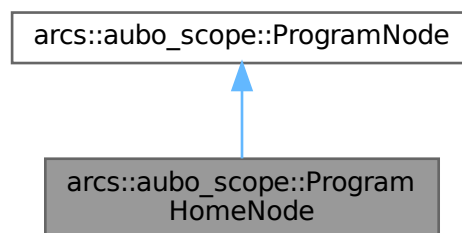
The documentation for this class was generated from the following file:

- include/aubo_caps/contribution/program/[program_creation_context.h](#)

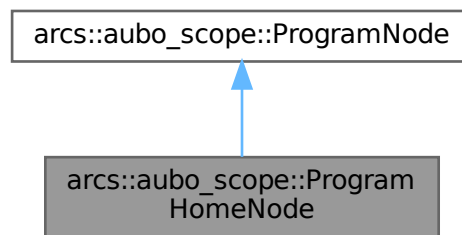
11.99 arcs::aubo_scope::ProgramHomeNode Class Reference

```
#include <program_home_node.h>
```

Inheritance diagram for arcs::aubo_scope::ProgramHomeNode:



Collaboration diagram for arcs::aubo_scope::ProgramHomeNode:



Public Member Functions

- [ProgramHomeNode](#) (ProgramHomeNode &f)
- [ProgramHomeNode](#) (ProgramHomeNode &&f)
- virtual [~ProgramHomeNode](#) ()

Public Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()
Get the program node contribution.

Private Member Functions

- [ProgramHomeNode](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Protected Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.99.1 Detailed Description

Definition at line 12 of file [program_home_node.h](#).

11.99.2 Constructor & Destructor Documentation

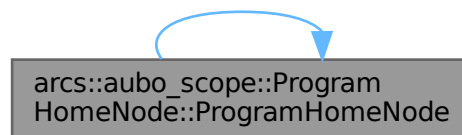
11.99.2.1 ProgramHomeNode() [1/3]

```
arcs::aubo_scope::ProgramHomeNode::ProgramHomeNode (
    ProgramHomeNode & f)
```

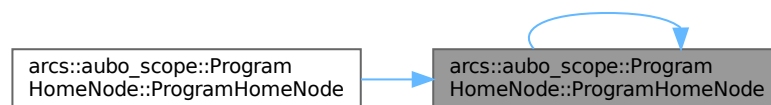
References [ProgramHomeNode\(\)](#).

Referenced by [ProgramHomeNode\(\)](#), and [ProgramHomeNode\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

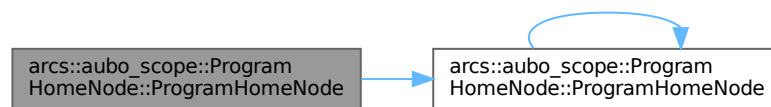


11.99.2.2 ProgramHomeNode() [2/3]

```
arcs::aubo_scope::ProgramHomeNode::ProgramHomeNode (
    ProgramHomeNode && f)
```

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

Here is the call graph for this function:



11.99.2.3 `~ProgramHomeNode()`

`virtual arcs::aubo_scope::ProgramHomeNode::~~ProgramHomeNode () [virtual]`

11.99.2.4 `ProgramHomeNode()` [3/3]

`arcs::aubo_scope::ProgramHomeNode::ProgramHomeNode () [private]`

11.99.3 Friends And Related Symbol Documentation

11.99.3.1 `DataSwitch`

`friend class DataSwitch [friend]`

Definition at line 22 of file [program_home_node.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.99.4 Member Data Documentation

11.99.4.1 `d_`

`void* arcs::aubo_scope::ProgramHomeNode::d_ { nullptr } [private]`

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

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

- [include/aubo_caps/domain/program/nodes/builtin/program_home_node.h](#)

11.100 arcs::aubo_scope::ProgramModel Class Reference

[ProgramModel](#)

`#include <program_model.h>`

Public Member Functions

- [ProgramModel](#) ([ProgramModel](#) &f)
- [ProgramModel](#) ([ProgramModel](#) &&f)
- virtual [~ProgramModel](#) ()
- [ProgramNodeFactoryPtr](#) [getProgramNodeFactory](#) ()
This method returns a [ProgramNodeFactory](#) to create program nodes.
- [TreeNodePtr](#) [getTreeNode](#) ([ProgramNodeContribution](#) *root)
Gets the [TreeNode](#) root from [ProgramNodeContribution](#).

Private Member Functions

- [ProgramModel](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

11.100.1 Detailed Description

[ProgramModel](#)

Summary

This interface makes it possible to modify sub-trees of aubo_studio plugins. The structure of trees is separated from the contents of the nodes. The [TreeNode](#) interface is concerned with the structure of trees whereas the [ProgramNode](#) interface deals with the concrete content of the nodes of the tree.

Getting started with program modification

A [ProgramModel](#) contains a:

- [getProgramNodeFactory\(\)](#) - that makes it possible to create new program nodes (both built-in AuboScope nodes and AuboCap program nodes).
- [getRootTreeNode\(ProgramNodeContribution root\)](#) - returns the sub-tree of a given AuboCap [ProgramNodeContribution](#)

The [TreeNode](#) interface supports insertion and removal of nodes. With it you can retrieve the list of children of particular nodes. Calling [lockChildSequence](#) on a tree node, locks the immediate children under the node. I.e. children can not be rearranged, deleted or have other nodes inserted into the child sequence by the end user.

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

11.100.2 Constructor & Destructor Documentation

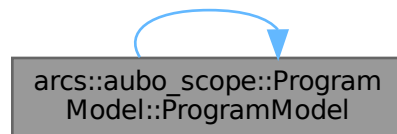
11.100.2.1 ProgramModel() [1/3]

arcs::aubo_scope::ProgramModel::ProgramModel (
ProgramModel & f)

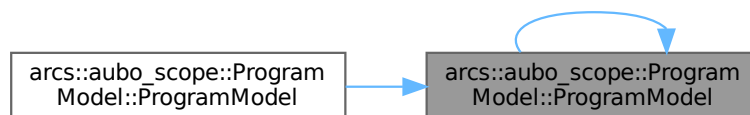
References [ProgramModel\(\)](#).

Referenced by [ProgramModel\(\)](#), and [ProgramModel\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

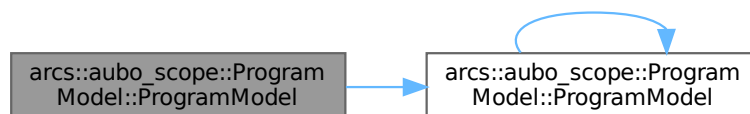


11.100.2.2 ProgramModel() [2/3]

arcs::aubo_scope::ProgramModel::ProgramModel (
ProgramModel && f)

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

Here is the call graph for this function:



11.100.2.3 ~ProgramModel()

virtual arcs::aubo_scope::ProgramModel::~~ProgramModel () [virtual]

11.100.2.4 ProgramModel() [3/3]

arcs::aubo_scope::ProgramModel::ProgramModel () [private]

11.100.3 Member Function Documentation

11.100.3.1 getProgramNodeFactory()

ProgramNodeFactoryPtr arcs::aubo_scope::ProgramModel::getProgramNodeFactory ()

This method returns a [ProgramNodeFactory](#) to create program nodes.

Returns

the factory to create program nodes.

11.100.3.2 getTreeNode()

TreeNodePtr arcs::aubo_scope::ProgramModel::getTreeNode (
[ProgramNodeContribution](#) * root)

Gets the [TreeNode](#) root from [ProgramNodeContribution](#).

From here, children can be added to form a sub-tree.

Parameters

root	The AuboCap program node where a sub-tree is to be rooted.

Returns

Returns a [TreeNode](#) root.

11.100.4 Friends And Related Symbol Documentation

11.100.4.1 DataSwitch

friend class DataSwitch [friend]

Definition at line 98 of file [program_model.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.100.5 Member Data Documentation

11.100.5.1 d_

`void* arcs::aubo_scope::ProgramModel::d_ { nullptr } [private]`

Definition at line 100 of file [program_model.h](#).

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

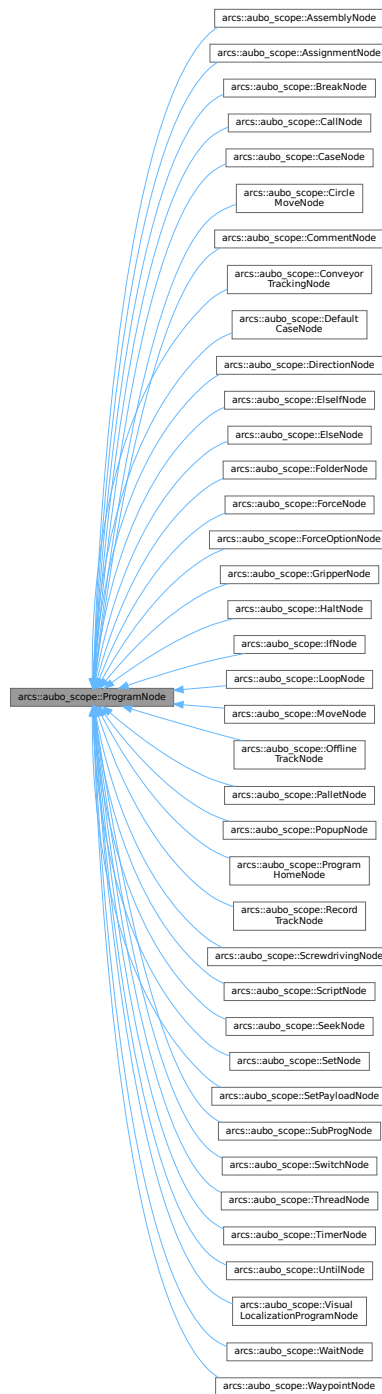
- [include/aubo_caps/domain/program/program_model.h](#)

11.101 arcs::aubo_scope::ProgramNode Class Reference

[ProgramNode](#) This interface represents AuboScope program nodes.

```
#include <program_node.h>
```

Inheritance diagram for `arcs::aubo_scope::ProgramNode`:



Public Member Functions

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()
Get the program node contribution.

Protected Member Functions

- [ProgramNode](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

11.101.1 Detailed Description

[ProgramNode](#) This interface represents AboScope program nodes.

The program nodes can be created using [ProgramNodeFactory](#).

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

11.101.2 Constructor & Destructor Documentation

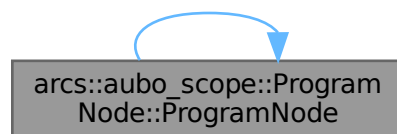
11.101.2.1 ProgramNode() [1/3]

```
arcs::aubo_scope::ProgramNode::ProgramNode (
    ProgramNode & f)
```

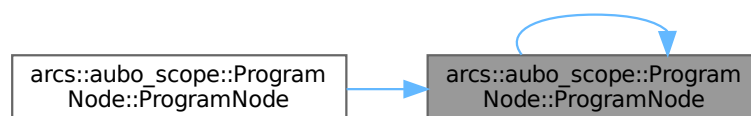
References [ProgramNode\(\)](#).

Referenced by [ProgramNode\(\)](#), and [ProgramNode\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

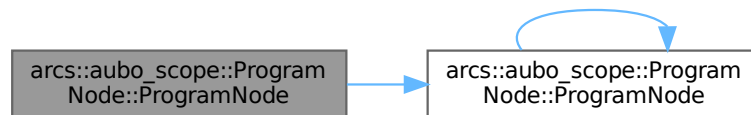


11.101.2.2 ProgramNode() [2/3]

```
arcs::aubo_scope::ProgramNode::ProgramNode (
    ProgramNode && f)
```

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

Here is the call graph for this function:



11.101.2.3 ~ProgramNode()

```
virtual arcs::aubo_scope::ProgramNode::~~ProgramNode () [virtual]
```

11.101.2.4 ProgramNode() [3/3]

```
arcs::aubo_scope::ProgramNode::ProgramNode () [protected]
```

11.101.3 Member Function Documentation

11.101.3.1 getProgramNodeContribution()

```
ProgramNodeContributionPtr arcs::aubo_scope::ProgramNode::getProgramNodeContribution ()
```

Get the program node contribution.

Not recommended to call [ProgramNodeContribution::generateScript](#) directly, use `ScriptWriter::writeChildren` instead.

Returns

the [ProgramNodeContribution](#)

11.101.4 Friends And Related Symbol Documentation

11.101.4.1 DataSwitch

```
friend class DataSwitch [friend]
```

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

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.101.5 Member Data Documentation

11.101.5.1 d_

void* arcs::aubo_scope::ProgramNode::d_ { nullptr } [private]

Definition at line 52 of file [program_node.h](#).

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

- include/aubo_caps/domain/program/nodes/[program_node.h](#)

11.102 arcs::aubo_scope::ProgramNodeContribution Class Reference

[ProgramNodeContribution](#) Defines an API required for specifying a AuboCap Program Node within AuboScope.

```
#include <program_node_contribution.h>
```

Public Member Functions

- virtual [~ProgramNodeContribution](#) ()=default
- virtual void [openView](#) ()=0
Called when this node is selected in the program tree.
- virtual void [closeView](#) ()=0
Called when this node is unselected in the program tree or when navigating to another view.
- virtual std::string [getTitle](#) ()=0
- virtual bool [isDefined](#) ()=0
- virtual void [generateScript](#) (ScriptWriterPtr script_writer)=0
Defines the script code to be generated by this program node within the program.
- virtual std::string [postAction](#) (const char *pattern, const std::string &body)
- virtual std::string [getAction](#) (const char *pattern, const std::string &body)

11.102.1 Detailed Description

[ProgramNodeContribution](#) Defines an API required for specifying a AuboCap Program Node within AuboScope.

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

11.102.2 Constructor & Destructor Documentation

11.102.2.1 ~ProgramNodeContribution()

virtual arcs::aubo_scope::ProgramNodeContribution::~ProgramNodeContribution () [virtual], [default]

11.102.3 Member Function Documentation

11.102.3.1 closeView()

virtual void arcs::aubo_scope::ProgramNodeContribution::closeView () [pure virtual]

Called when this node is unselected in the program tree or when navigating to another view.

11.102.3.2 generateScript()

virtual void arcs::aubo_scope::ProgramNodeContribution::generateScript (
ScriptWriterPtr script_writer) [pure virtual]

Defines the script code to be generated by this program node within the program.

Parameters

script_writer	serves to add script commands that define the behaviour of this program node.

11.102.3.3 getAction()

virtual std::string arcs::aubo_scope::ProgramNodeContribution::getAction (
const char * pattern,
const std::string & body) [inline], [virtual]

GET

Parameters

pattern	
body	

Returns

Definition at line 153 of file [program_node_contribution.h](#).

11.102.3.4 getTitle()

virtual std::string arcs::aubo_scope::ProgramNodeContribution::getTitle () [pure virtual]

Returns

The text displayed in the Program Tree for the program node.

Exceptions

IllegalStateException	if the data model or child program sub-tree is modified

11.102.3.5 isDefined()

virtual bool arcs::aubo_scope::ProgramNodeContribution::isDefined () [pure virtual]

Should return true if this program node is complete, and false otherwise.

Undefined program nodes will be marked with yellow in the program tree. The user cannot start the program as long as it contains one or more undefined program nodes.

If this program node stores an unresolvable entity (e.g. variable/PersistedVariable or tcp/TCP) in its [DataModel](#), the program node will be treated as undefined.

Returns

true if this program node is complete and otherwise false.

Exceptions

IllegalStateException	if the data model or child program sub-tree is modified.

11.102.3.6 openView()

virtual void arcs::aubo_scope::ProgramNodeContribution::openView () [pure virtual]

Called when this node is selected in the program tree.

11.102.3.7 postAction()

virtual std::string arcs::aubo_scope::ProgramNodeContribution::postAction (
const char * pattern,
const std::string & body) [inline], [virtual]

post

Parameters

pattern	

body	

Returns

Definition at line 134 of file [program_node_contribution.h](#).

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

- [include/aubo_caps/contribution/program/program_node_contribution.h](#)

11.103 arcs::aubo_scope::ProgramNodeFactory Class Reference

[ProgramNodeFactory](#) This interfaces supplies methods to create different types of program nodes.

```
#include <program_node_factory.h>
```

Public Member Functions

- [ProgramNodeFactory](#) ([ProgramNodeFactory](#) &f)
- [ProgramNodeFactory](#) ([ProgramNodeFactory](#) &&f)
- virtual [~ProgramNodeFactory](#) ()
- [ProgramNodePtr](#) [createUserProgramNode](#) (const char *typeid_service)
This method creates a AuboCap program node which is an instance of a contribution/ProgramNodeContribution.
- [MoveNodePtr](#) [createMoveNode](#) ()
Creates a default Move node with one default [Waypoint](#) node under it.
- [MoveNodePtr](#) [createMoveNodeNoTemplate](#) ()
- [WaypointNodePtr](#) [createWaypointNode](#) ()
Creates a [Waypoint](#) node with default waypoint name (corresponds to the name generated if the end user added a new node manually).
- [WaypointNodePtr](#) [createWaypointNode](#) (const std::string &suggested_name)
- [UntilNodePtr](#) [createUntilNode](#) ()
Creates an Until Node with no until type specified.
- [DirectionNodePtr](#) [createDirectionNode](#) ()
- [WaitNodePtr](#) [createWaitNode](#) ()
- [SetNodePtr](#) [createSetNode](#) ()
- [SetPayloadNodePtr](#) [createSetPayloadNode](#) ()
- [PopupNodePtr](#) [createPopupNode](#) ()
- [HaltNodePtr](#) [createHaltNode](#) ()
- [CommentNodePtr](#) [createCommentNode](#) ()
- [FolderNodePtr](#) [createFolderNode](#) ()
- [LoopNodePtr](#) [createLoopNode](#) ()
- [AssignmentNodePtr](#) [createAssignmentNode](#) ()
- [IfNodePtr](#) [createIfNode](#) ()
- [ElseIfNodePtr](#) [createElseIfNode](#) ()
- [ElseNodePtr](#) [createElseNode](#) ()

- CircleMoveNodePtr [createCircleMoveNode](#) ()
- PalletNodePtr [createPalletNode](#) ()
- SeekNodePtr [createSeekNode](#) ()
- ForceNodePtr [createForceNode](#) ()
- ForceOptionNodePtr [createForceOptionNode](#) ()
- AssemblyNodePtr [createAssemblyNode](#) ()
- ScrewdrivingNodePtr [createScrewdrivingNode](#) ()
- ScriptNodePtr [createScriptNode](#) ()
- TimerNodePtr [createTimerNode](#) ()
- SwitchNodePtr [createSwitchNode](#) ()
- CaseNodePtr [createCaseNode](#) ()
- DefaultCaseNodePtr [createDefaultCaseNode](#) ()
- BreakNodePtr [createBreakNode](#) ()
- SubProgNodePtr [createSubProgNode](#) ()
- CallNodePtr [createCallNode](#) ()
- OfflineTrackNodePtr [createOfflineTrackNode](#) ()
- RecordTrackNodePtr [createRecordTrackNode](#) ()
- ThreadNodePtr [createThreadNode](#) ()
- ProgramHomeNodePtr [createProgramHomeNode](#) ()
- VisualLocalizationProgramNodePtr [createVisualLocalizationProgramNode](#) ()
- ConveyorTrackingNodePtr [createConveyorTrackingNode](#) ()
- GripperNodePtr [createGripperNode](#) (GripperDevicePtr gripperDevice)

Private Member Functions

- [ProgramNodeFactory](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

11.103.1 Detailed Description

[ProgramNodeFactory](#) This interfaces supplies methods to create different types of program nodes.

Both built-in program nodes (provided by AUBO Robots) and AuboCap program nodes can be created. Most methods will create a node with the same default configuration as if the end user added the node manually.

Note that this default configuration may change between AuboScope versions.

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

11.103.2 Constructor & Destructor Documentation

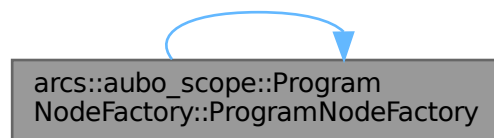
11.103.2.1 ProgramNodeFactory() [1/3]

```
arcs::aubo_scope::ProgramNodeFactory::ProgramNodeFactory (
    ProgramNodeFactory & f)
```

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

Referenced by [ProgramNodeFactory\(\)](#), and [ProgramNodeFactory\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



11.103.2.2 ProgramNodeFactory() [2/3]

```
arcs::aubo_scope::ProgramNodeFactory::ProgramNodeFactory (
    ProgramNodeFactory && f)
```

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

Here is the call graph for this function:



11.103.2.3 ~ProgramNodeFactory()

virtual arcs::aubo_scope::ProgramNodeFactory::~~ProgramNodeFactory () [virtual]

11.103.2.4 ProgramNodeFactory() [3/3]

arcs::aubo_scope::ProgramNodeFactory::ProgramNodeFactory () [private]

11.103.3 Member Function Documentation

11.103.3.1 createAssemblyNode()

AssemblyNodePtr arcs::aubo_scope::ProgramNodeFactory::createAssemblyNode ()

11.103.3.2 createAssignmentNode()

AssignmentNodePtr arcs::aubo_scope::ProgramNodeFactory::createAssignmentNode ()

11.103.3.3 createBreakNode()

BreakNodePtr arcs::aubo_scope::ProgramNodeFactory::createBreakNode ()

11.103.3.4 createCallNode()

CallNodePtr arcs::aubo_scope::ProgramNodeFactory::createCallNode ()

11.103.3.5 createCaseNode()

CaseNodePtr arcs::aubo_scope::ProgramNodeFactory::createCaseNode ()

11.103.3.6 createCircleMoveNode()

CircleMoveNodePtr arcs::aubo_scope::ProgramNodeFactory::createCircleMoveNode ()

11.103.3.7 createCommentNode()

CommentNodePtr arcs::aubo_scope::ProgramNodeFactory::createCommentNode ()

11.103.3.8 createConveyorTrackingNode()

ConveyorTrackingNodePtr arcs::aubo_scope::ProgramNodeFactory::createConveyorTrackingNode ()

11.103.3.9 createDefaultCaseNode()

DefaultCaseNodePtr arcs::aubo_scope::ProgramNodeFactory::createDefaultCaseNode ()

11.103.3.10 createDirectionNode()

DirectionNodePtr arcs::aubo_scope::ProgramNodeFactory::createDirectionNode ()

11.103.3.11 createElseIfNode()

ElseIfNodePtr arcs::aubo_scope::ProgramNodeFactory::createElseIfNode ()

11.103.3.12 createElseNode()

ElseNodePtr arcs::aubo_scope::ProgramNodeFactory::createElseNode ()

11.103.3.13 createFolderNode()

FolderNodePtr arcs::aubo_scope::ProgramNodeFactory::createFolderNode ()

11.103.3.14 createForceNode()

ForceNodePtr arcs::aubo_scope::ProgramNodeFactory::createForceNode ()

11.103.3.15 createForceOptionNode()

ForceOptionNodePtr arcs::aubo_scope::ProgramNodeFactory::createForceOptionNode ()

11.103.3.16 createGripperNode()

GripperNodePtr arcs::aubo_scope::ProgramNodeFactory::createGripperNode (
GripperDevicePtr gripperDevice)

This method creates a new Gripper program node. This node type can be used for programming grip and release actions with a selected gripper device.

The program node will be created with the same default configuration as if the end user added the node manually in AuboScope, i.e. an undefined configuration where the gripper action is unselected.

Parameters

gripper↵ Device	the gripper device to create a Gripper node for, not null. The created Gripper node will program gripper actions for this device.

Returns

a new Gripper program node (which can be inserted into the program tree using the [TreeNode](#) interface).

Exceptions

UnresolvedDeviceException	when the specified gripperDevice device is unresolved. See GripperDevice#isResolvable() .

11.103.3.17 createHaltNode()

HaltNodePtr arcs::aubo_scope::ProgramNodeFactory::createHaltNode ()

11.103.3.18 createIfNode()

IfNodePtr arcs::aubo_scope::ProgramNodeFactory::createIfNode ()

11.103.3.19 createLoopNode()

LoopNodePtr arcs::aubo_scope::ProgramNodeFactory::createLoopNode ()

11.103.3.20 createMoveNode()

MoveNodePtr arcs::aubo_scope::ProgramNodeFactory::createMoveNode ()

Creates a default Move node with one default [Waypoint](#) node under it.

Returns

new Move Node.

11.103.3.21 createMoveNodeNoTemplate()

MoveNodePtr arcs::aubo_scope::ProgramNodeFactory::createMoveNodeNoTemplate ()

Creates a default Move node without a [Waypoint](#) node under it (e.g. to be used for inserting named waypoints).

Note that at least one [Waypoint](#) node must be under it for a Move node to be valid.

Returns

new Move Node.

11.103.3.22 createOfflineTrackNode()

OfflineTrackNodePtr arcs::aubo_scope::ProgramNodeFactory::createOfflineTrackNode ()

11.103.3.23 createPalletNode()

PalletNodePtr arcs::aubo_scope::ProgramNodeFactory::createPalletNode ()

11.103.3.24 createPopupNode()

PopupNodePtr arcs::aubo_scope::ProgramNodeFactory::createPopupNode ()

11.103.3.25 createProgramHomeNode()

ProgramHomeNodePtr arcs::aubo_scope::ProgramNodeFactory::createProgramHomeNode ()

11.103.3.26 createRecordTrackNode()

RecordTrackNodePtr arcs::aubo_scope::ProgramNodeFactory::createRecordTrackNode ()

11.103.3.27 createScrewdrivingNode()

ScrewdrivingNodePtr arcs::aubo_scope::ProgramNodeFactory::createScrewdrivingNode ()

11.103.3.28 createScriptNode()

ScriptNodePtr arcs::aubo_scope::ProgramNodeFactory::createScriptNode ()

11.103.3.29 createSeekNode()

SeekNodePtr arcs::aubo_scope::ProgramNodeFactory::createSeekNode ()

11.103.3.30 createSetNode()

SetNodePtr arcs::aubo_scope::ProgramNodeFactory::createSetNode ()

11.103.3.31 createSetPayloadNode()

SetPayloadNodePtr arcs::aubo_scope::ProgramNodeFactory::createSetPayloadNode ()

11.103.3.32 createSubProgNode()

SubProgNodePtr arcs::aubo_scope::ProgramNodeFactory::createSubProgNode ()

11.103.3.33 createSwitchNode()

SwitchNodePtr arcs::aubo_scope::ProgramNodeFactory::createSwitchNode ()

11.103.3.34 createThreadNode()

ThreadNodePtr arcs::aubo_scope::ProgramNodeFactory::createThreadNode ()

11.103.3.35 createTimerNode()

TimerNodePtr arcs::aubo_scope::ProgramNodeFactory::createTimerNode ()

11.103.3.36 createUntilNode()

UntilNodePtr arcs::aubo_scope::ProgramNodeFactory::createUntilNode ()

Creates an Until Node with no until type specified.

Returns

a new Until Node.

11.103.3.37 createUserProgramNode()

ProgramNodePtr arcs::aubo_scope::ProgramNodeFactory::createUserProgramNode (
const char * typeid_service)

This method creates a AuboCap program node which is an instance of a contribution/ProgramNodeContribution.

Parameters

typeid_service	is the service creating the program node contribution

Returns

the [ProgramNode](#) which represents the contribution/ProgramNodeContribution created by [ProgramNodeService](#). Returns null if AuboCap providing [ProgramNodeService](#) is not installed.

Exceptions

IllegalArgumentException	if the argument ProgramNodeService is null or ProgramNodeService does not implement ProgramNodeService .

11.103.3.38 createVisualLocalizationProgramNode()

VisualLocalizationProgramNodePtr arcs::aubo_scope::ProgramNodeFactory::createVisualLocalizationProgramNode ()

11.103.3.39 createWaitNode()

WaitNodePtr arcs::aubo_scope::ProgramNodeFactory::createWaitNode ()

11.103.3.40 createWaypointNode() [1/2]

WaypointNodePtr arcs::aubo_scope::ProgramNodeFactory::createWaypointNode ()

Creates a [Waypoint](#) node with default waypoint name (corresponds to the name generated if the end user added a new node manually).

Returns

new [Waypoint](#) Node.

11.103.3.41 createWaypointNode() [2/2]

WaypointNodePtr arcs::aubo_scope::ProgramNodeFactory::createWaypointNode (
const std::string & suggested_name)

Creates a [Waypoint](#) node with a suggested name.

The final name can be different if the suggested name is already registered. The waypoint name is registered when it is inserted into the program tree.

[Variable](#) waypoints will use the name of the variable instead of the suggested name.

Parameters

suggested_name	the suggested name of the waypoint. Valid names must match regex <code>[a-zA-Z][a-zA-Z0-9_]{0,14}</code> for a total of 15 characters.

Returns

new [Waypoint](#) Node.

Exceptions

IllegalWaypointNameException	In case of an illegal name.

11.103.4 Friends And Related Symbol Documentation

11.103.4.1 DataSwitch

friend class DataSwitch [friend]

Definition at line 253 of file [program_node_factory.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.103.5 Member Data Documentation

11.103.5.1 d_

void* arcs::aubo_scope::ProgramNodeFactory::d_ { nullptr } [private]

Definition at line 255 of file [program_node_factory.h](#).

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

- [include/aubo_caps/domain/program/nodes/program_node_factory.h](#)

11.104 arcs::aubo_scope::ProgramNodeService Class Reference

[ProgramNodeService](#) Defines an API required for adding a new type of program node to AuboScope.

```
#include <program_node_service.h>
```

Public Member Functions

- virtual [~ProgramNodeService](#) ()=default
- virtual std::string [getId](#) ()=0
Get the unique identifier for this kind of program node.
- virtual std::string [getTitle](#) ()=0
- virtual std::string [getIcon](#) ()=0
- virtual void [configureContribution](#) (ProgramContributionConfigurationPtr configuration)=0
configureContribution
- virtual ProgramNodeViewPtr [createView](#) (ViewApiProviderPtr api_provider)=0
Creates a new View instance which implements the UI for your program node screen.
- virtual ProgramNodeContributionPtr [createNode](#) (ProgramApiProviderPtr api_provider, ProgramNodeViewPtr view, DataModelPtr model, ProgramCreationContextPtr context)=0

11.104.1 Detailed Description

[ProgramNodeService](#) Defines an API required for adding a new type of program node to AuboScope.

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

11.104.2 Constructor & Destructor Documentation

11.104.2.1 ~ProgramNodeService()

```
virtual arcs::aubo_scope::ProgramNodeService::~~ProgramNodeService () [virtual], [default]
```

11.104.3 Member Function Documentation

11.104.3.1 configureContribution()

```
virtual void arcs::aubo_scope::ProgramNodeService::configureContribution (
    ProgramContributionConfigurationPtr configuration) [pure virtual]
```

configureContribution

Parameters

configuration	

11.104.3.2 createNode()

```
virtual ProgramNodeContributionPtr arcs::aubo_scope::ProgramNodeService::createNode (
    ProgramApiProviderPtr api_provider,
    ProgramNodeViewPtr view,
    DataModelPtr model,
    ProgramCreationContextPtr context) [pure virtual]
```

Creates a new program node contribution instance.

Parameters

api_provider	the AuboCap API object with access to domain data
view	the View created by createView
model	object where all configuration data of the new program node instance is to be stored in and retrieved from
context	the creation context

Returns

the newly constructed program node contribution instance

11.104.3.3 createView()

```
virtual ProgramNodeViewPtr arcs::aubo_scope::ProgramNodeService::createView (  
    ViewApiProviderPtr api_provider) [pure virtual]
```

Creates a new View instance which implements the UI for your program node screen.

Called once when a new program is loaded or created.

Parameters

api_provider	Provides access to functionality and services available from within AuboScope related to user interface and end user interaction

Returns

the view

11.104.3.4 getIcon()

```
virtual std::string arcs::aubo_scope::ProgramNodeService::getIcon () [pure virtual]
```

Returns

The icon displayed for program nodes created by this factory.

11.104.3.5 getId()

virtual std::string arcs::aubo_scope::ProgramNodeService::getId () [pure virtual]

Get the unique identifier for this kind of program node.

The identifier will be used when storing programs that use the program node.

Returns

the unique identifier.

11.104.3.6 getTitle()

virtual std::string arcs::aubo_scope::ProgramNodeService::getTitle () [pure virtual]

Returns

The text displayed in the Structure Tab for program nodes created by this factory.

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

- include/aubo_caps/contribution/program/[program_node_service.h](#)

11.105 arcs::aubo_scope::ProgramNodeView Class Reference

[ProgramNodeView](#) API for a program node View which defines the UI for the corresponding [ProgramNodeContribution](#).

```
#include <program_node_view.h>
```

Public Member Functions

- virtual [~ProgramNodeView](#) ()=default
 - virtual void [buildUi](#) ([WidgetHandle](#) panel, [ContributionProviderPtr](#)< [ProgramNodeContribution](#) > provider)=0
- Build the UI for the corresponding [ProgramNodeContribution](#) on the provided panel.

11.105.1 Detailed Description

[ProgramNodeView](#) API for a program node View which defines the UI for the corresponding [ProgramNodeContribution](#).

The instance of [ProgramNodeView](#) is a shared entity between several contributions.

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

11.105.2 Constructor & Destructor Documentation

11.105.2.1 ~ProgramNodeView()

virtual arcs::aubo_scope::ProgramNodeView::~ProgramNodeView () [virtual], [default]

11.105.3 Member Function Documentation

11.105.3.1 buildUi()

virtual void arcs::aubo_scope::ProgramNodeView::buildUi (
[WidgetHandle](#) panel,
[ContributionProviderPtr](#)< [ProgramNodeContribution](#) > provider) [pure virtual]

Build the UI for the corresponding [ProgramNodeContribution](#) on the provided panel.

Use the contribution for your logic and data model manipulations.

Parameters

panel	the panel to build the UI on
provider	the provider for corresponding program node contribution currently selected (in the program tree)

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

- include/aubo_caps/contribution/program/[program_node_view.h](#)

11.106 arcs::aubo_scope::ProgramSelectionParameters Class Reference

```
#include <screw_driver_configuration.h>
```

Public Member Functions

- virtual [~ProgramSelectionParameters](#) ()=default
- virtual [ScrewdriverProgram](#) getScrewdriverProgram ()=0
- virtual [ScrewdriverOperationType](#) getScrewdriverOperationType ()=0

11.106.1 Detailed Description

This interface provides parameters relevant for generating script code for the Program Selection capability (registered using `ScrewdriverCapabilities#registerProgramSelectionCapability(ScrewdriverProgramListProvider, ScriptCodeGenerator)`).

These parameters are passed when AuboScope calls the implementation of the `ScriptCodeGenerator#generateScript(ScriptCodeGenerator)` method responsible for the generating the script code.

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

11.106.2 Constructor & Destructor Documentation

11.106.2.1 ~ProgramSelectionParameters()

virtual arcs::aubo_scope::ProgramSelectionParameters::~~ProgramSelectionParameters () [virtual], [default]

11.106.3 Member Function Documentation

11.106.3.1 getScrewdriverOperationType()

virtual [ScrewdriverOperationType](#) arcs::aubo_scope::ProgramSelectionParameters::getScrewdriverOperationType () [pure virtual]

Returns

The operation type selected by the end user.

Exceptions

UnsupportedOperationException	if the screwdriver has not registered this capability

11.106.3.2 getScrewdriverProgram()

virtual [ScrewdriverProgram](#) arcs::aubo_scope::ProgramSelectionParameters::getScrewdriverProgram () [pure virtual]

Returns

The selected screwdriver program to use for the operation.

Exceptions

UnsupportedOperationException	if the screwdriver has not registered this capability

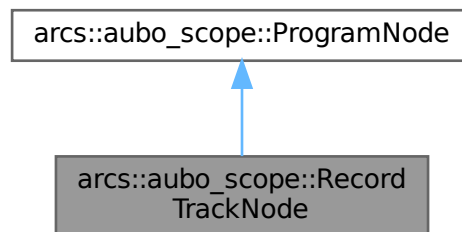
The documentation for this class was generated from the following file:

- include/aubo_caps/contribution/driver/screw_driver/[screw_driver_configuration.h](#)

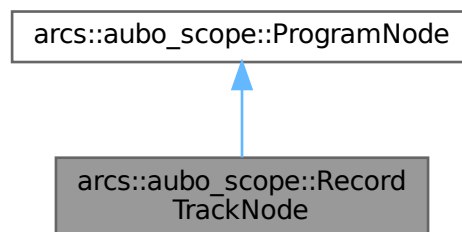
11.107 arcs::aubo_scope::RecordTrackNode Class Reference

```
#include <record_track_node.h>
```

Inheritance diagram for arcs::aubo_scope::RecordTrackNode:



Collaboration diagram for arcs::aubo_scope::RecordTrackNode:



Public Member Functions

- [RecordTrackNode](#) (RecordTrackNode &f)
- [RecordTrackNode](#) (RecordTrackNode &&f)
- virtual [~RecordTrackNode](#) ()

Public Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()
Get the program node contribution.

Private Member Functions

- [RecordTrackNode](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Protected Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.107.1 Detailed Description

Definition at line 10 of file [record_track_node.h](#).

11.107.2 Constructor & Destructor Documentation

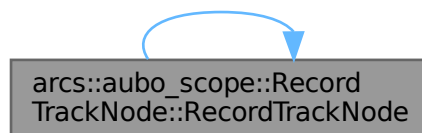
11.107.2.1 RecordTrackNode() [1/3]

```
arcs::aubo_scope::RecordTrackNode::RecordTrackNode (
    RecordTrackNode & f)
```

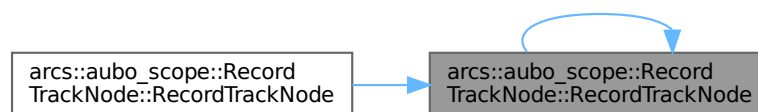
References [RecordTrackNode\(\)](#).

Referenced by [RecordTrackNode\(\)](#), and [RecordTrackNode\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

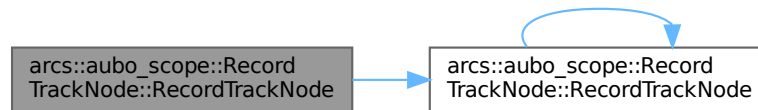


11.107.2.2 RecordTrackNode() [2/3]

arcs::aubo_scope::RecordTrackNode::RecordTrackNode (
RecordTrackNode && f)

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

Here is the call graph for this function:



11.107.2.3 ~RecordTrackNode()

virtual arcs::aubo_scope::RecordTrackNode::~~RecordTrackNode () [virtual]

11.107.2.4 RecordTrackNode() [3/3]

arcs::aubo_scope::RecordTrackNode::RecordTrackNode () [private]

11.107.3 Friends And Related Symbol Documentation

11.107.3.1 DataSwitch

friend class DataSwitch [friend]

Definition at line 18 of file [record_track_node.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.107.4 Member Data Documentation

11.107.4.1 d_

void* arcs::aubo_scope::RecordTrackNode::d_ { nullptr } [private]

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

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

- [include/aubo_caps/domain/program/nodes/builtin/record_track_node.h](#)

11.108 arcs::aubo_scope::Register Class Reference

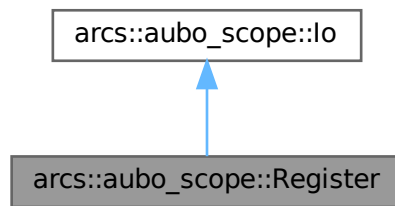
This interface provides support for registers.

```
#include <register.h>
```

Inheritance diagram for arcs::aubo_scope::Register:



Collaboration diagram for arcs::aubo_scope::Register:



Public Member Functions

- virtual [~Register](#) ()=default

Public Member Functions inherited from [arcs::aubo_scope::Io](#)

- [Io](#) (Io &f)
- [Io](#) (Io &&f)
- virtual [~Io](#) ()
- std::string [getDisplayName](#) () const
Returns the user defined name of the IO in the currently selected installation.
- std::string [getDefaultName](#) () const
Returns the robot default name of the I/O.
- std::string [getValueStr](#) () const
Returns the current reading of the I/O as a string.
- [IoType](#) [getType](#) () const

- Returns the I/O type.
- [InterfaceType](#) [getInterfaceType](#) () const
Returns the interface type this I/O belongs to.
- int [getAction](#) () const
Returns the action number of the I/O.
- bool [isInput](#) () const
Returns true if I/O supports reading of values and false otherwise.
- bool [isResolvable](#) () const
Some I/Os, such as MODBUS I/Os and analog Tool inputs, are in some situations not present in AuboScope or available for use.
- size_t [hashCode](#) ()

Additional Inherited Members

Public Types inherited from [arcs::aubo_scope::Io](#)

- enum [IoType](#) : int {
 [DIGITAL](#) , [ANALOG](#) , [INTEGER](#) , [BOOLEAN](#) ,
 [DOUBLE](#) }
- enum [InterfaceType](#) : int {
 [STANDARD](#) , [TOOL](#) , [CONFIGURABLE](#) , [LINKIO](#) ,
 [MODBUS](#) , [GENERAL_PURPOSE](#) , [PROFINET](#) , [ETHERNET_IP](#) }

Protected Member Functions inherited from [arcs::aubo_scope::Io](#)

- [Io](#) ()

11.108.1 Detailed Description

This interface provides support for registers.

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

11.108.2 Constructor & Destructor Documentation

11.108.2.1 ~Register()

virtual arcs::aubo_scope::Register::~~Register () [virtual], [default]

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

- include/aubo_caps/domain/io/[register.h](#)

11.109 arcs::aubo_scope::RegisteredCapabilities Class Reference

```
#include <registered_cpabilities.h>
```

Public Member Functions

- virtual [~RegisteredCapabilities](#) ()=default
- virtual WidthCapabilityPtr [getWidthCapability](#) ()=0
- virtual GripForceCapabilityPtr [getGripForceCapability](#) ()=0
- virtual SpeedCapabilityPtr [getSpeedCapability](#) ()=0
- virtual GripVacuumCapabilityPtr [getGripVacuumCapability](#) ()=0

11.109.1 Detailed Description

Each of the individual grippers/zones available in a multi-gripper device supports all capabilities that has been registered, i.e. all the individual grippers have the exact same set of capabilities. This interface gives access to all registered capabilities, such as the width capability (see [WidthCapability](#)), speed capability (see [SpeedCapability](#)), etc. This enables dynamic adjustment of capability properties, including the value range and default values, exclusively per individual gripper (independently of the other grippers).

This interface is only relevant if the multi-gripper capability has been registered (using the method `double, double, double, Length.Unit`)).

Each capability must be registered using the `GripperCapabilities` interface (when `GripperContribution#configureGripper()` is called) before they can be accessed using this interface.

NOTE: If an update of the properties of a capability needs to apply for all individual grippers, it is more convenient to perform the adjustment simultaneously for all grippers using the capability instance (e.g. [WidthCapability](#)) returned when the capability was registered (with the `GripperCapabilities` interface).

Definition at line 49 of file [registered_capabilities.h](#).

11.109.2 Constructor & Destructor Documentation

11.109.2.1 ~RegisteredCapabilities()

```
virtual arcs::aubo_scope::RegisteredCapabilities::~RegisteredCapabilities () [virtual], [default]
```

11.109.3 Member Function Documentation

11.109.3.1 getGripForceCapability()

```
virtual GripForceCapabilityPtr arcs::aubo_scope::RegisteredCapabilities::getGripForceCapability () [pure virtual]
```

Access the grip force capability interface which allows for dynamic adjustments of the capability properties exclusively for this specific individual gripper (independently of the other grippers), including the value range and default value.

This is only valid, if the force capability has been registered (using the method `double, double, Force.Unit`)).

Returns

The grip force capability for this individual gripper

Exceptions

UnsupportedOperationException	if the gripper has not registered the force capability

11.109.3.2 getGripVacuumCapability()

virtual GripVacuumCapabilityPtr arcs::aubo_scope::RegisteredCapabilities::getGripVacuumCapability () [pure virtual]

Access the grip vacuum capability interface which allows for dynamic adjustments of the capability properties exclusively for this specific individual gripper (independently of the other grippers), including the value range and default value.

This is only valid, if the vacuum capability has been registered (using the method double, double, Pressure.Unit)).

Returns

The grip vacuum capability for this individual gripper

Exceptions

UnsupportedOperationException	if the gripper has not registered the vacuum capability

11.109.3.3 getSpeedCapability()

virtual SpeedCapabilityPtr arcs::aubo_scope::RegisteredCapabilities::getSpeedCapability () [pure virtual]

Access the speed capability interface which allows for dynamic adjustments of the capability properties exclusively for this specific individual gripper (independently of the other grippers), including the value range and default value.

This is only valid, if the speed capability has been registered (using the method double, double, double, Speed.Unit)).

Returns

The speed capability for this individual gripper

Exceptions

UnsupportedOperationException	if the gripper has not registered the speed capability

11.109.3.4 getWidthCapability()

virtual WidthCapabilityPtr arcs::aubo_scope::RegisteredCapabilities::getWidthCapability () [pure virtual]

Exceptions

UnsupportedOperationException	if the gripper has not registered the width capability

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

- [include/aubo_caps/contribution/driver/gripper/capability/registered_cpabilities.h](#)

11.110 arcs::aubo_scope::ReleaseActionParameters Class Reference

```
#include <release_action_parameters.h>
```

Public Member Functions

- virtual [~ReleaseActionParameters](#) ()=default
- virtual double [getWidth](#) ()=0
- virtual double [getSpeed](#) ()=0
- virtual bool [isReleaseDetectionEnabled](#) ()=0

11.110.1 Detailed Description

This interface represents parameters for a release action defined/configured by the end user.

These parameters are used when generating script code for a release action (see GripperContribution#generateReleaseAction).

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

11.110.2 Constructor & Destructor Documentation

11.110.2.1 ~ReleaseActionParameters()

```
virtual arcs::aubo_scope::ReleaseActionParameters::~~ReleaseActionParameters () [virtual], [default]
```

11.110.3 Member Function Documentation

11.110.3.1 getSpeed()

```
virtual double arcs::aubo_scope::ReleaseActionParameters::getSpeed () [pure virtual]
```

Get the selected speed to be used for releasing the gripper.

This is only valid, if the speed capability has been registered (using the method double, double, double, Speed.Unit)).

Returns

The speed to use for the release action

Exceptions

UnsupportedOperationException	if the gripper has not registered the speed capability

11.110.3.2 getWidth()

virtual double arcs::aubo_scope::ReleaseActionParameters::getWidth () [pure virtual]

Get the selected width to be used for releasing the gripper.

This is only valid, if the width capability has been registered (using the method double, double, double, Length.Unit)).

Returns

The width to use for the release action

Exceptions

UnsupportedOperationException	if the gripper has not registered the width capability

11.110.3.3 isReleaseDetectionEnabled()

virtual bool arcs::aubo_scope::ReleaseActionParameters::isReleaseDetectionEnabled () [pure virtual]

Use this method to determine whether the release detection option in the Gripper program node is on or off.

Note that this method will report that release detection is off, when script code is to be generated due to the user testing a release configuration of a gripper program or operating the gripper using the toolbar.

This method should only be called, if the release detected feedback capability has been registered (using GripperFeedbackCapabilities#registerReleaseDetectedCapability(ScriptCodeGenerator)).

The purpose of this method is to ensure that the appropriate script code for performing a release action is generated when the end user has specified a new payload value. The user-defined payload value will be applied by AuboScope immediately after the generated release action script code has finished executing. Depending on the enablement state of release detection, the generated script should have the following properties:

- Release detection off: The script code should not finish earlier than when it is appropriate to apply the new payload value (i.e. when the object has been released). This typically means that the script code should wait a time period corresponding to opening the gripper's "fingers" fully (if the gripper is vacuum operated then long enough to achieve some level of vacuum).
- Release detection on: The payload will only be set if the release detection was successful, i.e. it will be applied immediately after a release was detected. This means that it is not necessary for the script code to wait (since the built-in detection timeout functionality will handle the waiting part).

Generated by Doxygen

Returns

Exceptions

UnsupportedOperationException	if the gripper has not registered the release detected feedback capability

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

- include/aubo_caps/contribution/driver/gripper/release_action_parameters.h

11.111 arcs::aubo_scope::RenderInterface Class Reference

[RenderInterface](#) Provides display option controls related to 3D rendering, including show/hide of ground, robot models, coordinate systems, trajectories and points.

```
#include <render_interface.h>
```

Public Types

- enum [ShowOptions](#) : uint16_t {
[None](#) = 0 , [Ground](#) = 0x01 , [TargetRobot](#) = 0x02 , [RealRobot](#) = 0x04 ,
[Coordinate](#) = 0x08 , [Track](#) = 0x10 , [Point](#) = 0x20 , [View](#) = 0x40 ,
[Obstacle](#) = 0x80 , [SafetyPlane](#) = 0x0100 , [ToolPosition](#) = 0x0200 , [PointCloud](#) = 0x0400 ,
[All](#) = 0xFFFF }

Public Member Functions

- [RenderInterface](#) (RenderInterface &f)
- [RenderInterface](#) (RenderInterface &&f)
- virtual [~RenderInterface](#) ()
- void [setRenderEmbedded](#) (bool is_embedded, QGridLayout *gly)
Set the render window to be embedded/removed in the layout gly.
- void [setShowOptions](#) (uint16_t options)
Set the display mode to control the visibility of various elements.
- void [addPointCloud](#) (const std::string &name, const std::vector< std::vector< double > > &points, const std::vector< double > &color)
Add/modify point cloud.
- void [removePointCloud](#) (const std::string &name)
Remove the point cloud with the specified name.
- bool [hasPointCloud](#) (const std::string &name)
Check whether the point cloud with the specified name exists.
- void [clearPointCloud](#) ()
Remove all point clouds.
- bool [addWaypoint](#) (void *wp, const std::vector< double > &pose, const std::vector< double > &color={ 0.0, 1.0, 0.0, 1.0 }, bool axis_visible=false)
Add/modify waypoint.
- void [removeWaypoint](#) (void *wp)
Remove the specified waypoint.

- bool [hasWaypoint](#) (void *wp)
Check whether the specified waypoint exists.
- void [clearWaypoints](#) ()
Remove all waypoints.
- void [setMoveTrack](#) (void *wp, const std::vector< std::vector< double > > &track)
Add/modify the move track for the specified waypoint.
- void [setBlendTrack](#) (void *wp, const std::vector< std::vector< double > > &track)
- void [setMoveTrack](#) (void *wp, const std::vector< std::vector< double > > &track, const std::vector< double > &color)
- void [setBlendTrack](#) (void *wp, const std::vector< std::vector< double > > &track, const std::vector< double > &color)
- void [addCoordinate](#) (const std::string &name, const std::vector< double > &pose)
Add/modify coordinate axis.
- void [removeCoordinate](#) (const std::string &name)
Remove the specified coordinate axis.
- bool [hasCoordinate](#) (const std::string &name)
Check whether the specified coordinate axis exists.
- void [addLine](#) (const std::string &name, const std::vector< std::vector< double > > &points, const std::vector< double > &color)
Add/modify line.
- void [addLine](#) (const std::string &name, int link_num, const std::vector< double > &pose, const std::vector< std::vector< double > > &points, const std::vector< double > &color)
- void [removeLine](#) (const std::string &name)
Remove the specified line.
- void [clearLines](#) ()
Remove all lines.
- void [addPlane](#) (const std::string &name, const std::vector< double > &pose, const std::vector< double > &size, const std::vector< double > &color)
Add/modify plane.
- void [addPlane](#) (const std::string &name, int link_num, const std::vector< double > &pose, const std::vector< double > &size, const std::vector< double > &color)
- void [removePlane](#) (const std::string &name)
Remove the specified plane.
- void [clearPlanes](#) ()
Remove all planes.
- void [addCube](#) (const std::string &name, const std::vector< double > &pose, const std::vector< double > &size, const std::vector< double > &color, int type=0)
Add/modify cube.
- void [addCube](#) (const std::string &name, int link_num, const std::vector< double > &pose, const std::vector< double > &size, const std::vector< double > &color, int type=0)
- void [addCube](#) (const std::string &name, const std::vector< double > &pose, float length_min, float length_max, float width_min, float width_max, float height_min, float height_max, const std::vector< double > &color, int type=0)
Add/modify cube with size ranges.
- void [addCube](#) (const std::string &name, int link_num, const std::vector< double > &pose, float length_min, float length_max, float width_min, float width_max, float height_min, float height_max, const std::vector< double > &color, int type=0)
- void [removeCube](#) (const std::string &name)
Remove the specified cube.
- void [clearCubes](#) ()
Remove all cubes.
- int [addMesh](#) (const std::string &name, int link_num, const std::string &model_path, const std::vector< double > &pose)

- addMesh
- `std::vector< std::string > getMeshes (int link_num)`
 - getMeshes link_num
- `int removeMesh (const std::string &name)`
 - removeMesh
- `int clearMeshes (int link_num)`
 - clearMeshes

Private Member Functions

- [RenderInterface](#) ()

Private Attributes

- `void * d\_ { nullptr }`

Friends

- class [DataSwitch](#)

11.111.1 Detailed Description

[RenderInterface](#) Provides display option controls related to 3D rendering, including show/hide of ground, robot models, coordinate systems, trajectories and points.

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

11.111.2 Member Enumeration Documentation

11.111.2.1 ShowOptions

enum [arcs::aubo_scope::RenderInterface::ShowOptions](#) : uint16_t

Enumerator

None	
Ground	
TargetRobot	
RealRobot	
Coordinate	
Track	
Point	
View	

Obstacle	
SafetyPlane	
ToolPosition	
PointCloud	
All	

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

11.111.3 Constructor & Destructor Documentation

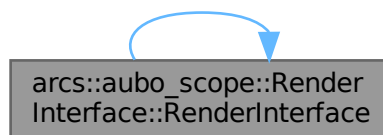
11.111.3.1 RenderInterface() [1/3]

```
arcs::aubo_scope::RenderInterface::RenderInterface (  
    RenderInterface & f)
```

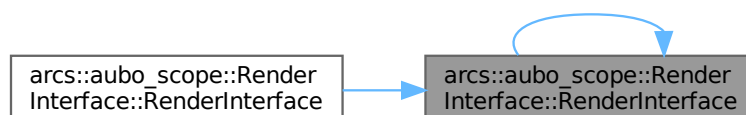
References [RenderInterface\(\)](#).

Referenced by [RenderInterface\(\)](#), and [RenderInterface\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

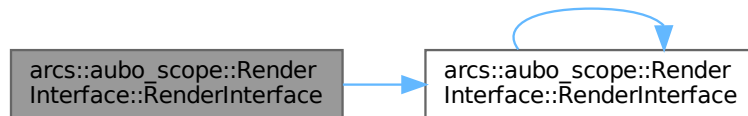


11.111.3.2 RenderInterface() [2/3]

```
arcs::aubo_scope::RenderInterface::RenderInterface (
    RenderInterface && f)
```

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

Here is the call graph for this function:



11.111.3.3 ~RenderInterface()

```
virtual arcs::aubo_scope::RenderInterface::~~RenderInterface () [virtual]
```

11.111.3.4 RenderInterface() [3/3]

```
arcs::aubo_scope::RenderInterface::RenderInterface () [private]
```

11.111.4 Member Function Documentation

11.111.4.1 addCoordinate()

```
void arcs::aubo_scope::RenderInterface::addCoordinate (
    const std::string & name,
    const std::vector< double > & pose)
```

Add/modify coordinate axis.

Parameters

name	coordinate name
pose	pose x,y,z,rx,ry,rz

11.111.4.2 addCube() [1/4]

```
void arcs::aubo_scope::RenderInterface::addCube (
    const std::string & name,
    const std::vector< double > & pose,
    const std::vector< double > & size,
```

Parameters

name	cube name
link_num	-1: TCP ; 0: base; 1~6: joints;
pose	position
size	cube size (length, width, height)
color	cube color {r, g, b, a}, range 0.0~1.0
type	cube rendering type: 0 for 6 planes, 1 for wireframe

11.111.4.3 addCube() [2/4]

```
void arcs::aubo_scope::RenderInterface::addCube (
    const std::string & name,
    const std::vector< double > & pose,
    float length_min,
    float length_max,
    float width_min,
    float width_max,
    float height_min,
    float height_max,
    const std::vector< double > & color,
    int type = 0)
```

Add/modify cube with size ranges.

Parameters

name	cube name
pose	position
length_min	
length_max	
width_min	
width_max	
height_min	
height_max	
color	cube color {r, g, b, a}, range 0.0~1.0
type	cube rendering type: 0 for 6 planes, 1 for wireframe

11.111.4.4 addCube() [3/4]

```
void arcs::aubo_scope::RenderInterface::addCube (  
    const std::string & name,  
    int link_num,  
    const std::vector< double > & pose,  
    const std::vector< double > & size,  
    const std::vector< double > & color,  
    int type = 0)
```

addCube

Parameters

name	
link_num	
pose	
size	
color	
type	

11.111.4.5 addCube() [4/4]

```
void arcs::aubo_scope::RenderInterface::addCube (  
    const std::string & name,  
    int link_num,  
    const std::vector< double > & pose,  
    float length_min,  
    float length_max,  
    float width_min,  
    float width_max,  
    float height_min,  
    float height_max,  
    const std::vector< double > & color,  
    int type = 0)
```

addCube

Parameters

name	
link_num	
pose	
length_min	
length_max	

width_min	
width_max	
height_min	
height_max	
color	
type	

11.111.4.6 addLine() [1/2]

```
void arcs::aubo_scope::RenderInterface::addLine (
    const std::string & name,
    const std::vector< std::vector< double > > & points,
    const std::vector< double > & color)
```

Add/modify line.

Parameters

name	line name
link_num	-1: TCP ; 0: base; 1~6: joints;
pose	reference coordinate system for points, default 6 zeros
points	point collection
color	line color {r, g, b, a}, range 0.0~1.0

11.111.4.7 addLine() [2/2]

```
void arcs::aubo_scope::RenderInterface::addLine (
    const std::string & name,
    int link_num,
    const std::vector< double > & pose,
    const std::vector< std::vector< double > > & points,
    const std::vector< double > & color)
```

addLine

Parameters

name	
link_num	

pose	
points	
color	

11.111.4.8 addMesh()

```
int arcs::aubo_scope::RenderInterface::addMesh (
    const std::string & name,
    int link_num,
    const std::string & model_path,
    const std::vector< double > & pose)
```

addMesh

Parameters

name	
link_num	-1: TCP ; 0: ; 1~6: ;
model_path	3ds
pose	{x, y, z, rx, ry, rz}

Returns

0: ;

11.111.4.9 addPlane() [1/2]

```
void arcs::aubo_scope::RenderInterface::addPlane (
    const std::string & name,
    const std::vector< double > & pose,
    const std::vector< double > & size,
    const std::vector< double > & color)
```

Add/modify plane.

Parameters

name	plane name
link_num	-1: TCP ; 0: base; 1~6: joints;
pose	position

size	plane size (length, width)
color	color {r, g, b, a}, range 0.0~1.0

11.111.4.10 addPlane() [2/2]

```
void arcs::aubo_scope::RenderInterface::addPlane (
    const std::string & name,
    int link_num,
    const std::vector< double > & pose,
    const std::vector< double > & size,
    const std::vector< double > & color)
```

addPlane

Parameters

name	
link_num	
pose	
size	
color	

11.111.4.11 addPointCloud()

```
void arcs::aubo_scope::RenderInterface::addPointCloud (
    const std::string & name,
    const std::vector< std::vector< double > > & points,
    const std::vector< double > & color)
```

Add/modify point cloud.

Parameters

name	point cloud name
points	point collection
color	point cloud color rgba

11.111.4.12 addWaypoint()

```
Generated by Doxygen
bool arcs::aubo_scope::RenderInterface::addWaypoint (
    void * wp,
    const std::vector< double > & pose,
    const std::vector< double > & color = { 0.0, 1.0, 0.0, 1.0 },
```

Parameters

wp	the unique object representing the waypoint
pose	waypoint pose x,y,z,rx,ry,rz
color	waypoint color {r, g, b, a}, range 0.0~1.0
axis_visible	whether to show pose direction arrows on the waypoint

Return values

true	success
false	failure

11.111.4.13 clearCubes()

```
void arcs::aubo_scope::RenderInterface::clearCubes ()
```

Remove all cubes.

11.111.4.14 clearLines()

```
void arcs::aubo_scope::RenderInterface::clearLines ()
```

Remove all lines.

11.111.4.15 clearMeshes()

```
int arcs::aubo_scope::RenderInterface::clearMeshes (
    int link_num)
```

clearMeshes

Parameters

link_num	

Returns

11.111.4.16 clearPlanes()

```
void arcs::aubo_scope::RenderInterface::clearPlanes ()
```

Remove all planes.

11.111.4.17 clearPointCloud()

```
void arcs::aubo_scope::RenderInterface::clearPointCloud ()
```

Remove all point clouds.

11.111.4.18 clearWaypoints()

```
void arcs::aubo_scope::RenderInterface::clearWaypoints ()
```

Remove all waypoints.

11.111.4.19 getMeshes()

```
std::vector< std::string > arcs::aubo_scope::RenderInterface::getMeshes (  
    int link_num)
```

```
getMeshes    link_num
```

Parameters

link_num	-1: TCP ; 0: ; 1~6: ;

Returns

11.111.4.20 hasCoordinate()

```
bool arcs::aubo_scope::RenderInterface::hasCoordinate (  
    const std::string & name)
```

Check whether the specified coordinate axis exists.

Parameters

name	coordinate name

Return values

true	exists
false	does not exist

11.111.4.21 hasPointCloud()

```
bool arcs::aubo_scope::RenderInterface::hasPointCloud (  
    const std::string & name)
```

Check whether the point cloud with the specified name exists.

Parameters

name	point cloud name

Return values

true	exists
false	does not exist

11.111.4.22 hasWaypoint()

```
bool arcs::aubo_scope::RenderInterface::hasWaypoint (  
    void * wp)
```

Check whether the specified waypoint exists.

Parameters

wp	the unique object representing the waypoint

Return values

true	exists
false	does not exist

11.111.4.23 removeCoordinate()

```
void arcs::aubo_scope::RenderInterface::removeCoordinate (  
    const std::string & name)
```

Remove the specified coordinate axis.

Parameters

name	coordinate name

11.111.4.24 removeCube()

```
void arcs::aubo_scope::RenderInterface::removeCube (  
    const std::string & name)
```

Remove the specified cube.

Parameters

name	cube name

11.111.4.25 removeLine()

```
void arcs::aubo_scope::RenderInterface::removeLine (  
    const std::string & name)
```

Remove the specified line.

Parameters

name	line name

11.111.4.26 removeMesh()

```
int arcs::aubo_scope::RenderInterface::removeMesh (  
    const std::string & name)
```

removeMesh

Parameters

name	

Returns

11.111.4.27 removePlane()

```
void arcs::aubo_scope::RenderInterface::removePlane (  
    const std::string & name)
```

Remove the specified plane.

Parameters

name	plane name

11.111.4.28 removePointCloud()

```
void arcs::aubo_scope::RenderInterface::removePointCloud (  
    const std::string & name)
```

Remove the point cloud with the specified name.

Parameters

name	point cloud name

11.111.4.29 removeWaypoint()

```
void arcs::aubo_scope::RenderInterface::removeWaypoint (  
    void * wp)
```

Remove the specified waypoint.

Parameters

wp	the unique object representing the waypoint

11.111.4.30 setBlendTrack() [1/2]

```
void arcs::aubo_scope::RenderInterface::setBlendTrack (  
    void * wp,  
    const std::vector< std::vector< double > > & track)
```

setBlendTrack

Parameters

wp	
track	

11.111.4.31 setBlendTrack() [2/2]

```
void arcs::aubo_scope::RenderInterface::setBlendTrack (  
    void * wp,  
    const std::vector< std::vector< double > > & track,  
    const std::vector< double > & color)
```

setBlendTrack

Parameters

wp	
track	
color	

11.111.4.32 setMoveTrack() [1/2]

```
void arcs::aubo_scope::RenderInterface::setMoveTrack (
    void * wp,
    const std::vector< std::vector< double > > & track)
```

Add/modify the move track for the specified waypoint.

Parameters

wp	the unique object representing the waypoint
track	waypoint pose collection x,y,z,rx,ry,rz

11.111.4.33 setMoveTrack() [2/2]

```
void arcs::aubo_scope::RenderInterface::setMoveTrack (
    void * wp,
    const std::vector< std::vector< double > > & track,
    const std::vector< double > & color)
```

setMoveTrack

Parameters

wp	
track	
color	

11.111.4.34 setRenderEmbedded()

```
void arcs::aubo_scope::RenderInterface::setRenderEmbedded (
    bool is_embedded,
    QGridLayout * gly)
```

Set the render window to be embedded/removed in the layout gly.

Parameters

is_embedded	whether to embed
gly	the layout

11.111.4.35 setShowOptions()

```
void arcs::aubo_scope::RenderInterface::setShowOptions (  
    uint16_t options)
```

Set the display mode to control the visibility of various elements.

Parameters

options	option value, see ShowOptions enum for details

11.111.5 Friends And Related Symbol Documentation

11.111.5.1 DataSwitch

```
friend class DataSwitch [friend]
```

Definition at line 682 of file [render_interface.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.111.6 Member Data Documentation

11.111.6.1 d_

```
void* arcs::aubo_scope::RenderInterface::d_ { nullptr } [private]
```

Definition at line 684 of file [render_interface.h](#).

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

- [include/aubo_caps/domain/user_interface/render_interface.h](#)

11.112 arcs::aubo_scope::ScrewdriverApiProvider Class Reference

[ScrewdriverApiProvider](#) This interface provides access to functionality and services available from within AuboScope relevant for screwdriver contributions.

```
#include <screw_driver_api_provider.h>
```

Public Member Functions

- virtual [~ScrewdriverApiProvider](#) ()=default
- virtual SystemApiPtr [getSystemAPI](#) ()=0
- virtual IoModelPtr [getIOModel](#) ()=0

11.112.1 Detailed Description

[ScrewdriverApiProvider](#) This interface provides access to functionality and services available from within AuboScope relevant for screwdriver contributions.

Definition at line 22 of file [screw_driver_api_provider.h](#).

11.112.2 Constructor & Destructor Documentation

11.112.2.1 ~ScrewdriverApiProvider()

virtual arcs::aubo_scope::ScrewdriverApiProvider::~~ScrewdriverApiProvider () [virtual], [default]

11.112.3 Member Function Documentation

11.112.3.1 getIOModel()

virtual IoModelPtr arcs::aubo_scope::ScrewdriverApiProvider::getIOModel () [pure virtual]

Returns

an I/O model interface (e.g. for getting I/Os to display in the custom user input for setting up the screwdriver)

11.112.3.2 getSystemAPI()

virtual SystemApiPtr arcs::aubo_scope::ScrewdriverApiProvider::getSystemAPI () [pure virtual]

Returns

an interface providing access to system related functionality.

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

- include/aubo_caps/contribution/driver/screw_driver/[screw_driver_api_provider.h](#)

11.113 arcs::aubo_scope::ScrewdriverConfiguration Class Reference

This interface provides access to register or setup various properties and capabilities of a screwdriver.

```
#include <screw_driver_configuration.h>
```

Public Member Functions

- virtual [~ScrewdriverConfiguration](#) ()=default
- virtual void [addScrewdriverProgram](#) (ScrewdriverProgramPtr screwdriverProgram)=0
Add a screwdriver program to the list of programs.
- virtual void [addScrewdriverPrograms](#) (std::vector< ScrewdriverProgramPtr > screwdriverProgramList)=0
Add a list of screwdriver programs to the list of programs.
- virtual void [registerProgramSelectionCapability](#) (std::function< void(ScriptWriterPtr, ProgramSelectionParametersPtr)> scriptCodeGenerator)=0
- virtual void [registerOperationTypeCapability](#) ()=0
[Register](#) an operation type capability for a screwdriver that supports/requires the parameter for the screwdriving operation type (i.e.
- virtual void [registerPrepareToStartScrewdriverCapability](#) (std::function< void(ScriptWriterPtr, PrepareToStartScrewdriverParametersPtr)> scriptCodeGenerator)=0
- virtual void [registerFeedScrewCapability](#) (std::function< void(ScriptWriterPtr, FeedScrewParametersPtr)> scriptCodeGenerator)=0
- virtual void [registerDriveScrewOKCapability](#) (std::function< void(ScriptWriterPtr, DriveScrewOKParametersPtr)> scriptCodeGenerator)=0
- virtual void [registerDriveScrewNotOKCapability](#) (std::function< void(ScriptWriterPtr, DriveScrewNotOKParametersPtr)> scriptCodeGenerator)=0
- virtual void [registerScrewdriverReadyCapability](#) (std::function< void(ScriptWriterPtr, ScrewdriverReadyParametersPtr)> scriptCodeGenerator)=0

11.113.1 Detailed Description

This interface provides access to register or setup various properties and capabilities of a screwdriver.

It is optional to register these capabilities and properties.

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

11.113.2 Constructor & Destructor Documentation

11.113.2.1 ~ScrewdriverConfiguration()

```
virtual arcs::aubo_scope::ScrewdriverConfiguration::~~ScrewdriverConfiguration () [virtual], [default]
```

11.113.3 Member Function Documentation

11.113.3.1 addScrewdriverProgram()

```
virtual void arcs::aubo_scope::ScrewdriverConfiguration::addScrewdriverProgram (ScrewdriverProgramPtr screwdriverProgram) [pure virtual]
```

Add a screwdriver program to the list of programs.

Parameters

screwdriverProgram	program to add, not null

Exceptions

DuplicateScrewdriverProgramID	if the ids of the screwdriver programs (returned by the method ScrewdriverProgram#getId()) are not unique.
NameOfScrewdriverProgramCannotBeNullOrEmpty	if the display name of a program (returned by the method ScrewdriverProgram#getDisplayName()) is null or an empty string.

11.113.3.2 addScrewdriverPrograms()

```
virtual void arcs::aubo_scope::ScrewdriverConfiguration::addScrewdriverPrograms (
    std::vector< ScrewdriverProgramPtr > screwdriverProgramList) [pure virtual]
```

Add a list of screwdriver programs to the list of programs.

Parameters

screwdriverProgramList	list of programs to add, not null

Exceptions

DuplicateScrewdriverProgramID	if the ids of the screwdriver programs (returned by the method ScrewdriverProgram#getId()) are not unique.
NameOfScrewdriverProgramCannotBeNullOrEmpty	if the display name of a program (returned by the method ScrewdriverProgram#getDisplayName()) is null or an empty string.

11.113.3.3 registerDriveScrewNotOKCapability()

```
virtual void arcs::aubo_scope::ScrewdriverConfiguration::registerDriveScrewNotOKCapability (
    std::function< void(ScriptWriterPtr, DriveScrewNotOKParametersPtr)> scriptCodeGenerator) [pure
virtual]
```

[Register](#) a feedback capability for a screwdriver that is capable of providing information about whether or not an ongoing screwdriving operation (driving the screw) failed, i.e. ended in Not Ok (NOK) state.

Note: This capability must only be used to provide feedback about the failure of the screwing operation. It should not be used to provide feedback about successes. To support feedback about when the operation succeeds, register the Drive Screw Ok feedback capability the using the registerDriveScrewOKCapability(ScriptCodeGenerator) method.

The provided implementation of the ScriptCodeGenerator interface must generate the script code for determining if the screwing operation ended in failure. The return value of the script code must be a boolean, i.e. 'True' or 'False'. 'True' must be returned if the screwing operation failed, otherwise 'False' must be returned.

Note: A return value of 'False' must not be used to indicate that the operation succeeded, but simply that a failure was not (yet) detected. If the screwdriver can provide feedback on whether or not the screwing operation succeeded, the Drive Screw Ok feedback capability should be used.

Example:

The digital tool input 1 will go high if the operation ended in NOT OK-state. The script code is:

```
return get_tool_digital_in(1)
```

So the implementation of ScriptCodeGenerator#generateScript(ScriptWriter, Object) could be:

```
scriptWriter.appendLine("return get_tool_digital_in(1)");
```

Parameters

scriptCodeGenerator	script code generator responsible for generating the script code for determining if the screwing operation failed. When the script code needs to be generated, the ScriptCodeGenerator#generateScript(ScriptWriter, Object) method will be called (by AuboScope). Return value for the generated script code must be a boolean, i.e. 'True' or 'False'.

Exceptions

CapabilityAlreadyRegistered	if this capability has already been registered

CalledOutsideScrewdriverConfigurationPhase	if this method is called at the wrong time, i.e. outside the scope of the ScrewdriverContribution↵ #configureScrewdriver(ScrewdriverConfiguration, ScrewdriverApiProvider) method.

11.113.3.4 registerDriveScrewOKCapability()

```
virtual void arcs::aubo_scope::ScrewdriverConfiguration::registerDriveScrewOKCapability (
    std::function< void(ScriptWriterPtr, DriveScrewOKParametersPtr)> scriptCodeGenerator) [pure virtual]
```

[Register](#) a feedback capability for a screwdriver that is capable of providing information about whether or not an ongoing screwdriving operation (driving the screw) ended successfully (ended in OK-state).

Note: This capability must only be used to provide feedback about the success of the screwing operation. It should not be used to provide feedback about failures. To support feedback about when the operation fails, register the Drive Screw Not Ok feedback capability the using the registerDriveScrewNotOKCapability(ScriptCodeGenerator) method.

The provided implementation of the ScriptCodeGenerator interface must generate the script code for determining if the screwing operation succeeded. The return value of the script code must be a boolean, i.e. 'True' or 'False'. 'True' must be returned if the screwing operation was successful, otherwise 'False' must be returned.

Note: A return value of 'False' must not be used to indicate that the operation failed, but simply that a success was not (yet) detected. If the screwdriver can provide feedback on whether or not the screwing operation failed, the Drive Screw Not Ok feedback capability must be used.

Example:

The digital tool input 0 will go high if the operation ended in OK-state. The script code is:

```
return get_tool_digital_in(0)
```

So the implementation of ScriptCodeGenerator#generateScript(ScriptWriter, Object) could be:

```
scriptWriter.appendLine("return get_tool_digital_in(0)");
```

Parameters

scriptCodeGenerator	script code generator responsible for generating the script code for determining if the screwing operation succeeded. When the script code needs to be generated, the ScriptCodeGenerator#generateScript(ScriptWriter, Object) method will be called (by AuboScope). Return value for the generated script code must be a boolean, i.e. 'True' or 'False'.

Exceptions

CapabilityAlreadyRegistered	if this capability has already been registered
CalledOutsideScrewdriverConfigurationPhase	if this method is called at the wrong time, i.e. outside the scope of the ScrewdriverContribution↵ #configureScrewdriver(ScrewdriverConfiguration, ScrewdriverApiProvider) method.

11.113.3.5 registerFeedScrewCapability()

```
virtual void arcs::aubo_scope::ScrewdriverConfiguration::registerFeedScrewCapability (
    std::function< void(ScriptWriterPtr, FeedScrewParametersPtr)> scriptCodeGenerator) [pure virtual]
```

[Register](#) a capability for a screwdriver that supports automatic screw feeding.

The provided implementation of the ScriptCodeGenerator interface must generate the script code for feeding a screw to the screwdriver. The return value of the script code must be a boolean, i.e. 'True' or 'False'. 'True' must be returned if the screw feed operation was successful and 'False' must be returned if the operation failed.

The generated script code is executed before the script code for preparing the screwdriver, i.e the script code generated by registerPrepareToStartScrewdriverCapability(ScriptCodeGenerator).

Parameters

scriptCodeGenerator	script code generator responsible for generating the script code for feeding a screw to the screwdriver. When the script code needs to be generated, the ScriptCodeGenerator#generateScript(ScriptWriter, Object) method will be called (by AuboScope). Return value for the generated script code must be a boolean, i.e. 'True' or 'False'

Exceptions

CapabilityAlreadyRegistered	if this capability has already been registered
CalledOutsideScrewdriverConfigurationPhase	if this method is called at the wrong time, i.e. outside the scope of the ScrewdriverContribution↵ #configureScrewdriver(ScrewdriverConfiguration, ScrewdriverApiProvider) method.

11.113.3.6 registerOperationTypeCapability()

```
virtual void arcs::aubo_scope::ScrewdriverConfiguration::registerOperationTypeCapability () [pure virtual]
```

Generated by Doxygen

[Register](#) an operation type capability for a screwdriver that supports/requires the parameter for the screwdriving operation type (i.e.

if the operation is Tighten or Loosen).

Exceptions

CapabilityAlreadyRegistered	if this capability has already been registered
CalledOutsideScrewdriverConfigurationPhase	if this method is called at the wrong time, i.e. outside the scope of the ScrewdriverContribution↵ #configureScrewdriver(ScrewdriverConfiguration, ScrewdriverApiProvider) method.

11.113.3.7 registerPrepareToStartScrewdriverCapability()

```
virtual void arcs::aubo_scope::ScrewdriverConfiguration::registerPrepareToStartScrewdriverCapability (
    std::function< void(ScriptWriterPtr, PrepareToStartScrewdriverParametersPtr)> scriptCodeGenerator)
[pure virtual]
```

Register a capability/requirement for a screwdriver that needs to generate script code for preparing the screwdriver for a screwdriving operation (before starting the screw driver).

The provided implementation of the ScriptCodeGenerator interface must generate the script code for preparing the screwdriver. The return value of the script code must be a boolean, i.e. 'True' or 'False'. 'True' must be returned if the preparation of the screwdriver was successful and 'False' must be returned if the preparation failed.

The generated script code is executed before the script code for starting the screwdriver (could be from a different node), i.e the script code generated by ScrewdriverContribution#generateStartScrewdriverScript(ScriptWriter, S

Parameters

scriptCodeGenerator	script code generator responsible for generating the script code for preparing the screwdriver. When the script code needs to be generated, the ScriptCodeGenerator#generateScript(ScriptWriter, Object) method will be called (by AuboScope). Return value for the generated script code must be a boolean, i.e. 'True' or 'False'.

Exceptions

CapabilityAlreadyRegistered	if this capability has already been registered
CalledOutsideScrewdriverConfigurationPhase	if this method is called at the wrong time, i.e. outside the scope of the ScrewdriverContribution↵ #configureScrewdriver(ScrewdriverConfiguration, ScrewdriverApiProvider) method.

11.113.3.8 registerProgramSelectionCapability()

```
virtual void arcs::aubo_scope::ScrewdriverConfiguration::registerProgramSelectionCapability (
    std::function< void(ScriptWriterPtr, ProgramSelectionParametersPtr)> scriptCodeGenerator) [pure
virtual]
```

[Register](#) a program selection capability for a screwdriver that supports selecting between a set of screwdriver programs, typically defined on an external control box for the screwdriver.

The provided implementation of the ScriptCodeGenerator interface must generate the script code for selecting the specified screwdriver program. The return value of the script code must be a boolean, i.e. 'True' or 'False'. 'True' must be returned if selecting the program succeeded and 'False' must be returned if the program selection failed.

The generated script code is executed before the script code for starting the screwdriver (could be from a different node), i.e the script code generated by ScrewdriverContribution#generateStartScrewdriverScript(ScriptWriter, S

Parameters

programListProvider	provider of the list of screwdriver programs, not null
scriptCodeGenerator	script code generator responsible for generating the script code for selecting the specified screwdriver program. When the script code needs to be generated, the ScriptCodeGenerator#generateScript(ScriptWriter, Object) method will be called (by AuboScope). Return value for the generated script code must be a boolean, i.e. 'True' or 'False'.

Exceptions

CapabilityAlreadyRegistered	if this capability has already been registered
CalledOutsideScrewdriverConfigurationPhase	if this method is called at the wrong time, i.e. outside the scope of the ScrewdriverContribution#configureScrewdriver(ScrewdriverConfiguration, ScrewdriverApiProvider) method.

11.113.3.9 registerScrewdriverReadyCapability()

```
virtual void arcs::aubo_scope::ScrewdriverConfiguration::registerScrewdriverReadyCapability (
    std::function< void(ScriptWriterPtr, ScrewdriverReadyParametersPtr)> scriptCodeGenerator) [pure
virtual]
```

[Register](#) a feedback capability for a screwdriver that is capable of providing information about whether or not the screwdriver device is ready to operate (receive commands). This capability is used (by AuboScope) in a program to ensure that the screwdriver is ready before sending commands to it.

The provided implementation of the ScriptCodeGenerator interface must generate the script code for determining if the screwdriver is ready. The return value of the script code must be a boolean, i.e. 'True' or 'False'. 'True' must be returned if the screwdriver is ready, otherwise 'False' must be returned. —————
Generated by Doxygen

The generated script code is executed before the script code for the program selected capability (if that capability has been registered).

Parameters

scriptCodeGenerator	script code generator responsible for generating the script code for determining if the screwdriver is ready. When the script code needs to be generated, the <code>ScriptCodeGenerator#generateScript(ScriptWriter, Object)</code> method will be called (by <code>AuboScope</code>). Return value for the generated script code must be a boolean, i.e. 'True' or 'False'.

Exceptions

CapabilityAlreadyRegistered	if this capability has already been registered
CalledOutsideScrewdriverConfigurationPhase	if this method is called at the wrong time, i.e. outside the scope of the <code>ScrewdriverContribution#configureScrewdriver(ScrewdriverConfiguration, ScrewdriverApiProvider)</code> method.

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

- `include/aubo_caps/contribution/driver/screw_driver/screw_driver_configuration.h`

11.114 arcs::aubo_scope::ScrewdriverContribution Class Reference

[ScrewdriverContribution](#) This interface defines an API for adding functionality for screwdrivers to `AuboScope`.

```
#include <screw_driver_contribution.h>
```

Public Member Functions

- virtual [~ScrewdriverContribution](#) ()=default
- virtual std::string [getTitle](#) ()=0

This method must return the title of the screwdriver contribution.
- virtual void [configureScrewdriver](#) ([ScrewdriverConfiguration](#) screwdriverConfiguration, [ScrewdriverApiProviderPtr](#) apiProvider)
- virtual void [generateStartScrewdriverScript](#) ([ScriptWriterPtr](#) scriptWriter, [ScrewdriverParametersPtr](#) parameters)=0
- virtual void [generateStopScrewdriverScript](#) ([ScriptWriterPtr](#) scriptWriter, [ScrewdriverParametersPtr](#) parameters)=0

11.114.1 Detailed Description

ScrewdriverContribution This interface defines an API for adding functionality for screwdrivers to AuboScope.

Implementing this interface will "integrate" (hook) this screwdriving contribution into the builtin (native) AuboScope Screwdriving program node and Screwdriving installation screen. This will expose the screwdriving functionality supported by the screwdriver to the end user.

Note: Both the builtin (native) AuboScope Screwdriving program node and Screwdriver contributions are not supported on CB3 robots. If a AuboCap registers a Screwdriver contribution on a CB3 robot, the AuboCap will fail to be activated.

The Screwdriving program node allows the end user to program screwdriving operations. The configuration options available to the end user are based on the properties and capabilities supported by the screwdriver. As a minimum, all screwdrivers must support "default" screw (start screwdriver) and stop (stop screwdriver) operations. All other capabilities are optional to support and can be registered when the method `ScrewdriverContribution#configureScrewdriver(ScrewdriverConfiguration, ScrewdriverApiProvider)` is called.

The integration in the Screwdriving installation screen allows the user to select the screwdriver for programming screwing operations (using the Screwdriving program node). It also offers the possibility to define any custom inputs from the end user that are required to setup the screwdriver, e.g. specifying an IP address. This functionality is optional, since such configuration properties are specific for each screwdriver device. The corresponding generated custom UI will be accessible in the screen when the screwdriver is selected (by the end user) and replace the default Screwdriving installation UI. Access to create a custom configuration based on user inputs is provided when the method `ScrewdriverContribution#configureInstallation(CustomUserInputConfiguration, SystemConfiguration, TCPConfiguration, ScrewdriverApiProvider)` is called.

Definition at line 78 of file `screw_driver_contribution.h`.

11.114.2 Constructor & Destructor Documentation

11.114.2.1 ~ScrewdriverContribution()

```
virtual arcs::aubo_scope::ScrewdriverContribution::~~ScrewdriverContribution () [virtual], [default]
```

11.114.3 Member Function Documentation

11.114.3.1 configureScrewdriver()

```
virtual void arcs::aubo_scope::ScrewdriverContribution::configureScrewdriver (
    ScrewdriverConfiguration screwdriverConfiguration,
    ScrewdriverAPIProviderPtr apiProvider) [virtual]
```

When this method is called, use the configuration parameter to register or setup optional properties and capabilities of the screwdriver.

...

Parameters

screwdriverConfiguration	a configuration instance that can be used for registering or setting up the properties and capabilities of the screwdriver.
apiProvider	provides access to functionality and services available from within AuboScope which can be relevant for setting up the screwdriver capabilities (e.g. the SystemAPI interface with functionality for querying information about the robot).

11.114.3.2 generateStartScrewdriverScript()

```
virtual void arcs::aubo_scope::ScrewdriverContribution::generateStartScrewdriverScript (
    ScriptWriterPtr scriptWriter,
    ScrewdriverParametersPtr parameters) [pure virtual]
```

When this method is called, the script code for driving the screw must be generated.

The relevant parameters for the registered optional screwdriver capabilities/requirements defined/↵ configured by the end user are provided as input to the script code generation.

Parameters

scriptWriter	use this script writer instance to generate the script code for driving the screw
parameters	the parameters for the screwdriving operation defined/configured by the end user

11.114.3.3 generateStopScrewdriverScript()

```
virtual void arcs::aubo_scope::ScrewdriverContribution::generateStopScrewdriverScript (
    ScriptWriterPtr scriptWriter,
    ScrewdriverParametersPtr parameters) [pure virtual]
```

When this method is called, the script code for stopping the screw driver must be generated (i.e. the screwdriver must stop the current screwdriving operation).

The relevant parameters for the registered optional screwdriver capabilities/requirements defined/↵ configured by the end user are provided as input to the script code generation.

Parameters

scriptWriter	use this script writer instance to generate the script code for stopping the screwdriver
parameters	the parameters for the screwdriving operation defined/configured by the end user

11.114.3.4 getTitle()

virtual std::string arcs::aubo_scope::ScrewdriverContribution::getTitle () [pure virtual]

This method must return the title of the screwdriver contribution.

The title is displayed in:

- The builtin (native) AuboScope Screwdriving installation screen at the top where the screwdriver is selected
- The screen for the builtin (native) AuboScope Screwdriving program node (if this screwdriver has been selected in the Screwdriving installation screen)

This method is called once upon startup.

Parameters

locale	the current locale of AuboScope. Can be used for supporting titles in several languages.

Returns

the title of this screw driver contribution, not null nor an empty string.

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

- include/aubo_caps/contribution/driver/screw_driver/[screw_driver_contribution.h](#)

11.115 arcs::aubo_scope::ScrewdriverParameters Class Reference

This interface represents parameters for a screwdriver operation defined/configured by the end user.

#include <screw_driver_parameters.h>

Public Member Functions

- virtual [~ScrewdriverParameters](#) ()=default
- virtual [ScrewdriverProgram](#) getScrewdriverProgram ()=0
- virtual [ScrewdriverOperationType](#) getScrewdriverOperationType ()=0

11.115.1 Detailed Description

This interface represents parameters for a screwdriver operation defined/configured by the end user.

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

11.115.2 Constructor & Destructor Documentation

11.115.2.1 ~ScrewdriverParameters()

virtual arcs::aubo_scope::ScrewdriverParameters::~~ScrewdriverParameters () [virtual], [default]

11.115.3 Member Function Documentation

11.115.3.1 getScrewdriverOperationType()

virtual [ScrewdriverOperationType](#) arcs::aubo_scope::ScrewdriverParameters::getScrewdriverOperationType () [pure virtual]

Get the selected screwdriver operation type.

This is only valid, if the operation type capability has been registered (using the method `ScrewdriverCapabilities#registerOperationTypeCapability()`).

Returns

The operation type selected by the user.

Exceptions

UnsupportedOperationException	if the screwdriver has not registered the operation type capability

11.115.3.2 getScrewdriverProgram()

virtual [ScrewdriverProgram](#) arcs::aubo_scope::ScrewdriverParameters::getScrewdriverProgram () [pure virtual]

Get the selected screwdriver program to be used for the screwdriver operation.

This is only valid, if the program selection capability has been registered (using the method `ScrewdriverCapabilities#registerProgramSelectionCapability(ScrewdriverProgramListProvider, ScriptCodeGenerator)`).

Returns

The selected screwdriver program to use for the operation.

Exceptions

UnsupportedOperationException	if the screwdriver has not registered the program selection capability

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

- include/aubo_caps/contribution/driver/screw_driver/[screw_driver_parameters.h](#)

11.116 arcs::aubo_scope::ScrewdriverProgram Class Reference

This abstract class represents a screwdriver program, typically defined on an external control box for the screwdriver.

```
#include <screw_driver_parameters.h>
```

Public Member Functions

- virtual [~ScrewdriverProgram](#) ()=0
- virtual std::string [getId](#) ()=0
- virtual std::string [getDisplayName](#) ()=0

11.116.1 Detailed Description

This abstract class represents a screwdriver program, typically defined on an external control box for the screwdriver.

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

11.116.2 Constructor & Destructor Documentation

11.116.2.1 ~ScrewdriverProgram()

```
virtual arcs::aubo_scope::ScrewdriverProgram::~~ScrewdriverProgram () [pure virtual]
```

11.116.3 Member Function Documentation

11.116.3.1 getDisplayName()

```
virtual std::string arcs::aubo_scope::ScrewdriverProgram::getDisplayName () [pure virtual]
```

This method should return the name of the screwdriver program. The name will be displayed in the UI, e.g. in a combo box. The name returned is allowed to change over versions of the driver AuboCap as well as be different depending on the current language selected in AuboScope, in case translations are supported.

This method must be implemented.

Returns

the name of the screwdriver program to display in the UI.

11.116.3.2 getId()

virtual std::string arcs::aubo_scope::ScrewdriverProgram::getId () [pure virtual]

This method must return an identifier to uniquely identify this screwdriver program. The identifier will be used for persistence and resolving the correct screwdriver program. Therefore the identifier must remain constant over time and versions of the screwdriver AuboCap.

This method must be implemented.

Returns

a unique identifier string which will remain constant over time and versions of the screwdriver AuboCap.

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

- include/aubo_caps/contribution/driver/screw_driver/[screw_driver_parameters.h](#)

11.117 arcs::aubo_scope::ScrewdriverReadyParameters Class Reference

This interface provides the parameters relevant for generating script for the Screwdriver Ready feedback operation capability (registered using `ScrewdriverFeedbackCapabilities#registerScrewdriverReadyCapability(ScriptCodeC`

`#include <screw_driver_configuration.h>`

Public Member Functions

- virtual [~ScrewdriverReadyParameters](#) ()=default

11.117.1 Detailed Description

This interface provides the parameters relevant for generating script for the Screwdriver Ready feedback operation capability (registered using `ScrewdriverFeedbackCapabilities#registerScrewdriverReadyCapability(ScriptCodeC`

These parameters are passed when AuboScope calls the implementation of the `ScriptCodeGenerator#generateScript(Script` method responsible for the generating the script code.

The interface is intentionally empty, but could have parameters in future releases.

Definition at line 151 of file [screw_driver_configuration.h](#).

11.117.2 Constructor & Destructor Documentation

11.117.2.1 ~ScrewdriverReadyParameters()

virtual arcs::aubo_scope::ScrewdriverReadyParameters::~~ScrewdriverReadyParameters () [virtual], [default]

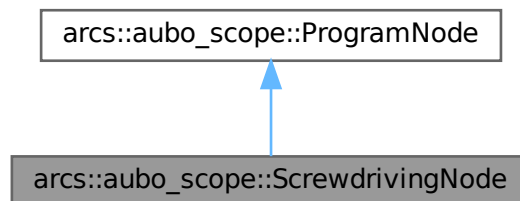
The documentation for this class was generated from the following file:

- include/aubo_caps/contribution/driver/screw_driver/[screw_driver_configuration.h](#)

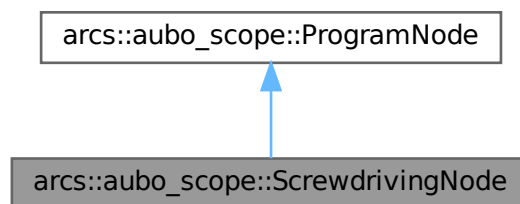
11.118 arcs::aubo_scope::ScrewdrivingNode Class Reference

```
#include <screw_driving_node.h>
```

Inheritance diagram for arcs::aubo_scope::ScrewdrivingNode:



Collaboration diagram for arcs::aubo_scope::ScrewdrivingNode:



Public Member Functions

- [ScrewdrivingNode](#) (ScrewdrivingNode &f)
- [ScrewdrivingNode](#) (ScrewdrivingNode &&f)
- virtual [~ScrewdrivingNode](#) ()

Public Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()
Get the program node contribution.

Private Member Functions

- [ScrewdrivingNode](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Protected Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.118.1 Detailed Description

Definition at line 10 of file [screw_driving_node.h](#).

11.118.2 Constructor & Destructor Documentation

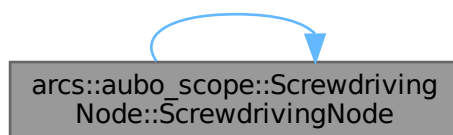
11.118.2.1 ScrewdrivingNode() [1/3]

`arcs::aubo_scope::ScrewdrivingNode::ScrewdrivingNode (ScrewdrivingNode & f)`

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

Referenced by [ScrewdrivingNode\(\)](#), and [ScrewdrivingNode\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

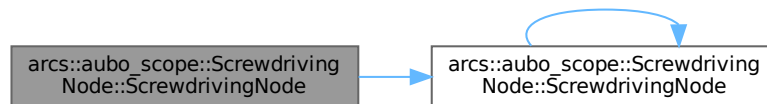


11.118.2.2 ScrewdrivingNode() [2/3]

```
arcs::aubo_scope::ScrewdrivingNode::ScrewdrivingNode (
    ScrewdrivingNode && f)
```

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

Here is the call graph for this function:



11.118.2.3 ~ScrewdrivingNode()

```
virtual arcs::aubo_scope::ScrewdrivingNode::~~ScrewdrivingNode () [virtual]
```

11.118.2.4 ScrewdrivingNode() [3/3]

```
arcs::aubo_scope::ScrewdrivingNode::ScrewdrivingNode () [private]
```

11.118.3 Friends And Related Symbol Documentation

11.118.3.1 DataSwitch

```
friend class DataSwitch [friend]
```

Definition at line 18 of file [screw_driving_node.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.118.4 Member Data Documentation

11.118.4.1 d_

```
void* arcs::aubo_scope::ScrewdrivingNode::d_ { nullptr } [private]
```

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

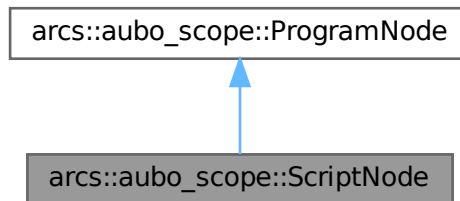
The documentation for this class was generated from the following file:

- [include/aubo_caps/domain/program/nodes/builtin/screw_driving_node.h](#)

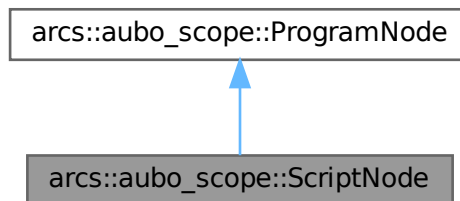
11.119 arcs::aubo_scope::ScriptNode Class Reference

#include <script_node.h>

Inheritance diagram for arcs::aubo_scope::ScriptNode:



Collaboration diagram for arcs::aubo_scope::ScriptNode:



Public Member Functions

- [ScriptNode](#) (ScriptNode &f)
- [ScriptNode](#) (ScriptNode &&f)
- virtual [~ScriptNode](#) ()

Public Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()
Get the program node contribution.

Private Member Functions

- [ScriptNode\(\)](#)

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Protected Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode\(\)](#)

11.119.1 Detailed Description

Definition at line 10 of file [script_node.h](#).

11.119.2 Constructor & Destructor Documentation

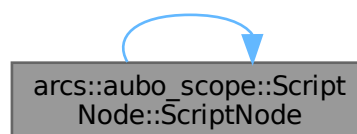
11.119.2.1 ScriptNode() [1/3]

```
arcs::aubo_scope::ScriptNode::ScriptNode (  
    ScriptNode & f)
```

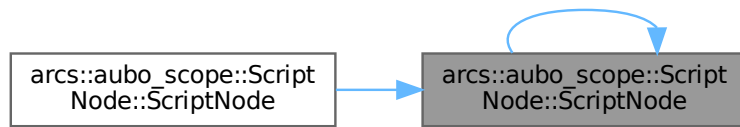
References [ScriptNode\(\)](#).

Referenced by [ScriptNode\(\)](#), and [ScriptNode\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

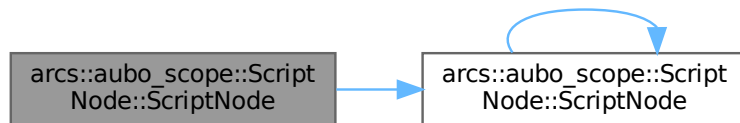


11.119.2.2 ScriptNode() [2/3]

```
arcs::aubo_scope::ScriptNode::ScriptNode (
    ScriptNode && f)
```

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

Here is the call graph for this function:



11.119.2.3 ~ScriptNode()

```
virtual arcs::aubo_scope::ScriptNode::~~ScriptNode () [virtual]
```

11.119.2.4 ScriptNode() [3/3]

```
arcs::aubo_scope::ScriptNode::ScriptNode () [private]
```

11.119.3 Friends And Related Symbol Documentation

11.119.3.1 DataSwitch

```
friend class DataSwitch [friend]
```

Definition at line 18 of file [script_node.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.119.4 Member Data Documentation

11.119.4.1 d_

void* arcs::aubo_scope::ScriptNode::d_ { nullptr } [private]

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

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

- include/aubo_caps/domain/program/nodes/builtin/[script_node.h](#)

11.120 arcs::aubo_scope::ScriptWriter Class Reference

[ScriptWriter](#) This interface provides support for generating Script code.

```
#include <script_writer.h>
```

Public Member Functions

- [ScriptWriter](#) (ScriptWriter &f)
- [ScriptWriter](#) (ScriptWriter &&f)
- virtual [~ScriptWriter](#) ()
- void [setLabel](#) (int lineno, const std::string &comment)
- void [appendLine](#) (const std::string &script_line)

Adds a single line of script code using auto-indentation.
- void [appendVectorDouble](#) (const std::vector< double > &value)
- void [appendRaw](#) (const std::string &script)

Adds script code appending the script code as is without using auto-indentation.
- std::string [generateScript](#) ()

Generate a string with the full script code.
- void [writeChildren](#) ()

Use this method when generating script code for a node that has children nodes.
- void [assign](#) (VariablePtr variable, const std::string &expression, bool [sync](#)=false)
- void [assign](#) (const std::string &variableName, const ExpressionPtr &expression, bool [sync](#)=false)
- void [assign](#) (VariablePtr variable, const ExpressionPtr &expression, bool [sync](#)=false)
- void [incrementVariable](#) (const std::string &variable_name)

Add 1 to the variable value.
- void [note](#) (const std::string &expression)

Add a note.
- void [sleep](#) (double seconds)

Sleep for a number of seconds.
- void [sync](#) ()

Sync.
- void [defineFunction](#) (const std::string &func_name)

Define a function.
- void [anonyFunction](#) (const std::string &func_name)
- void [setRobotIndex](#) (int robot_index)
- void [returnMethod](#) ()

Return from method.

- void `end` ()
Insert an end.
- void `lineFeed` ()
Insert an empty line.
- void `setPayload` (double mass, double x, double y, double z)
Sets the mass and Center of Gravity of the payload.
- void `setTcp` (const std::vector< double > &pose)
Set the Tool Center Point.
- void `ifCondition` (const ExpressionPtr &expression)
Start an if-conditional.
- void `ifNotCondition` (const ExpressionPtr &expression)
Start a negated if-conditional.
- void `elseifCondition` (const ExpressionPtr &expression)
Adds an else-if branch.
- void `elseCondition` ()
Adds an else branch.
- void `forCondition` (int count, int step)
Starts a for-loop.
- void `whileTrue` ()
Starts a while true loop.
- void `whileCondition` (const ExpressionPtr &expression)
Starts a while-loop.
- void `whileNot` (const ExpressionPtr &expression)
Starts a negated while-loop.
- void `defineThread` (const std::string &thread_name, bool loop_or_not)
Start a thread definition.
- void `runThread` (const std::string &thread_name)
Start a thread.
- void `killThread` (const std::string &thread_name)
Kill a thread.
- std::string `getResolvedVariable` (const std::string &variable_name)
Returns a registered variable name that can be used in a script.
- void `increaseIndent` ()
- void `decreaseIndent` ()

Private Member Functions

- `ScriptWriter` ()

Private Attributes

- void * `d_` { nullptr }

Friends

- class `DataSwitch`

11.120.1 Detailed Description

[ScriptWriter](#) This interface provides support for generating Script code.

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

11.120.2 Constructor & Destructor Documentation

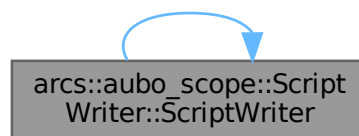
11.120.2.1 ScriptWriter() [1/3]

```
arcs::aubo_scope::ScriptWriter::ScriptWriter (  
    ScriptWriter & f)
```

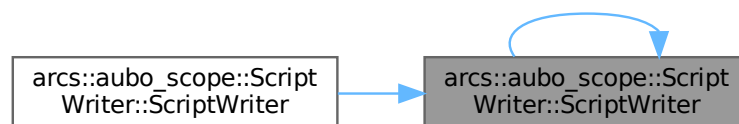
References [ScriptWriter\(\)](#).

Referenced by [ScriptWriter\(\)](#), and [ScriptWriter\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

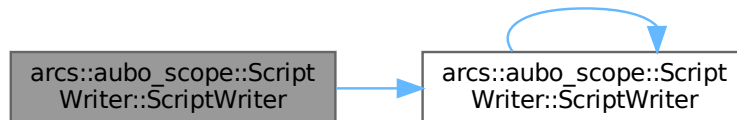


11.120.2.2 ScriptWriter() [2/3]

```
arcs::aubo_scope::ScriptWriter::ScriptWriter (
    ScriptWriter && f)
```

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

Here is the call graph for this function:



11.120.2.3 ~ScriptWriter()

```
virtual arcs::aubo_scope::ScriptWriter::~~ScriptWriter () [virtual]
```

11.120.2.4 ScriptWriter() [3/3]

```
arcs::aubo_scope::ScriptWriter::ScriptWriter () [private]
```

11.120.3 Member Function Documentation

11.120.3.1 anonyFunction()

```
void arcs::aubo_scope::ScriptWriter::anonyFunction (
    const std::string & func_name)
```

11.120.3.2 appendLine()

```
void arcs::aubo_scope::ScriptWriter::appendLine (
    const std::string & script_line)
```

Adds a single line of script code using auto-indentation.

Parameters

<code>script_line</code>	single line of script code to append.

11.120.3.3 appendRaw()

Parameters

script	script code to append.

11.120.3.4 appendVectorDouble()

```
void arcs::aubo_scope::ScriptWriter::appendVectorDouble (
    const std::vector< double > & value)
```

11.120.3.5 assign() [1/3]

```
void arcs::aubo_scope::ScriptWriter::assign (
    const std::string & variableName,
    const ExpressionPtr & expression,
    bool sync = false)
```

[Variable](#) assignment. Assigns the variable without a local or global qualifier.

Parameters

variableName	name of the variable, not null.
expression	expression that is assigned to the variable, not null.

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

Here is the call graph for this function:



11.120.3.6 assign() [2/3]

```
void arcs::aubo_scope::ScriptWriter::assign (
    VariablePtr variable,
    const ExpressionPtr & expression,
    bool sync = false)
```

[Variable](#) assignment.

Parameters

variable	the variable to assign an expression to, not null.
expression	expression that is assigned to the variable, not null.

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

Here is the call graph for this function:

11.120.3.7 `assign()` [3/3]

```
void arcs::aubo_scope::ScriptWriter::assign (
    VariablePtr variable,
    const std::string & expression,
    bool sync = false)
```

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

Here is the call graph for this function:

11.120.3.8 `decreaseIndent()`

```
void arcs::aubo_scope::ScriptWriter::decreaseIndent ()
```

11.120.3.9 `defineFunction()`

```
void arcs::aubo_scope::ScriptWriter::defineFunction (
    const std::string & func_name)
```

Define a function.

Parameters

func_name	function name.

11.120.3.10 defineThread()

```
void arcs::aubo_scope::ScriptWriter::defineThread (  
    const std::string & thread_name,  
    bool loop_or_not)
```

Start a thread definition.

Parameters

thread_name	the thread name,
loop_or_not	whether loops.

11.120.3.11 elseCondition()

```
void arcs::aubo_scope::ScriptWriter::elseCondition ()
```

Adds an else branch.

11.120.3.12 elseIfCondition()

```
void arcs::aubo_scope::ScriptWriter::elseIfCondition (  
    const ExpressionPtr & expression)
```

Adds an else-if branch.

Parameters

expression	the expression.

11.120.3.13 end()

```
void arcs::aubo_scope::ScriptWriter::end ()
```

Insert an end.

11.120.3.14 forCondition()

Parameters

count	loop count,
step	loop step.

11.120.3.15 generateScript()

```
std::string arcs::aubo_scope::ScriptWriter::generateScript ()
```

Generate a string with the full script code.

Returns

the resulting script as a string.

11.120.3.16 getResolvedVariable()

```
std::string arcs::aubo_scope::ScriptWriter::getResolvedVariable (  
    const std::string & variable_name)
```

Returns a registered variable name that can be used in a script.

Parameters

variable_name	a variable name.

Returns

variable name that can be used in a script.

11.120.3.17 ifCondition()

```
void arcs::aubo_scope::ScriptWriter::ifCondition (  
    const ExpressionPtr & expression)
```

Start an if-conditional.

Parameters

expression	the expression.

11.120.3.18 ifNotCondition()

Parameters

expression	the expression.

11.120.3.19 increaseIndent()

```
void arcs::aubo_scope::ScriptWriter::increaseIndent ()
```

11.120.3.20 incrementVariable()

```
void arcs::aubo_scope::ScriptWriter::incrementVariable (  
    const std::string & variable_name)
```

Add 1 to the variable value.

Parameters

variable_name	the variable to increment.

11.120.3.21 killThread()

```
void arcs::aubo_scope::ScriptWriter::killThread (  
    const std::string & thread_name)
```

Kill a thread.

Parameters

thread_name	the thread name.

11.120.3.22 lineFeed()

```
void arcs::aubo_scope::ScriptWriter::lineFeed ()
```

Insert an empty line.

11.120.3.23 note()

```
void arcs::aubo_scope::ScriptWriter::note (  
    const std::string & expression)
```

Add a note.

Parameters

expression	the note expression.

11.120.3.24 returnMethod()

```
void arcs::aubo_scope::ScriptWriter::returnMethod ()
```

Return from method.

11.120.3.25 runThread()

```
void arcs::aubo_scope::ScriptWriter::runThread (  
    const std::string & thread_name)
```

Start a thread.

Parameters

thread_name	the thread name.

11.120.3.26 setLabel()

```
void arcs::aubo_scope::ScriptWriter::setLabel (  
    int lineno,  
    const std::string & comment)
```

11.120.3.27 setPayload()

```
void arcs::aubo_scope::ScriptWriter::setPayload (  
    double mass,  
    double x,  
    double y,  
    double z)
```

Sets the mass and Center of Gravity of the payload.

11.120.3.28 setRobotIndex()

```
void arcs::aubo_scope::ScriptWriter::setRobotIndex (  
    int robot_index)
```

11.120.3.29 setTcp()

```
void arcs::aubo_scope::ScriptWriter::setTcp (
    const std::vector< double > & pose)
```

Set the Tool Center Point.

11.120.3.30 sleep()

```
void arcs::aubo_scope::ScriptWriter::sleep (
    double seconds)
```

Sleep for a number of seconds.

Parameters

seconds	amount of time to sleep in seconds.

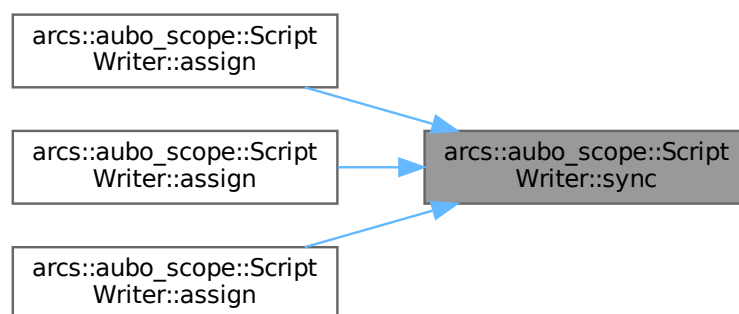
11.120.3.31 sync()

```
void arcs::aubo_scope::ScriptWriter::sync ()
```

Sync.

Referenced by [assign\(\)](#), [assign\(\)](#), and [assign\(\)](#).

Here is the caller graph for this function:



11.120.3.32 whileCondition()

```
void arcs::aubo_scope::ScriptWriter::whileCondition (
    const ExpressionPtr & expression)
```

Starts a while-loop.

Parameters

expression	the loop invariant.

11.120.3.33 whileNot()

```
void arcs::aubo_scope::ScriptWriter::whileNot (
    const ExpressionPtr & expression)
```

Starts a negated while-loop.

Parameters

expression	the loop invariant.

11.120.3.34 whileTrue()

```
void arcs::aubo_scope::ScriptWriter::whileTrue ()
```

Starts a while true loop.

11.120.3.35 writeChildren()

```
void arcs::aubo_scope::ScriptWriter::writeChildren ()
```

Use this method when generating script code for a node that has children nodes.

Let children nodes generate code.

11.120.4 Friends And Related Symbol Documentation

11.120.4.1 DataSwitch

```
friend class DataSwitch [friend]
```

Definition at line 223 of file [script_writer.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.120.5 Member Data Documentation

11.120.5.1 d_

```
void* arcs::aubo_scope::ScriptWriter::d_ { nullptr } [private]
```

Definition at line 225 of file [script_writer.h](#).

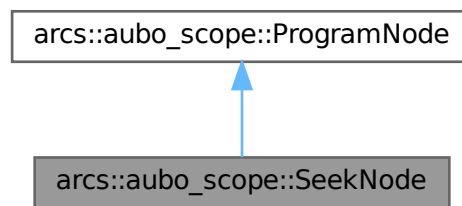
The documentation for this class was generated from the following file:

- include/aubo_caps/domain/script/[script_writer.h](#)

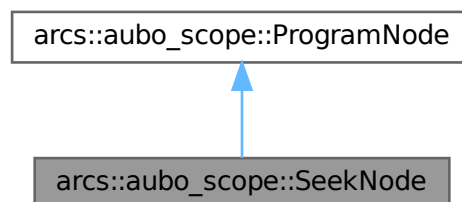
11.121 arcs::aubo_scope::SeekNode Class Reference

```
#include <seek_node.h>
```

Inheritance diagram for arcs::aubo_scope::SeekNode:



Collaboration diagram for arcs::aubo_scope::SeekNode:



Public Member Functions

- [SeekNode](#) (SeekNode &f)
- [SeekNode](#) (SeekNode &&f)
- virtual [~SeekNode](#) ()

Public Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()
Get the program node contribution.

Private Member Functions

- [SeekNode](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Protected Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.121.1 Detailed Description

Definition at line 10 of file [seek_node.h](#).

11.121.2 Constructor & Destructor Documentation

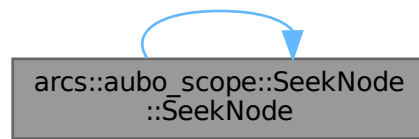
11.121.2.1 [SeekNode\(\)](#) [1/3]

```
arcs::aubo_scope::SeekNode::SeekNode (  
    SeekNode & f)
```

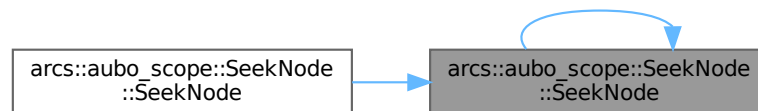
References [SeekNode\(\)](#).

Referenced by [SeekNode\(\)](#), and [SeekNode\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

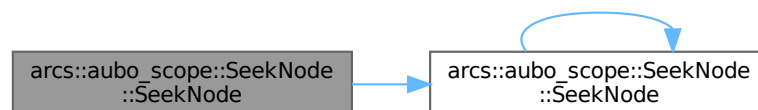


11.121.2.2 SeekNode() [2/3]

```
arcs::aubo_scope::SeekNode::SeekNode (
    SeekNode && f)
```

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

Here is the call graph for this function:



11.121.2.3 ~SeekNode()

```
virtual arcs::aubo_scope::SeekNode::~~SeekNode () [virtual]
```

11.121.2.4 SeekNode() [3/3]

```
arcs::aubo_scope::SeekNode::SeekNode () [private]
```

11.121.3 Friends And Related Symbol Documentation

11.121.3.1 DataSwitch

```
friend class DataSwitch [friend]
```

Definition at line 18 of file [seek_node.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.121.4 Member Data Documentation

11.121.4.1 d_

```
void* arcs::aubo_scope::SeekNode::d_ { nullptr } [private]
```

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

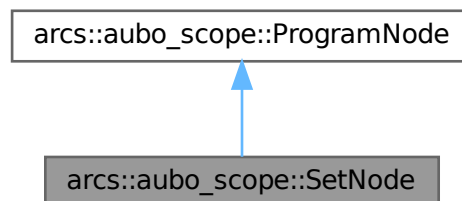
The documentation for this class was generated from the following file:

- [include/aubo_caps/domain/program/nodes/builtin/seek_node.h](#)

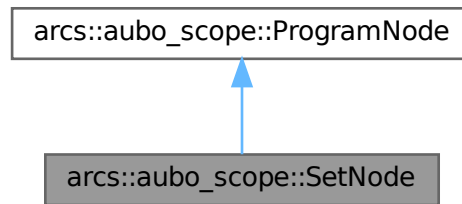
11.122 arcs::aubo_scope::SetNode Class Reference

```
#include <set_node.h>
```

Inheritance diagram for arcs::aubo_scope::SetNode:



Collaboration diagram for arcs::aubo_scope::SetNode:



Public Types

- enum `ConfigType` {
`NO_TYPE` , `NO_ACTION` , `DIGITAL_OUTPUT` , `ANALOG_OUTPUT_CURRENT` ,
`ANALOG_OUTPUT_VOLTAGE` , `FLOAT_REGISTER_OUTPUT` , `EXPRESSION_OUTPUT` ,
`SINGLE_DIGITAL_PULSE` ,
`INCREMENT_VARIABLE` }
- The configuration type used to determine which type of configuration this instance is.
- enum `OutputSelectionType` { `NO_OUTPUT` , `UNRESOLVED_OUTPUT` , `OUTPUT` }

Public Member Functions

- `SetNode` (`SetNode &f`)
- `SetNode` (`SetNode &&f`)
- virtual `~SetNode` ()
- `ConfigType` `getConfigType` ()
 This method returns the type of configuration.
- void `setConfigType` (`ConfigType` type)
- double `getPulseTime` ()
- void `setPulseTime` (double time)
- float `getFloatValueToSet` ()
- void `setFloatValueToSet` (float value)
- double `getAnalogOutput` ()
- void `setAnalogOutput` (double value)
- `OutputSelectionType` `getOutputSelectionType` ()
- `ExpressionPtr` `getExpression` ()
- void `setExpression` (`ExpressionPtr` expression)
- `IoPtr` `getOutput` ()
 The output selected can be an analog output, a digital output, a MODBUS output or a register output.
- void `setOutput` (`IoPtr` io)
- bool `getBoolValueToSet` ()
 Depending on the type of output this is interpreted in different ways.
- void `setBoolValueToSet` (bool value)
- double `getVoltage` ()
- void `setVoltage` (double voltage)
- double `getCurrent` ()
- void `setCurrent` (double curr)

Public Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()
Get the program node contribution.

Private Member Functions

- [SetNode](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Protected Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.122.1 Detailed Description

Definition at line 12 of file [set_node.h](#).

11.122.2 Member Enumeration Documentation

11.122.2.1 ConfigType

enum [arcs::aubo_scope::SetNode::ConfigType](#)

The configuration type used to determine which type of configuration this instance is.

Enumerator

NO_TYPE	No selection has been made. This type has no further information.
NO_ACTION	No action has been selected. This type has no further information.

DIGITAL_OUTPUT	Digital output has been selected. The config instance can be cast to DigitalOutputSetNodeConfig.
ANALOG_OUTPUT_CURRENT	Analog output configured for electric current or no output has been selected. The config instance can be cast to AnalogOutputCurrentSetNodeConfig.
ANALOG_OUTPUT_VOLTAGE	Analog output configured for voltage has been selected. The config instance can be cast to AnalogOutputVoltageSetNodeConfig.
FLOAT_REGISTER_OUTPUT	Register using float type values has been selected. The config instance can be cast to FloatRegisterOutputSetNodeConfig.
EXPRESSION_OUTPUT	Expression output has been selected. The config instance can be cast to ExpressionOutputSetNodeConfig.
SINGLE_DIGITAL_PULSE	Single digital pulse output has been selected. The config instance can be cast to SingleDigitalPulseSetNodeConfig.
INCREMENT_VARIABLE	Increment variable has been selected. This type has no further information. Setting this type of config will be ignored and will have no effect on the node.

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

11.122.2.2 OutputSelectionType

enum [arcs::aubo_scope::SetNode::OutputSelectionType](#)

Enumerator

NO_OUTPUT	No output is selected.
UNRESOLVED_OUTPUT	The selected output is no longer available (e.g. because the end user has removed the output in the Installation or the needed Installation is not currently loaded).
OUTPUT	Output is selected. The Selection instance can be cast to ExpressionOutput.

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

11.122.3 Constructor & Destructor Documentation

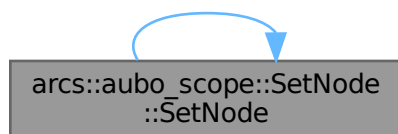
11.122.3.1 SetNode() [1/3]

```
arcs::aubo_scope::SetNode::SetNode (
    SetNode & f)
```

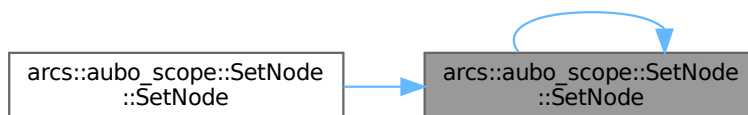
References [SetNode\(\)](#).

Referenced by [SetNode\(\)](#), and [SetNode\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

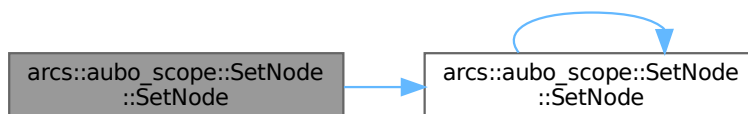


11.122.3.2 SetNode() [2/3]

```
arcs::aubo_scope::SetNode::SetNode (
    SetNode && f)
```

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

Here is the call graph for this function:



11.122.3.3 ~SetNode()

virtual arcs::aubo_scope::SetNode::~~SetNode () [virtual]

11.122.3.4 SetNode() [3/3]

arcs::aubo_scope::SetNode::SetNode () [private]

11.122.4 Member Function Documentation

11.122.4.1 getAnalogOutput()

double arcs::aubo_scope::SetNode::getAnalogOutput ()

Returns

the analog output value.

11.122.4.2 getBoolValueToSet()

bool arcs::aubo_scope::SetNode::getBoolValueToSet ()

Depending on the type of output this is interpreted in different ways.

For digital output and MODBUS output, true means the output will be set to HIGH, false means the output will be set to LOW. For boolean register the value is interpreted literally.

Returns

the output value.

11.122.4.3 getConfigType()

[ConfigType](#) arcs::aubo_scope::SetNode::getConfigType ()

This method returns the type of configuration.

Cast this instance appropriately to have access to specific getters.

Returns

the type of this config.

11.122.4.4 `getCurrent()`

`double arcs::aubo_scope::SetNode::getCurrent ()`

Returns

the electric current value/level to be applied to the output.

11.122.4.5 `getExpression()`

`ExpressionPtr arcs::aubo_scope::SetNode::getExpression ()`

Returns

the expression whose evaluation will be applied to the output.

11.122.4.6 `getFloatValueToSet()`

`float arcs::aubo_scope::SetNode::getFloatValueToSet ()`

Returns

the float value to be applied to the float register output.

11.122.4.7 `getOutput()`

`IoPtr arcs::aubo_scope::SetNode::getOutput ()`

The output selected can be an analog output, a digital output, a MODBUS output or a register output.

Returns

the output to which the result of the evaluation of the expression will be applied.

11.122.4.8 `getOutputSelectionType()`

`OutputSelectionType arcs::aubo_scope::SetNode::getOutputSelectionType ()`

11.122.4.9 `getPulseTime()`

`double arcs::aubo_scope::SetNode::getPulseTime ()`

Returns

the time for the length of the pulse.

11.122.4.10 getVoltage()

```
double arcs::aubo_scope::SetNode::getVoltage ()
```

Returns

the voltage value/level to be applied to the output

11.122.4.11 setAnalogOutput()

```
void arcs::aubo_scope::SetNode::setAnalogOutput (  
    double value)
```

11.122.4.12 setBoolValueToSet()

```
void arcs::aubo_scope::SetNode::setBoolValueToSet (  
    bool value)
```

11.122.4.13 setConfigType()

```
void arcs::aubo_scope::SetNode::setConfigType (  
    ConfigType type)
```

11.122.4.14 setCurrent()

```
void arcs::aubo_scope::SetNode::setCurrent (  
    double curr)
```

11.122.4.15 setExpression()

```
void arcs::aubo_scope::SetNode::setExpression (  
    ExpressionPtr expression)
```

11.122.4.16 setFloatValueToSet()

```
void arcs::aubo_scope::SetNode::setFloatValueToSet (  
    float value)
```

11.122.4.17 setOutput()

```
void arcs::aubo_scope::SetNode::setOutput (  
    IoPtr io)
```

11.122.4.18 setPulseTime()

```
void arcs::aubo_scope::SetNode::setPulseTime (  
    double time)
```

11.122.4.19 setVoltage()

```
void arcs::aubo_scope::SetNode::setVoltage (  
    double voltage)
```

11.122.5 Friends And Related Symbol Documentation

11.122.5.1 DataSwitch

```
friend class DataSwitch [friend]
```

Definition at line 211 of file [set_node.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.122.6 Member Data Documentation

11.122.6.1 d_

```
void* arcs::aubo_scope::SetNode::d_ { nullptr } [private]
```

Definition at line 213 of file [set_node.h](#).

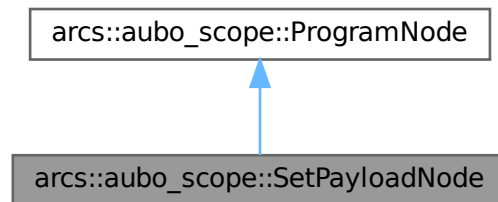
The documentation for this class was generated from the following file:

- [include/aubo_caps/domain/program/nodes/builtin/set_node.h](#)

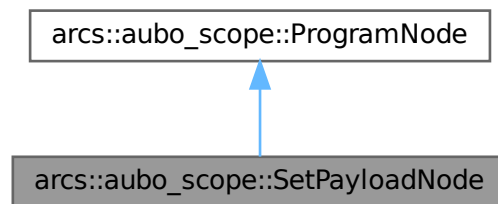
11.123 arcs::aubo_scope::SetPayloadNode Class Reference

```
#include <set_payload_node.h>
```

Inheritance diagram for arcs::aubo_scope::SetPayloadNode:



Collaboration diagram for arcs::aubo_scope::SetPayloadNode:



Public Types

- enum `ConfigType` { `PAYLOAD_SELECTION` , `CUSTOM_PARAMETERS` }
- This base interface represents a configuration of the Set `Payload` program node (i.e.

Public Member Functions

- `SetPayloadNode` (`SetPayloadNode &f`)
- `SetPayloadNode` (`SetPayloadNode &&f`)
- virtual `~SetPayloadNode` ()
- `ConfigType getConfigType` ()

This method returns the type of configuration.
- void `setConfigType` (`ConfigType type`)
- double `getMass` ()
- void `setMass` (double mass)
- `std::vector< double >` `getCenterOfGravity` ()

Gets the payload's center of gravity (CoG), also referred to as center of mass.
- void `setCenterOfGravity` (const `std::vector< double >` &cog)

Public Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()
Get the program node contribution.

Private Member Functions

- [SetPayloadNode](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Protected Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.123.1 Detailed Description

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

11.123.2 Member Enumeration Documentation

11.123.2.1 ConfigType

enum [arcs::aubo_scope::SetPayloadNode::ConfigType](#)

This base interface represents a configuration of the Set [Payload](#) program node (i.e. a [SetPayloadNode](#) instance).

Enumerator

PAYLOAD_SELECTION	A payload (from the installation) is selected. The config instance can be cast to SelectionPayloadNodeConfig.

CUSTOM_PARAMETERS	Custom payload parameters are specified. The config instance can be cast to CustomParametersPayloadNodeConfig.

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

11.123.3 Constructor & Destructor Documentation

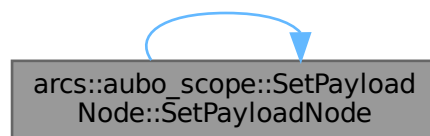
11.123.3.1 SetPayloadNode() [1/3]

arcs::aubo_scope::SetPayloadNode::SetPayloadNode (
SetPayloadNode & f)

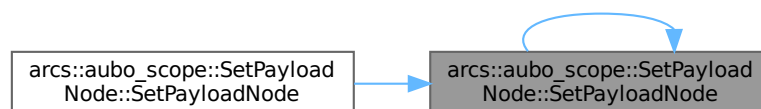
References [SetPayloadNode\(\)](#).

Referenced by [SetPayloadNode\(\)](#), and [SetPayloadNode\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

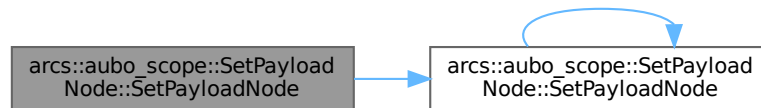


11.123.3.2 SetPayloadNode() [2/3]

```
arcs::aubo_scope::SetPayloadNode::SetPayloadNode (
    SetPayloadNode && f)
```

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

Here is the call graph for this function:



11.123.3.3 ~SetPayloadNode()

```
virtual arcs::aubo_scope::SetPayloadNode::~~SetPayloadNode () [virtual]
```

11.123.3.4 SetPayloadNode() [3/3]

```
arcs::aubo_scope::SetPayloadNode::SetPayloadNode () [private]
```

11.123.4 Member Function Documentation

11.123.4.1 getCenterOfGravity()

```
std::vector< double > arcs::aubo_scope::SetPayloadNode::getCenterOfGravity ()
```

Gets the payload's center of gravity (CoG), also referred to as center of mass.

It is defined as the offset between the center of the tool output flange and the center of gravity of the attached payload.

Returns

The center of gravity (CoG) of the payload relative to the center of the tool output flange

11.123.4.2 getConfigType()

```
ConfigType arcs::aubo_scope::SetPayloadNode::getConfigType ()
```

This method returns the type of configuration.

Cast this instance appropriately to have access to specific getters.

Returns

The type of this config.

11.123.4.3 getMass()

double arcs::aubo_scope::SetPayloadNode::getMass ()

Returns

The total mass of the payload

11.123.4.4 setCenterOfGravity()

void arcs::aubo_scope::SetPayloadNode::setCenterOfGravity (
 const std::vector< double > & cog)

11.123.4.5 setConfigType()

void arcs::aubo_scope::SetPayloadNode::setConfigType (
 [ConfigType](#) type)

11.123.4.6 setMass()

void arcs::aubo_scope::SetPayloadNode::setMass (
 double mass)

11.123.5 Friends And Related Symbol Documentation

11.123.5.1 DataSwitch

friend class DataSwitch [friend]

Definition at line 88 of file [set_payload_node.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.123.6 Member Data Documentation

11.123.6.1 d_

void* arcs::aubo_scope::SetPayloadNode::d_ { nullptr } [private]

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

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

- include/aubo_caps/domain/program/nodes/builtin/[set_payload_node.h](#)

11.124 arcs::aubo_scope::SpeedCapability Class Reference

[SpeedCapability](#) This interface represents a registered speed capability for a gripper which supports operating using a user configurable speed.

```
#include <speed_capability.h>
```

Public Member Functions

- virtual [~SpeedCapability](#) ()=default
- virtual void [updateCapability](#) (double minSpeed, double maxSpeed, double defaultGripSpeed, double defaultReleaseSpeed)=0

11.124.1 Detailed Description

[SpeedCapability](#) This interface represents a registered speed capability for a gripper which supports operating using a user configurable speed.

Through this interface, it is possible to adjust the range and default values for the speed after the capability has been registered. This is typically not applicable for most grippers, but can be relevant when the range depends on another custom gripper setting (which can be configured by the user).

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

11.124.2 Constructor & Destructor Documentation

11.124.2.1 ~SpeedCapability()

```
virtual arcs::aubo_scope::SpeedCapability::~~SpeedCapability () [virtual], [default]
```

11.124.3 Member Function Documentation

11.124.3.1 updateCapability()

```
virtual void arcs::aubo_scope::SpeedCapability::updateCapability (
    double minSpeed,
    double maxSpeed,
    double defaultGripSpeed,
    double defaultReleaseSpeed) [pure virtual]
```

Update the range and default values for the speed supported by the gripper.

Updating the range could result in an already entered value in the configuration of a gripper program node being outside the range. This will make the given program node undefined, and force the end user to adjust the value to be valid again.

Parameters

minSpeed	minimum operating speed supported by the gripper
maxSpeed	maximum operating speed supported by the gripper
defaultGripSpeed	default value for the speed for a grip action. This value is used for, e.g. performing a "default" grip action using the toolbar
defaultReleaseSpeed	default value for the speed for a release action. This value is used for, e.g. performing a "default" release action using the toolbar
unit	the unit for all specified values, not null

Exceptions

InvalidCapabilityRange	if minSpeed > maxSpeed
InvalidCapabilityRange	if defaultGripSpeed or defaultReleaseSpeed is outside the range defined by minSpeed and maxSpeed.

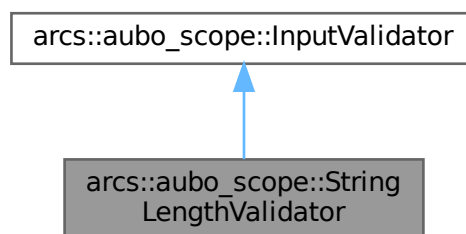
The documentation for this class was generated from the following file:

- include/aubo_caps/contribution/driver/gripper/capability/[speed_capability.h](#)

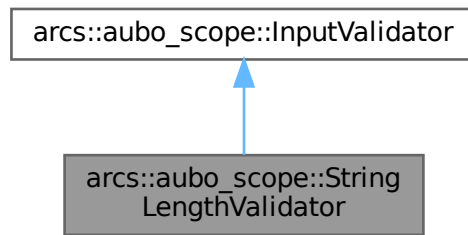
11.125 arcs::aubo_scope::StringLengthValidator Struct Reference

```
#include <input_validator.h>
```

Inheritance diagram for arcs::aubo_scope::StringLengthValidator:



Collaboration diagram for `arcs::aubo_scope::StringLengthValidator`:



Public Member Functions

- [StringLengthValidator](#) (`size_t min_length`, `size_t max_length`)
- [StringLengthValidator](#) (`std::function< std::pair< size_t, size_t >()> range_func`)
- `bool` [isValid](#) (`std::string value`) override
Validates whether the input value is valid.
- `std::string` [getMessage](#) (`std::string value`) override
Returns a meaningful message in case the value is not valid.

Private Member Functions

- `void` [updateValue](#) ()

Private Attributes

- `size_t` [min_length](#) { 0 }
- `size_t` [max_length](#) { ULONG_MAX }
- `std::function< std::pair< size_t, size_t >()>` [range_func](#) { nullptr }

11.125.1 Detailed Description

Definition at line 146 of file [input_validator.h](#).

11.125.2 Constructor & Destructor Documentation

11.125.2.1 StringLengthValidator() [1/2]

```

arcs::aubo_scope::StringLengthValidator::StringLengthValidator (
    size_t min_length,
    size_t max_length)
  
```

11.125.2.2 StringLengthValidator() [2/2]

```
arcs::aubo_scope::StringLengthValidator::StringLengthValidator (  
    std::function< std::pair< size_t, size_t >()> range_func)
```

11.125.3 Member Function Documentation

11.125.3.1 getMessage()

```
std::string arcs::aubo_scope::StringLengthValidator::getMessage (  
    std::string value)  [override], [virtual]
```

Returns a meaningful message in case the value is not valid.

Parameters

value	the invalid value. Can be included in the message if it makes sense.

Returns

message.

Implements [arcs::aubo_scope::InputValidator](#).

11.125.3.2 isValid()

```
bool arcs::aubo_scope::StringLengthValidator::isValid (  
    std::string value)  [override], [virtual]
```

Validates whether the input value is valid.

Parameters

value	to be validated.

Returns

true if value is valid.

Implements [arcs::aubo_scope::InputValidator](#).

11.125.3.3 updateValue()

```
void arcs::aubo_scope::StringLengthValidator::updateValue () [private]
```

11.125.4 Member Data Documentation

11.125.4.1 max_length__

```
size_t arcs::aubo_scope::StringLengthValidator::max_length__ { ULONG_MAX } [private]
```

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

11.125.4.2 min_length__

```
size_t arcs::aubo_scope::StringLengthValidator::min_length__ { 0 } [private]
```

Definition at line 161 of file [input_validator.h](#).

11.125.4.3 range_func__

```
std::function<std::pair<size_t, size_t>()> arcs::aubo_scope::StringLengthValidator::range_func__ { nullptr } [private]
```

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

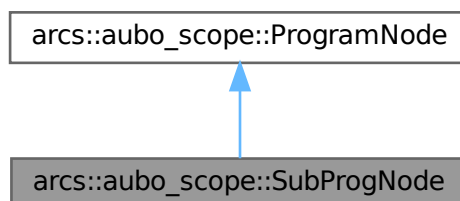
The documentation for this struct was generated from the following file:

- [include/aubo_caps/domain/user_interface/input_validator.h](#)

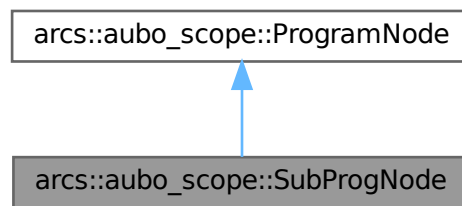
11.126 arcs::aubo_scope::SubProgNode Class Reference

```
#include <sub_prog_node.h>
```

Inheritance diagram for arcs::aubo_scope::SubProgNode:



Collaboration diagram for arcs::aubo_scope::SubProgNode:



Public Member Functions

- [SubProgNode](#) (SubProgNode &f)
- [SubProgNode](#) (SubProgNode &&f)
- virtual [~SubProgNode](#) ()

Public Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()
Get the program node contribution.

Private Member Functions

- [SubProgNode](#) ()

Private Attributes

- void * [d__](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Protected Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.126.1 Detailed Description

Definition at line 10 of file [sub_prog_node.h](#).

11.126.2 Constructor & Destructor Documentation

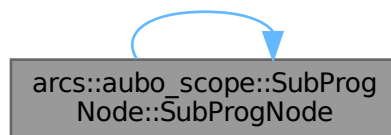
11.126.2.1 SubProgNode() [1/3]

```
arcs::aubo_scope::SubProgNode::SubProgNode (
    SubProgNode & f)
```

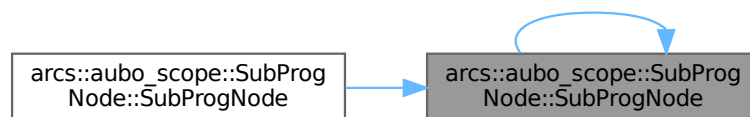
References [SubProgNode\(\)](#).

Referenced by [SubProgNode\(\)](#), and [SubProgNode\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

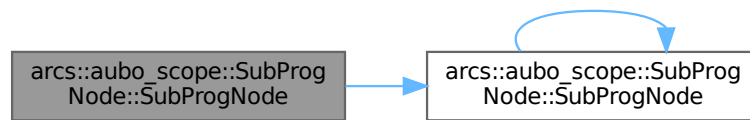


11.126.2.2 SubProgNode() [2/3]

```
arcs::aubo_scope::SubProgNode::SubProgNode (
    SubProgNode && f)
```

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

Here is the call graph for this function:



11.126.2.3 ~SubProgNode()

virtual arcs::aubo_scope::SubProgNode::~~SubProgNode () [virtual]

11.126.2.4 SubProgNode() [3/3]

arcs::aubo_scope::SubProgNode::SubProgNode () [private]

11.126.3 Friends And Related Symbol Documentation

11.126.3.1 DataSwitch

friend class DataSwitch [friend]

Definition at line 18 of file [sub_prog_node.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.126.4 Member Data Documentation

11.126.4.1 d_

void* arcs::aubo_scope::SubProgNode::d_ { nullptr } [private]

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

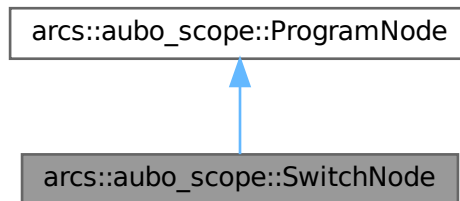
The documentation for this class was generated from the following file:

- [include/aubo_caps/domain/program/nodes/builtin/sub_prog_node.h](#)

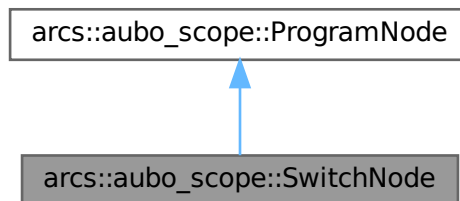
11.127 arcs::aubo_scope::SwitchNode Class Reference

#include <switch_node.h>

Inheritance diagram for arcs::aubo_scope::SwitchNode:



Collaboration diagram for arcs::aubo_scope::SwitchNode:



Public Member Functions

- [SwitchNode](#) (SwitchNode &f)
- [SwitchNode](#) (SwitchNode &&f)
- virtual [~SwitchNode](#) ()

Public Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()
Get the program node contribution.

Private Member Functions

- [SwitchNode](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Protected Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.127.1 Detailed Description

Definition at line 10 of file [switch_node.h](#).

11.127.2 Constructor & Destructor Documentation

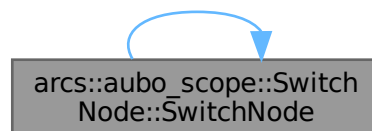
11.127.2.1 SwitchNode() [1/3]

```
arcs::aubo_scope::SwitchNode::SwitchNode (  
    SwitchNode & f)
```

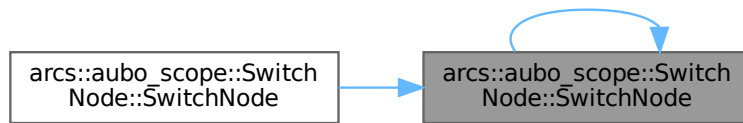
References [SwitchNode\(\)](#).

Referenced by [SwitchNode\(\)](#), and [SwitchNode\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

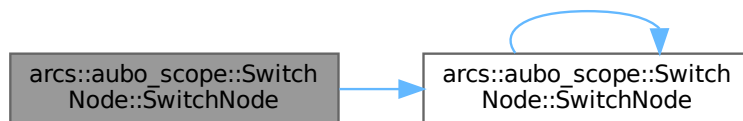


11.127.2.2 SwitchNode() [2/3]

```
arcs::aubo_scope::SwitchNode::SwitchNode (
    SwitchNode && f)
```

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

Here is the call graph for this function:



11.127.2.3 ~SwitchNode()

```
virtual arcs::aubo_scope::SwitchNode::~~SwitchNode () [virtual]
```

11.127.2.4 SwitchNode() [3/3]

```
arcs::aubo_scope::SwitchNode::SwitchNode () [private]
```

11.127.3 Friends And Related Symbol Documentation

11.127.3.1 DataSwitch

```
friend class DataSwitch [friend]
```

Definition at line 18 of file [switch_node.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.127.4 Member Data Documentation

11.127.4.1 d__

```
void* arcs::aubo_scope::SwitchNode::d__ { nullptr } [private]
```

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

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

- include/aubo_caps/domain/program/nodes/builtin/[switch_node.h](#)

11.128 arcs::aubo_scope::SystemApi Class Reference

```
#include <system_api.h>
```

Public Member Functions

- [SystemApi](#) (SystemApi &f)
- [SystemApi](#) (SystemApi &&f)
- virtual [~SystemApi](#) ()
- int [getMajorVersion](#) ()
Get the version of Software.
- int [getMinorVersion](#) ()
Get the minor version of Software.
- int [getBugfixVersion](#) ()
Get the bug fix version of Software.
- int [getBuildNumber](#) ()
Get the build number of Software.
- std::string [getLocale](#) ()
Returns the Locale for the system.
- std::string [getLocaleForProgrammingLanguage](#) ()
Returns the Locale for the programming language used in the system.
- [UnitType](#) [getUnitType](#) ()
Returns the [UnitType](#) of the system as an enum.
- std::string [getSerialNumber](#) ()
Returns the serial number of the robot.
- bool [isRealRobot](#) ()
Returns true if controlling the real, physical robot and false if in simulation mode.
- std::vector< double > [getJointMaxPositions](#) ()
Returns the max positions of robot joints.
- std::vector< double > [getJointMinPositions](#) ()
Returns the min positions of robot joints.
- double [getJointDefaultVelocity](#) ()
Returns the default velocity of robot joints.
- double [getJointDefaultAcceleration](#) ()
Returns the default acceleration of robot joints.
- std::vector< double > [getJointMaxVelocity](#) ()
Returns the max velocity of robot joints.

- `std::vector< double > getJointMaxAcceleration ()`
Returns the max acceleration of robot joints.
- `double getTcpDefaultVelocity ()`
Returns the default velocity of TCP.
- `double getTcpDefaultAcceleration ()`
Returns the default acceleration of TCP.
- `double getTcpMaxVelocity ()`
Returns the max velocity of TCP.
- `double getTcpMaxAcceleration ()`
Returns the max acceleration of TCP.
- `double getTcpMaxDistance ()`
Returns the max distance of TCP.
- `double getMaxPayloadMass ()`
Returns the max payload mass.
- `double getMaxPayloadDistance ()`
Returns the max payload distance.
- `std::vector< double > getHomeDefaultPosition ()`
Returns the default home position.
- `std::vector< double > getHomePackagePosition ()`
Returns the package home position.
- `std::unordered_map< std::string, std::vector< double > > getRobotDhParams (bool is_real)`
Returns the robot DH parameters.
- `IoConfigure getToolDigitalIOConfigure ()`
Returns the tool digital I/O configuration.
- `IoConfigure getStandardDigitalInputConfigure ()`
Returns the standard digital input configuration.
- `IoConfigure getConfigurableDigitalInputConfigure ()`
Returns the configurable digital input configuration.
- `IoConfigure getStandardDigitalOutputConfigure ()`
Returns the standard digital output configuration.
- `IoConfigure getConfigurableDigitalOutputConfigure ()`
Returns the configurable digital output configuration.
- `IoConfigure getStandardAnalogInputConfigure ()`
Returns the standard analog input configuration.
- `IoConfigure getToolAnalogInputConfigure ()`
Returns the tool analog input configuration.
- `IoConfigure getStandardAnalogOutputConfigure ()`
Returns the standard analog output configuration.
- `IoConfigure getToolAnalogOutputConfigure ()`
Returns the tool analog output configuration.
- `IoConfigure getStaticLinkInputConfigure ()`
Returns the static link input configuration.
- `IoConfigure getStaticLinkOutputConfigure ()`
Returns the static link output configuration.
- `IoConfigure getENCConfigure ()`
Returns the encoder configuration.

Private Member Functions

- `SystemApi ()`

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

11.128.1 Detailed Description

Definition at line 54 of file [system_api.h](#).

11.128.2 Constructor & Destructor Documentation

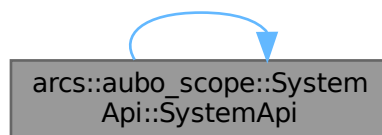
11.128.2.1 SystemApi() [1/3]

```
arcs::aubo_scope::SystemApi::SystemApi (  
    SystemApi & f)
```

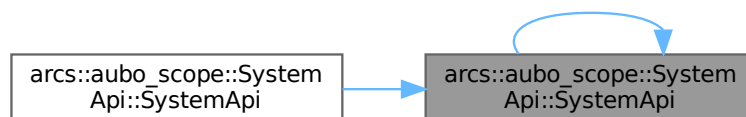
References [SystemApi\(\)](#).

Referenced by [SystemApi\(\)](#), and [SystemApi\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

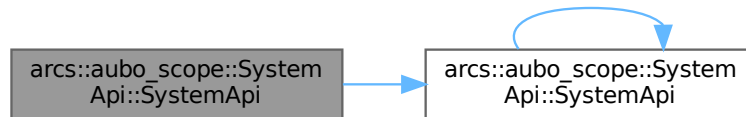


11.128.2.2 SystemApi() [2/3]

```
arcs::aubo_scope::SystemApi::SystemApi (
    SystemApi && f)
```

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

Here is the call graph for this function:



11.128.2.3 ~SystemApi()

```
virtual arcs::aubo_scope::SystemApi::~~SystemApi () [virtual]
```

11.128.2.4 SystemApi() [3/3]

```
arcs::aubo_scope::SystemApi::SystemApi () [private]
```

11.128.3 Friends And Related Symbol Documentation

11.128.3.1 DataSwitch

```
friend class DataSwitch [friend]
```

Definition at line 556 of file [system_api.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.128.4 Member Data Documentation

11.128.4.1 d__

```
void* arcs::aubo_scope::SystemApi::d__ { nullptr } [private]
```

Definition at line 558 of file [system_api.h](#).

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

- [include/aubo_caps/domain/system_api.h](#)

11.129 arcs::aubo_scope::TCP Class Reference

TCP This interface represents a **TCP** in AuboScope.

```
#include <tcp.h>
```

Public Member Functions

- **TCP** (TCP &f)
- **TCP** (TCP &&f)
- **~TCP** ()
- std::vector< double > **getOffset** ()
Returns the offset of the **TCP** with respect to the tool output flange.
- std::string **getDisplayname** ()
Note: The **TCP** can be renamed at any time.
- bool **isResolvable** ()
A **TCP** cannot be guaranteed to be present in AuboScope.
- size_t **hashCode** ()
Returns the hash code.

Private Member Functions

- **TCP** ()

Private Attributes

- void * **d_** { nullptr }

Friends

- class **DataSwitch**

11.129.1 Detailed Description

TCP This interface represents a **TCP** in AuboScope.

This can either be a user-defined **TCP** or a **TCP** added by a AuboCap.

Definition at line 25 of file **tcp.h**.

11.129.2 Constructor & Destructor Documentation

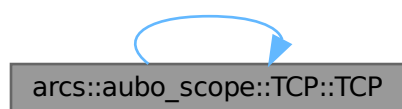
11.129.2.1 TCP() [1/3]

```
arcs::aubo_scope::TCP::TCP (  
    TCP & f)
```

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

Referenced by [TCP\(\)](#), and [TCP\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



11.129.2.2 TCP() [2/3]

```
arcs::aubo_scope::TCP::TCP (  
    TCP && f)
```

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

Here is the call graph for this function:



11.129.2.3 ~TCP()

arcs::aubo_scope::TCP::~~TCP ()

11.129.2.4 TCP() [3/3]

arcs::aubo_scope::TCP::TCP () [private]

11.129.3 Member Function Documentation

11.129.3.1 getDisplayName()

std::string arcs::aubo_scope::TCP::getDisplayName ()

Note: The [TCP](#) can be renamed at any time.

Returns

The name of the [TCP](#).

11.129.3.2 getOffset()

std::vector< double > arcs::aubo_scope::TCP::getOffset ()

Returns the offset of the [TCP](#) with respect to the tool output flange.

Returns

The offset of the [TCP](#).

11.129.3.3 hashCode()

size_t arcs::aubo_scope::TCP::hashCode ()

Returns the hash code.

Returns

the hash code.

11.129.3.4 isResolvable()

```
bool arcs::aubo_scope::TCP::isResolvable ()
```

A [TCP](#) cannot be guaranteed to be present in AboScope.

The end user can load a different installation (not containing the [TCP](#)) or the [TCP](#) was removed by the end user or the AboCap that added the tcp. This method can be used to determine, if the [TCP](#) is present.

Returns

true if this [TCP](#) is present in AboScope, otherwise false.

11.129.4 Friends And Related Symbol Documentation

11.129.4.1 DataSwitch

```
friend class DataSwitch [friend]
```

Definition at line 91 of file [tcp.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.129.5 Member Data Documentation

11.129.5.1 d__

```
void* arcs::aubo_scope::TCP::d__ { nullptr } [private]
```

Definition at line 93 of file [tcp.h](#).

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

- [include/aubo_caps/domain/tcp/tcp.h](#)

11.130 arcs::aubo_scope::TcpContributionModel Class Reference

[TcpContributionModel](#) This interface provides functionality for adding, updating and removing TCPs in AboScope.

```
#include <tcp_contribution_model.h>
```

Public Member Functions

- [TcpContributionModel](#) ([TcpContributionModel](#) &f)
- [TcpContributionModel](#) ([TcpContributionModel](#) &&f)
- virtual [~TcpContributionModel](#) ()
- [TCPPtr](#) [addTcp](#) (const std::string &idKey, const std::string &suggested_name, const std::vector< double > &offset)
[TCP](#) Add a [TCP](#) to the current AuboScope installation.
- [TCPPtr](#) [getTcp](#) (const std::string &idKey)
[TCP](#) Returns the [TCP](#) previously added by this AuboCap using the same idKey.
- void [updateTcp](#) (const std::string &idKey, const std::vector< double > &newOffset)
Update the offset of a [TCP](#) already added by this AuboCap.
- void [removeTcp](#) (const std::string &idKey)
Remove a [TCP](#) added by this AuboCap from AuboScope.

Private Member Functions

- [TcpContributionModel](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

11.130.1 Detailed Description

[TcpContributionModel](#) This interface provides functionality for adding, updating and removing TCPs in AuboScope.

Note: This interface must only be used in an installation contribution.

Definition at line 27 of file [tcp_contribution_model.h](#).

11.130.2 Constructor & Destructor Documentation

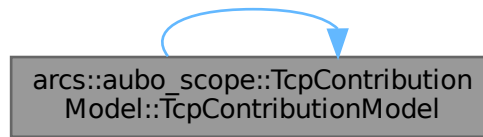
11.130.2.1 TcpContributionModel() [1/3]

```
arcs::aubo_scope::TcpContributionModel::TcpContributionModel (  
    TcpContributionModel & f)
```

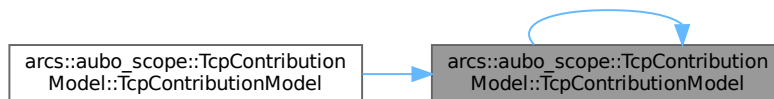
References [TcpContributionModel\(\)](#).

Referenced by [TcpContributionModel\(\)](#), and [TcpContributionModel\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

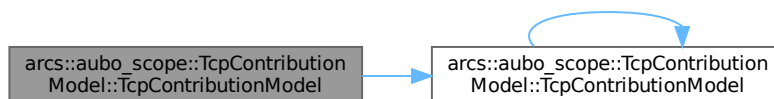


11.130.2.2 `TcpContributionModel()` [2/3]

```
arcs::aubo_scope::TcpContributionModel::TcpContributionModel (
    TcpContributionModel && f)
```

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

Here is the call graph for this function:



11.130.2.3 `~TcpContributionModel()`

```
virtual arcs::aubo_scope::TcpContributionModel::~~TcpContributionModel () [virtual]
```

11.130.2.4 `TcpContributionModel()` [3/3]

```
arcs::aubo_scope::TcpContributionModel::TcpContributionModel () [private]
```

11.130.3 Member Function Documentation

11.130.3.1 addTcp()

```
TCPPtr arcs::aubo_scope::TcpContributionModel::addTcp (
    const std::string & idKey,
    const std::string & suggested_name,
    const std::vector< double > & offset)
```

TCP Add a **TCP** to the current AuboScope installation.

This makes it selectable by the end user. The **TCP** is not modifiable by the end user.

Parameters

idKey	The key to identify this TCP by.
suggested_name	Suggested name for the TCP .
offset	The offset of the TCP with respect to the tool output flange.

Returns

The **TCP** created and registered in AuboScope.

Exceptions

TCPAlreadyAddedException	If called twice using the same idKey.
IllegalTCPNameException	If the suggested name does not match regex.
IllegalArgumentException	If the offset is invalid.

11.130.3.2 getTcp()

```
TCPPtr arcs::aubo_scope::TcpContributionModel::getTcp (
    const std::string & idKey)
```

TCP Returns the **TCP** previously added by this AuboCap using the same idKey.

Parameters

idKey	The key to identify this TCP by.

The **TCP** previously added by this AuboCap.

Parameters

idKey	The key used to add the TCP with.

Exceptions

TCPNotFoundException	If no TCP exists with the provided idKey.

11.130.3.4 updateTcp()

```
void arcs::aubo_scope::TcpContributionModel::updateTcp (
    const std::string & idKey,
    const std::vector< double > & newOffset)
```

Update the offset of a [TCP](#) already added by this AuboCap.

Parameters

idKey	The key to identify the TCP .
newOffset	The new offset to set for the TCP .

Exceptions

TCPNotFoundException	If no TCP exists with the provided idKey.
IllegalArgumentException	If the offset is invalid.

11.130.4 Friends And Related Symbol Documentation

11.130.4.1 DataSwitch

```
friend class DataSwitch [friend]
```

Definition at line 116 of file [tcp_contribution_model.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.130.5 Member Data Documentation

11.130.5.1 d_

void* arcs::aubo_scope::TcpContributionModel::d_ { nullptr } [private]

Definition at line 118 of file [tcp_contribution_model.h](#).

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

- include/aubo_caps/domain/tcp/[tcp_contribution_model.h](#)

11.131 arcs::aubo_scope::TcpModel Class Reference

[TcpModel](#) This interface provides access to the TCPs that are currently available in AuboScope.

```
#include <tcp_model.h>
```

Public Member Functions

- [TcpModel](#) (TcpModel &f)
- [TcpModel](#) (TcpModel &&f)
- virtual [~TcpModel](#) ()
- std::vector< TCPPtr > [getAll](#) ()
Returns a collection of all the TCPs in AuboScope.
- TCPPtr [getTcp](#) (const std::string &name)
[TCP](#) Get a [TCP](#) by name.
- TCPPtr [getDefaultTcp](#) ()
[TCP](#) Returns the default [TCP](#).
- std::string [connectedTo](#) (const std::string &receiver, const std::function< void([TcpModel](#) *)> &slot)
[Register](#) callback for changed.
- void [disconnectFrom](#) (const std::string &receiver)
Disconnect a callback.

Private Member Functions

- [TcpModel](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

11.131.1 Detailed Description

[TcpModel](#) This interface provides access to the TCPs that are currently available in AuboScope.

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

11.131.2 Constructor & Destructor Documentation

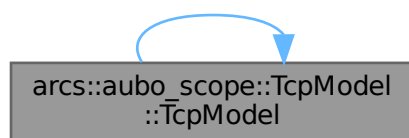
11.131.2.1 TcpModel() [1/3]

```
arcs::aubo_scope::TcpModel::TcpModel (  
    TcpModel & f)
```

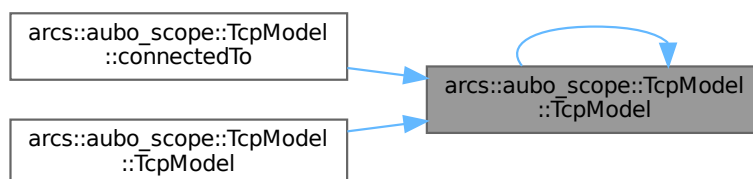
References [TcpModel\(\)](#).

Referenced by [connectedTo\(\)](#), [TcpModel\(\)](#), and [TcpModel\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



11.131.2.2 TcpModel() [2/3]

```
arcs::aubo_scope::TcpModel::TcpModel (
    TcpModel && f)
```

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

Here is the call graph for this function:



11.131.2.3 ~TcpModel()

```
virtual arcs::aubo_scope::TcpModel::~~TcpModel () [virtual]
```

11.131.2.4 TcpModel() [3/3]

```
arcs::aubo_scope::TcpModel::TcpModel () [private]
```

11.131.3 Member Function Documentation

11.131.3.1 connectedTo()

```
std::string arcs::aubo_scope::TcpModel::connectedTo (
    const std::string & receiver,
    const std::function< void(TcpModel *)> & slot)
```

[Register](#) callback for changed.

Parameters

receiver	the receiver identifier.
slot	the callback function.

Returns

the connection identifier.

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

Here is the call graph for this function:



11.131.3.2 disconnectFrom()

```
void arcs::aubo_scope::TcpModel::disconnectFrom (
    const std::string & receiver)
```

Disconnect a callback.

Parameters

receiver	the receiver identifier.

11.131.3.3 getAll()

```
std::vector< TCPPtr > arcs::aubo_scope::TcpModel::getAll ()
```

Returns a collection of all the TCPs in AuboScope.

Returns

A collection of all the TCPs in AuboScope.

11.131.3.4 getDefaultTcp()

```
TCPPtr arcs::aubo_scope::TcpModel::getDefaultTcp ()
```

[TCP](#) Returns the default [TCP](#).

Returns

the default [TCP](#).

Parameters

name	the name of the TCP .

Returns

the [TCP](#) with the given name.

11.131.4 Friends And Related Symbol Documentation

11.131.4.1 DataSwitch

friend class DataSwitch [friend]

Definition at line [105](#) of file [tcp_model.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.131.5 Member Data Documentation

11.131.5.1 d_

void* arcs::aubo_scope::TcpModel::d_ { nullptr } [private]

Definition at line [107](#) of file [tcp_model.h](#).

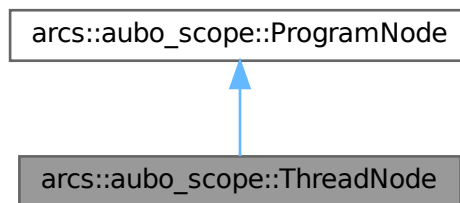
The documentation for this class was generated from the following file:

- [include/aubo_caps/domain/tcp/tcp_model.h](#)

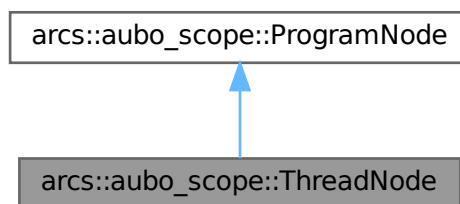
11.132 arcs::aubo_scope::ThreadNode Class Reference

#include <thread_node.h>

Inheritance diagram for arcs::aubo_scope::ThreadNode:



Collaboration diagram for arcs::aubo_scope::ThreadNode:



Public Member Functions

- [ThreadNode](#) (ThreadNode &f)
- [ThreadNode](#) (ThreadNode &&f)
- virtual [~ThreadNode](#) ()

Public Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()
Get the program node contribution.

Private Member Functions

- [ThreadNode](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Protected Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.132.1 Detailed Description

Definition at line 10 of file [thread_node.h](#).

11.132.2 Constructor & Destructor Documentation

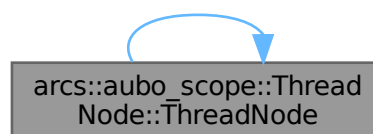
11.132.2.1 ThreadNode() [1/3]

```
arcs::aubo_scope::ThreadNode::ThreadNode (  
    ThreadNode & f)
```

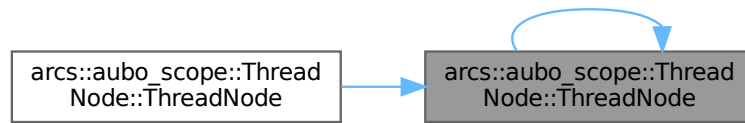
References [ThreadNode\(\)](#).

Referenced by [ThreadNode\(\)](#), and [ThreadNode\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

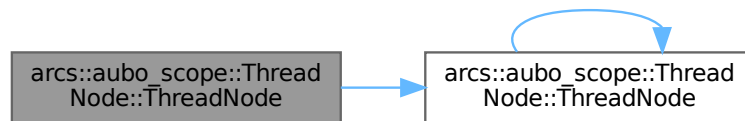


11.132.2.2 ThreadNode() [2/3]

```
arcs::aubo_scope::ThreadNode::ThreadNode (
    ThreadNode && f)
```

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

Here is the call graph for this function:



11.132.2.3 ~ThreadNode()

```
virtual arcs::aubo_scope::ThreadNode::~~ThreadNode () [virtual]
```

11.132.2.4 ThreadNode() [3/3]

```
arcs::aubo_scope::ThreadNode::ThreadNode () [private]
```

11.132.3 Friends And Related Symbol Documentation

11.132.3.1 DataSwitch

```
friend class DataSwitch [friend]
```

Definition at line 18 of file [thread_node.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.132.4 Member Data Documentation

11.132.4.1 d_

```
void* arcs::aubo_scope::ThreadNode::d_ { nullptr } [private]
```

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

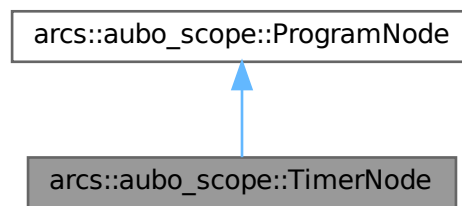
The documentation for this class was generated from the following file:

- [include/aubo_caps/domain/program/nodes/builtin/thread_node.h](#)

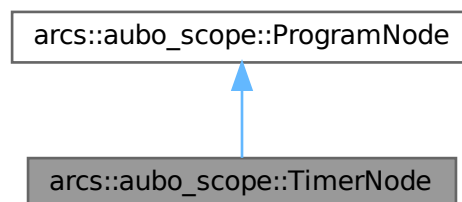
11.133 arcs::aubo_scope::TimerNode Class Reference

```
#include <timer_node.h>
```

Inheritance diagram for arcs::aubo_scope::TimerNode:



Collaboration diagram for arcs::aubo_scope::TimerNode:



Public Member Functions

- [TimerNode](#) (TimerNode &f)
- [TimerNode](#) (TimerNode &&f)
- virtual [~TimerNode](#) ()

Public Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()
Get the program node contribution.

Private Member Functions

- [TimerNode](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Protected Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.133.1 Detailed Description

Definition at line 10 of file [timer_node.h](#).

11.133.2 Constructor & Destructor Documentation

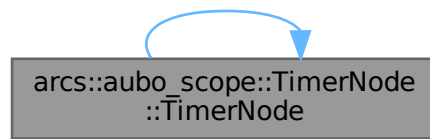
11.133.2.1 [TimerNode](#)() [1/3]

```
arcs::aubo_scope::TimerNode::TimerNode (  
    TimerNode & f)
```

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

Referenced by [TimerNode\(\)](#), and [TimerNode\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

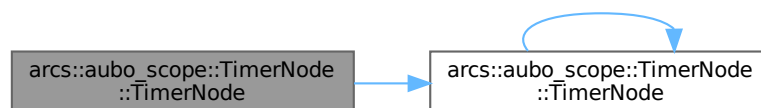


11.133.2.2 TimerNode() [2/3]

```
arcs::aubo_scope::TimerNode::TimerNode (
    TimerNode && f)
```

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

Here is the call graph for this function:



11.133.2.3 ~TimerNode()

```
virtual arcs::aubo_scope::TimerNode::~~TimerNode () [virtual]
```

11.133.2.4 TimerNode() [3/3]

arcs::aubo_scope::TimerNode::TimerNode () [private]

11.133.3 Friends And Related Symbol Documentation

11.133.3.1 DataSwitch

friend class DataSwitch [friend]

Definition at line 18 of file [timer_node.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.133.4 Member Data Documentation

11.133.4.1 d_

void* arcs::aubo_scope::TimerNode::d_ { nullptr } [private]

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

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

- include/aubo_caps/domain/program/nodes/builtin/[timer_node.h](#)

11.134 arcs::aubo_scope::TreeNode Class Reference

[TreeNode](#) This interface represents a node in the program tree that can be used to construct a sub-tree rooted in a AuboCap program node.

```
#include <tree_node.h>
```

Public Member Functions

- [TreeNode](#) (TreeNode &f)
- [TreeNode](#) (TreeNode &&f)
- virtual [~TreeNode](#) ()
- TreeNodePtr [addChild](#) (ProgramNodePtr program_node)

Add a child program node to the sub-tree.
- TreeNodePtr [insertChildBefore](#) (TreeNodePtr existingChildNode, ProgramNodePtr program_node)

Inserts a child program node in the sub-tree directly before the existing selected child node.
- TreeNodePtr [insertChildAfter](#) (TreeNodePtr existingChildNode, ProgramNodePtr program_node)

Inserts a child program node under in the sub-tree directly after the existing selected child node.
- TreeNodePtr [cutChildNode](#) (TreeNodePtr child)

Cut a child program node under in the sub-tree directly after the existing selected child node.
- bool [removeChild](#) (TreeNodePtr program_node)

Removes a child node from the sub-tree.
- std::vector< TreeNodePtr > [getChildren](#) ()
- std::vector< TreeNodePtr > [getChildrenNotSuppressed](#) ()
- TreeNodePtr [getParent](#) ()
- ProgramNodePtr [getProgramNode](#) ()
- bool [isSuppressed](#) ()
- bool [isInProgramTree](#) ()

Whether the node is in the program tree; nodes not in the program tree may have been cut or deleted.
- TreeNodePtr [locateDescendantTreeNode](#) (ProgramNodePtr program_node)
- void [setChildSequenceLocked](#) (bool isChildSequenceLocked)

Configures whether or not child nodes can be rearranged, deleted or have other nodes inserted into the child sequence by the end user.
- int [traverse](#) (std::function< int(ProgramNodePtr, int, int)> nodeVisitor)

Private Member Functions

- [TreeNode](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

11.134.1 Detailed Description

[TreeNode](#) This interface represents a node in the program tree that can be used to construct a sub-tree rooted in a AuboCap program node.

Using the ProgramModel#getRootTreeNode(ProgramNodeContribution) to obtain a root for the sub-tree, it is possible to add children. For each call to TreeNode#addChild(ProgramNode), a new [TreeNode](#) is returned, that can, in turn, act as a root for yet another sub-tree.

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

11.134.2 Constructor & Destructor Documentation

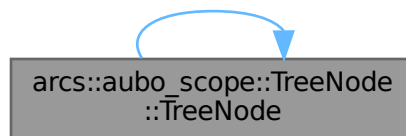
11.134.2.1 `TreeNode()` [1/3]

`arcs::aubo_scope::TreeNode::TreeNode (`
`TreeNode & f)`

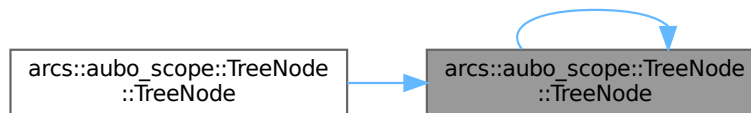
References [TreeNode\(\)](#).

Referenced by [TreeNode\(\)](#), and [TreeNode\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

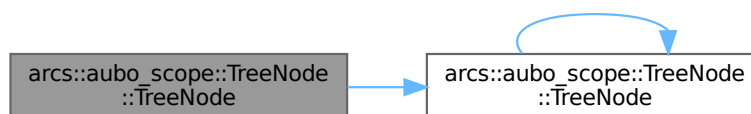


11.134.2.2 `TreeNode()` [2/3]

`arcs::aubo_scope::TreeNode::TreeNode (`
`TreeNode && f)`

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

Here is the call graph for this function:



11.134.2.3 ~TreeNode()

virtual arcs::aubo_scope::TreeNode::~TreeNode () [virtual]

11.134.2.4 TreeNode() [3/3]

arcs::aubo_scope::TreeNode::TreeNode () [private]

11.134.3 Member Function Documentation

11.134.3.1 addChild()

TreeNodePtr arcs::aubo_scope::TreeNode::addChild (
ProgramNodePtr program_node)

Add a child program node to the sub-tree.

Parameters

program_node	the ProgramNode constructed using the ProgramNodeFactory

Returns

Returns a [TreeNode](#) that can be used to add children to the newly added child.

Exceptions

TreeStructureException	If it is not allowed to insert the ProgramNode at this position a TreeStructureException will be thrown.
IllegalStateException	if called from a Qt-based AuboCap outside of an UndoableChanges scope (see also UndoRedoManager).

11.134.3.2 cutChildNode()

TreeNodePtr arcs::aubo_scope::TreeNode::cutChildNode (
TreeNodePtr child)

Cut a child program node under in the sub-tree directly after the existing selected child node.

Shifts any subsequent nodes to positions after the newly added child.

Parameters

child	existing TreeNode child of this TreeNode .

Returns

Returns the [TreeNode](#) that has been cut off.

Exceptions

TreeStructureException	If it is not allowed to cut the ProgramNode or if the selected child node is not a child of this TreeNode a TreeStructureException will be thrown.
IllegalStateException	if called from a Qt-based AuboCap outside of an UndoableChanges scope (see also UndoRedoManager).

11.134.3.3 getChildren()

```
std::vector< TreeNodePtr > arcs::aubo_scope::TreeNode::getChildren ()
```

Returns

a list of [TreeNode](#) objects that represents all the children of this [TreeNode](#) (both programmatically added as well as inserted by the end user).

11.134.3.4 getChildrenNotSuppressed()

```
std::vector< TreeNodePtr > arcs::aubo_scope::TreeNode::getChildrenNotSuppressed ()
```

11.134.3.5 getParent()

```
TreeNodePtr arcs::aubo_scope::TreeNode::getParent ()
```

Returns

Parent treenode of the node

11.134.3.6 getProgramNode()

```
ProgramNodePtr arcs::aubo_scope::TreeNode::getProgramNode ()
```

Returns

Returns the [ProgramNode](#) at this position in the sub-tree. Can either be a built-in AuboScope program node (provided by AUBO Robots) or a AuboCap program node.

Parameters

existingChildNode	existing TreeNode child of this TreeNode .
program_node	the ProgramNode constructed using the ProgramNodeFactory .

Returns

Returns a [TreeNode](#) that can be used to add children to the newly added child.

Exceptions

TreeStructureException	If it is not allowed to insert the ProgramNode at this position or if the selected child node is not a child of this TreeNode a TreeStructureException will be thrown.
IllegalStateException	if called from a Qt-based AuboCap outside of an UndoableChanges scope (see also UndoRedoManager).

11.134.3.8 insertChildBefore()

```
TreeNodePtr arcs::aubo_scope::TreeNode::insertChildBefore (
    TreeNodePtr existingChildNode,
    ProgramNodePtr program_node)
```

Inserts a child program node in the sub-tree directly before the existing selected child node.

Shifts the selected child node and any subsequent nodes to positions after the newly added child.

Parameters

existingChildNode	existing TreeNode child of this TreeNode .
program_node	the ProgramNode constructed using the ProgramNodeFactory .

Returns

Returns a [TreeNode](#) that can be used to add children to the newly added child.

Exceptions

TreeStructureException	If it is not allowed to insert the ProgramNode at this position or if the selected child node is not a child of this TreeNode a TreeStructureException will be thrown.
IllegalStateException	if called from a Qt-based AuboCap outside of an UndoableChanges scope (see also UndoRedoManager).

11.134.3.9 isInProgramTree()

```
bool arcs::aubo_scope::TreeNode::isInProgramTree ()
```

Whether the node is in the program tree; nodes not in the program tree may have been cut or deleted.

Returns

Returns true The node is in the program tree. false The node is not in the program tree; such nodes may have been cut or deleted.

11.134.3.10 isSuppressed()

```
bool arcs::aubo_scope::TreeNode::isSuppressed ()
```

11.134.3.11 locateDescendantTreeNode()

```
TreeNodePtr arcs::aubo_scope::TreeNode::locateDescendantTreeNode (
    ProgramNodePtr program_node)
```

Gets a corresponding [TreeNode](#) instance for a child program node (a [ProgramNode](#) instance) in the sub-tree under this [TreeNode](#).

This method can for instance be used to gain access to the sub-tree under a child program node encountered when iterating the sub-tree of the parent node using the ProgramNodeVisitor (or any derived sub class thereof).

Parameters

program_node	program node to get a corresponding TreeNode representation for, not null. Can either be a built-in AuboScope program node (provided by AUBO Robots) or a AuboCap program node.

Returns

the [TreeNode](#) instance for the specified program node.

Exceptions

ProgramNodeNotInSubTreeException	when the program node cannot be found because it is not in the sub-tree.

11.134.3.12 removeChild()

```
bool arcs::aubo_scope::TreeNode::removeChild (
    TreeNodePtr program_node)
```

Removes a child node from the sub-tree.

Be aware that removing the last child will trigger the insertion of an <empty> child node.

Parameters

program_node	The TreeNode child to be removed.

Returns

Returns true if removed successfully. false otherwise.

Exceptions

TreeStructureException	If the removed child would leave the tree in an illegal state a TreeStructureException will be thrown.
IllegalStateException	if called from a Qt-based AuboCap outside of an UndoableChanges scope (see also UndoRedoManager).

11.134.3.13 setChildSequenceLocked()

```
void arcs::aubo_scope::TreeNode::setChildSequenceLocked (
    bool isChildSequenceLocked)
```

Configures whether or not child nodes can be rearranged, deleted or have other nodes inserted into the child sequence by the end user.

Parameters

isChildSequenceLocked	If true then the immediate children under this TreeNode will be locked.

11.134.3.14 traverse()

```
int arcs::aubo_scope::TreeNode::traverse (
    std::function< int(ProgramNodePtr, int, int)> nodeVisitor)
```

This method traverses the entire sub-tree under this tree node in a depth-first fashion (this corresponds to a top-down approach in the program tree).

A node visitor is used for callbacks to the visit-overloads you choose to override. The overload called depends on the node type encountered. Override the overloads for the node types you are concerned with. All visit-methods have the program node, sibling index and depth as arguments to help filter nodes if needed.

The node visitor can be either a `ProgramNodeVisitor` implementation with optional overrides or a `ProgramNodeInterfaceVisitor` implementation. In the latter case, the `ProgramNodeInterfaceVisitor#visitAs(Object, int, int)` method must be implemented.

The `ProgramNodeInterfaceVisitor` can be used when targeting AuboCap program nodes implementing the (generic) type parameter specified in `ProgramNodeInterfaceVisitor` (see also `ProgramNode#getAs(Class)`).

Note that this method is sometimes called `accept()` in the Visitor software design pattern.

Parameters

node↔ Visitor	the instance callbacks are made to.

Returns

0 for normal exit. non-zero for exception or error causing premature exit.

11.134.4 Friends And Related Symbol Documentation

11.134.4.1 DataSwitch

```
friend class DataSwitch [friend]
```

Definition at line 342 of file [tree_node.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.134.5 Member Data Documentation

11.134.5.1 d_

`void* arcs::aubo_scope::TreeNode::d_ { nullptr } [private]`

Definition at line 344 of file [tree_node.h](#).

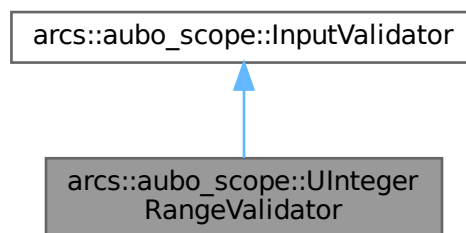
The documentation for this class was generated from the following file:

- [include/aubo_caps/domain/program/structure/tree_node.h](#)

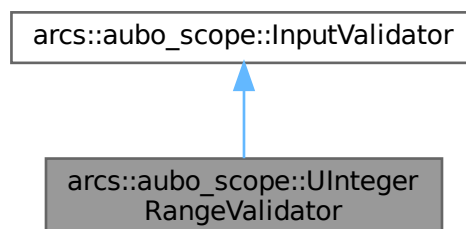
11.135 arcs::aubo_scope::UIntegerRangeValidator Struct Reference

```
#include <input_validator.h>
```

Inheritance diagram for arcs::aubo_scope::UIntegerRangeValidator:



Collaboration diagram for arcs::aubo_scope::UIntegerRangeValidator:



Public Member Functions

- [UIntegerRangeValidator](#) (uint32_t min_value, uint32_t max_value)
- [UIntegerRangeValidator](#) (std::function< std::pair< uint32_t, uint32_t >()> range_func)
- bool [isValid](#) (std::string value) override
Validates whether the input value is valid.
- std::string [getMessage](#) (std::string value) override
Returns a meaningful message in case the value is not valid.

Private Member Functions

- void [updateValue](#) ()

Private Attributes

- uint32_t [min_value_](#) { 0 }
- uint32_t [max_value_](#) { UINT_MAX }
- std::function< std::pair< uint32_t, uint32_t >()> [range_func_](#) { nullptr }

11.135.1 Detailed Description

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

11.135.2 Constructor & Destructor Documentation

11.135.2.1 UIntegerRangeValidator() [1/2]

```
arcs::aubo_scope::UIntegerRangeValidator::UIntegerRangeValidator (
    uint32_t min_value,
    uint32_t max_value)
```

11.135.2.2 UIntegerRangeValidator() [2/2]

```
arcs::aubo_scope::UIntegerRangeValidator::UIntegerRangeValidator (
    std::function< std::pair< uint32_t, uint32_t >()> range_func)
```

11.135.3 Member Function Documentation

11.135.3.1 getMessage()

```
std::string arcs::aubo_scope::UIntegerRangeValidator::getMessage (
    std::string value)  [override], [virtual]
```

Returns a meaningful message in case the value is not valid.

Parameters

value	the invalid value. Can be included in the message if it makes sense.

Returns

message.

Implements [arcs::aubo_scope::InputValidator](#).

11.135.3.2 isValid()

```
bool arcs::aubo_scope::UIntegerRangeValidator::isValid (
    std::string value)  [override], [virtual]
```

Validates whether the input value is valid.

Parameters

value	to be validated.

Returns

true if value is valid.

Implements [arcs::aubo_scope::InputValidator](#).

11.135.3.3 updateValue()

```
void arcs::aubo_scope::UIntegerRangeValidator::updateValue ()  [private]
```

11.135.4 Member Data Documentation

11.135.4.1 max_value__

```
uint32_t arcs::aubo_scope::UIntegerRangeValidator::max_value__ { UINT_MAX }  [private]
```

Definition at line 113 of file [input_validator.h](#).

11.135.4.2 min_value__

```
uint32_t arcs::aubo_scope::UIntegerRangeValidator::min_value__ { 0 }  [private]
```

Definition at line 112 of file [input_validator.h](#).

11.135.4.3 range_func_

```
std::function<std::pair<uint32_t, uint32_t>()> arcs::aubo_scope::UIntegerRangeValidator::range_func_ { nullptr }
[private]
```

Definition at line 115 of file [input_validator.h](#).

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

- [include/aubo_caps/domain/user_interface/input_validator.h](#)

11.136 arcs::aubo_scope::UndoRedoManager Class Reference

[UndoRedoManager](#) This interface is used to record changes triggered by an end user in the screen of a AboCap program node contribution (e.g.

```
#include <undo_redo_manager.h>
```

Public Member Functions

- [UndoRedoManager](#) (UndoRedoManager &f)
- [UndoRedoManager](#) (UndoRedoManager &&f)
- virtual [~UndoRedoManager](#) ()
- void [recordChanges](#) (const std::string &text, std::function< void()> undoableChanges)

Private Member Functions

- [UndoRedoManager](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

11.136.1 Detailed Description

[UndoRedoManager](#) This interface is used to record changes triggered by an end user in the screen of a AboCap program node contribution (e.g.

when a button is clicked). It should be used in a AboCap program node to support the Undo/Redo functionality in AboScope.

NOTE: Do not use this functionality for AboCap installation nodes, since it is not supported (and has no effect).

NOTE: This functionality is not relevant for AboCap program nodes with a HTML-based user interface, since automatic undo is supported for these.

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

11.136.2 Constructor & Destructor Documentation

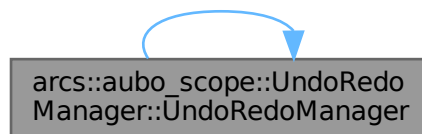
11.136.2.1 UndoRedoManager() [1/3]

arcs::aubo_scope::UndoRedoManager::UndoRedoManager (
UndoRedoManager & f)

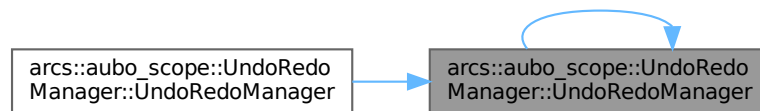
References [UndoRedoManager\(\)](#).

Referenced by [UndoRedoManager\(\)](#), and [UndoRedoManager\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

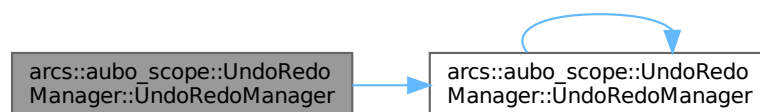


11.136.2.2 UndoRedoManager() [2/3]

arcs::aubo_scope::UndoRedoManager::UndoRedoManager (
UndoRedoManager && f)

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

Here is the call graph for this function:



11.136.2.3 ~UndoRedoManager()

virtual arcs::aubo_scope::UndoRedoManager::~~UndoRedoManager () [virtual]

11.136.2.4 UndoRedoManager() [3/3]

arcs::aubo_scope::UndoRedoManager::UndoRedoManager () [private]

11.136.3 Member Function Documentation

11.136.3.1 recordChanges()

```
void arcs::aubo_scope::UndoRedoManager::recordChanges (
    const std::string & text,
    std::function< void()> undoableChanges)
```

Group changes into one Undo/Redo step.

Currently Undo/Redo supports changes to the program tree and the data model.

Parameters

undoableChanges	Code to be recorded as one Undo/Redo step. Pressing Undo in AuboScope will undo all changes done to the program tree and the data model inside this block of code.

11.136.4 Friends And Related Symbol Documentation

11.136.4.1 DataSwitch

friend class DataSwitch [friend]

Definition at line 69 of file [undo_redo_manager.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.136.5 Member Data Documentation

11.136.5.1 d_

void* arcs::aubo_scope::UndoRedoManager::d_ { nullptr } [private]

Definition at line 71 of file [undo_redo_manager.h](#).

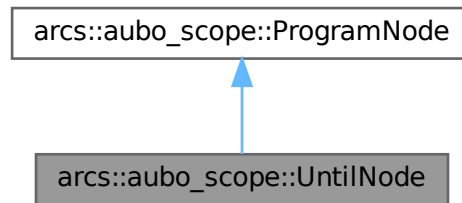
The documentation for this class was generated from the following file:

- [include/aubo_caps/domain/undoredo/undo_redo_manager.h](#)

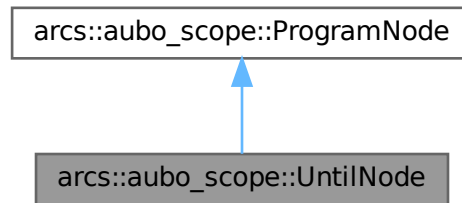
11.137 arcs::aubo_scope::UntilNode Class Reference

```
#include <until_node.h>
```

Inheritance diagram for arcs::aubo_scope::UntilNode:



Collaboration diagram for arcs::aubo_scope::UntilNode:



Public Types

- enum [ConfigType](#) : short { [EXPRESSION](#) , [TOOL_CONTACT](#) , [IO_INPUT](#) , [DISTANCE](#) , [REACHED_WAYPOINT](#) , [NONE_CONFIG_TYPE](#) }
The configuration type used to determine which type of configuration this instance is.
- enum [CompareOperator](#) : int { [LESS_THAN](#) , [GREATER_THAN](#) }
Compare operator types LESS_THAN and GREATER_THAN used when waiting for an analog input to go lower or higher, respectively, than a threshold.
- enum [BlendParameterType](#) { [NO_BLEND](#) , [CUSTOM_BLEND](#) }
Blend Parameters can be of different types The parameter type is used to determine the explicit blend parameters (can be used in place of instanceof operations).
- enum [CustomDecelerationParameterType](#) { [Unknown](#) , [Angular](#) , [Cartesian](#) }
- enum [DecelerationParameterType](#) { [SHARED_DECELERATION](#) , [CUSTOM_DECELERATION](#) }
The deceleration parameter type used to determine which type of deceleration parameter this instance is.
- enum [IOInputSelectionType](#) { [NONE_INPUT_SELECT](#) , [DIGITAL_INPUT](#) , [ANALOG_INPUT](#) , [FLOAT_REGISTER_INPUT](#) }
The type of the input selection.

Public Member Functions

- [UntilNode](#) (UntilNode &f)
- [UntilNode](#) (UntilNode &&f)
- virtual [~UntilNode](#) ()
- [ConfigType](#) [getConfigType](#) ()

This method returns the type of configuration.
- void [setConfigType](#) ([ConfigType](#) type)
- [BlendParameterType](#) [getBlendParameterType](#) ()
- void [setBlendParameterType](#) ([BlendParameterType](#) type)
- void [setWaypointBlendParameter](#) (double suggested_blend)

Creates blend parameters for an Until node.
- double [getWaypointBlendParameter](#) ()
- [DecelerationParameterType](#) [getDecelerationParameterType](#) ()
- void [setDecelerationParameterType](#) ([DecelerationParameterType](#) type)
- void [setDecelerationParameters](#) (double deceleration)
- double [getDecelerationParameters](#) ()
- [CustomDecelerationParameterType](#) [getCustomDecelerationParameterType](#) ()
- void [setCustomDecelerationParameterType](#) ([CustomDecelerationParameterType](#) type)
- double [getAngularDeceleration](#) ()

The rate at which to decelerate when the until condition is met.
- void [setAngularDeceleration](#) (double dec)
- double [getCartesianDeceleration](#) ()

The rate at which to decelerate when the until condition is met.
- void [setCartesianDeceleration](#) (double dec)
- void [setExpressionConfig](#) ([ExpressionPtr](#) expression)
- [ExpressionPtr](#) [getExpressionConfig](#) ()
- double [getDistance](#) ()

The distance the robot will travel.
- void [setDistance](#) (double dis)
- [IOInputSelectionType](#) [getIOInputSelectionType](#) ()
- void [setIOInputSelectionType](#) ([IOInputSelectionType](#) type)
- void [setDigitalInputValueToProceed](#) (bool value_to_proceed_until)

Creates an I/O Input configuration for an Until node.
- bool [getDigitalInputValueToProceed](#) ()
- void [setAnalogInputValueToProceed](#) (double value)

Creates an I/O Input configuration for an Until node.
- double [getAnalogInputValueToProceed](#) ()
- void [setCompareOperator](#) ([CompareOperator](#) compare_operator)
- [CompareOperator](#) [getCompareOperator](#) ()

Public Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- [ProgramNodeContributionPtr](#) [getProgramNodeContribution](#) ()

Get the program node contribution.

Private Member Functions

- [UntilNode](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Protected Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.137.1 Detailed Description

The Until node is a type of program node that must be under a [Waypoint](#) node or Direction node.

It has special behavior when calling `TreeNode#addChild(ProgramNode)` and `TreeNode#removeChild(TreeNode)`. Until nodes by default are leaf nodes, however they can have actions added to them. When that happens they are transformed into branch nodes that allow child nodes. When the last child is removed they become leaf nodes again. This is all automatically handled when `TreeNode#addChild(ProgramNode)` and `TreeNode#removeChild(TreeNode)` are called.

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

11.137.2 Member Enumeration Documentation

11.137.2.1 BlendParameterType

enum [arcs::aubo_scope::UntilNode::BlendParameterType](#)

Blend Parameters can be of different types The parameter type is used to determine the explicit blend parameters (can be used in place of instanceof operations).

Enumerator

NO_BLEND	The robot arm will stop without performing a blend.
CUSTOM_BLEND	Custom blend has been selected. The blend instance can be cast to CustomBlendParameters.

Definition at line 111 of file [until_node.h](#).

11.137.2.2 CompareOperator

Generated by Doxygen
enum [arcs::aubo_scope::UntilNode::CompareOperator](#) : int

Compare operator types LESS_THAN and GREATER_THAN used when waiting for an analog input to go lower or higher, respectively, than a threshold.

Enumerator

LESS_THAN	
GREATER_THAN	

Definition at line 100 of file [until_node.h](#).

11.137.2.3 ConfigType

enum [arcs::aubo_scope::UntilNode::ConfigType](#) : short

The configuration type used to determine which type of configuration this instance is.

Enumerator

EXPRESSION	An Until node where the until condition is true once the evaluation of an expression is true. The config instance can be cast to ExpressionUntilNodeConfig.
TOOL_CONTACT	An Until node where the until condition is true once an impact on the Tool is detected. This functionality is not supported by the API.
IO_INPUT	An Until node where the until condition is true once a given I/O reaches a certain value. The config can be cast to IOInputUntilNodeConfig.
DISTANCE	An Until node where the until condition is true once a given distance is reached. The config can be cast to DistanceUntilNodeConfig.
REACHED_WAYPOINT	An Until node where the until condition is true once its parent Waypoint is reached. The config can be cast to ReachedWaypointUntilNodeConfig.
NONE_CONFIG_TYPE	An undefined Until node with no type set yet.

Definition at line 38 of file [until_node.h](#).

11.137.2.4 CustomDecelerationParameterType

enum [arcs::aubo_scope::UntilNode::CustomDecelerationParameterType](#)

Enumerator

Unknown	This deceleration was either created specifying values for both AngularAcceleration and Acceleration and/or is in use in an Until node that is not yet in the program tree.

Angular	This deceleration is currently used under a MoveJ node.
Cartesian	This deceleration is currently used under a MoveL or MoveP node.

Definition at line 129 of file [until_node.h](#).

11.137.2.5 DecelerationParameterType

enum [arcs::aubo_scope::UntilNode::DecelerationParameterType](#)

The deceleration parameter type used to determine which type of deceleration parameter this instance is.

Enumerator

SHARED_DECELERATION	Shared deceleration has been selected. The acceleration values from the parent Move node will be used as deceleration (absolute value).
CUSTOM_DECELERATION	Custom deceleration has been selected. This instance can be cast to CustomDecelerationParameters.

Definition at line 154 of file [until_node.h](#).

11.137.2.6 IOInputSelectionType

enum [arcs::aubo_scope::UntilNode::IOInputSelectionType](#)

The type of the input selection.

Enumerator

NONE_INPUT_SELECT	The node is not fully configured.
DIGITAL_INPUT	Digital input is selected. The config instance can be cast to DigitalInputUntilNodeConfig.
ANALOG_INPUT	Analog input is selected. The config instance can be cast to AnalogInputUntilNodeConfig.
FLOAT_REGISTER_INPUT	A float input register is selected. The config instance can be cast to FloatRegisterInputUntilNodeConfig.

Definition at line 175 of file [until_node.h](#).

11.137.3 Constructor & Destructor Documentation

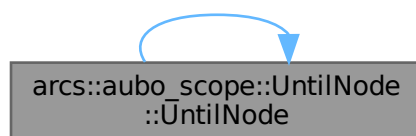
11.137.3.1 UntilNode() [1/3]

```
arcs::aubo_scope::UntilNode::UntilNode (
    UntilNode & f)
```

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

Referenced by [UntilNode\(\)](#), and [UntilNode\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



11.137.3.2 UntilNode() [2/3]

```
arcs::aubo_scope::UntilNode::UntilNode (
    UntilNode && f)
```

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

Here is the call graph for this function:



11.137.3.3 ~UntilNode()

virtual arcs::aubo_scope::UntilNode::~~UntilNode () [virtual]

11.137.3.4 UntilNode() [3/3]

arcs::aubo_scope::UntilNode::UntilNode () [private]

11.137.4 Member Function Documentation

11.137.4.1 getAnalogInputValueToProceed()

double arcs::aubo_scope::UntilNode::getAnalogInputValueToProceed ()

11.137.4.2 getAngularDeceleration()

double arcs::aubo_scope::UntilNode::getAngularDeceleration ()

The rate at which to decelerate when the until condition is met.

Returns

the specified angular deceleration of the until node (absolute value).

11.137.4.3 getBlendParameterType()

[BlendParameterType](#) arcs::aubo_scope::UntilNode::getBlendParameterType ()

11.137.4.4 getCartesianDeceleration()

double arcs::aubo_scope::UntilNode::getCartesianDeceleration ()

The rate at which to decelerate when the until condition is met.

Returns

the specified Cartesian deceleration of the until node (absolute value).

11.137.4.5 getCompareOperator()

[CompareOperator](#) arcs::aubo_scope::UntilNode::getCompareOperator ()

11.137.4.6 getConfigType()

[ConfigType](#) arcs::aubo_scope::UntilNode::getConfigType ()

This method returns the type of configuration.

Cast this instance appropriately to have access to specific getters.

Returns

the type of this config.

11.137.4.7 getCustomDecelerationParameterType()

[CustomDecelerationParameterType](#) arcs::aubo_scope::UntilNode::getCustomDecelerationParameterType ()

Returns

the deceleration type. This can change depending on the Move node type above the Until node.

11.137.4.8 getDecelerationParameters()

double arcs::aubo_scope::UntilNode::getDecelerationParameters ()

11.137.4.9 getDecelerationParameterType()

[DecelerationParameterType](#) arcs::aubo_scope::UntilNode::getDecelerationParameterType ()

11.137.4.10 getDigitalInputValueToProceed()

bool arcs::aubo_scope::UntilNode::getDigitalInputValueToProceed ()

11.137.4.11 getDistance()

double arcs::aubo_scope::UntilNode::getDistance ()

The distance the robot will travel.

Returns

The distance.

11.137.4.12 getExpressionConfig()

ExpressionPtr arcs::aubo_scope::UntilNode::getExpressionConfig ()

11.137.4.13 getIOInputSelectionType()

[IOInputSelectionType](#) arcs::aubo_scope::UntilNode::getIOInputSelectionType ()

11.137.4.14 getWaypointBlendParameter()

double arcs::aubo_scope::UntilNode::getWaypointBlendParameter ()

11.137.4.15 setAnalogInputValueToProceed()

void arcs::aubo_scope::UntilNode::setAnalogInputValueToProceed (
double value)

Creates an I/O Input configuration for an Until node.

The until condition for this configuration is true once the the signal value of a given analog input goes past a specified electric Current threshold.

Parameters

input	the analog input whose signal value will determine whether the until condition is met, not null.
compareOperator	the operator to use when comparing the signal value of the selected analog input to the threshold value, not null. Available options are "less than" (<) or "greater than" (>).
currentThreshold	the electric Current threshold for the signal value of the analog input. Execution will proceed until the signal value of the selected analog input goes past this threshold.
errorHandler	the error handler for validation errors. If using ErrorHandler#AUTO_CORRECT this will clamp the value to the nearest valid current value.

Returns

the Until node configuration.

Exceptions

IllegalArgumentException	if input is an output (does not support reading of values). See IO#isInput().

11.137.4.16 setAngularDeceleration()

```
void arcs::aubo_scope::UntilNode::setAngularDeceleration (
    double dec)
```

11.137.4.17 setBlendParameterType()

```
void arcs::aubo_scope::UntilNode::setBlendParameterType (
    BlendParameterType type)
```

11.137.4.18 setCartesianDeceleration()

```
void arcs::aubo_scope::UntilNode::setCartesianDeceleration (
    double dec)
```

11.137.4.19 setCompareOperator()

```
void arcs::aubo_scope::UntilNode::setCompareOperator (
    CompareOperator compare_operator)
```

11.137.4.20 setConfigType()

```
void arcs::aubo_scope::UntilNode::setConfigType (
    ConfigType type)
```

11.137.4.21 setCustomDecelerationParameterType()

```
void arcs::aubo_scope::UntilNode::setCustomDecelerationParameterType (
    CustomDecelerationParameterType type)
```

11.137.4.22 setDecelerationParameters()

```
void arcs::aubo_scope::UntilNode::setDecelerationParameters (
    double deceleration)
```

Creates deceleration parameter for an Until node. Can be used in an [Expression](#) configuration type.

An angular deceleration is used when the type of the parent Move node is a MoveJ, whereas a Cartesian deceleration is used when the move type is a MoveL or MoveP.

Parameters

deceleration	the angular deceleration (the value for the Cartesian deceleration will be the default value that AboScope uses), not null.

decelerationErrorHandler	error handler for handling validation. If using ErrorHandler#AUTO_CORRECT this will clamp the value to the nearest valid angular deceleration value.

Returns

the deceleration parameter for the Until node.

11.137.4.23 setDecelerationParameterType()

```
void arcs::aubo_scope::UntilNode::setDecelerationParameterType (
    DecelerationParameterType type)
```

11.137.4.24 setDigitalInputValueToProceed()

```
void arcs::aubo_scope::UntilNode::setDigitalInputValueToProceed (
    bool value_to_proceed_until)
```

Creates an I/O Input configuration for an Until node.

The until condition for this configuration is true once the signal value of a given digital input goes HIGH or LOW.

Parameters

input	the digital input whose signal value will determine whether the until condition is met, not null.
valueToProceedUntil	true, if the until condition is met when the input signal value goes HIGH, false otherwise (condition met when signal value is LOW).

Returns

the Until node configuration.

Exceptions

IllegalArgumentException	if input is an output (does not support reading of values). See IO#isInput().

11.137.4.25 setDistance()

```
void arcs::aubo_scope::UntilNode::setDistance (  
    double dis)
```

11.137.4.26 setExpressionConfig()

```
void arcs::aubo_scope::UntilNode::setExpressionConfig (  
    ExpressionPtr expression)
```

Creates an [Expression](#) configuration for an Until node with an expression and default value for the deceleration parameter. The until condition for this configuration is true once the evaluation of the expression is true.

Note that this default value may change between AuboScope versions.

Parameters

expression	the expression whose evaluation will determine whether the until condition is met.

Returns

the Until node configuration.

11.137.4.27 setIOInputSelectionType()

```
void arcs::aubo_scope::UntilNode::setIOInputSelectionType (  
    IOInputSelectionType type)
```

11.137.4.28 setWaypointBlendParameter()

```
void arcs::aubo_scope::UntilNode::setWaypointBlendParameter (  
    double suggested_blend)
```

Creates blend parameters for an Until node.

Can be used in a Distance configuration type.

Parameters

suggestedBlend	This blend value may be (automatically) corrected depending on the value of the distance parameter the Distance configuration.

Returns

the blend parameters for the Until node.

11.137.5 Friends And Related Symbol Documentation

11.137.5.1 DataSwitch

friend class DataSwitch [friend]

Definition at line 367 of file [until_node.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.137.6 Member Data Documentation

11.137.6.1 d_

void* arcs::aubo_scope::UntilNode::d_ { nullptr } [private]

Definition at line 369 of file [until_node.h](#).

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

- include/aubo_caps/domain/program/nodes/builtin/[until_node.h](#)

11.138 arcs::aubo_scope::UserInteraction Class Reference

[UserInteraction](#) This interface provides functionality for requesting input or actions from end users.

```
#include <user_interaction.h>
```

Public Member Functions

- [UserInteraction](#) (UserInteraction &f)
- [UserInteraction](#) (UserInteraction &&f)
- virtual [~UserInteraction](#) ()
- void [getUserDefinedRobotPosition](#) ([RobotPositionCallback](#) callback)
- void [getUserDefinedRobotPosition](#) (const std::vector< double > &tcpOffset, [RobotPositionCallback](#) callback)
- Request the end user to define a robot position with specified tcpOffset.
- void [getUserDefinedRobotPosition](#) (const std::string &tcpOffset, [RobotPositionCallback](#) callback)
- Request the end user to define a robot position with specified tcpOffset string.
- void [getUserDefinedRobotPosition](#) (const TCPPtr tcpOffset, [RobotPositionCallback](#) callback)
- Request the end user to define a robot position with specified TCP object.
- KeyboardManagerPtr [getKeyboardManager](#) ()
- int [requestUserToMoveJoint](#) (const std::vector< double > &q, [RobotMovementCallback](#) callback)
- int [requestUserToMoveJoint](#) (const std::string &q, [RobotMovementCallback](#) callback)
- [requestUserToMoveJoint](#)
- int [requestUserToMoveJoint](#) (const std::vector< double > &pose, const std::vector< double > &q, [RobotMovementCallback](#) callback)
- QT_DEPRECATED int [requestUserToMoveLine](#) (const std::vector< double > &pose, [RobotMovementCallback](#) callback)
- int [requestUserToMoveLine](#) (const std::vector< double > &pose, const std::vector< double > &tcpOffset, [RobotMovementCallback](#) callback)
- QPixmap * [getRobotThumb](#) (const std::vector< double > &q, const std::vector< double > &pose)
- Get robot thumbnail.
- void [setOperationLogDisplayName](#) (QWidget *pbn, const std::string &name)
- RenderInterfacePtr [getRenderInterface](#) ()
- PopupManagerPtr [getPopupManager](#) ()

Private Member Functions

- [UserInteraction](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

11.138.1 Detailed Description

[UserInteraction](#) This interface provides functionality for requesting input or actions from end users.

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

11.138.2 Constructor & Destructor Documentation

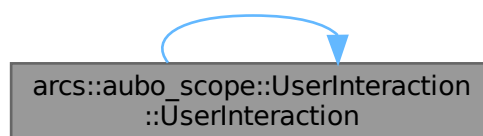
11.138.2.1 UserInteraction() [1/3]

```
arcs::aubo_scope::UserInteraction::UserInteraction (
    UserInteraction & f)
```

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

Referenced by [UserInteraction\(\)](#), and [UserInteraction\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

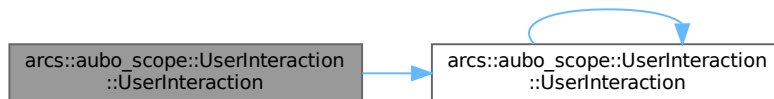


11.138.2.2 UserInteraction() [2/3]

```
arcs::aubo_scope::UserInteraction::UserInteraction (
    UserInteraction && f)
```

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

Here is the call graph for this function:



11.138.2.3 ~UserInteraction()

```
virtual arcs::aubo_scope::UserInteraction::~~UserInteraction () [virtual]
```

11.138.2.4 UserInteraction() [3/3]

```
arcs::aubo_scope::UserInteraction::UserInteraction () [private]
```

11.138.3 Member Function Documentation

11.138.3.1 getKeyboardManager()

```
KeyboardManagerPtr arcs::aubo_scope::UserInteraction::getKeyboardManager ()
```

This method provides a factory for creating keyboard inputs which are used to configure a virtual keyboard/keypad and to request it to be displayed for a Qt GUI component.

NOTE: This functionality is only relevant for AuboCap nodes with a Qt-based user interface (see [ProgramNodeService](#), [InstallationNodeService](#) and [ToolbarService](#)).

Returns

factory providing access to keyboard inputs.

11.138.3.2 getPopupManager()

```
PopupManagerPtr arcs::aubo_scope::UserInteraction::getPopupManager ()
```

Returns

This method provides popup manager.

11.138.3.3 getRenderInterface()

RenderInterfacePtr arcs::aubo_scope::UserInteraction::getRenderInterface ()

Returns

This method provides robot model drawing objects.

11.138.3.4 getRobotThumb()

QPixmap * arcs::aubo_scope::UserInteraction::getRobotThumb (
 const std::vector< double > & q,
 const std::vector< double > & pose)

Get robot thumbnail.

Parameters

q	joint angles
pose	pose

Returns

robot thumbnail

11.138.3.5 getUserDefinedRobotPosition() [1/4]

void arcs::aubo_scope::UserInteraction::getUserDefinedRobotPosition (
 const std::string & tcpOffset,
 [RobotPositionCallback](#) callback)

Request the end user to define a robot position with specified tcpOffset string.

Parameters

tcpOffset	
callback	

11.138.3.6 getUserDefinedRobotPosition() [2/4]

void arcs::aubo_scope::UserInteraction::getUserDefinedRobotPosition (
 const std::vector< double > & tcpOffset,
 [RobotPositionCallback](#) callback)

Request the end user to define a robot position with specified tcpOffset.

Parameters

tcpOffset	
callback	

11.138.3.7 getUserDefinedRobotPosition() [3/4]

```
void arcs::aubo_scope::UserInteraction::getUserDefinedRobotPosition (
    const TCPPtr tcpOffset,
    RobotPositionCallback callback)
```

Request the end user to define a robot position with specified [TCP](#) object.

Parameters

tcpOffset	
callback	

11.138.3.8 getUserDefinedRobotPosition() [4/4]

```
void arcs::aubo_scope::UserInteraction::getUserDefinedRobotPosition (
    RobotPositionCallback callback)
```

Request the end user to use the robot to define a robot position. Override the [RobotPositionCallback2#onOk\(PositionParameters\)](#) method to execute code once the end user is done. Optionally override the [RobotPositionCallback2#onCancel\(\)](#) method to execute code if the end user cancels the operation.

This method is asynchronous, i.e. the method will return immediately. Only when the end user is done, either [RobotPositionCallback2#onOk\(PositionParameters\)](#) or [RobotPositionCallback2#onCancel\(\)](#) will be called.

NOTE: This functionality should not be used in a toolbar contribution (see [ToolbarService](#)).

Parameters

callback	the instance callbacks will be made to.

11.138.3.9 requestUserToMoveJoint() [1/3]

```
int arcs::aubo_scope::UserInteraction::requestUserToMoveJoint (
    const std::string & q,
    RobotMovementCallback callback)
```

requestUserToMoveJoint

Parameters

q	
callback	

Returns

11.138.3.10 requestUserToMoveJoint() [2/3]

```
int arcs::aubo_scope::UserInteraction::requestUserToMoveJoint (
    const std::vector< double > & pose,
    const std::vector< double > & q,
    RobotMovementCallback callback)
```

Parameters

pose	The target pose the robot TCP will move to.
q	The reference joint Angle.
callback	The instance which callbacks will be made to.

Exceptions

IllegalStateException	If the plugin screen is not active (visible).

11.138.3.11 requestUserToMoveJoint() [3/3]

```
int arcs::aubo_scope::UserInteraction::requestUserToMoveJoint (
    const std::vector< double > & q,
    RobotMovementCallback callback)
```

Transition to the move tab widget to allow/request the end user to move the robot (either automatically or manually) to a desired target position specified by joint positions.

The Automove function will move the robot to the target joint positions linearly in joint space.

NOTE: This method does not take the currently active [TCP](#) offset into account, i.e. the Automove function will have the specified joint positions as target.

Parameters

q	The target joint positions to move to.
callback	The instance which callbacks will be made to.

Exceptions

IllegalStateException	If the plugin screen is not active (visible).

11.138.3.12 requestUserToMoveLine() [1/2]

```
int arcs::aubo_scope::UserInteraction::requestUserToMoveLine (
    const std::vector< double > & pose,
    const std::vector< double > & tcpOffset,
    RobotMovementCallback callback)
```

Transition to the move tab widget to allow/request the end user to move the robot (either automatically or manually) to a desired target position specified by a pose.

The Automove function will move the robot [TCP](#) to the target position linearly in Cartesian space. If this it not possible, the robot will move to the target linearly in joint space.

The current joint positions of the robot will be used as starting point for inverse kinematics to calculate a target joint vector corresponding to desired pose, taking the currently active [TCP](#) offset into account.

Parameters

pose	The target pose the robot TCP will move to.
tcpOffset	
callback	The instance which callbacks will be made to.

Exceptions

IllegalStateException	If the plugin screen is not active (visible).

11.138.3.13 requestUserToMoveLine() [2/2]

```
QT_DEPRECATED int arcs::aubo_scope::UserInteraction::requestUserToMoveLine (
    const std::vector< double > & pose,
    RobotMovementCallback callback)
```

Transition to the move tab widget to allow/request the end user to move the robot (either automatically or manually) to a desired target position specified by a pose.

The Automove function will move the robot **TCP** to the target position linearly in Cartesian space. If this it not possible, the robot will move to the target linearly in joint space.

The current joint positions of the robot will be used as starting point for inverse kinematics to calculate a target joint vector corresponding to desired pose, taking the currently active **TCP** offset into account.

Parameters

pose	The target pose the robot TCP will move to.
callback	The instance which callbacks will be made to.

Exceptions

IllegalStateException	If the plugin screen is not active (visible).

11.138.3.14 setOperationLogDisplayName()

```
void arcs::aubo_scope::UserInteraction::setOperationLogDisplayName (
    QWidget * pbn,
    const std::string & name)
```

setOperationLogDisplayName

Parameters

pbn	
name	

11.138.4 Friends And Related Symbol Documentation

11.138.4.1 DataSwitch

friend class DataSwitch [friend]

Definition at line 283 of file [user_interaction.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.138.5 Member Data Documentation

11.138.5.1 d_

void* arcs::aubo_scope::UserInteraction::d_ { nullptr } [private]

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

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

- include/aubo_caps/domain/user_interface/[user_interaction.h](#)

11.139 arcs::aubo_scope::UserInterfaceApi Class Reference

```
#include <user_interface_api.h>
```

Public Member Functions

- [UserInterfaceApi](#) (UserInterfaceApi &f)
- [UserInterfaceApi](#) (UserInterfaceApi &&f)
- virtual [~UserInterfaceApi](#) ()
- UserInteractionPtr [getUserInteraction](#) ()
[UserInteraction](#)

Private Member Functions

- [UserInterfaceApi](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

11.139.1 Detailed Description

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

11.139.2 Constructor & Destructor Documentation

11.139.2.1 `UiThreadApi()` [1/3]

```
arcs::aubo_scope::UiThreadApi::UiThreadApi (
    UiInterfaceApi & f)
```

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

Referenced by [UiThreadApi\(\)](#), and [UiThreadApi\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

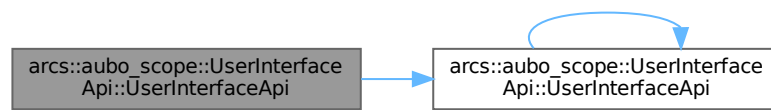


11.139.2.2 `UiThreadApi()` [2/3]

```
arcs::aubo_scope::UiThreadApi::UiThreadApi (
    UiInterfaceApi && f)
```

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

Here is the call graph for this function:



11.139.2.3 `~UserInterfaceApi()`

virtual arcs::aubo_scope::UserInterfaceApi::~UserInterfaceApi () [virtual]

11.139.2.4 `UserInterfaceApi()` [3/3]

arcs::aubo_scope::UserInterfaceApi::UserInterfaceApi () [private]

11.139.3 Friends And Related Symbol Documentation

11.139.3.1 `DataSwitch`

friend class DataSwitch [friend]

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

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.139.4 Member Data Documentation

11.139.4.1 `d_`

void* arcs::aubo_scope::UserInterfaceApi::d_ { nullptr } [private]

Definition at line 48 of file [user_interface_api.h](#).

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

- [include/aubo_caps/domain/user_interface/user_interface_api.h](#)

11.140 arcs::aubo_scope::ValueFactory Class Reference

[ValueFactory](#)

```
#include <value_factory_provider.h>
```

Public Member Functions

- [ValueFactory](#) (ValueFactory &f)
- [ValueFactory](#) (ValueFactory &&f)
- virtual [~ValueFactory](#) ()
- ExpressionPtr [createExpression](#) ()

This method instantiates a new ExpressionBuilder to create a single valid expression.
- ExpressionPtr [createExpression](#) (const std::string &str)

Private Member Functions

- [ValueFactory](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

11.140.1 Detailed Description

[ValueFactory](#)

Summary

This interface provides a mean to create immutable value objects using factories. The SimpleValueFactory helps creating simple value objects. The ExpressionBuilder helps building valid expressions. The BlendFactory helps creating blend objects. The JointPositionFactory helps creating joint positions objects. The PoseFactory helps creating Pose objects.

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

11.140.2 Constructor & Destructor Documentation

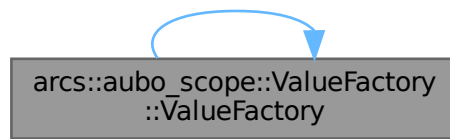
11.140.2.1 [ValueFactory](#)() [1/3]

```
arcs::aubo_scope::ValueFactory::ValueFactory (
    ValueFactory & f)
```

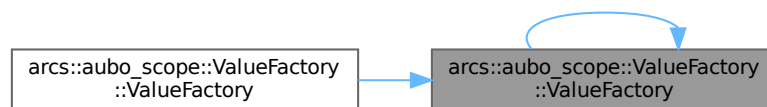
References [ValueFactory](#)().

Referenced by [ValueFactory\(\)](#), and [ValueFactory\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

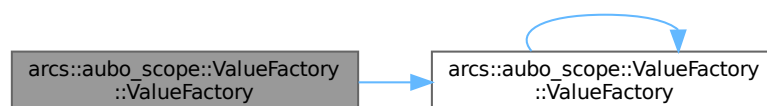


11.140.2.2 ValueFactory() [2/3]

```
arcs::aubo_scope::ValueFactory::ValueFactory (
    ValueFactory && f)
```

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

Here is the call graph for this function:



11.140.2.3 ~ValueFactory()

```
virtual arcs::aubo_scope::ValueFactory::~ValueFactory () [virtual]
```

11.140.2.4 ValueFactory() [3/3]

`arcs::aubo_scope::ValueFactory::ValueFactory () [private]`

11.140.3 Member Function Documentation

11.140.3.1 createExpression() [1/2]

`ExpressionPtr arcs::aubo_scope::ValueFactory::createExpression ()`

This method instantiates a new `ExpressionBuilder` to create a single valid expression.

Returns

the builder to build a single expression.

11.140.3.2 createExpression() [2/2]

`ExpressionPtr arcs::aubo_scope::ValueFactory::createExpression (
const std::string & str)`

11.140.4 Friends And Related Symbol Documentation

11.140.4.1 DataSwitch

`friend class DataSwitch [friend]`

Definition at line 55 of file [value_factory_provider.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.140.5 Member Data Documentation

11.140.5.1 d_

`void* arcs::aubo_scope::ValueFactory::d_ { nullptr } [private]`

Definition at line 57 of file [value_factory_provider.h](#).

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

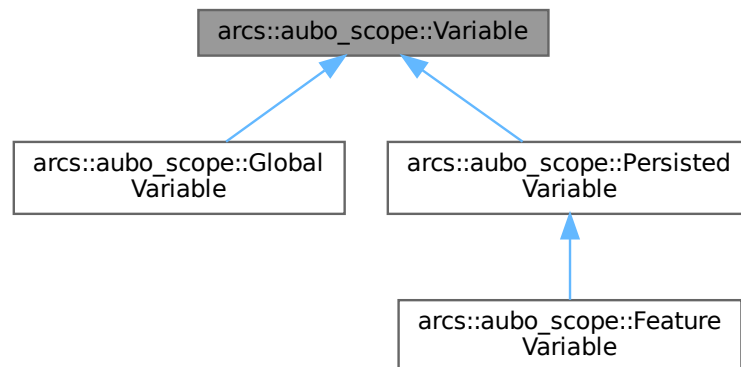
- [include/aubo_caps/domain/value/value_factory_provider.h](#)

11.141 arcs::aubo_scope::Variable Class Reference

Variable Represents a variable in AuboScope.

```
#include <variable.h>
```

Inheritance diagram for arcs::aubo_scope::Variable:



Public Types

- enum **Type** : int { **GLOBAL** , **VALUE_PERSISTED** , **FEATURE** }
The variable type used to determine which type of variable this instance is.

Public Member Functions

- Variable** (Variable &f)
- Variable** (Variable &&f)
- virtual **~Variable** ()
- Type** **getType** ()
Returns the type of variable.
- std::string **getDisplayname** ()
Note: The variable can be renamed at any time.
- bool **equals** (VariablePtr object)
Checks if the object references the same internal **Variable**.
- size_t **hashCode** ()
Returns the hash code for this object.
- std::string **toString** ()
Returns a string representation of the variable.

Protected Member Functions

- Variable** ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

11.141.1 Detailed Description

[Variable](#) Represents a variable in AuboScope.

Variables can be used to store and pass data in programs.

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

11.141.2 Member Enumeration Documentation

11.141.2.1 Type

enum [arcs::aubo_scope::Variable::Type](#) : int

The variable type used to determine which type of variable this instance is.

Enumerator

GLOBAL	GLOBAL means that the variable is available everywhere in a program. This variable instance can be cast to GlobalVariable .
VALUE_PERSISTED	VALUE_PERSISTED means that the variable is stored with the installation. These variables are defined in the installation and will retain their values even when a robot is turned off and on. This variable instance can be cast to ValuePersistedVariable .
FEATURE	FEATURE means that a Feature is marked as a variable. The variable is defined in the installation and will exist even when a robot is turned off and on. This variable instance can be cast to FeatureVariable .

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

11.141.3 Constructor & Destructor Documentation

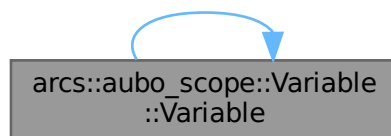
11.141.3.1 Variable() [1/3]

arcs::aubo_scope::Variable::Variable (
Variable & f)

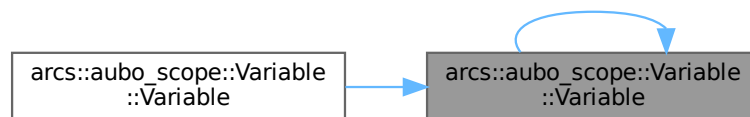
References [Variable\(\)](#).

Referenced by [Variable\(\)](#), and [Variable\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

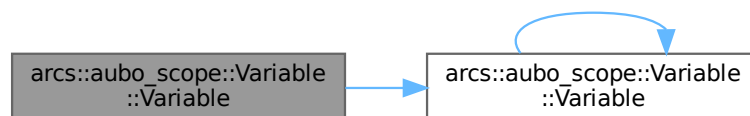


11.141.3.2 Variable() [2/3]

arcs::aubo_scope::Variable::Variable (
Variable && f)

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

Here is the call graph for this function:



11.141.3.3 ~Variable()

virtual arcs::aubo_scope::Variable::~~Variable () [virtual]

11.141.3.4 Variable() [3/3]

arcs::aubo_scope::Variable::Variable () [protected]

11.141.4 Member Function Documentation

11.141.4.1 equals()

bool arcs::aubo_scope::Variable::equals (
VariablePtr object)

Checks if the object references the same internal [Variable](#).

Returns

true if object references the same internal [Variable](#).

11.141.4.2 getDisplayName()

std::string arcs::aubo_scope::Variable::getDisplayName ()

Note: The variable can be renamed at any time.

Returns

the display name of the variable.

11.141.4.3 getType()

[Type](#) arcs::aubo_scope::Variable::getType ()

Returns the type of variable.

Returns

the type of variable.

11.141.4.4 hashCode()

size_t arcs::aubo_scope::Variable::hashCode ()

Returns the hash code for this object.

Returns

hashCode for this object.

11.141.4.5 toString()

```
std::string arcs::aubo_scope::Variable::toString ()
```

Returns a string representation of the variable.

Returns

a string representation of the variable.

11.141.5 Friends And Related Symbol Documentation

11.141.5.1 DataSwitch

```
friend class DataSwitch [friend]
```

Definition at line 139 of file [variable.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.141.6 Member Data Documentation

11.141.6.1 d__

```
void* arcs::aubo_scope::Variable::d__ { nullptr } [private]
```

Definition at line 140 of file [variable.h](#).

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

- include/aubo_caps/domain/variable/[variable.h](#)

11.142 arcs::aubo_scope::VariableFactory Class Reference

```
#include <variable_factory.h>
```

Public Member Functions

- [VariableFactory](#) (VariableFactory &f)
- [VariableFactory](#) (VariableFactory &&f)
- [~VariableFactory](#) ()
- GlobalVariablePtr [createGlobalVariable](#) (const std::string &suggested_name)
- GlobalVariablePtr [createGlobalVariable](#) (const std::string &suggested_name, ExpressionPtr initialValue)
- FeatureVariablePtr [createFeatureVariable](#) (const std::string &suggested_name, const std::vector<double> &pose)

Private Member Functions

- [VariableFactory](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

11.142.1 Detailed Description

This interface supplies methods to create variable objects.

NOTE: Only use this functionality in a AuboCap program node (not relevant for an installation node).

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

11.142.2 Constructor & Destructor Documentation

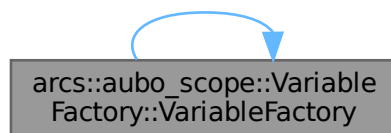
11.142.2.1 VariableFactory() [1/3]

```
arcs::aubo_scope::VariableFactory::VariableFactory (
    VariableFactory & f)
```

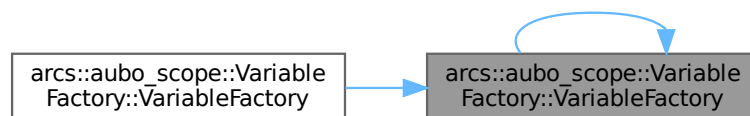
References [VariableFactory\(\)](#).

Referenced by [VariableFactory\(\)](#), and [VariableFactory\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

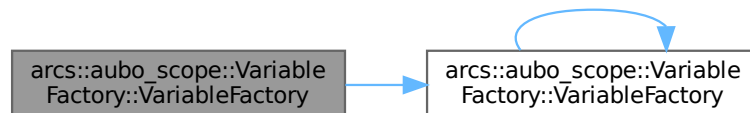


11.142.2.2 VariableFactory() [2/3]

```
arcs::aubo_scope::VariableFactory::VariableFactory (
    VariableFactory && f)
```

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

Here is the call graph for this function:



11.142.2.3 ~VariableFactory()

```
arcs::aubo_scope::VariableFactory::~~VariableFactory ()
```

11.142.2.4 VariableFactory() [3/3]

```
arcs::aubo_scope::VariableFactory::VariableFactory () [private]
```

11.142.3 Member Function Documentation

11.142.3.1 createFeatureVariable()

```
FeatureVariablePtr arcs::aubo_scope::VariableFactory::createFeatureVariable (
    const std::string & suggested_name,
    const std::vector< double > & pose)
```

Creates a feature variable.

Parameters

suggested_name	Suggested name of the variable.
pose	the pose of the feature.

Returns

[FeatureVariable](#).

Exceptions

VariableException	In case of an illegal name.

11.142.3.2 createGlobalVariable() [1/2]

```
GlobalVariablePtr arcs::aubo_scope::VariableFactory::createGlobalVariable (
    const std::string & suggested_name)
```

Creates a global variable. Registering the variable in AuboScope (by storing it in a [DataModel](#) instance or using it for the configuration of a built-in AuboScope program node) will make it accessible to all program nodes.

Parameters

suggested_name	Suggested name of the variable.

Returns

[GlobalVariable](#).

Exceptions

VariableException	In case of an illegal name.

11.142.3.3 createGlobalVariable() [2/2]

```
GlobalVariablePtr arcs::aubo_scope::VariableFactory::createGlobalVariable (
    const std::string & suggested_name,
    ExpressionPtr initialValue)
```

Creates a global variable with an initial value.

Parameters

suggested_name	Suggested name of the variable.
initialValue	The expression that will be evaluated in order to initialize the value of the variable.

Returns

[GlobalVariable](#)

Exceptions

VariableException	In case of an illegal name.

11.142.4 Friends And Related Symbol Documentation

11.142.4.1 DataSwitch

friend class DataSwitch [friend]

Definition at line 98 of file [variable_factory.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.142.5 Member Data Documentation

11.142.5.1 d_

void* arcs::aubo_scope::VariableFactory::d_ { nullptr } [private]

Definition at line 100 of file [variable_factory.h](#).

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

- include/aubo_caps/domain/variable/[variable_factory.h](#)

11.143 arcs::aubo_scope::VariableModel Class Reference

[VariableModel](#) Provides methods that returns the variables that currently are registered in the system.

#include <variable_model.h>

Public Member Functions

- [VariableModel](#) (VariableModel &f)
- [VariableModel](#) (VariableModel &&f)
- [~VariableModel](#) ()
- std::vector< VariablePtr > [getAll](#) ()
Returns the collection of all variables that currently are registered in the system.
- VariablePtr [get](#) (const std::string &name)
- std::vector< VariablePtr > [get](#) (std::function< bool(VariablePtr)> filter=[]) (VariablePtr) { return true;})
Get a subset of all the variables registered in the system using a filter.
- VariableFactoryPtr [getVariableFactory](#) ()
Returns a [VariableFactory](#) to create variable objects.
- FeatureVariablePtr [getFeatureVariable](#) (FeaturePtr feature)
Get the feature variable for a [Feature](#).
- std::string [connectedTo](#) (const std::string &receiver, const std::function< void(VariableModel *)> &slot)
[Register](#) callback for changed.
- void [disconnectFrom](#) (const std::string &receiver)
Disconnect a callback.

Private Member Functions

- [VariableModel](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

11.143.1 Detailed Description

[VariableModel](#) Provides methods that returns the variables that currently are registered in the system.

A variable is registered if it has been stored in a [DataModel](#) instance or used for the configuration of a built-in AboScope program node.

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

11.143.2 Constructor & Destructor Documentation

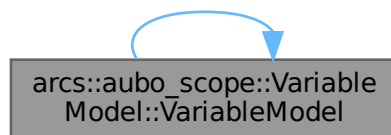
11.143.2.1 VariableModel() [1/3]

```
arcs::aubo_scope::VariableModel::VariableModel (
    VariableModel & f)
```

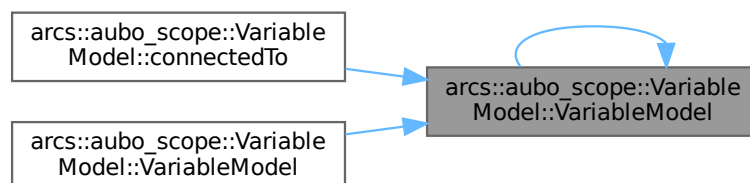
References [VariableModel\(\)](#).

Referenced by [connectedTo\(\)](#), [VariableModel\(\)](#), and [VariableModel\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

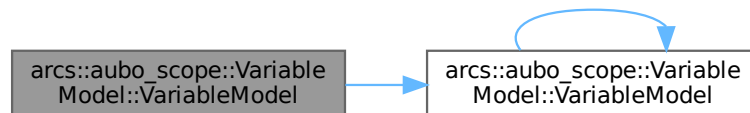


11.143.2.2 VariableModel() [2/3]

```
arcs::aubo_scope::VariableModel::VariableModel (
    VariableModel && f)
```

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

Here is the call graph for this function:



11.143.2.3 ~VariableModel()

```
arcs::aubo_scope::VariableModel::~~VariableModel ()
```

11.143.2.4 VariableModel() [3/3]

```
arcs::aubo_scope::VariableModel::VariableModel () [private]
```

11.143.3 Member Function Documentation

11.143.3.1 connectedTo()

```
std::string arcs::aubo_scope::VariableModel::connectedTo (
    const std::string & receiver,
    const std::function< void(VariableModel *)> & slot)
```

[Register](#) callback for changed.

Parameters

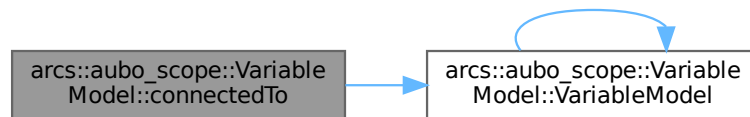
receiver	the receiver identifier.
slot	the callback function.

Returns

the connection identifier.

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

Here is the call graph for this function:



11.143.3.2 disconnectFrom()

```
void arcs::aubo_scope::VariableModel::disconnectFrom (
    const std::string & receiver)
```

Disconnect a callback.

Parameters

receiver	the receiver identifier.

11.143.3.3 get() [1/2]

```
VariablePtr arcs::aubo_scope::VariableModel::get (
    const std::string & name)
```

11.143.3.4 get() [2/2]

```
std::vector< VariablePtr > arcs::aubo_scope::VariableModel::get (
    std::function< bool(VariablePtr)> filter = [](VariablePtr) { return true;})
```

Get a subset of all the variables registered in the system using a filter.

Parameters

filter	see VariableFilterFactory for examples.

Returns

the collection of Variables that are accepted by the filter.

11.143.3.5 getAll()

```
std::vector< VariablePtr > arcs::aubo_scope::VariableModel::getAll ()
```

Returns the collection of all variables that currently are registered in the system.

Returns

the collection of all registered variables.

11.143.3.6 getFeatureVariable()

```
FeatureVariablePtr arcs::aubo_scope::VariableModel::getFeatureVariable (  
    FeaturePtr feature)
```

Get the feature variable for a [Feature](#).

Parameters

feature	the feature with a variable.

Returns

the variable for the feature.

Exceptions

IllegalArgumentException	If feature not is present in the installation or feature has no variable, see Feature#isVariable() .

11.143.3.7 getVariableFactory()

VariableFactoryPtr arcs::aubo_scope::VariableModel::getVariableFactory ()

Returns a [VariableFactory](#) to create variable objects.

Returns

a [VariableFactory](#) to create variable objects.

11.143.4 Friends And Related Symbol Documentation

11.143.4.1 DataSwitch

friend class DataSwitch [friend]

Definition at line 133 of file [variable_model.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.143.5 Member Data Documentation

11.143.5.1 d_

void* arcs::aubo_scope::VariableModel::d_ { nullptr } [private]

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

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

- include/aubo_caps/domain/variable/[variable_model.h](#)

11.144 arcs::aubo_scope::ViewApiProvider Class Reference

[ViewApiProvider](#) Provides access to functionality relevant for the View (UI) of an installation or a program node contribution.

```
#include <view_api_provider.h>
```

Public Member Functions

- [ViewApiProvider](#) ()
- [ViewApiProvider](#) (ViewApiProvider &f)
- [ViewApiProvider](#) (ViewApiProvider &&f)
- virtual [~ViewApiProvider](#) ()
- SystemApiPtr [getSystemApi](#) ()

Provides access to system related functionality.
- UserInterfaceApiPtr [getUserInterfaceApi](#) ()

Provides access to functionality related to user interface and user interaction.

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

11.144.1 Detailed Description

[ViewApiProvider](#) Provides access to functionality relevant for the View (UI) of an installation or a program node contribution.

NOTE: This interface is not relevant for AuboCap program and installation nodes with a HTML-based user interface.

Definition at line 27 of file [view_api_provider.h](#).

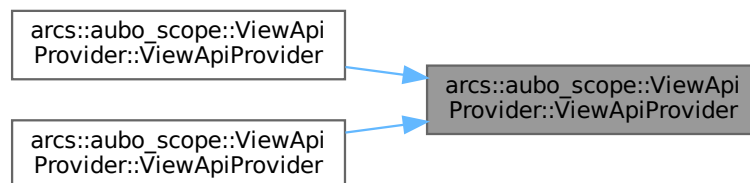
11.144.2 Constructor & Destructor Documentation

11.144.2.1 ViewApiProvider() [1/3]

arcs::aubo_scope::ViewApiProvider::ViewApiProvider ()

Referenced by [ViewApiProvider\(\)](#), and [ViewApiProvider\(\)](#).

Here is the caller graph for this function:



11.144.2.2 ViewApiProvider() [2/3]

arcs::aubo_scope::ViewApiProvider::ViewApiProvider (
ViewApiProvider & f)

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

Here is the call graph for this function:



11.144.2.3 ViewApiProvider() [3/3]

```
arcs::aubo_scope::ViewApiProvider::ViewApiProvider (
    ViewApiProvider && f)
```

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

Here is the call graph for this function:



11.144.2.4 ~ViewApiProvider()

```
virtual arcs::aubo_scope::ViewApiProvider::~~ViewApiProvider () [virtual]
```

11.144.3 Member Function Documentation

11.144.3.1 getSystemApi()

```
SystemApiPtr arcs::aubo_scope::ViewApiProvider::getSystemApi ()
```

Provides access to system related functionality.

Returns

an instance of SystemAPI.

11.144.3.2 getUserInterfaceApi()

```
UserInterfaceApiPtr arcs::aubo_scope::ViewApiProvider::getUserInterfaceApi ()
```

Provides access to functionality related to user interface and user interaction.

Returns

an instance of UserInterfaceAPI.

11.144.4 Friends And Related Symbol Documentation

11.144.4.1 DataSwitch

friend class DataSwitch [friend]

Definition at line 65 of file [view_api_provider.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.144.5 Member Data Documentation

11.144.5.1 d_

void* arcs::aubo_scope::ViewApiProvider::d_ { nullptr } [private]

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

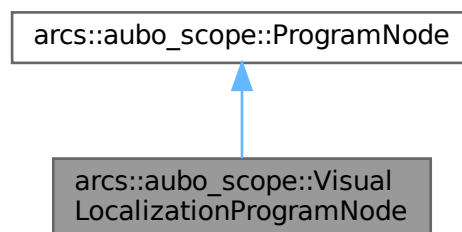
The documentation for this class was generated from the following file:

- [include/aubo_caps/contribution/view_api_provider.h](#)

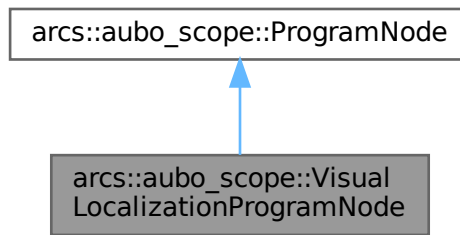
11.145 arcs::aubo_scope::VisualLocalizationProgramNode Class Reference

```
#include <visual_localization_program_node.h>
```

Inheritance diagram for arcs::aubo_scope::VisualLocalizationProgramNode:



Collaboration diagram for `arcs::aubo_scope::VisualLocalizationProgramNode`:



Public Member Functions

- [VisualLocalizationProgramNode](#) ([VisualLocalizationProgramNode](#) &f)
- [VisualLocalizationProgramNode](#) ([VisualLocalizationProgramNode](#) &&f)
- virtual [~VisualLocalizationProgramNode](#) ()

Public Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ([ProgramNode](#) &f)
- [ProgramNode](#) ([ProgramNode](#) &&f)
- virtual [~ProgramNode](#) ()
- [ProgramNodeContributionPtr](#) [getProgramNodeContribution](#) ()
Get the program node contribution.

Private Member Functions

- [VisualLocalizationProgramNode](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Protected Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.145.1 Detailed Description

Definition at line 10 of file [visual_localization_program_node.h](#).

11.145.2 Constructor & Destructor Documentation

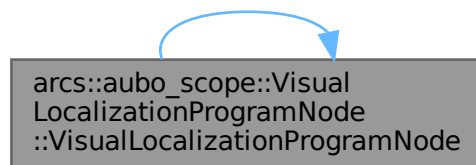
11.145.2.1 VisualLocalizationProgramNode() [1/3]

```
arcs::aubo_scope::VisualLocalizationProgramNode::VisualLocalizationProgramNode (
    VisualLocalizationProgramNode & f)
```

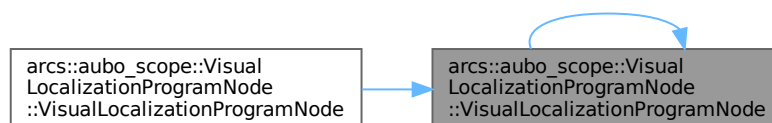
References [VisualLocalizationProgramNode\(\)](#).

Referenced by [VisualLocalizationProgramNode\(\)](#), and [VisualLocalizationProgramNode\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

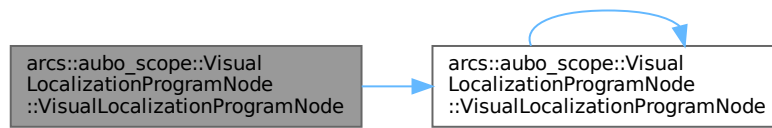


11.145.2.2 VisualLocalizationProgramNode() [2/3]

```
arcs::aubo_scope::VisualLocalizationProgramNode::VisualLocalizationProgramNode (
    VisualLocalizationProgramNode && f)
```

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

Here is the call graph for this function:



11.145.2.3 ~VisualLocalizationProgramNode()

virtual arcs::aubo_scope::VisualLocalizationProgramNode::~~VisualLocalizationProgramNode () [virtual]

11.145.2.4 VisualLocalizationProgramNode() [3/3]

arcs::aubo_scope::VisualLocalizationProgramNode::VisualLocalizationProgramNode () [private]

11.145.3 Friends And Related Symbol Documentation

11.145.3.1 DataSwitch

friend class DataSwitch [friend]

Definition at line 18 of file [visual_localization_program_node.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.145.4 Member Data Documentation

11.145.4.1 d_

void* arcs::aubo_scope::VisualLocalizationProgramNode::d_ { nullptr } [private]

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

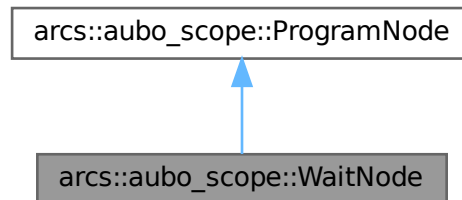
The documentation for this class was generated from the following file:

- [include/aubo_caps/domain/program/nodes/builtin/visual_localization_program_node.h](#)

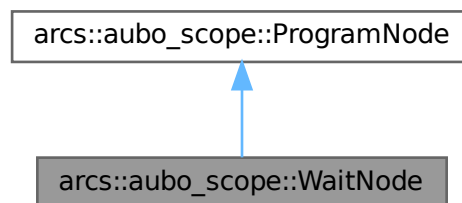
11.146 arcs::aubo_scope::WaitNode Class Reference

#include <wait_node.h>

Inheritance diagram for arcs::aubo_scope::WaitNode:



Collaboration diagram for arcs::aubo_scope::WaitNode:



Public Types

- enum `ConfigType` : int {
`NO_TYPE` , `NO_WAIT` , `TIME` , `DIGITAL_INPUT` ,
`ANALOG_INPUT_CURRENT` , `ANALOG_INPUT_VOLTAGE` , `FLOAT_REGISTER_INPUT`
`, EXPRESSION_INPUT` }

The configuration type used to determine which type of configuration this instance is.

- enum `CompareOperator` : int { `LESS_THAN` , `GREATER_THAN` }

Compare operator types `LESS_THAN` and `GREATER_THAN` used when waiting for an analog input to go lower or higher, respectively, than a threshold.

Public Member Functions

- `WaitNode` (`WaitNode &f`)
- `WaitNode` (`WaitNode &&f`)
- virtual `~WaitNode` ()

- [ConfigType getConfigType \(\)](#)

This method returns the type of configuration.

- void [setConfigType \(ConfigType type\)](#)
- double [getWaitTime \(\)](#)
- void [setWaitTime \(double time\)](#)
- bool [getBoolValueToWaitFor \(\)](#)

Depending on the type of input this is interpreted in different ways.

- void [setBoolValueToWaitFor \(bool value\)](#)
- float [getFloatValueToWaitFor \(\)](#)
- void [setFloatValueToWaitFor \(float value\)](#)
- void [setDigitalInput \(DigitalIoPtr input\)](#)

Creates a configuration for waiting for a digital input to go high or low.

- DigitalIoPtr [getDigitalInput \(\)](#)
- void [setModbusInput \(ModbusIoPtr input\)](#)

Creates a configuration for waiting for a digital MODBUS input to go high or low.

- ModbusIoPtr [getModbusInput \(\)](#)
- void [setBoolRegister \(BoolRegisterPtr input\)](#)

Creates a configuration for waiting for a boolean register to go true or false.

- BoolRegisterPtr [getBoolRegister \(\)](#)
- void [setCompareOperator \(CompareOperator compareOperator\)](#)
- [CompareOperator getCompareOperator \(\)](#)
- void [setAnalogInput \(AnalogIoPtr input\)](#)

Creates a configuration for waiting for an analog input to go past an electric current threshold.

- AnalogIoPtr [getAnalogInput \(\)](#)
- void [setDoubleRegisterInput \(DoubleRegisterPtr input\)](#)

Creates a configuration for waiting for a register to go past a float threshold.

- DoubleRegisterPtr [getDoubleRegisterInput \(\)](#)
- void [setExpressionInputConfig \(ExpressionPtr expression\)](#)

Create a configuration for waiting for an expression to be evaluated to true.

- ExpressionPtr [getExpressionInputConfig \(\)](#)

Public Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode \(ProgramNode &f\)](#)
- [ProgramNode \(ProgramNode &&f\)](#)
- virtual [~ProgramNode \(\)](#)
- ProgramNodeContributionPtr [getProgramNodeContribution \(\)](#)

Get the program node contribution.

Private Member Functions

- [WaitNode \(\)](#)

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Protected Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.146.1 Detailed Description

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

11.146.2 Member Enumeration Documentation

11.146.2.1 CompareOperator

enum [arcs::aubo_scope::WaitNode::CompareOperator](#) : int

Compare operator types LESS_THAN and GREATER_THAN used when waiting for an analog input to go lower or higher, respectively, than a threshold.

Enumerator

LESS_THAN	
GREATER_THAN	

Definition at line 101 of file [wait_node.h](#).

11.146.2.2 ConfigType

enum [arcs::aubo_scope::WaitNode::ConfigType](#) : int

The configuration type used to determine which type of configuration this instance is.

Enumerator

NO_TYPE	
NO_WAIT	Nothing to wait.
TIME	Wait type is time. The config instance can be cast to TimeWaitNodeConfig.
DIGITAL_INPUT	Wait type is digital_input. The config instance can be cast to DigitalInputWaitNodeConfig.

ANALOG_INPUT_CURRENT	Wait type is analog_input_current. The config instance can be cast to AnalogInputCurrentWaitNodeConfig.
ANALOG_INPUT_VOLTAGE	Wait type is analog_input_voltage. The config instance can be cast to AnalogInputVoltageWaitNodeConfig.
FLOAT_REGISTER_INPUT	Wait type is float_register_input. The config instance can be cast to FloatRegisterInputWaitNodeConfig.
EXPRESSION_INPUT	Wait type is expression_input. The config instance can be cast to ExpressionInputWaitNodeConfig.

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

11.146.3 Constructor & Destructor Documentation

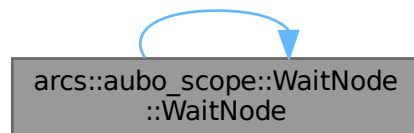
11.146.3.1 WaitNode() [1/3]

```
arcs::aubo_scope::WaitNode::WaitNode (
    WaitNode & f)
```

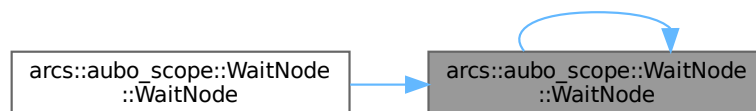
References [WaitNode\(\)](#).

Referenced by [WaitNode\(\)](#), and [WaitNode\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

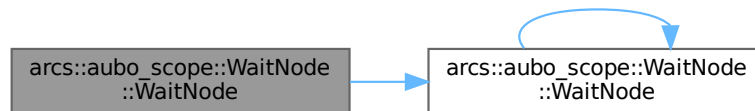


11.146.3.2 WaitNode() [2/3]

```
arcs::aubo_scope::WaitNode::WaitNode (
    WaitNode && f)
```

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

Here is the call graph for this function:



11.146.3.3 ~WaitNode()

```
virtual arcs::aubo_scope::WaitNode::~~WaitNode () [virtual]
```

11.146.3.4 WaitNode() [3/3]

```
arcs::aubo_scope::WaitNode::WaitNode () [private]
```

11.146.4 Member Function Documentation

11.146.4.1 getAnalogInput()

```
AnalogIoPtr arcs::aubo_scope::WaitNode::getAnalogInput ()
```

11.146.4.2 getBoolRegister()

```
BoolRegisterPtr arcs::aubo_scope::WaitNode::getBoolRegister ()
```

11.146.4.3 getBoolValueToWaitFor()

```
bool arcs::aubo_scope::WaitNode::getBoolValueToWaitFor ()
```

Depending on the type of input this is interpreted in different ways.

For digital inputs and MODBUS inputs, true means the input to wait for must be HIGH, false means the input to wait for must be LOW. For boolean registers the value is interpreted literally.

Returns

the input value to wait for.

11.146.4.4 getCompareOperator()

[CompareOperator](#) arcs::aubo_scope::WaitNode::getCompareOperator ()

11.146.4.5 getConfigType()

[ConfigType](#) arcs::aubo_scope::WaitNode::getConfigType ()

This method returns the type of configuration.

Cast this instance appropriately to have access to specific getters.

Returns

the type of this config.

11.146.4.6 getDigitalInput()

DigitalIoPtr arcs::aubo_scope::WaitNode::getDigitalInput ()

11.146.4.7 getDoubleRegisterInput()

DoubleRegisterPtr arcs::aubo_scope::WaitNode::getDoubleRegisterInput ()

11.146.4.8 getExpressionInputConfig()

ExpressionPtr arcs::aubo_scope::WaitNode::getExpressionInputConfig ()

11.146.4.9 getFloatValueToWaitFor()

float arcs::aubo_scope::WaitNode::getFloatValueToWaitFor ()

11.146.4.10 getModbusInput()

ModbusIoPtr arcs::aubo_scope::WaitNode::getModbusInput ()

11.146.4.11 getWaitTime()

double arcs::aubo_scope::WaitNode::getWaitTime ()

Returns

the time to wait for. second

11.146.4.12 setAnalogInput()

void arcs::aubo_scope::WaitNode::setAnalogInput (

Parameters

input	on which to wait, not null.
compareOperator	the operator to use when comparing input to value, not null. Available options are "less than" (<) or "greater than" (>).
current	the electric current threshold to wait for, not null.
errorHandler	the error handler for validation errors. If using ErrorHandler#AUTO_CORRECT this will clamp the value to the nearest valid current value.

Returns

the configuration.

Exceptions

IllegalArgumentException	if input is an output (does not support reading of values). See IO#isInput().

11.146.4.13 setBoolRegister()

```
void arcs::aubo_scope::WaitNode::setBoolRegister (  
    BoolRegisterPtr input)
```

Creates a configuration for waiting for a boolean register to go true or false.

Parameters

input	the boolean register input on which to wait, not null.
value_to_wait_for	true, if waiting for the register to be true}, false} otherwise.

Returns

the configuration.

Exceptions

IllegalArgumentException	if input is an output (does not support reading of values). See IO#isInput() .

11.146.4.14 setBoolValueToWaitFor()

```
void arcs::aubo_scope::WaitNode::setBoolValueToWaitFor (
    bool value)
```

11.146.4.15 setCompareOperator()

```
void arcs::aubo_scope::WaitNode::setCompareOperator (
    CompareOperator compareOperator)
```

11.146.4.16 setConfigType()

```
void arcs::aubo_scope::WaitNode::setConfigType (
    ConfigType type)
```

11.146.4.17 setDigitalInput()

```
void arcs::aubo_scope::WaitNode::setDigitalInput (
    DigitalIoPtr input)
```

Creates a configuration for waiting for a digital input to go high or low.

Parameters

input	the digital input on which to wait, not null.
value_to_wait_for	true, if waiting for the input signal value/level to go HIGH, false otherwise (waiting for LOW signal value/level).

Returns

the configuration.

Exceptions

IllegalArgumentException	if input is an output (does not support reading of values). See IO#isInput().

11.146.4.18 setDoubleRegisterInput()

```
void arcs::aubo_scope::WaitNode::setDoubleRegisterInput (  
    DoubleRegisterPtr input)
```

Creates a configuration for waiting for a register to go past a float threshold.

Parameters

input	the float register input on which to wait, not null.
compareOperator	the operator to use when comparing register to value, not null. Available options are "less than" (<) or "greater than" (>).
value_to_wait_for	the float threshold to wait for.

Returns

the configuration.

Exceptions

IllegalArgumentException	if input is an output (does not support reading of values). See IO#isInput().

11.146.4.19 setExpressionInputConfig()

```
void arcs::aubo_scope::WaitNode::setExpressionInputConfig (  
    ExpressionPtr expression)
```

Create a configuration for waiting for an expression to be evaluated to true.

Parameters

expression	the expression to evaluate, not null.

Returns

the configuration.

11.146.4.20 setFloatValueToWaitFor()

```
void arcs::aubo_scope::WaitNode::setFloatValueToWaitFor (
    float value)
```

11.146.4.21 setModbusInput()

```
void arcs::aubo_scope::WaitNode::setModbusInput (
    ModbusIoPtr input)
```

Creates a configuration for waiting for a digital MODBUS input to go high or low.

Parameters

input	the digital MODBUS input on which to wait, not null.
value_to_wait_for	true, if waiting for the input signal value/level to go HIGH, false otherwise (waiting for LOW signal value/level).

Returns

the configuration.

Exceptions

IllegalArgumentException	if input is an output (does not support reading of values) or if output is not a digital output. See IO#isInput() and IO#getType().

11.146.4.22 setWaitTime()

```
void arcs::aubo_scope::WaitNode::setWaitTime (
    double time)
```

11.146.5 Friends And Related Symbol Documentation

11.146.5.1 DataSwitch

friend class DataSwitch [friend]

Definition at line 239 of file [wait_node.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.146.6 Member Data Documentation

11.146.6.1 d_

void* arcs::aubo_scope::WaitNode::d_ { nullptr } [private]

Definition at line 241 of file [wait_node.h](#).

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

- include/aubo_caps/domain/program/nodes/builtin/[wait_node.h](#)

11.147 arcs::aubo_scope::Waypoint Class Reference

[Waypoint](#) This interface composes parameters that define a position of the robot.

```
#include <waypoint.h>
```

Public Member Functions

- [Waypoint](#) (Waypoint &f)
- [Waypoint](#) (Waypoint &&f)
- [~Waypoint](#) ()
- std::string [getDisplayName](#) ()
Returns the display name of the [Waypoint](#).
- std::vector< double > [getPose](#) ()
Returns the pose for the position of the robot (including the [TCP](#) offset).
- std::vector< double > [getJointPositions](#) ()
Returns the joint positions corresponding to the pose of the robot (when taking the [TCP](#) offset into account).
- std::vector< double > [getTcpOffset](#) ()
Returns the pose for the [TCP](#) offset (used when defining the robot position).
- bool [isDefined](#) ()
Returns whether the [Waypoint](#) is defined.
- size_t [hashCode](#) ()
Returns the hash code of the [Waypoint](#).

Private Member Functions

- [Waypoint](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

11.147.1 Detailed Description

[Waypoint](#) This interface composes parameters that define a position of the robot.

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

11.147.2 Constructor & Destructor Documentation

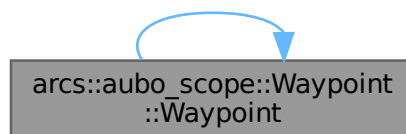
11.147.2.1 [Waypoint](#)() [1/3]

```
arcs::aubo_scope::Waypoint::Waypoint (  
    Waypoint & f)
```

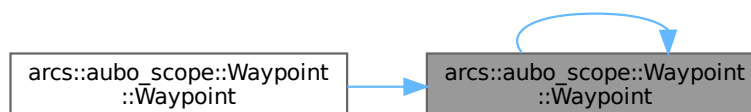
References [Waypoint\(\)](#).

Referenced by [Waypoint\(\)](#), and [Waypoint\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

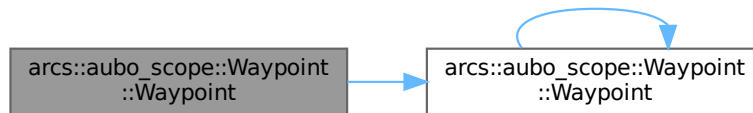


11.147.2.2 Waypoint() [2/3]

```
arcs::aubo_scope::Waypoint::Waypoint (
    Waypoint && f)
```

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

Here is the call graph for this function:



11.147.2.3 ~Waypoint()

```
arcs::aubo_scope::Waypoint::~~Waypoint ()
```

11.147.2.4 Waypoint() [3/3]

```
arcs::aubo_scope::Waypoint::Waypoint () [private]
```

11.147.3 Member Function Documentation

11.147.3.1 getDisplayName()

```
std::string arcs::aubo_scope::Waypoint::getDisplayName ()
```

Returns the display name of the [Waypoint](#).

Returns

Display name.

11.147.3.2 getJointPositions()

```
std::vector< double > arcs::aubo_scope::Waypoint::getJointPositions ()
```

Returns the joint positions corresponding to the pose of the robot (when taking the [TCP](#) offset into account).

Returns

the joint positions.

11.147.3.3 `getPose()`

`std::vector< double > arcs::aubo_scope::Waypoint::getPose ()`

Returns the pose for the position of the robot (including the [TCP](#) offset).

Returns

the pose for the position of the robot.

11.147.3.4 `getTcpOffset()`

`std::vector< double > arcs::aubo_scope::Waypoint::getTcpOffset ()`

Returns the pose for the [TCP](#) offset (used when defining the robot position).

Returns

the [TCP](#) offset.

11.147.3.5 `hashCode()`

`size_t arcs::aubo_scope::Waypoint::hashCode ()`

Returns the hash code of the [Waypoint](#).

Returns

the hash code.

11.147.3.6 `isDefined()`

`bool arcs::aubo_scope::Waypoint::isDefined ()`

Returns whether the [Waypoint](#) is defined.

Returns

true if defined.

11.147.4 Friends And Related Symbol Documentation

11.147.4.1 `DataSwitch`

`friend class DataSwitch [friend]`

Definition at line [111](#) of file [waypoint.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.147.5 Member Data Documentation

11.147.5.1 d__

void* arcs::aubo_scope::Waypoint::d__ { nullptr } [private]

Definition at line 113 of file [waypoint.h](#).

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

- include/aubo_caps/domain/waypoint/[waypoint.h](#)

11.148 arcs::aubo_scope::WaypointModel Class Reference

[WaypointModel](#) Provides methods that returns Waypoints from the current robot installation.

```
#include <waypoint_model.h>
```

Public Member Functions

- [WaypointModel](#) (WaypointModel &f)
- [WaypointModel](#) (WaypointModel &&f)
- [~WaypointModel](#) ()
- std::vector< WaypointPtr > [getWaypoints](#) ()
Returns the geometric Waypoints of the current installation.
- WaypointPtr [getByName](#) (const std::string &name)
Returns the [Waypoint](#) by name.
- WaypointPtr [addWaypoint](#) (const std::string &suggestedName, const std::vector< double > &pose)
Add a [Waypoint](#) to the current AuboScope installation.
- WaypointPtr [addWaypoint](#) (const std::string &suggestedName, const std::vector< double > &pose, const std::vector< double > &joint, const std::vector< double > &offset)
Add a [Waypoint](#) to the current AuboScope installation with joint angles and offset.
- void [removeWaypoint](#) (const std::string &name)
Remove a [Waypoint](#) added by this AuboCap from AuboScope.
- void [renameWaypoint](#) (const std::string &name, const std::string &newName)
Rename the waypoint in the model by the name.

Private Member Functions

- [WaypointModel](#) ()

Private Attributes

- void * [d__](#) { nullptr }

Friends

- class [DataSwitch](#)

11.148.1 Detailed Description

[WaypointModel](#) Provides methods that returns Waypoints from the current robot installation.

Definition at line 22 of file [waypoint_model.h](#).

11.148.2 Constructor & Destructor Documentation

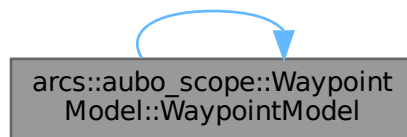
11.148.2.1 WaypointModel() [1/3]

```
arcs::aubo_scope::WaypointModel::WaypointModel (  
    WaypointModel & f)
```

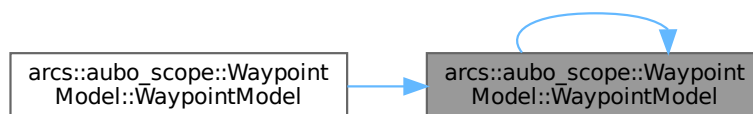
References [WaypointModel\(\)](#).

Referenced by [WaypointModel\(\)](#), and [WaypointModel\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

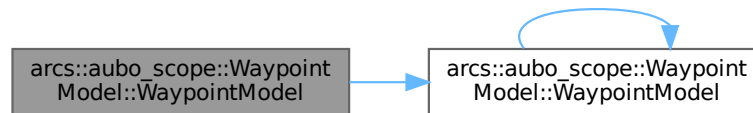


11.148.2.2 WaypointModel() [2/3]

```
arcs::aubo_scope::WaypointModel::WaypointModel (
    WaypointModel && f)
```

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

Here is the call graph for this function:



11.148.2.3 ~WaypointModel()

```
arcs::aubo_scope::WaypointModel::~~WaypointModel ()
```

11.148.2.4 WaypointModel() [3/3]

```
arcs::aubo_scope::WaypointModel::WaypointModel () [private]
```

11.148.3 Member Function Documentation

11.148.3.1 addWaypoint() [1/2]

```
WaypointPtr arcs::aubo_scope::WaypointModel::addWaypoint (
    const std::string & suggestedName,
    const std::vector< double > & pose)
```

Add a [Waypoint](#) to the current AuboScope installation.

This makes it selectable by the end user. The [Waypoint](#) is not modifiable by the end user.

Parameters

suggestedName	Suggested name for the Waypoint . Valid names must match regex <code>[a-zA-Z][a-zA-Z0-9_]{0,14}</code> for a total of 15 characters. The final name can be retrieved from the returned Waypoint instance.
pose	The pose of the Waypoint with respect to the robot base.

Exceptions

WaypointAlreadyAddedException	If a Waypoint has previously been added the same idKey identifier. Use <code>getWaypoint(String)</code> to check if the Waypoint has already been added to the current installation. Use <code>updateWaypoint(String, Pose)</code> to update the Waypoint .
IllegalWaypointNameException	If the suggested name does not match required regex.

11.148.3.2 `addWaypoint()` [2/2]

```
WaypointPtr arcs::aubo_scope::WaypointModel::addWaypoint (
    const std::string & suggestedName,
    const std::vector< double > & pose,
    const std::vector< double > & joint,
    const std::vector< double > & offset)
```

Add a [Waypoint](#) to the current AuboScope installation with joint angles and offset.

Parameters

suggestedName	Suggested name for the Waypoint . Valid names must match regex <code>[a-zA-Z][a-zA-Z0-9_]{0,14}</code> for a total of 15 characters. The final name can be retrieved from the returned Waypoint instance.
pose	The pose of the Waypoint with respect to the robot base.
joint	The joint of the Waypoint with respect to the robot base.
offset	The tcp of the Waypoint with respect to the robot base.

Returns

The [Waypoint](#) created and registered in AuboScope.

Exceptions

WaypointAlreadyAddedException	If a Waypoint has previously been added the same idKey identifier. Use <code>getWaypoint(String)</code> to check if the Waypoint has already been added to the current installation. Use <code>updateWaypoint(String, Pose)</code> to update the Waypoint .
IllegalWaypointNameException	If the suggested name does not match required regex.

11.148.3.3 `getByName()`

```
WaypointPtr arcs::aubo_scope::WaypointModel::getByName (
    const std::string & name)
```

Returns the [Waypoint](#) by name.

Parameters

name	the name of the Waypoint .

Returns

the [Waypoint](#) with the given name.

11.148.3.4 getWaypoints()

```
std::vector< WaypointPtr > arcs::aubo_scope::WaypointModel::getWaypoints ()
```

Returns the geometric Waypoints of the current installation.

Returns

the geometric Waypoints of the current installation.

11.148.3.5 removeWaypoint()

```
void arcs::aubo_scope::WaypointModel::removeWaypoint (
    const std::string & name)
```

Remove a [Waypoint](#) added by this AuboCap from AuboScope.

Program nodes using the [Waypoint](#) will be become undefined because the [Waypoint](#) is no longer resolvable.

Parameters

name	The key used to add the Waypoint with.

Exceptions

WaypointNotFoundException	If no Waypoint exists with the provided idKey.

11.148.3.6 renameWaypoint()

```
void arcs::aubo_scope::WaypointModel::renameWaypoint (
    const std::string & name,
```

```
    const std::string & newName)
```

Rename the waypoint in the model by the name.

Parameters

name	the original name.
newName	the new name.

11.148.4 Friends And Related Symbol Documentation

11.148.4.1 DataSwitch

friend class DataSwitch [friend]

Definition at line 169 of file [waypoint_model.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.148.5 Member Data Documentation

11.148.5.1 d_

void* arcs::aubo_scope::WaypointModel::d_ { nullptr } [private]

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

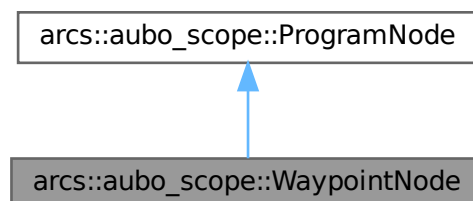
The documentation for this class was generated from the following file:

- [include/aubo_caps/domain/waypoint/waypoint_model.h](#)

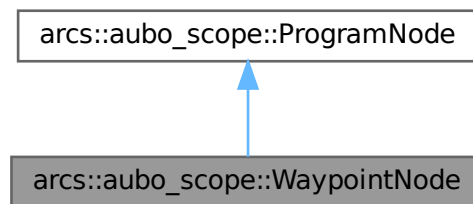
11.149 arcs::aubo_scope::WaypointNode Class Reference

```
#include <waypoint_node.h>
```

Inheritance diagram for arcs::aubo_scope::WaypointNode:



Collaboration diagram for arcs::aubo_scope::WaypointNode:



Public Types

- enum `ConfigType` : int { `FIXED_POSITION` = 0 , `VARIABLE_POSITION` , `RELATIVE_POSITION` }
- The configuration type used to determine which type of configuration this instance is.
- enum `BlendType` : int { `NO_BLEND` = 0 , `SHARED_BLEND` , `WAYPOINT_BLEND` }
- The blend parameters type used to determine which type of blend parameters this instance is.
- enum `MotionType` : int { `TIME_MOTION` , `JOINT_MOTION` , `TOOL_MOTION` , `SHARED_MOTION` }
- The blend parameters type used to determine which type of blend parameters this instance is.
- enum `VariableType` : int { `VARIABLE_POSE` , `VARIABLE_JOINT` }
- enum `RelativePositionType` : int { `TEACHING` , `INPUT` }
- enum `RelativeOffsetType` : int { `CUSTOM` , `VARIABLE` }
- enum class `PositionDefinition` : int { `UNDEFINED` , `DEFINED` }
- enum `VariableDefinition` : int { `UNDEFINED` , `DEFINED` }

Public Member Functions

- `WaypointNode` (`WaypointNode` &f)
- `WaypointNode` (`WaypointNode` &&f)
- virtual `~WaypointNode` ()
- `ConfigType` `getConfigType` ()
- This method returns the type of configuration.
- `BlendType` `getBlendType` ()
- Get blend type of the waypoint.
- `MotionType` `getMotionType` ()
- `PositionDefinition` `getPositionDefinition` ()
- Cast this instance appropriately to have access to specific getters.
- `VariableDefinition` `getVariableDefinition` ()
- Cast this instance appropriately to have access to specific getters.
- double `getBlend` ()
- double `getJointSpeed` ()
- double `getJointAcceleration` ()
- double `getTime` ()
- double `getToolSpeed` ()
- double `getToolAcceleration` ()

- `std::vector< double > getPose ()`
- `std::vector< double > getJointPositions ()`
- `std::vector< double > getTcpOffset ()`
- `VariablePtr getVariable ()`
- `void setSharedMotionParameters ()`
- `void setTimeMotionParameters (unsigned int duration)`
- `void setJointMotionParameters (double joint_speed, double joint_acceleration)`
- `void setToolMotionParameters (double tool_speed, double tool_acceleration)`
- `void setFixedPositionConfig ()`
- `void setFixedPositionConfig (const std::vector< double > &pose_including_tcp, const std::vector< double > &qNear)`
- `void setVariablePositionConfig ()`
- `void setVariablePositionConfig (VariablePtr variable)`

Creates a variable position configuration for a waypoint with a defined variable.

Public Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) (ProgramNode &f)
- [ProgramNode](#) (ProgramNode &&f)
- virtual [~ProgramNode](#) ()
- ProgramNodeContributionPtr [getProgramNodeContribution](#) ()

Get the program node contribution.

Private Member Functions

- [WaypointNode](#) ()

Private Attributes

- void * [d_](#) { nullptr }

Friends

- class [DataSwitch](#)

Additional Inherited Members

Protected Member Functions inherited from [arcs::aubo_scope::ProgramNode](#)

- [ProgramNode](#) ()

11.149.1 Detailed Description

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

11.149.2 Member Enumeration Documentation

11.149.2.1 BlendType

enum [arcs::aubo_scope::WaypointNode::BlendType](#) : int

Enumerator

NO_BLEND	No blend has been selected (the robot arm will stop at this point).
SHARED_BLEND	Shared blend has been selected.
WAYPOINT_BLEND	> Waypoint blend has been selected. The blend instance can be cast to WaypointBlendParameters.

Definition at line 52 of file [waypoint_node.h](#).

11.149.2.2 ConfigType

```
enum arcs::aubo_scope::WaypointNode::ConfigType : int
```

The configuration type used to determine which type of configuration this instance is.

Enumerator

FIXED_POSITION	Fixed position has been selected (fixed waypoint). The config instance can be cast to FixedPositionWaypointNodeConfig.
VARIABLE_POSITION	Variable position has been selected (variable waypoint). The config instance can be cast to VariablePositionWaypointNodeConfig.
RELATIVE_POSITION	Relative position has been selected (relative waypoint).

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

11.149.2.3 MotionType

```
enum arcs::aubo_scope::WaypointNode::MotionType : int
```

The blend parameters type used to determine which type of blend parameters this instance is.

Enumerator

TIME_MOTION	Time motion parameters has been selected. This instance can be cast to TimeMotionParameters.
JOINT_MOTION	Joint motion parameters has been selected. This instance can be cast to JointMotionParameters.
TOOL_MOTION	Tool motion parameters has been selected. This instance can be cast to ToolMotionParameters.
SHARED_MOTION	Shared motion parameters has been selected.

Enumerator

UNDEFINED	The position has not been defined.
DEFINED	The position of the waypoint has been defined. Cast this instance to FixedPositionDefinedWaypointNodeConfig.

Definition at line 152 of file [waypoint_node.h](#).

11.149.2.5 RelativeOffsetType

enum [arcs::aubo_scope::WaypointNode::RelativeOffsetType](#) : int

Enumerator

CUSTOM	Custom offset has been selected.
VARIABLE	Variable type has been selected.

Definition at line 139 of file [waypoint_node.h](#).

11.149.2.6 RelativePositionType

enum [arcs::aubo_scope::WaypointNode::RelativePositionType](#) : int

Enumerator

TEACHING	The teaching location has been selected.
INPUT	The input offset has been selected.

Definition at line 126 of file [waypoint_node.h](#).

11.149.2.7 VariableDefinition

enum [arcs::aubo_scope::WaypointNode::VariableDefinition](#) : int

Enumerator

UNDEFINED	A variable has not been selected. This instance has no further information.
DEFINED	A variable has been selected making this config defined. Cast this instance to VariablePositionDefinedWaypointNodeConfig.

Definition at line 169 of file [waypoint_node.h](#).

11.149.2.8 VariableType

enum [arcs::aubo_scope::WaypointNode::VariableType](#) : int

Enumerator

VARIABLE_POSE	This variable type pose parameters has been selected.
VARIABLE_JOINT	This variable type joint parameters has been selected.

Definition at line 113 of file [waypoint_node.h](#).

11.149.3 Constructor & Destructor Documentation

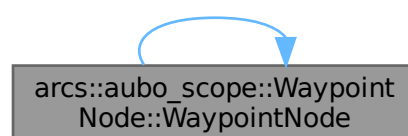
11.149.3.1 WaypointNode() [1/3]

[arcs::aubo_scope::WaypointNode::WaypointNode](#) (
WaypointNode & f)

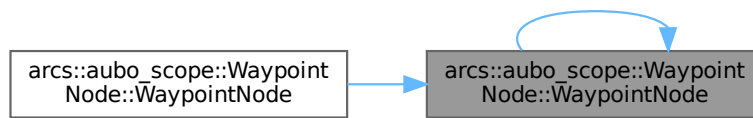
References [WaypointNode\(\)](#).

Referenced by [WaypointNode\(\)](#), and [WaypointNode\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:

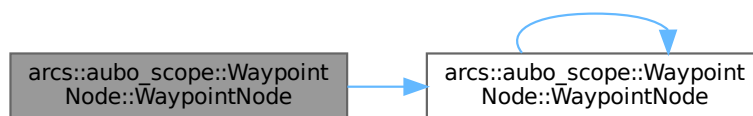


11.149.3.2 `WaypointNode()` [2/3]

```
arcs::aubo_scope::WaypointNode::WaypointNode (
    WaypointNode && f)
```

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

Here is the call graph for this function:



11.149.3.3 `~WaypointNode()`

```
virtual arcs::aubo_scope::WaypointNode::~~WaypointNode () [virtual]
```

11.149.3.4 `WaypointNode()` [3/3]

```
arcs::aubo_scope::WaypointNode::WaypointNode () [private]
```

11.149.4 Member Function Documentation

11.149.4.1 `getBlend()`

```
double arcs::aubo_scope::WaypointNode::getBlend ()
```

Get the blend to be used for the movement.

Note that, if the configuration has not been applied to a waypoint node in the program tree, the blend value returned is a "suggested" value, since the blend may be (automatically) corrected depending on surrounding waypoints in the program tree (when it is applied to a waypoint node).

Returns

the blend for the waypoint.

11.149.4.2 getBlendType()

[BlendType](#) arcs::aubo_scope::WaypointNode::getBlendType ()

Get blend type of the waypoint.

Returns

the blend type.

11.149.4.3 getConfigType()

[ConfigType](#) arcs::aubo_scope::WaypointNode::getConfigType ()

This method returns the type of configuration.

Cast this instance appropriately to have access to specific getters.

Returns

the type of this config.

11.149.4.4 getJointAcceleration()

double arcs::aubo_scope::WaypointNode::getJointAcceleration ()

Returns

the joint acceleration to use for the movement.

11.149.4.5 getJointPositions()

std::vector< double > arcs::aubo_scope::WaypointNode::getJointPositions ()

Returns

the joint positions corresponding to the pose of the robot (when taking the [TCP](#) offset into account)

11.149.4.6 getJointSpeed()

double arcs::aubo_scope::WaypointNode::getJointSpeed ()

Returns

the joint speed to be achieved.

11.149.4.7 getMotionType()

[MotionType](#) arcs::aubo_scope::WaypointNode::getMotionType ()

11.149.4.8 getPose()

std::vector< double > arcs::aubo_scope::WaypointNode::getPose ()

Returns

the waypoint position and orientation.

11.149.4.9 getPositionDefinition()

[PositionDefinition](#) arcs::aubo_scope::WaypointNode::getPositionDefinition ()

Cast this instance appropriately to have access to specific getters.

Returns

the position definition.

11.149.4.10 getTcpOffset()

std::vector< double > arcs::aubo_scope::WaypointNode::getTcpOffset ()

Returns

the pose for the [TCP](#) offset (used when defining the robot position)

11.149.4.11 getTime()

double arcs::aubo_scope::WaypointNode::getTime ()

Returns

the time for the waypoint movement.

11.149.4.12 getToolAcceleration()

double arcs::aubo_scope::WaypointNode::getToolAcceleration ()

Returns

the tool acceleration to use for the movement

11.149.4.13 getToolSpeed()

double arcs::aubo_scope::WaypointNode::getToolSpeed ()

Returns

the tool speed to be achieved.

11.149.4.14 getVariable()

VariablePtr arcs::aubo_scope::WaypointNode::getVariable ()

Returns

the variable for the waypoint.

11.149.4.15 getVariableDefinition()

[VariableDefinition](#) arcs::aubo_scope::WaypointNode::getVariableDefinition ()

Cast this instance appropriately to have access to specific getters.

Returns

the variable definition.

11.149.4.16 setFixedPositionConfig() [1/2]

void arcs::aubo_scope::WaypointNode::setFixedPositionConfig ()

Creates a fixed position configuration with default values for a waypoint.

Note that these default values may change between AuboScope versions.

Returns

the waypoint configuration.

11.149.4.17 setFixedPositionConfig() [2/2]

void arcs::aubo_scope::WaypointNode::setFixedPositionConfig (
 const std::vector< double > & pose_including_tcp,
 const std::vector< double > & qNear)

Creates a fixed position configuration for a waypoint with a defined pose.

Parameters

pose_including_tcp	the position and orientation for the waypoint, not null.
qNear	the joint positions that will be used for the inverse kinematics calculations as a robot configuration near the waypoint pose, not null.
blendParameters	the blend parameters for the waypoint, not null.
waypoint_motion_parameters	the motion parameters for the waypoint, not null.

Returns

the waypoint configuration.

11.149.4.18 setJointMotionParameters()

```
void arcs::aubo_scope::WaypointNode::setJointMotionParameters (
    double joint_speed,
    double joint_acceleration)
```

Creates joint motion parameters for a waypoint.

Can be used for waypoints under a MoveJ.

Parameters

jointSpeed	the joint speed to be achieved, not null.
jointSpeedErrorHandler	error handler for handling validation. If using ErrorHandler#AUTO_CORRECT this will clamp the value to the nearest valid joint speed value.
joint_acceleration	the joint acceleration to be used for the movement, not null.
joint_accelerationErrorHandler	error handler for handling validation. If using ErrorHandler#AUTO_CORRECT this will clamp the value to the nearest valid joint acceleration value.

Returns

the joint motion parameters for the waypoint.

11.149.4.19 setSharedMotionParameters()

```
void arcs::aubo_scope::WaypointNode::setSharedMotionParameters ()
```

Creates shared motion parameters for a waypoint.

Can be used for waypoints under a MoveJ, MoveL or a MoveP.

Returns

the motion parameters for the waypoint.

11.149.4.20 setTimeMotionParameters()

```
void arcs::aubo_scope::WaypointNode::setTimeMotionParameters (  
    unsigned int duration)
```

Creates time motion parameters for a waypoint.

Can be used for waypoints under a MoveJ or a MoveL.

Parameters

duration	the time the movement should take, not null.
errorHandler	error handler for handling validation. If using ErrorHandler#AUTO_CORRECT this will clamp the value to the nearest valid time value.

Returns

the time motion parameters for the waypoint.

11.149.4.21 setToolMotionParameters()

```
void arcs::aubo_scope::WaypointNode::setToolMotionParameters (  
    double tool_speed,  
    double tool_acceleration)
```

Creates tool motion parameters for a waypoint.

Can be used for waypoints under a MoveL or a MoveP.

Parameters

tool_speed	the tool speed to be achieved, not null.
tool_speedErrorHandler	error handler for handling validation. If using ErrorHandler#AUTO_CORRECT this will clamp the value to the nearest valid tool speed value.
tool_acceleration	the tool acceleration to be used for the movement, not null.
tool_accelerationErrorHandler	error handler for handling validation. If using ErrorHandler#AUTO_CORRECT this will clamp the value to the nearest valid tool acceleration value.

Returns

the tool motion parameters for the waypoint.

11.149.4.22 `setVariablePositionConfig()` [1/2]

```
void arcs::aubo_scope::WaypointNode::setVariablePositionConfig ()
```

Creates a variable position configuration with default values for a waypoint

Note that these default values may change between AuboScope versions.

Returns

the waypoint configuration.

11.149.4.23 `setVariablePositionConfig()` [2/2]

```
void arcs::aubo_scope::WaypointNode::setVariablePositionConfig (
    VariablePtr variable)
```

Creates a variable position configuration for a waypoint with a defined variable.

Parameters

variable	the variable position and orientation for the waypoint, not null.
blendParameters	the blend parameters for the waypoint, not null.
waypoint_motion_parameters	the motion parameters for the waypoint, not null.

Returns

the waypoint configuration.

11.149.5 Friends And Related Symbol Documentation

11.149.5.1 `DataSwitch`

```
friend class DataSwitch [friend]
```

Definition at line 409 of file [waypoint_node.h](#).

References [DataSwitch](#).

Referenced by [DataSwitch](#).

11.149.6 Member Data Documentation

11.149.6.1 d_

void* arcs::aubo_scope::WaypointNode::d_ { nullptr } [private]

Definition at line 411 of file [waypoint_node.h](#).

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

- include/aubo_caps/domain/program/nodes/builtin/[waypoint_node.h](#)

11.150 arcs::aubo_scope::WidthCapability Class Reference

[WidthCapability](#) This interface represents a registered width/position capability for a gripper which supports moving to a user configurable position (open/close to a configurable width).

```
#include <width_capability.h>
```

Public Member Functions

- virtual [~WidthCapability](#) ()=default
- virtual void [updateCapability](#) (double minWidth, double maxWidth, double defaultGripWidth, double defaultReleaseWidth)=0

11.150.1 Detailed Description

[WidthCapability](#) This interface represents a registered width/position capability for a gripper which supports moving to a user configurable position (open/close to a configurable width).

Through this interface, it is possible to adjust the range and default values for the width after the capability has been registered. This is typically not applicable for most grippers, but can be relevant when the range depends on another custom gripper setting (which can be configured by the user).

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

11.150.2 Constructor & Destructor Documentation

11.150.2.1 ~WidthCapability()

virtual arcs::aubo_scope::WidthCapability::~~WidthCapability () [virtual], [default]

11.150.3 Member Function Documentation

11.150.3.1 updateCapability()

```
virtual void arcs::aubo_scope::WidthCapability::updateCapability (
    double minWidth,
    double maxWidth,
    double defaultGripWidth,
    double defaultReleaseWidth) [pure virtual]
```

Parameters

minWidth	minimum width supported by the gripper
maxWidth	maximum width supported by the gripper
defaultGripWidth	default value for the width for a grip action. This value is used for, e.g. performing a "default" grip action using the toolbar
defaultReleaseWidth	default value for the width for a release action. This value is used for, e.g. performing a "default" release action using the toolbar
unit	the unit for all specified values, not null

Exceptions

InvalidCapabilityRange	if minWidth > maxWidth
InvalidCapabilityRange	if defaultGripWidth or defaultReleaseWidth is outside the range defined by minWidth and maxWidth.

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

- include/aubo_caps/contribution/driver/gripper/capability/[width_capability.h](#)

Chapter 12

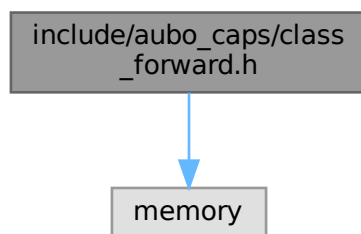
File Documentation

12.1 doc/SDK_overview_en.md File Reference

12.2 include/aubo_caps/class_forward.h File Reference

```
#include <memory>
```

Include dependency graph for class_forward.h:



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



Macros

- #define [ARCS_ABI_EXPORT](#)
- #define [ARCS_ABI_IMPORT](#)
- #define [ARCS_ABI_LOCAL](#)
- #define [ARCS_DECLARE_PTR](#)(Name, Type)

- `#define ARCS_DECLARE_PTR_MEMBER(Type)`
- `#define ARCS_CLASS_FORWARD(C)`
Macro that forward declares a class and defines the respective smartpointers through `ARCS_DECLARE_PTR`.
- `#define ARCS_STRUCT_FORWARD(C)`
Like `ARCS_CLASS_FORWARD`, but forward declares the type as a struct instead of a class.

12.2.1 Macro Definition Documentation

12.2.1.1 ARCS_ABI_EXPORT

```
#define ARCS_ABI_EXPORT
```

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

12.2.1.2 ARCS_ABI_IMPORT

```
#define ARCS_ABI_IMPORT
```

Definition at line 17 of file [class_forward.h](#).

12.2.1.3 ARCS_ABI_LOCAL

```
#define ARCS_ABI_LOCAL
```

Definition at line 18 of file [class_forward.h](#).

12.2.1.4 ARCS_CLASS_FORWARD

```
#define ARCS_CLASS_FORWARD(  
    C)
```

Value:

```
class C;  
ARCS_DECLARE_PTR(C, C)
```

Macro that forward declares a class and defines the respective smartpointers through `ARCS_DECLARE_PTR`.

Definition at line 68 of file [class_forward.h](#).

12.2.1.5 ARCS_DECLARE_PTR

```
#define ARCS_DECLARE_PTR(  
    Name,  
    Type)
```

Value:

```
typedef std::shared_ptr<Type> Name##Ptr; \
typedef std::shared_ptr<const Type> Name##ConstPtr; \
typedef std::weak_ptr<Type> Name##WeakPtr; \
typedef std::weak_ptr<const Type> Name##ConstWeakPtr; \
typedef std::unique_ptr<Type> Name##UniquePtr; \
typedef std::unique_ptr<const Type> Name##ConstUniquePtr
```

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

12.2.1.6 ARCS_DECLARE_PTR_MEMBER

```
#define ARCS_DECLARE_PTR_MEMBER(  
    Type)
```

Value:

```
typedef std::shared_ptr<Type> Ptr; \
typedef std::shared_ptr<const Type> ConstPtr; \
typedef std::weak_ptr<Type> WeakPtr; \
typedef std::weak_ptr<const Type> ConstWeakPtr; \
typedef std::unique_ptr<Type> UniquePtr; \
typedef std::unique_ptr<const Type> ConstUniquePtr
```

Definition at line 54 of file [class_forward.h](#).

12.2.1.7 ARCS_STRUCT_FORWARD

```
#define ARCS_STRUCT_FORWARD(  
    C)
```

Value:

```
struct C; \
ARCS_DECLARE_PTR(C, C)
```

Like ARCS_CLASS_FORWARD, but forward declares the type as a struct instead of a class.

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

12.3 class_forward.h

[Go to the documentation of this file.](#)

```
00001 #ifndef AUBO_SCOPE_CLASS_FORWARD_H
00002 #define AUBO_SCOPE_CLASS_FORWARD_H
00003
00004 #include <memory>
00005
00006 #if defined(ARCS_ABI_EXPORT)
00007 #elif defined(ARCS_PLATFORM_WINDOWS)
00008 #define ARCS_ABI_EXPORT __declspec(dllexport)
00009 #define ARCS_ABI_IMPORT __declspec(dllimport)
00010 #define ARCS_ABI_LOCAL
00011 #elif defined(ARCS_HAVE_VISIBILITY_ATTRIBUTE)
00012 #define ARCS_ABI_EXPORT __attribute__((visibility("default")))
00013 #define ARCS_ABI_IMPORT __attribute__((visibility("default")))
00014 #define ARCS_ABI_LOCAL __attribute__((visibility("hidden")))
00015 #else
00016 #define ARCS_ABI_EXPORT
00017 #define ARCS_ABI_IMPORT
00018 #define ARCS_ABI_LOCAL
00019 #endif
00020
00021 /**
00022  * @def ARCS_DECLARE_PTR
00023  * Macro that given a Name and a Type declares the following types:
00024  * - ${Name}Ptr = shared_ptr<${Type}>
00025  * - ${Name}ConstPtr = shared_ptr<const ${Type}>
00026  * - ${Name}WeakPtr = weak_ptr<${Type}>
00027  * - ${Name}ConstWeakPtr = weak_ptr<const ${Type}>
00028  * - ${Name}UniquePtr = unique_ptr<${Type}>
00029  * - ${Name}ConstUniquePtr = unique_ptr<const ${Type}>
00030  *
00031  * For best portability the exact type of shared_ptr declared by the macro
00032  * should be considered to be an implementation detail, liable to change in
00033  * future releases.
00034  */
00035
00036 #define ARCS_DECLARE_PTR(Name, Type) \
00037     typedef std::shared_ptr<Type> Name##Ptr; \
```

```

00038 typedef std::shared_ptr<const Type> Name##ConstPtr; \
00039 typedef std::weak_ptr<Type> Name##WeakPtr; \
00040 typedef std::weak_ptr<const Type> Name##ConstWeakPtr; \
00041 typedef std::unique_ptr<Type> Name##UniquePtr; \
00042 typedef std::unique_ptr<const Type> Name##ConstUniquePtr
00043
00044 /**
00045  * @def ARCS_DECLARE_PTR_MEMBER
00046  * The macro defines the same typedefs as ARCS_DECLARE_PTR, but shortens the
00047  * new names to their suffix.
00048  *
00049  * This can be used to create `Classname::Ptr` style names, but in most
00050  * situations in ARCS's codebase, ARCS_CLASS_FORWARD and ARCS_DECLARE_PTR
00051  * should be preferred.
00052  */
00053
00054 #define ARCS_DECLARE_PTR_MEMBER(Type) \
00055     typedef std::shared_ptr<Type> Ptr; \
00056     typedef std::shared_ptr<const Type> ConstPtr; \
00057     typedef std::weak_ptr<Type> WeakPtr; \
00058     typedef std::weak_ptr<const Type> ConstWeakPtr; \
00059     typedef std::unique_ptr<Type> UniquePtr; \
00060     typedef std::unique_ptr<const Type> ConstUniquePtr
00061
00062 /**
00063  * @def ARCS_CLASS_FORWARD
00064  * Macro that forward declares a class and defines the respective smartpointers
00065  * through ARCS_DECLARE_PTR.
00066  */
00067
00068 #define ARCS_CLASS_FORWARD(C) \
00069     class C; \
00070     ARCS_DECLARE_PTR(C, C)
00071
00072 /**
00073  * @def ARCS_STRUCT_FORWARD
00074  * Like ARCS_CLASS_FORWARD, but forward declares the type as a struct
00075  * instead of a class.
00076  */
00077 #define ARCS_STRUCT_FORWARD(C) \
00078     struct C; \
00079     ARCS_DECLARE_PTR(C, C)
00080
00081 #endif

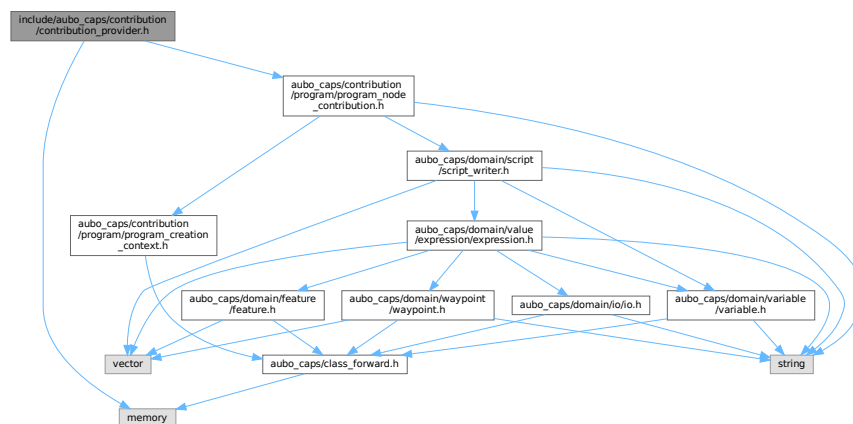
```

12.4 include/aubo_caps/contribution/contribution_provider.h File Reference

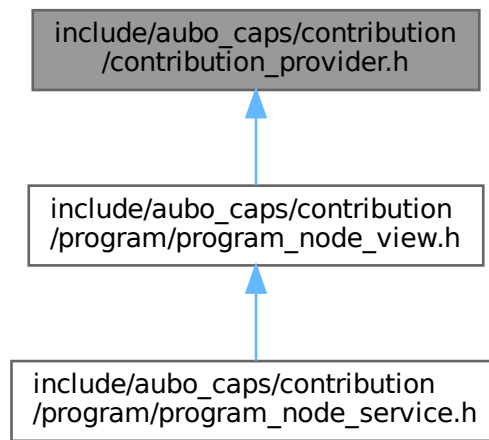
#include <memory>

#include <aubo_caps/contribution/program/program_node_contribution.h>

Include dependency graph for contribution_provider.h:



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



Classes

- class [arcs::aubo_scope::ContributionProvider< T >](#)
[ContributionProvider](#) This interfaces provides access to a contribution.

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Typedefs

- template<class T>
using [arcs::aubo_scope::ContributionProviderPtr](#) = std::shared_ptr<[ContributionProvider](#)<T>>

12.5 contribution_provider.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_CONTRIBUTION_PROVIDER_H
00002 #define AUBO_SCOPE_CONTRIBUTION_PROVIDER_H
00003
00004 #include <memory>
00005 #include <aubo_caps/contribution/program/program_node_contribution.h>
00006
00007 namespace arcs {
00008 namespace aubo_scope {
00009
00010 /**
00011  * \chinese
00012  *
00013  *
00014  *
00015  *      HTML
  
```

```

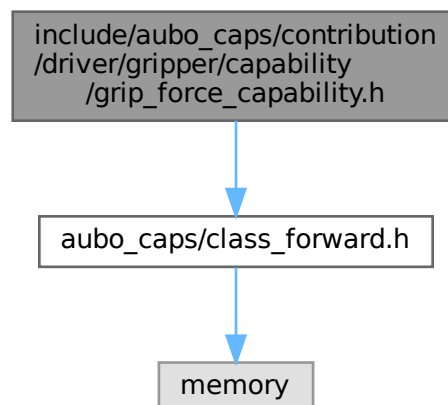
00016 *
00017 * @tparam <T>
00018 * \endchinese
00019 * \english
00020 * ContributionProvider
00021 * This interfaces provides access to a contribution.
00022 *
00023 * NOTE: This interface is not relevant for contributions with a HTML-based user
00024 * interface.
00025 *
00026 * @tparam <T> the (generic) type parameter for the interface representing the
00027 * type of contribution the interface provides access to
00028 * \endenglish
00029 */
00030 template <class T>
00031 class ContributionProvider
00032 {
00033 public:
00034     virtual ~ContributionProvider() = default;
00035
00036     /**
00037      * \chinese
00038      *
00039      * @return
00040      * \endchinese
00041      * \english
00042      * Gets a contribution. Call this as the result of a user action to get the
00043      * currently selected node.
00044      *
00045      * @return the relevant contribution
00046      * \endenglish
00047      */
00048     virtual T *get() = 0;
00049 };
00050
00051 template <class T>
00052 using ContributionProviderPtr = std::shared_ptr<ContributionProvider<T>;
00053
00054 } // namespace aubo_scope
00055 } // namespace arcs
00056
00057 #endif // AUBO_SCOPE_CONTRIBUTION_PROVIDER_H

```

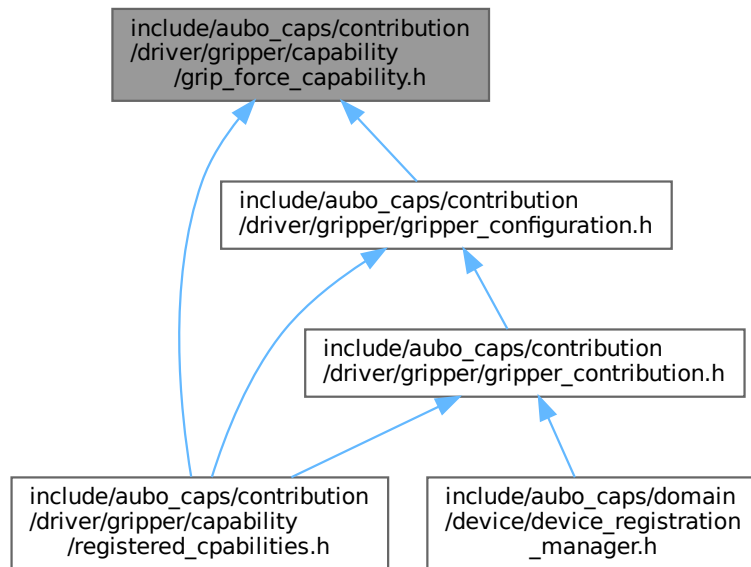
12.6 include/aubo_caps/contribution/driver/gripper/capability/grip_force_capability.h File Reference

#include <aubo_caps/class_forward.h>

Include dependency graph for grip_force_capability.h:



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



Classes

- class [arcs::aubo_scope::GripForceCapability](#)
[GripForceCapability](#) This interface represents a registered force capability for a gripper which supports gripping using a user configurable force.

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([GripForceCapability](#))

12.7 grip_force_capability.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_GRIP_FORCE_CAPABILITY_H
00002 #define AUBO_SCOPE_GRIP_FORCE_CAPABILITY_H
00003
00004 #include <aubo_caps/class_forward.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(GripForceCapability);
00009
00010 /**

```

```

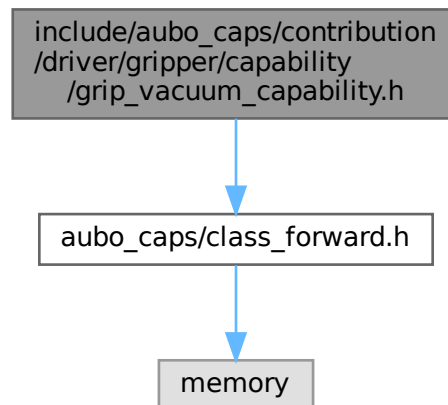
00011 * \chinese
00012 *
00013 *
00014 *
00015 *
00016 *
00017 * \endchinese
00018 * \english
00019 * GripForceCapability
00020 * This interface represents a registered force capability for a gripper which
00021 * supports gripping using a user configurable force.
00022 *
00023 * Through this interface, it is possible to adjust the range and default value
00024 * for the force after the capability has been registered. This is typically not
00025 * applicable for most grippers, but can be relevant when the range depends on
00026 * another custom gripper setting (which can be configured by the user).
00027 * \endenglish
00028 */
00029 class GripForceCapability
00030 {
00031 public:
00032     virtual ~GripForceCapability() = 0;
00033
00034     /**
00035      * <p>
00036      * Update the range and default value for the force supported by the
00037      * gripper.
00038      * </p>
00039      *
00040      * Updating the range could result in an already entered value in the
00041      * configuration of a gripper program node being outside the range. This
00042      * will make the given program node undefined, and force the end user to
00043      * adjust the value to be valid again.
00044      *
00045      * @param minForce minimum force supported by the gripper
00046      * @param maxForce maximum force supported by the gripper
00047      * @param defaultGripForce default value for the force for a grip action.
00048      * This value is used for, e.g. performing a "default" grip action using the
00049      * toolbar
00050      * @param unit the unit for all specified values, <code>null</code>
00051      * @throws InvalidCapabilityRange if <code>minForce</code> > <code>maxForce</code>
00052      * or <code>defaultGripForce</code> is outside the range defined by
00053      * <code>minForce</code> and <code>maxForce</code>
00054      */
00055     virtual void updateCapability(double minForce, double maxForce,
00056                                   double defaultGripForce) = 0;
00057 };
00058 } // namespace aubo_scope
00059 } // namespace arcs
00060 #endif

```

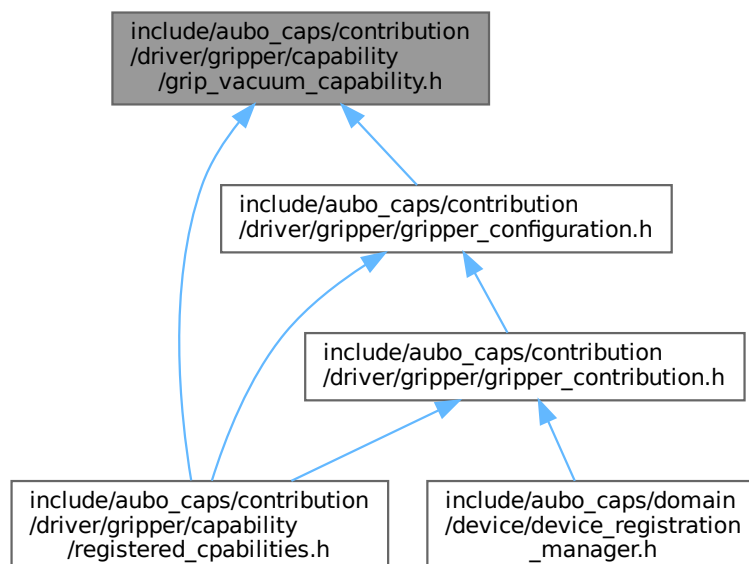
12.8 include/aubo_caps/contribution/driver/gripper/capability/grip_vacuum_capability.h File Reference

```
#include <aubo_caps/class_forward.h>
```

Include dependency graph for grip_vacuum_capability.h:



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



Classes

- class [arcs::aubo_scope::GripVacuumCapability](#)
[GripVacuumCapability](#) This interface represents a registered vacuum capability for a gripper which supports gripping using a user configurable vacuum level.

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([GripVacuumCapability](#))

12.9 [grip_vacuum_capability.h](#)

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_GRIP_VACUUM_CAPABILITY_H
00002 #define AUBO_SCOPE_GRIP_VACUUM_CAPABILITY_H
00003
00004 #include <aubo_caps/class_forward.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(GripVacuumCapability);
00009
00010 /**
00011  * \chinese
00012  *
00013  *
00014  *
00015  *
00016  *
00017  * \endchinese
00018  * \english
00019  * GripVacuumCapability
00020  * This interface represents a registered vacuum capability for a gripper which
00021  * supports gripping using a user configurable vacuum level.
00022  *
00023  * Through this interface, it is possible to adjust the range and default value
00024  * for the vacuum level after the capability has been registered. This is
00025  * typically not applicable for most grippers, but can be relevant when the
00026  * range depends on another custom gripper setting (which can be configured by
00027  * the user).
00028  * \endenglish
00029  */
00030 class GripVacuumCapability
00031 {
00032 public:
00033     virtual ~GripVacuumCapability() = 0;
00034
00035     /**
00036      * <p>
00037      * Update the range and default value for the vacuum level supported by the
00038      * gripper.
00039      * </p>
00040      *
00041      * <p>
00042      * Updating the range could result in an already entered value in the
00043      * configuration of a gripper program node being outside the range. This
00044      * will make the given program node undefined, and force the end user to
00045      * adjust the value to be valid again.
00046      * </p>
00047      *
00048      * If the gripper works with absolute pressure, specify a positive range for
00049      * the supported vacuum level. If the gripper works with relative pressure
00050      * (vacuum), specify a negative range for the supported vacuum level.
00051      *
00052      * @param minVacuum minimum vacuum level supported by the gripper
00053      * @param maxVacuum maximum vacuum level supported by the gripper

```

```

00054 * @param defaultGripVacuum default value for the vacuum level for a grip
00055 * action. This value is used for, e.g. performing a "default" grip action
00056 * using the toolbar
00057 * @param unit the unit for all specified values, not <code>null</code>
00058 * @throws InvalidCapabilityRange if <code>minVacuum</code> >
00059 * <code>maxVacuum</code> or <code>defaultGripVacuum</code> is outside the
00060 * range defined by <code>minVacuum</code> and <code>maxVacuum</code>.
00061 */
00062 virtual void updateCapability(double minVacuum, double maxVacuum,
00063                             double defaultGripVacuum) = 0;
00064 };
00065 } // namespace aubo_scope
00066 } // namespace arcs
00067
00068 #endif

```

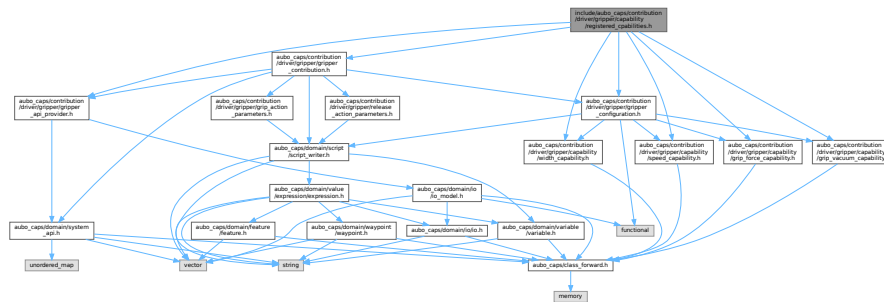
12.10 include/aubo_caps/contribution/driver/gripper/capability/registered_cpabilities.h File Reference

```

#include <aubo_caps/contribution/driver/gripper/gripper_api_provider.h>
#include <aubo_caps/contribution/driver/gripper/gripper_configuration.h>
#include <aubo_caps/contribution/driver/gripper/gripper_contribution.h>
#include <aubo_caps/contribution/driver/gripper/capability/grip_force_capability.h>
#include <aubo_caps/contribution/driver/gripper/capability/grip_vacuum_capability.h>
#include <aubo_caps/contribution/driver/gripper/capability/speed_capability.h>
#include <aubo_caps/contribution/driver/gripper/capability/width_capability.h>

```

Include dependency graph for registered_cpabilities.h:



Classes

- class [arcs::aubo_scope::RegisteredCapabilities](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(RegisteredCapabilities\)](#)

12.11 registered_cpabilities.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_REGISTERED_CAPABILITIES_H
00002 #define AUBO_SCOPE_REGISTERED_CAPABILITIES_H
00003
00004 #include <aubo_caps/contribution/driver/gripper/gripper_api_provider.h>
00005 #include <aubo_caps/contribution/driver/gripper/gripper_configuration.h>
00006 #include <aubo_caps/contribution/driver/gripper/gripper_contribution.h>
00007 #include <aubo_caps/contribution/driver/gripper/capability/grip_force_capability.h>
00008 #include <aubo_caps/contribution/driver/gripper/capability/grip_vacuum_capability.h>
00009 #include <aubo_caps/contribution/driver/gripper/capability/speed_capability.h>
00010 #include <aubo_caps/contribution/driver/gripper/capability/width_capability.h>
00011
00012 namespace arcs {
00013 namespace aubo_scope {
00014 ARCS_CLASS_FORWARD(RegisteredCapabilities);
00015
00016 /**
00017  * <p>
00018  * Each of the individual grippers/zones available in a multi-gripper device
00019  * supports all capabilities that has been registered, i.e. all the individual
00020  * grippers have the exact same set of capabilities. This interface gives access
00021  * to all registered capabilities, such as the width capability (see {@link
00022  * WidthCapability}), speed capability (see
00023  * {@link SpeedCapability}), etc. This enables dynamic adjustment of capability
00024  * properties, including the value range and default values, exclusively per
00025  * individual gripper (independently of the other grippers).
00026  * </p>
00027  *
00028  * <p>
00029  * This interface is only relevant if the multi-gripper capability has been
00030  * registered (using the method
00031  * {@link GripperCapabilities#registerWidthCapability(double, double, double,
00032  * double, Length.Unit)}).
00033  * </p>
00034  *
00035  * <p>
00036  * Each capability <b>must</b> be registered using the {@link
00037  * GripperCapabilities} interface (when
00038  * {@link GripperContribution#configureGripper(GripperConfiguration,
00039  * GripperAPIProvider)} is called) before they can be accessed using this
00040  * interface.
00041  * </p>
00042  *
00043  * <b>NOTE:</b> If an update of the properties of a capability needs to apply
00044  * for all individual grippers, it is more convenient to perform the adjustment
00045  * simultaneously for all grippers using the capability instance (e.g.
00046  * {@link WidthCapability}) returned when the capability was registered (with
00047  * the {@link GripperCapabilities} interface).
00048  */
00049 class RegisteredCapabilities
00050 {
00051 public:
00052     virtual ~RegisteredCapabilities() = default;
00053
00054     /**
00055      * <p>
00056      * Access the width capability interface which allows for dynamic
00057      * adjustments of the capability properties exclusively for this
00058      * specific individual gripper (independently of the other grippers),
00059      * including the value range and default value.
00060      * </p>
00061      *
00062      * This is only valid, if the width capability has been registered
00063      * (using the method
00064      * {@link GripperCapabilities#registerWidthCapability(double, double,
00065      * double, double, Length.Unit)}).
00066      *
00067      * @return The width capability for this individual gripper
00068      * @throws UnsupportedOperationException if the gripper has not
00069      * registered the width capability
00070      */
00071     virtual WidthCapabilityPtr getWidthCapability() = 0;
00072
00073     /**
00074      * <p>
00075      * Access the grip force capability interface which allows for dynamic
00076      * adjustments of the capability properties exclusively for this specific
00077      * individual gripper (independently of the other grippers), including the
00078      * value range and default value.
00079      * </p>
00080      *
00081      *
00082      * This is only valid, if the force capability has been registered (using

```

```

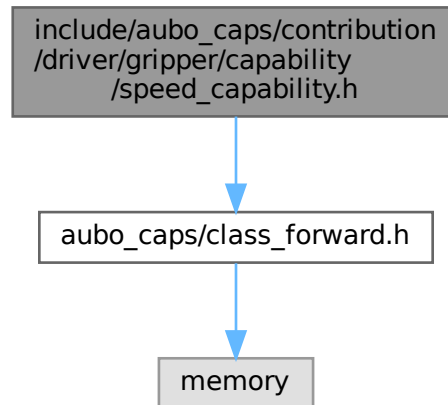
00083     * the method
00084     * {@link GripperCapabilities#registerGrippingForceCapability(double,
00085     * double, double, Force.Unit)}}.
00086     *
00087     * @return The grip force capability for this individual gripper
00088     * @throws UnsupportedOperationException if the gripper has not registered
00089     * the force capability
00090     */
00091     virtual GripForceCapabilityPtr getGripForceCapability() = 0;
00092
00093     /**
00094     * <p>
00095     * Access the speed capability interface which allows for dynamic
00096     * adjustments of the capability properties exclusively for this specific
00097     * individual gripper (independently of the other grippers), including the
00098     * value range and default value.
00099     * </p>
00100     *
00101     * This is only valid, if the speed capability has been registered (using
00102     * the method
00103     * {@link GripperCapabilities#registerSpeedCapability(double, double,
00104     * double, double, Speed.Unit)}}.
00105     *
00106     * @return The speed capability for this individual gripper
00107     * @throws UnsupportedOperationException if the gripper has not registered
00108     * the speed capability
00109     */
00110     virtual SpeedCapabilityPtr getSpeedCapability() = 0;
00111
00112     /**
00113     * <p>
00114     * Access the grip vacuum capability interface which allows for dynamic
00115     * adjustments of the capability properties exclusively for this specific
00116     * individual gripper (independently of the other grippers), including the
00117     * value range and default value.
00118     * </p>
00119     *
00120     * This is only valid, if the vacuum capability has been registered (using
00121     * the method
00122     * {@link GripperCapabilities#registerGrippingVacuumCapability(double,
00123     * double, double, Pressure.Unit)}}.
00124     *
00125     * @return The grip vacuum capability for this individual gripper
00126     * @throws UnsupportedOperationException if the gripper has not registered
00127     * the vacuum capability
00128     */
00129     virtual GripVacuumCapabilityPtr getGripVacuumCapability() = 0;
00130 };
00131 } // namespace aubo_scope
00132 } // namespace arcs
00133 #endif

```

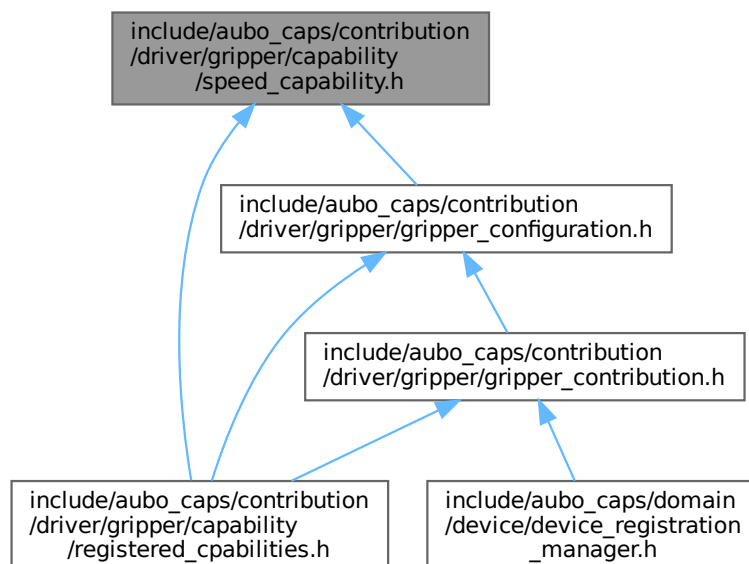
12.12 include/aubo_caps/contribution/driver/gripper/capability/speed_capability.h File Reference

```
#include <aubo_caps/class_forward.h>
```

Include dependency graph for speed_capability.h:



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



Classes

- class [arcs::aubo_scope::SpeedCapability](#)
[SpeedCapability](#) This interface represents a registered speed capability for a gripper which supports operating using a user configurable speed.

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([SpeedCapability](#))

12.13 speed_capability.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_SPEED_CAPABILITY_H
00002 #define AUBO_SCOPE_SPEED_CAPABILITY_H
00003
00004 #include <aubo_caps/class_forward.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(SpeedCapability);
00009
00010 /**
00011  * \chinese
00012  *
00013  *
00014  *
00015  *
00016  *
00017  * \endchinese
00018  * \english
00019  * SpeedCapability
00020  * This interface represents a registered speed capability for a gripper which
00021  * supports operating using a user configurable speed.
00022  *
00023  * Through this interface, it is possible to adjust the range and default values
00024  * for the speed after the capability has been registered. This is typically not
00025  * applicable for most grippers, but can be relevant when the range depends on
00026  * another custom gripper setting (which can be configured by the user).
00027  * \endenglish
00028  */
00029 class SpeedCapability
00030 {
00031 public:
00032     virtual ~SpeedCapability() = default;
00033
00034     /**
00035      * <p>
00036      * Update the range and default values for the speed supported by the
00037      * gripper.
00038      * </p>
00039      *
00040      * Updating the range could result in an already entered value in the
00041      * configuration of a gripper program node being outside the range. This
00042      * will make the given program node undefined, and force the end user to
00043      * adjust the value to be valid again.
00044      *
00045      * @param minSpeed minimum operating speed supported by the gripper
00046      * @param maxSpeed maximum operating speed supported by the gripper
00047      * @param defaultGripSpeed default value for the speed for a grip action.
00048      * This value is used for, e.g. performing a "default" grip action using the
00049      * toolbar
00050      * @param defaultReleaseSpeed default value for the speed for a release
00051      * action. This value is used for, e.g. performing a "default" release
00052      * action using the toolbar
00053      * @param unit the unit for all specified values, not <code>null</code>

```

```

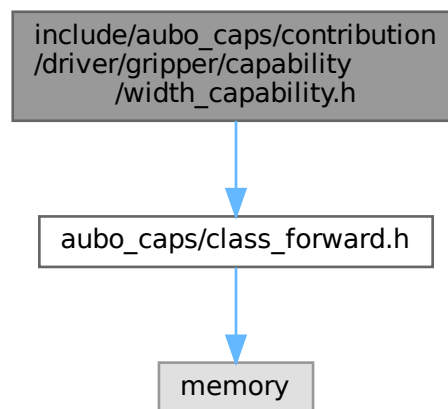
00054 * @throws InvalidCapabilityRange if <code>minSpeed</code> >
00055 * <code>maxSpeed</code>
00056 * @throws InvalidCapabilityRange if <code>defaultGripSpeed</code> or
00057 * <code>defaultReleaseSpeed</code> is outside the range defined by
00058 * <code>minSpeed</code> and <code>maxSpeed</code>.
00059 */
00060 virtual void updateCapability(double minSpeed, double maxSpeed,
00061                             double defaultGripSpeed,
00062                             double defaultReleaseSpeed) = 0;
00063 };
00064 } // namespace aubo_scope
00065 } // namespace arcs
00066 #endif

```

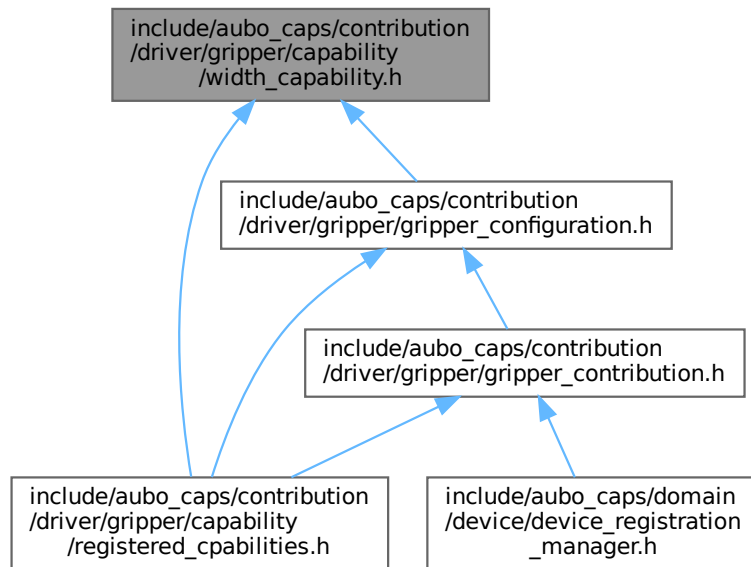
12.14 include/aubo_caps/contribution/driver/gripper/capability/↵ width_capability.h File Reference

#include <aubo_caps/class_forward.h>

Include dependency graph for width_capability.h:



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



Classes

- class `arcs::aubo_scope::WidthCapability`
`WidthCapability` This interface represents a registered width/position capability for a gripper which supports moving to a user configurable position (open/close to a configurable width).

Namespaces

- namespace `arcs`
- namespace `arcs::aubo_scope`

Functions

- `arcs::aubo_scope::ARCS_CLASS_FORWARD` (`WidthCapability`)

12.15 width_capability.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_WIDTH_CAPABILITY_H
00002 #define AUBO_SCOPE_WIDTH_CAPABILITY_H
00003
00004 #include <aubo_caps/class_forward.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(WidthCapability);
00009
00010 /**

```

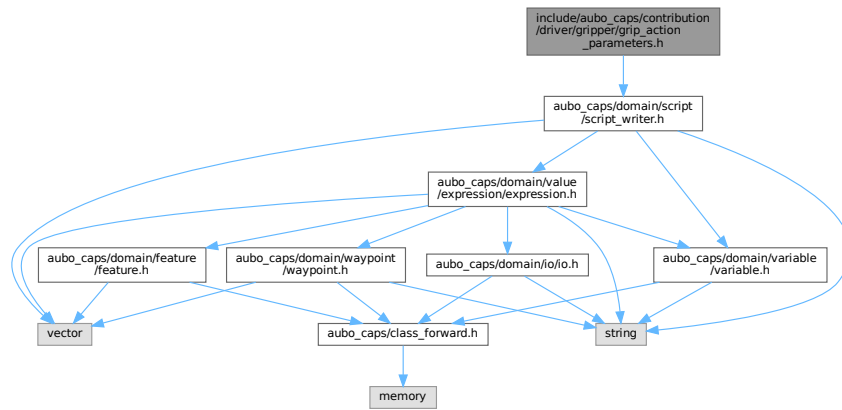
```

00011 * \chinese
00012 *
00013 *      /      /
00014 *
00015 *
00016 *
00017 *
00018 * \endchinese
00019 * \english
00020 * WidthCapability
00021 * This interface represents a registered width/position capability for a
00022 * gripper which supports moving to a user configurable position (open/close to
00023 * a configurable width).
00024 *
00025 * Through this interface, it is possible to adjust the range and default values
00026 * for the width after the capability has been registered. This is typically not
00027 * applicable for most grippers, but can be relevant when the range depends on
00028 * another custom gripper setting (which can be configured by the user).
00029 * \endenglish
00030 */
00031 class WidthCapability
00032 {
00033 public:
00034     virtual ~WidthCapability() = default;
00035
00036     /**
00037      * <p>
00038      * Update the range and default values for the width supported by the
00039      * gripper.
00040      * </p>
00041      *
00042      * Updating the range could result in an already entered value in the
00043      * configuration of a gripper program node being outside the range. This
00044      * will make the given program node undefined, and force the end user to
00045      * adjust the value to be valid again.
00046      *
00047      * @param minWidth minimum width supported by the gripper
00048      * @param maxWidth maximum width supported by the gripper
00049      * @param defaultGripWidth default value for the width for a grip
00050      * action. This value is used for, e.g. performing a "default" grip
00051      * action using the toolbar
00052      * @param defaultReleaseWidth default value for the width for a release
00053      * action. This value is used for, e.g. performing a "default" release
00054      * action using the toolbar
00055      * @param unit the unit for all specified values, not <code>null</code>
00056      * @throws InvalidCapabilityRange if <code>minWidth</code> > <code>maxWidth</code>
00057      * @throws InvalidCapabilityRange if <code>defaultGripWidth</code> or
00058      * <code>defaultReleaseWidth</code> is outside the range defined by
00059      * <code>minWidth</code> and <code>maxWidth</code>.
00060      */
00061     virtual void updateCapability(double minWidth, double maxWidth,
00062                                   double defaultGripWidth,
00063                                   double defaultReleaseWidth) = 0;
00064 };
00065 } // namespace aubo_scope
00066 } // namespace arcs
00067 #endif

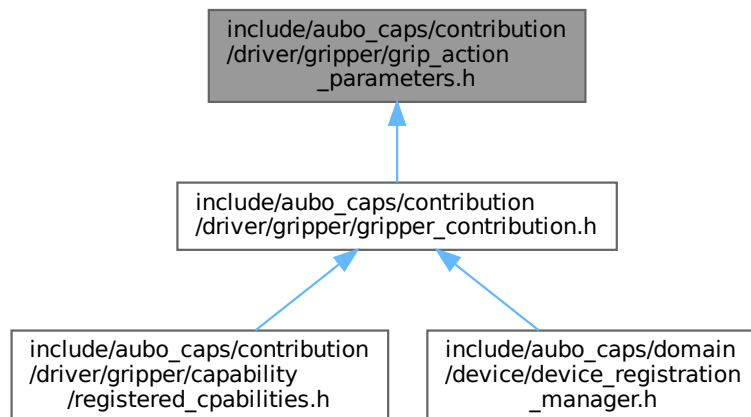
```

12.16 include/aubo_caps/contribution/driver/gripper/grip_action_parameters.h File Reference

#include <aubo_caps/domain/script/script_writer.h>
 Include dependency graph for grip_action_parameters.h:



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



Classes

- class [arcs::aubo_scope::GripActionParameters](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([GripActionParameters](#))

12.17 `grip_action_parameters.h`

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_GRIP_ACTION_PARAMETERS_H
00002 #define AUBO_SCOPE_GRIP_ACTION_PARAMETERS_H
00003
00004 #include <aubo_caps/domain/script/script_writer.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(GripActionParameters);
00009
00010 /**
00011  * <p>
00012  * This interface represents parameters for a gripping action defined/configured
00013  * by the end user.
00014  * </p>
00015  *
00016  * These parameters are used when generating script code for a gripping action
00017  * (see
00018  * {@link GripperContribution#generateGripActionScript(ScriptWriter,
00019  * GripActionParameters)}).
00020  */
00021 class GripActionParameters
00022 {
00023 public:
00024     virtual ~GripActionParameters() = default;
00025
00026     /**
00027      * <p>
00028      * Get the selected force to be used the gripping action.
00029      * </p>
00030      *
00031      * This is only valid, if the force capability has been registered
00032      * (using the method
00033      * {@link GripperCapabilities#registerGrippingForceCapability(double,
00034      * double, double, Force.Unit)}).
00035      *
00036      * @return The force to use for the gripping action
00037      * @throws UnsupportedOperationException if the gripper has not
00038      * registered the force capability
00039      */
00040     virtual double getForce() = 0;
00041
00042     /**
00043      * <p>
00044      * Get the selected width to be used for the gripping action.
00045      * </p>
00046      *
00047      * This is only valid, if the width capability has been registered (using
00048      * the method
00049      * {@link GripperCapabilities#registerWidthCapability(double, double,
00050      * double, double, Length.Unit)}).
00051      *
00052      * @return The width to use for the gripping action
00053      * @throws UnsupportedOperationException if the gripper has not registered
00054      * the width capability
00055      */
00056     virtual double getWidth() = 0;
00057
00058     /**
00059      * <p>
00060      * Get the selected vacuum level to be used for the gripping action.
00061      * </p>
00062      *
00063      * This is only valid, if the vacuum capability has been registered (using
00064      * the method
00065      * {@link GripperCapabilities#registerGrippingVacuumCapability(double,
00066      * double, double, Pressure.Unit)}).
00067      *
00068      * @return The vacuum level to use for the gripping action
00069      * @throws UnsupportedOperationException if the gripper has not registered
00070      * the vacuum capability
00071      */
00072     virtual double getVacuum() = 0;
00073

```

```

00074  /**
00075   * <p>
00076   * Get the selected speed to be used for the gripping action.
00077   * </p>
00078   *
00079   * This is only valid, if the speed capability has been registered (using
00080   * the method
00081   * {@link GripperCapabilities#registerSpeedCapability(double, double,
00082   * double, double, Speed.Unit)}).
00083   *
00084   * @return The speed to use for the gripping action
00085   * @throws UnsupportedOperationException if the gripper has not registered
00086   * the speed capability
00087   */
00088  virtual double getSpeed() = 0;
00089
00090  /**
00091   * <p>
00092   * Use this method to determine whether the grip detection option in the
00093   * Gripper program node is on or off.
00094   * </p>
00095   *
00096   * <p>
00097   * Note that this method will report that grip detection is off, when script
00098   * code is to be generated due to the user testing a grip configuration of a
00099   * gripper program or operating the gripper using the toolbar.
00100   * </p>
00101   *
00102   * <p>
00103   * This method should only be called, if the grip detected feedback
00104   * capability has been registered (using
00105   * {@link
00106   * GripperFeedbackCapabilities#registerGripDetectedCapability(ScriptCodeGenerator)}).
00107   * </p>
00108   *
00109   * <p>
00110   * The purpose of this method is to ensure that the appropriate script code
00111   * for performing a grip action is generated when the end user has specified
00112   * a new payload value. The user-defined payload value will be applied by
00113   * AuboScope immediately after the generated grip action script code has
00114   * finished executing. Depending on the enablement state of grip detection,
00115   * the generated script should have the following properties: <ul> <li>
00116   * <i>Grip detection off:</i> The script code should not finish
00117   * earlier than when it is appropriate to apply the new payload value (i.e.
00118   * when the object has been gripped). This typically means that the script
00119   * code should wait a time period corresponding to closing the gripper's
00120   * "fingers" fully (if the gripper is vacuum operated then long enough to
00121   * achieve some level of vacuum).
00122   * </li>
00123   * <li>
00124   * <i>Grip detection on:</i> The payload will only be set if the
00125   * grip detection was successful, i.e. it will be applied immediately after
00126   * a grip was detected. This means that it is not necessary for the script
00127   * code to wait (since the built-in detection timeout functionality will
00128   * handle the waiting part).
00129   * </li>
00130   * </ul>
00131   * </p>
00132   *
00133   * @return <code>true</code> if the end user has enabled grip detection, and
00134   * <code>false</code> if grip detection is disabled.
00135   * @throws UnsupportedOperationException if the gripper has not registered
00136   * the grip detected feedback capability
00137   */
00138  virtual bool isGripDetectionEnabled() = 0;
00139 };
00140 } // namespace aubo_scope
00141 } // namespace arcs
00142 #endif

```

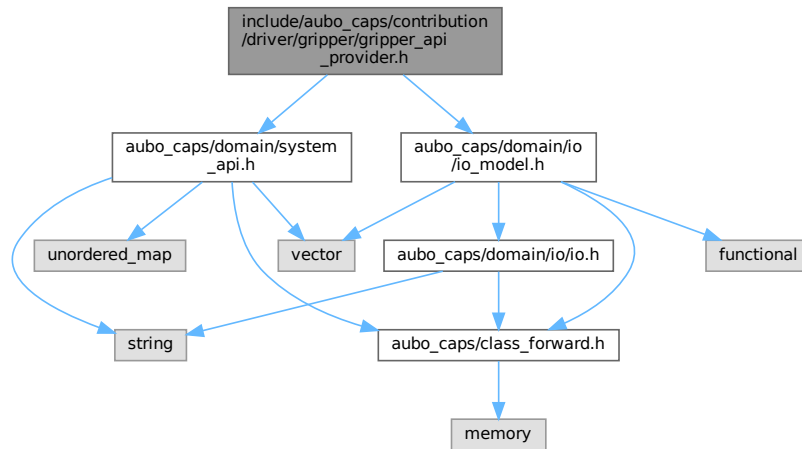
12.18 include/aubo_caps/contribution/driver/gripper/gripper_api_provider.h File Reference

```

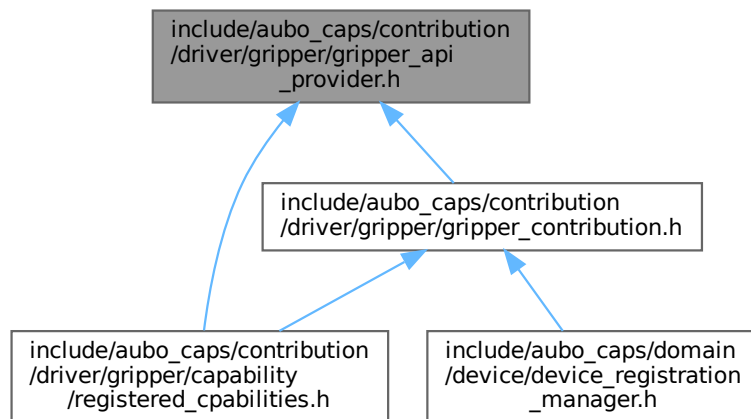
#include <aubo_caps/domain/system_api.h>
#include <aubo_caps/domain/io/io_model.h>

```

Include dependency graph for gripper_api_provider.h:



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



Classes

- class [arcs::aubo_scope::GripperApiProvider](#)
[GripperApiProvider](#) This interface provides access to functionality and services available from within AuboScope relevant for gripper contributions.

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(GripperApiProvider\)](#)

12.19 gripper_api_provider.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_GRIPPER_API_PROVIDER_H
00002 #define AUBO_SCOPE_GRIPPER_API_PROVIDER_H
00003
00004 #include <aubo_caps/domain/system_api.h>
00005 #include <aubo_caps/domain/io/io_model.h>
00006
00007 namespace arcs {
00008 namespace aubo_scope {
00009 ARCS_CLASS_FORWARD(GripperApiProvider);
00010
00011 /**
00012  * \chinese
00013  *   API
00014  *   AuboScope
00015  * \endchinese
00016  * \english
00017  * GripperApiProvider
00018  * This interface provides access to functionality and services available from
00019  * within AuboScope relevant for gripper contributions.
00020  * \endenglish
00021  */
00022 class GripperApiProvider
00023 {
00024 public:
00025     virtual ~GripperApiProvider() = default;
00026
00027     /**
00028      * \chinese
00029      * @return
00030      * \endchinese
00031      * \english
00032      * @return An interface providing access to system related functionality.
00033      * \endenglish
00034      */
00035     virtual SystemApiPtr getSystemApi() = 0;
00036
00037     /**
00038      * \chinese
00039      * @return I/O          I/O
00040      * \endchinese
00041      * \english
00042      * @return An I/O model interface (e.g. for getting I/Os to display in the
00043      * custom user input for setting up the gripper)
00044      * \endenglish
00045      */
00046     virtual IoModelPtr getIoModel() = 0;
00047 };
00048 } // namespace aubo_scope
00049 } // namespace arcs
00050
00051 #endif

```

12.20 include/aubo_caps/contribution/driver/gripper/gripper_↵ configuration.h File Reference

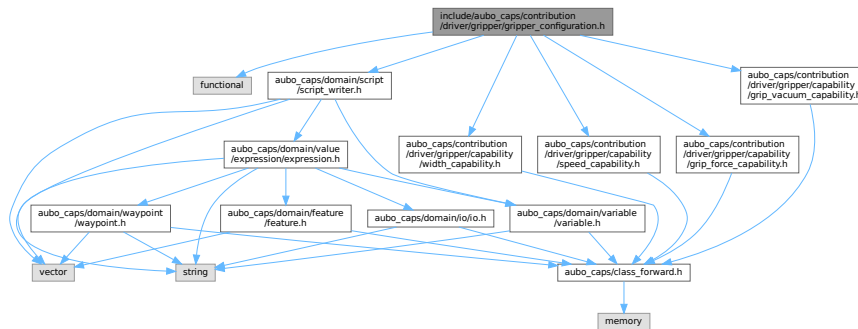
```

#include <functional>
#include <aubo_caps/domain/script/script_writer.h>
#include <aubo_caps/contribution/driver/gripper/capability/width_capability.h>
#include <aubo_caps/contribution/driver/gripper/capability/speed_capability.h>
#include <aubo_caps/contribution/driver/gripper/capability/grip_force_capability.h>

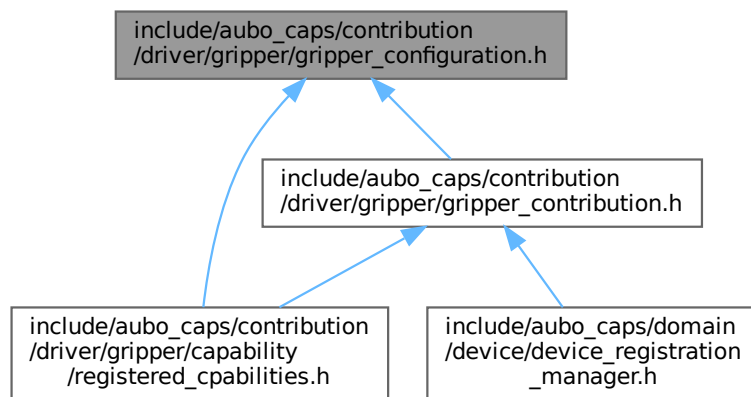
```

```
#include <aubo_caps/contribution/driver/gripper/capability/grip_vacuum_capability.h>
```

Include dependency graph for gripper_configuration.h:



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



Classes

- class [arcs::aubo_scope::GripperConfiguration](#)
This interface provides access to register or setup various properties and capabilities of a gripper.

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(GripperConfiguration\)](#)

12.21 gripper_configuration.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_GRIPPER_CONFIGURATION_H
00002 #define AUBO_SCOPE_GRIPPER_CONFIGURATION_H
00003
00004 #include <functional>
00005 #include <aubo_caps/domain/script/script_writer.h>
00006 #include <aubo_caps/contribution/driver/gripper/capability/width_capability.h>
00007 #include <aubo_caps/contribution/driver/gripper/capability/speed_capability.h>
00008 #include <aubo_caps/contribution/driver/gripper/capability/grip_force_capability.h>
00009 #include <aubo_caps/contribution/driver/gripper/capability/grip_vacuum_capability.h>
00010
00011 namespace arcs {
00012 namespace aubo_scope {
00013 ARCS_CLASS_FORWARD(GripperConfiguration);
00014
00015 /**
00016  * This interface provides access to register or setup various properties and
00017  * capabilities of a gripper. It is optional to register these capabilities and
00018  * properties.
00019  */
00020 class GripperConfiguration
00021 {
00022 public:
00023     virtual ~GripperConfiguration() = default;
00024
00025     /**
00026      * Register a width/position capability for a gripper that supports moving
00027      * to a user configurable position (open/close to a configurable width).
00028      *
00029      * @param minWidth The minimum width supported by the gripper
00030      * @param maxWidth The maximum width supported by the gripper
00031      * @param defaultGripWidth default value for the width for a grip action.
00032      * This value is used for, e.g. performing a "default" grip action using the
00033      * toolbar
00034      * @param defaultReleaseWidth default value for the width for a release
00035      * action. This value is used for, e.g. performing a "default" release
00036      * action using the toolbar
00037      * @param unit the unit for all specified values, not <code>null</code>
00038      * @return a width capability allowing for dynamic adjustments of the
00039      * capability properties including the value range and default values
00040      * @throws InvalidCapabilityRange if <code>minWidth</code> >
00041      * <code>maxWidth</code>
00042      * @throws InvalidCapabilityRange if <code>defaultGripWidth</code> or
00043      * <code>defaultReleaseWidth</code> is outside the range defined by
00044      * <code>minWidth</code> and <code>maxWidth</code>.
00045      * @throws CapabilityAlreadyRegistered if this capability has already been
00046      * registered
00047      * @throws CalledOutsideGripperConfigurationPhase if this method is called
00048      * at the wrong time by a AuboCap implementing the {@link
00049      * GripperContribution} interface, i.e. called outside the scope of the
00050      * configuration of the gripper.
00051      * @throws CalledOutsideMethodScope if this method is called at the wrong
00052      * time by a (regular) AuboCap which has registered its program node
00053      * contribution/service as conformant with a AuboScope gripper (using
00054      * {@link GripperRegistrationManager#registerAsGripper(Class)}),
00055      * i.e. called outside the scope of the call to either {@link
00056      * ProgramNodeService#configureContribution(ContributionConfiguration)}
00057      * (for a Qt-based AuboCap) or {@link
00058      * ProgramNodeServiceConfigurable#configureContribution(ProgramNodeConfiguration)}
00059      * (for a HTML-based AuboCap).
00060      */
00061     virtual WidthCapabilityPtr registerWidthCapability(
00062         double minWidth, double maxWidth, double defaultGripWidth,
00063         double defaultReleaseWidth) = 0;
00064
00065     /**
00066      * Register a force capability for a gripper that supports gripping using a
00067      * user configurable force.
00068      *
00069      * @param minForce minimum force supported by the gripper
00070      * @param maxForce maximum force supported by the gripper
00071      * @param defaultGripForce default value for the force for a grip action.
00072      * This value is used for, e.g. performing a "default" grip action using the
00073      * toolbar
00074      * @param unit the unit for all specified values, <code>null</code>
00075      * @return a force capability allowing for dynamic adjustments of the
00076      * capability properties including the value range and default value
00077      * @throws InvalidCapabilityRange if <code>minForce</code> >
00078      * <code>maxForce</code> or <code>defaultGripForce</code> is outside the
00079      * range defined by <code>minForce</code> and <code>maxForce</code>.
00080      * @throws CapabilityAlreadyRegistered if this capability has already been
00081      * registered
00082      * @throws CalledOutsideGripperConfigurationPhase if this method is called

```

```

00083 * at the wrong time by a AuboCap implementing the {@link
00084 * GripperContribution} interface, i.e. called outside the scope of the
00085 * configuration of the gripper.
00086 * @throws CalledOutsideMethodScope if this method is called at the wrong
00087 * time by a (regular) AuboCap which has registered its program node
00088 * contribution/service as conformant with a AuboScope gripper (using
00089 * {@link GripperRegistrationManager#registerAsGripper(Class)}),
00090 * i.e. called outside the scope of the call to either {@link
00091 * ProgramNodeService#configureContribution(ContributionConfiguration)}
00092 * (for a Qt-based AuboCap) or {@link
00093 * ProgramNodeServiceConfigurable#configureContribution(ProgramNodeConfiguration)}
00094 * (for a HTML-based AuboCap).
00095 */
00096 virtual GripForceCapabilityPtr registerGrippingForceCapability(
00097     double minForce, double maxForce, double defaultGripForce) = 0;
00098
00099 /**
00100 * Register a speed capability for a gripper that supports operating using a
00101 * user configurable speed.
00102 *
00103 * @param minSpeed minimum speed supported by the gripper
00104 * @param maxSpeed maximum speed supported by the gripper
00105 * @param defaultGripSpeed default value for the speed for a grip action.
00106 * This value is used for, e.g. performing a "default" grip action using the
00107 * toolbar
00108 * @param defaultReleaseSpeed default value for the speed for a grip action.
00109 * This value is used for, e.g. performing a "default" release action using
00110 * the toolbar
00111 * @param unit the unit for all specified values, not <code>null</code>
00112 * @return a speed capability allowing for dynamic adjustments of the
00113 * capability properties including the value range and default values
00114 * @throws InvalidCapabilityRange if <code>minSpeed</code> >
00115 * <code>maxSpeed</code>
00116 * @throws InvalidCapabilityRange if <code>defaultGripSpeed</code> or
00117 * <code>defaultReleaseSpeed</code> is outside the range defined by
00118 * <code>minSpeed</code> and <code>maxSpeed</code>.
00119 * @throws CapabilityAlreadyRegistered if this capability has already been
00120 * registered
00121 * @throws CalledOutsideGripperConfigurationPhase if this method is called
00122 * at the wrong time by a AuboCap implementing the {@link
00123 * GripperContribution} interface, i.e. called outside the scope of the
00124 * configuration of the gripper.
00125 * @throws CalledOutsideMethodScope if this method is called at the wrong
00126 * time by a (regular) AuboCap which has registered its program node
00127 * contribution/service as conformant with a AuboScope gripper (using
00128 * {@link GripperRegistrationManager#registerAsGripper(Class)}),
00129 * i.e. called outside the scope of the call to either {@link
00130 * ProgramNodeService#configureContribution(ContributionConfiguration)}
00131 * (for a Qt-based AuboCap) or {@link
00132 * ProgramNodeServiceConfigurable#configureContribution(ProgramNodeConfiguration)}
00133 * (for a HTML-based AuboCap).
00134 */
00135 virtual SpeedCapabilityPtr registerSpeedCapability(
00136     double minSpeed, double maxSpeed, double defaultGripSpeed,
00137     double defaultReleaseSpeed) = 0;
00138
00139 /**
00140 * <p>
00141 * Register a vacuum capability for a gripper that supports gripping using a
00142 * user configurable vacuum level.
00143 * </p>
00144 *
00145 * If the gripper works with absolute pressure, specify a positive range for
00146 * the supported vacuum level. If the gripper works with relative pressure
00147 * (vacuum), specify a negative range for the supported vacuum level.
00148 *
00149 * @param minVacuum minimum vacuum level supported by the gripper
00150 * @param maxVacuum maximum vacuum level supported by the gripper
00151 * @param defaultGripVacuum default value for the vacuum level for a grip
00152 * action. This value is used for, e.g. performing a "default" grip action
00153 * using the toolbar
00154 * @param unit the unit for all specified values, not <code>null</code>
00155 * @return a vacuum capability allowing for dynamic adjustments of the
00156 * capability properties including the value range and default value
00157 * @throws InvalidCapabilityRange if <code>minVacuum</code> >
00158 * <code>maxVacuum</code> or <code>defaultGripVacuum</code> is outside the
00159 * range defined by <code>minVacuum</code> and <code>maxVacuum</code>.
00160 * @throws CapabilityAlreadyRegistered if this capability has already been
00161 * registered
00162 * @throws CalledOutsideGripperConfigurationPhase if this method is called
00163 * at the wrong time by a AuboCap implementing the {@link
00164 * GripperContribution} interface, i.e. called outside the scope of the
00165 * configuration of the gripper.
00166 * @throws CalledOutsideMethodScope if this method is called at the wrong
00167 * time by a (regular) AuboCap which has registered its program node
00168 * contribution/service as conformant with a AuboScope gripper (using
00169 * {@link GripperRegistrationManager#registerAsGripper(Class)}),

```

```

00170 * i.e. called outside the scope of the call to either {@link
00171 * ProgramNodeService#configureContribution(ContributionConfiguration)}
00172 * (for a Qt-based AuboCap) or {@link
00173 * ProgramNodeServiceConfigurable#configureContribution(ProgramNodeConfiguration)}
00174 * (for a HTML-based AuboCap).
00175 */
00176 virtual GripVacuumCapabilityPtr registerGrippingVacuumCapability(
00177     double minVacuum, double maxVacuum, double defaultGripVacuum) = 0;
00178
00179 /**
00180 * <p>
00181 * Register this feedback capability if the gripper can inform AuboScope
00182 * whether it has detected that an object has been gripped, after a grip
00183 * action has been triggered. This will enable the end user to configure the
00184 * actions to take when the gripper has detected that an object has been
00185 * gripped and/or when the grip action timed out.
00186 * </p>
00187 *
00188 * <p>
00189 * The provided implementation of the {@link ScriptCodeGenerator} interface
00190 * must generate the script code for determining if a grip was detected. The
00191 * return value of the script code <b>must</b> be a boolean, i.e. 'True' or
00192 * 'False'.
00193 * </p>
00194 *
00195 * <b>Note:</b> When this capability has been registered, the
00196 * {@link GripperContribution#generateGripActionScript(ScriptWriter,
00197 * GripActionParameters)} method should generate different grip action
00198 * script code depending on whether the end user has enabled or disabled the
00199 * grip detection option (in the Gripper program node). For more details,
00200 * see Javadoc for
00201 * {@link GripActionParameters#isGripDetectionEnabled()}.
00202 *
00203 * @param scriptCodeGenerator script generator that generates the script
00204 * code for determining if a grip was detected. When the script code needs
00205 * to be generated, the method
00206 * {@link
00207 * ScriptCodeGenerator#generateScript(ScriptWriter, Object)} will be called
00208 * (by AuboScope). The resulting script code will be embedded in a script
00209 * function and <b>must</b> return a boolean, i.e. 'True' or 'False'.
00210 * @throws CapabilityAlreadyRegistered if this capability has already been
00211 * registered
00212 * @throws CalledOutsideGripperConfigurationPhase if this method is called
00213 * at the wrong time, i.e. outside the scope of the {@link
00214 * GripperContribution#configureGripper(GripperConfiguration,
00215 * GripperAPIProvider)} method.
00216 */
00217 virtual void registerGripDetectedCapability(
00218     std::function<void(ScriptWriterPtr scriptWriter)>
00219     scriptCodeGenerator) = 0;
00220
00221 /**
00222 * <p>
00223 * Register this feedback capability if the gripper can inform AuboScope
00224 * whether it has detected an object has been released, after a release
00225 * action has been triggered. This will enable the end user to configure the
00226 * actions to take when the gripper has detected that an object has been
00227 * released and/or when the release action timed out.
00228 * </p>
00229 *
00230 * <p>
00231 * The provided implementation of the {@link ScriptCodeGenerator} interface
00232 * must generate the script code for determining if a release was detected.
00233 * The return value of the script code <b>must</b> be a boolean, i.e. 'True'
00234 * or 'False'.
00235 * </p>
00236 *
00237 * <b>Note:</b> When this capability has been registered, the
00238 * {@link GripperContribution#generateReleaseActionScript(ScriptWriter,
00239 * ReleaseActionParameters)} method should generate different release action
00240 * script code depending on whether the end user has enabled or disabled the
00241 * release detection option (in the Gripper program node). For more details,
00242 * see Javadoc for
00243 * {@link GripActionParameters#isGripDetectionEnabled()}.
00244 *
00245 * @param scriptCodeGenerator script generator that generates the script
00246 * code for determining if a release was detected. When the script code
00247 * needs to be generated, the method
00248 * {@link
00249 * ScriptCodeGenerator#generateScript(ScriptWriter, Object)} will be called
00250 * (by AuboScope). The resulting script code will be embedded in a script
00251 * function and <b>must</b> return a boolean, i.e. 'True' or 'False'.
00252 * @throws CapabilityAlreadyRegistered if this capability has already been
00253 * registered
00254 * @throws CalledOutsideGripperConfigurationPhase if this method is called
00255 * at the wrong time, i.e. outside the scope of the {@link
00256 * GripperContribution#configureGripper(GripperConfiguration,

```


Namespaces

- namespace `arcs`
- namespace `arcs::aubo_scope`

Functions

- `arcs::aubo_scope::ARCS_CLASS_FORWARD` (`GripperContribution`)

12.23 gripper_contribution.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_GRIPPER_CONTRIBUTION_H
00002 #define AUBO_SCOPE_GRIPPER_CONTRIBUTION_H
00003
00004 #include <aubo_caps/contribution/driver/gripper/gripper_api_provider.h>
00005 #include <aubo_caps/contribution/driver/gripper/gripper_configuration.h>
00006 #include <aubo_caps/contribution/driver/gripper/release_action_parameters.h>
00007 #include <aubo_caps/contribution/driver/gripper/grip_action_parameters.h>
00008
00009 #include <aubo_caps/domain/system_api.h>
00010 #include <aubo_caps/domain/script/script_writer.h>
00011
00012 namespace arcs {
00013 namespace aubo_scope {
00014 ARCS_CLASS_FORWARD(GripperContribution);
00015
00016 /**
00017  * \chinese
00018  *
00019  *      AuboScope      API
00020  *
00021  *
00022  *      CB3
00023  *
00024  *
00025  *      " "
00026  *      {@link GripperContribution#configureGripper(GripperConfiguration,
00027  * GripperAPIProvider)}
00028  *
00029  *      IP
00030  *      UI
00031  *
00032  * {@link GripperContribution#configureInstallation(CustomUserInputConfiguration,
00033  * SystemConfiguration, TCPConfiguration, GripperAPIProvider)}
00034  *
00035  *      AuboScope
00036  * {@link #generatePreambleScript(ScriptWriter)}
00037  *
00038  *      " "
00039  * \endchinese
00040  * \english
00041  * GripperContribution
00042  * This interface defines an API for adding functionality for gripping devices
00043  * to AuboScope.
00044  *
00045  * Implementing this interface will create a program node contribution, an
00046  * installation contribution and a toolbar contribution exposing gripper
00047  * functionality to the end user. Note that the toolbar contribution is not
00048  * supported on CB3 robots.
00049  *
00050  * The program node contribution allows the end user to program grip and release
00051  * actions. The configuration options available to the end user are based on the
00052  * properties and capabilities supported by the gripper. As a minimum, all
00053  * grippers must support "default" grip and release actions. All other
00054  * capabilities are optional to support and can be registered when the {@link
00055  * GripperContribution#configureGripper(GripperConfiguration,
00056  * GripperAPIProvider)} method is called.
00057  *
00058  * The installation node contribution can be used, if any custom input from the
00059  * end user is required to setup the gripper, e.g. specifying an IP address. The
00060  * corresponding generated UI will be accessible in the installation node. This
00061  * functionality is optional, since such configuration properties are specific
00062  * for each gripper. Access to create a custom configuration based on user
00063  * inputs is provided when the method
00064  * {@link

```

```

00065 * GripperContribution#configureInstallation(CustomUserInputConfiguration,
00066 * SystemConfiguration, TCPConfiguration, GripperAPIProvider)} is called. If no
00067 * custom user inputs, texts, etc. are defined, the installation node will not
00068 * be visible in AuboScope. Preamble script code (generated by {@link
00069 * #generatePreambleScript(ScriptWriter)}) will still be added to the robot
00070 * program.
00071 *
00072 * The toolbar contribution provides the end user quick access to perform grip
00073 * and release actions using "default" gripper action parameters.
00074 * \endenglish
00075 */
00076 class GripperContribution
00077 {
00078 public:
00079     virtual ~GripperContribution() = default;
00080
00081     /**
00082      * \chinese
00083      *
00084      * <ul>
00085      * <li>      "      </li>
00086      * <li>      </li>
00087      * <li>      </li>
00088      * <li>      </li>
00089      * <li>      </li>
00090      * <li>      CB3      </li>
00091      * </ul>
00092      *
00093      * <p>      <code>null</code>      </p>
00094      *
00095      *
00096      * @param locale AuboScope
00097      * @return      <code>null</code>
00098      * \endchinese
00099      * \english
00100      * This method must return the title of the gripper contribution. The title
00101      * is displayed in: <ul> <li>The Structure section of the Program Tab for
00102      * this type of gripper program node (from which the Gripper program node
00103      * can be inserted)</li> <li>The program node screen for this gripper
00104      * contribution.</li> <li>The Program Tree as title for the program node for
00105      * this gripper contribution</li> <li>The left-hand side navigation of the
00106      * Installation Tab for this gripper contribution.</li> <li>The installation
00107      * node screen for this gripper contribution.</li> <li>The toolbar
00108      * contribution for this gripper. Note: The toolbar contribution is not
00109      * available on CB3 robots.</li>
00110      * </ul>
00111      *
00112      * <p>
00113      * The return value of this method cannot be <code>null</code> nor an empty
00114      * string.
00115      * </p>
00116      *
00117      * This method is called once upon startup.
00118      *
00119      * @param locale The current locale of AuboScope. Can be used for supporting
00120      * titles in several languages.
00121      * @return The title of this gripper contribution, not <code>null</code> nor
00122      * an empty string.
00123      * \endenglish
00124      */
00125     virtual std::string getTitle(/*Locale locale*/) = 0;
00126
00127     /**
00128      * \chinese
00129      * <p>      configuration      </p>
00130      * <p>
00131      * <b> </b>      "      </p>
00132      *
00133      * @param gripperConfiguration
00134      * @param apiProvider AuboScope
00135      * \endchinese
00136      * \english
00137      * <p>
00138      * When this method is called, use the configuration parameter to register
00139      * or setup optional properties and capabilities of the gripper.
00140      * </p>
00141      *
00142      * <p>
00143      * The method is called in the Gripper Configuration phase after this
00144      * contribution has been registered when a new installation is loaded or
00145      * created.
00146      * </p>
00147      *
00148      * <b>Note:</b> If the gripper only supports basic "default" grip and
00149      * release actions, leave the implementation of this method empty.
00150      *
00151      * @param gripperConfiguration A configuration instance that can be used for

```



```

00152 * registering or setting up the properties and capabilities of the gripper.
00153 * @param apiProvider Provides access to functionality and services
00154 * available from within AuboScope which can be relevant for setting up the
00155 * gripper capabilities (e.g. the {@link SystemAPI} interface with
00156 * functionality for querying information about the robot).
00157 * \endenglish
00158 */
00159 virtual void configureGripper(GripperConfigurationPtr gripperConfiguration,
00160                               GripperApiProviderPtr apiProvider) = 0;
00161
00162 /**
00163  * \chinese
00164  * <p>
00165  * <p> /
00166  * <p> </p>
00167  * <ul>
00168  * <li>
00169  * <li> </li>
00170  * <li> </li>
00171  * </ul>
00172  *
00173  * <b> </b>
00174  * <ul>
00175  * <li> <b> </b> AuboScope </li>
00176  * <li>
00177  * AuboScope
00178  * </li>
00179  * </ul>
00180  *
00181  * @param scriptWriter
00182  * @param parameters /
00183  * \endchinese
00184  * \english
00185  * <p>
00186  * When this method is called, the script code for performing a grip action
00187  * with the gripper must be generated.
00188  * </p>
00189  *
00190  * ...
00191  * @param scriptWriter Use this script writer instance to generate the
00192  * script code for gripping
00193  * @param parameters The parameters for the gripper action
00194  * defined/configured by the end user
00195  * \endenglish
00196  */
00197 virtual void generateGripActionScript(
00198     ScriptWriterPtr scriptWriter, GripActionParametersPtr parameters) = 0;
00199
00200 /**
00201  * \chinese
00202  * <p>
00203  * <p> /
00204  * <p> </p>
00205  * <ul>
00206  * <li>
00207  * <li> </li>
00208  * <li> </li>
00209  * </ul>
00210  *
00211  * <b> </b>
00212  * <ul>
00213  * <li> <b> </b> AuboScope </li>
00214  * <li>
00215  * AuboScope
00216  * </li>
00217  * </ul>
00218  *
00219  * @param scriptWriter
00220  * @param parameters /
00221  * \endchinese
00222  * \english
00223  * <p>
00224  * When this method is called, the script code for performing a release
00225  * action with the gripper must be generated.
00226  * </p>
00227  * ...
00228  * @param scriptWriter Use this script writer instance to generate the
00229  * script code for releasing the gripper
00230  * @param parameters The parameters for the gripper action
00231  * defined/configured by the end user
00232  * \endenglish
00233  */
00234 virtual void generateReleaseActionScript(
00235     ScriptWriterPtr scriptWriter,
00236     ReleaseActionParametersPtr parameters) = 0;
00237 };
00238 } // namespace aubo_scope
00239 } // namespace arcs

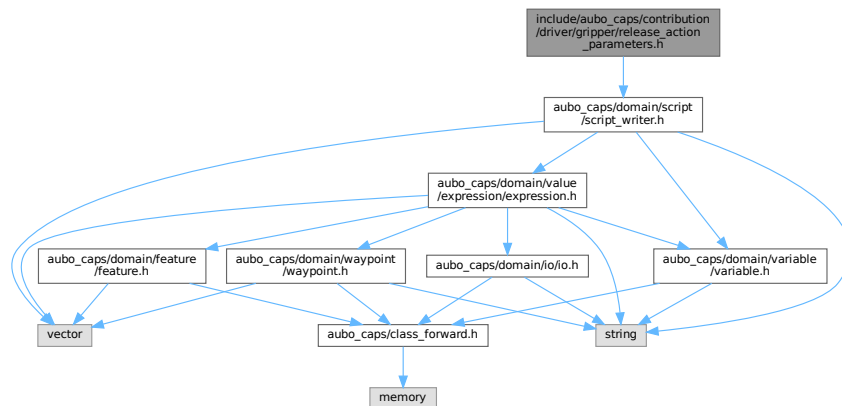
```

00239 `#endif`

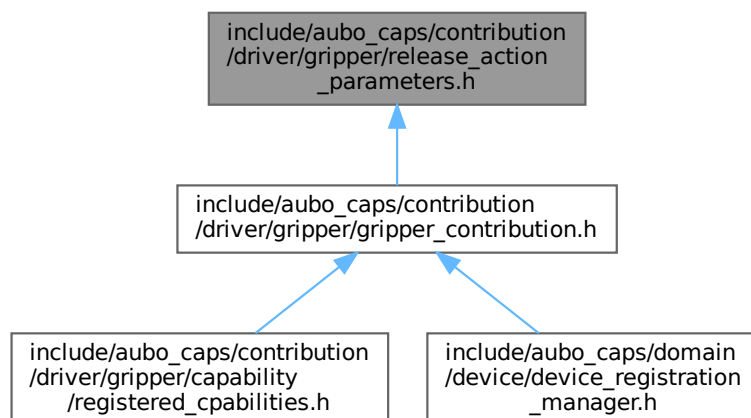
12.24 `include/aubo_caps/contribution/driver/gripper/release_action_parameters.h` File Reference

```
#include <aubo_caps/domain/script/script_writer.h>
```

Include dependency graph for `release_action_parameters.h`:



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



Classes

- class [arcs::aubo_scope::ReleaseActionParameters](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([ReleaseActionParameters](#))

12.25 release_action_parameters.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_RELEASE_ACTION_PARAMETERS_H
00002 #define AUBO_SCOPE_RELEASE_ACTION_PARAMETERS_H
00003
00004 #include <aubo_caps/domain/script/script_writer.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008
00009 ARCS_CLASS_FORWARD(ReleaseActionParameters);
00010 /**
00011  * <p>
00012  * This interface represents parameters for a release action defined/configured
00013  * by the end user.
00014  * </p>
00015  *
00016  * These parameters are used when generating script code for a release action
00017  * (see
00018  * {@link GripperContribution#generateReleaseActionScript(ScriptWriter,
00019  * ReleaseActionParameters)}).
00020  */
00021 class ReleaseActionParameters
00022 {
00023 public:
00024     virtual ~ReleaseActionParameters() = default;
00025
00026     /**
00027      * <p>
00028      * Get the selected width to be used for releasing the gripper.
00029      * </p>
00030      *
00031      * This is only valid, if the width capability has been registered (using
00032      * the method
00033      * {@link GripperCapabilities#registerWidthCapability(double, double,
00034      * double, double, Length.Unit)}).
00035      *
00036      * @return The width to use for the release action
00037      * @throws UnsupportedOperationException if the gripper has not registered
00038      * the width capability
00039      */
00040     virtual double getWidth() = 0;
00041
00042     /**
00043      * <p>
00044      * Get the selected speed to be used for releasing the gripper.
00045      * </p>
00046      *
00047      * This is only valid, if the speed capability has been registered (using
00048      * the method
00049      * {@link GripperCapabilities#registerSpeedCapability(double, double,
00050      * double, double, Speed.Unit)}).
00051      *
00052      * @return The speed to use for the release action
00053      * @throws UnsupportedOperationException if the gripper has not registered
00054      * the speed capability
00055      */
00056     virtual double getSpeed() = 0;
00057
00058     /**
00059      * <p>
00060      * Use this method to determine whether the release detection option in the
00061      * Gripper program node is on or off.
00062      * </p>
00063      *
00064      * <p>

```

```

00065 * Note that this method will report that release detection is off, when
00066 * script code is to be generated due to the user testing a release
00067 * configuration of a gripper program or operating the gripper using the
00068 * toolbar.
00069 * </p>
00070 *
00071 * <p>
00072 * This method should only be called, if the release detected feedback
00073 * capability has been registered (using
00074 * {@link
00075 * GripperFeedbackCapabilities#registerReleaseDetectedCapability(ScriptCodeGenerator)}).
00076 * </p>
00077 *
00078 * <p>
00079 * The purpose of this method is to ensure that the appropriate script code
00080 * for performing a release action is generated when the end user has
00081 * specified a new payload value. The user-defined payload value will be
00082 * applied by AuboScope immediately after the generated release action
00083 * script code has finished executing. Depending on the enablement state of
00084 * release detection, the generated script should have the following
00085 * properties: <ul> <li> <i>Release detection off:</i> The script code
00086 * should not finish earlier than when it is appropriate to apply the new
00087 * payload value (i.e. when the object has been released). This typically
00088 * means that the script code should wait a time period corresponding to
00089 * opening the gripper's "fingers" fully (if the gripper is vacuum operated
00090 * then long enough to achieve some level of vacuum).
00091 * </li>
00092 * <li>
00093 * <i>Release detection on:</i> The payload will only be set if the
00094 * release detection was successful, i.e. it will be applied immediately
00095 * after a release was detected. This means that it is not necessary for the
00096 * script code to wait (since the built-in detection timeout
00097 * functionality will handle the waiting part).
00098 * </li>
00099 * </ul>
00100 * </p>
00101 *
00102 * @return <code>true</code> if the end user has enabled release detection,
00103 * and <code>false</code> if release detection is disabled.
00104 * @throws UnsupportedOperationException if the gripper has not registered
00105 * the release detected feedback capability
00106 */
00107 virtual bool isReleaseDetectionEnabled() = 0;
00108 };
00109 } // namespace aubo_scope
00110 } // namespace arcs
00111 #endif

```

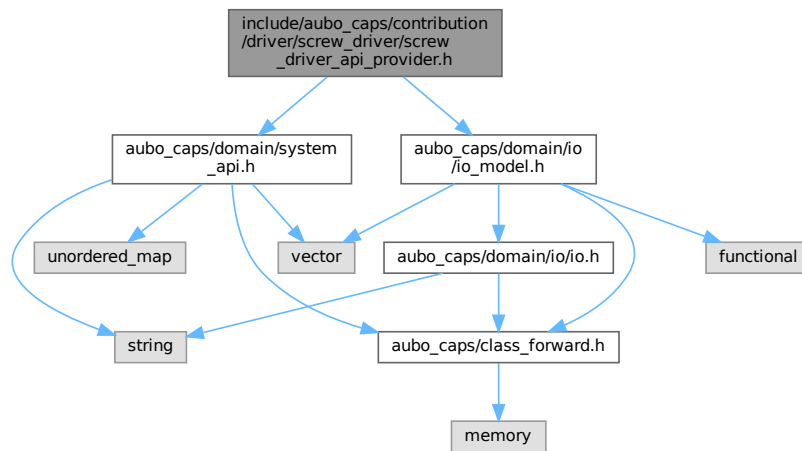
12.26 include/aubo_caps/contribution/driver/screw_driver/screw_↵ driver_api_provider.h File Reference

```

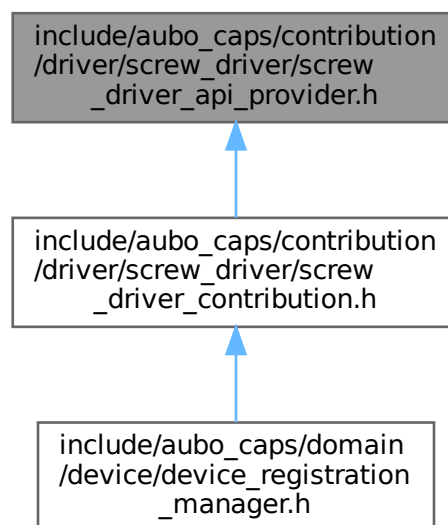
#include <aubo_caps/domain/system_api.h>
#include <aubo_caps/domain/io/io_model.h>

```

Include dependency graph for screw_driver_api_provider.h:



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



Classes

- class [arcs::aubo_scope::ScrewdriverApiProvider](#)

[ScrewdriverApiProvider](#) This interface provides access to functionality and services available from within AuboScope relevant for screwdriver contributions.

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([ScrewdriverApiProvider](#))

12.27 screw_driver_api_provider.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_SCREW_DRIVER_API_PROVIDER_H
00002 #define AUBO_SCOPE_SCREW_DRIVER_API_PROVIDER_H
00003
00004 #include <aubo_caps/domain/system_api.h>
00005 #include <aubo_caps/domain/io/io_model.h>
00006
00007 namespace arcs {
00008     namespace aubo_scope {
00009         ARCS_CLASS_FORWARD(ScrewdriverApiProvider);
00010
00011         /**
00012          * \chinese
00013          *   API
00014          *   AuboScope
00015          * \endchinese
00016          * \english
00017          * ScrewdriverApiProvider
00018          * This interface provides access to functionality and services available from
00019          * within AuboScope relevant for screwdriver contributions.
00020          * \endenglish
00021          */
00022         class ScrewdriverApiProvider
00023         {
00024         public:
00025             virtual ~ScrewdriverApiProvider() = default;
00026
00027             /**
00028              * \chinese
00029              * @return
00030              * \endchinese
00031              * \english
00032              * @return an interface providing access to system related functionality.
00033              * \endenglish
00034              */
00035             virtual SystemApiPtr getSystemAPI() = 0;
00036
00037             /**
00038              * \chinese
00039              * @return I/O
00040              * \endchinese
00041              * \english
00042              * @return an I/O model interface (e.g. for getting I/Os to display in the
00043              * custom user input for setting up the screwdriver)
00044              * \endenglish
00045              */
00046             virtual IoModelPtr getIOModel() = 0;
00047         };
00048     } // namespace aubo_scope
00049 } // namespace arcs
00050 #endif

```

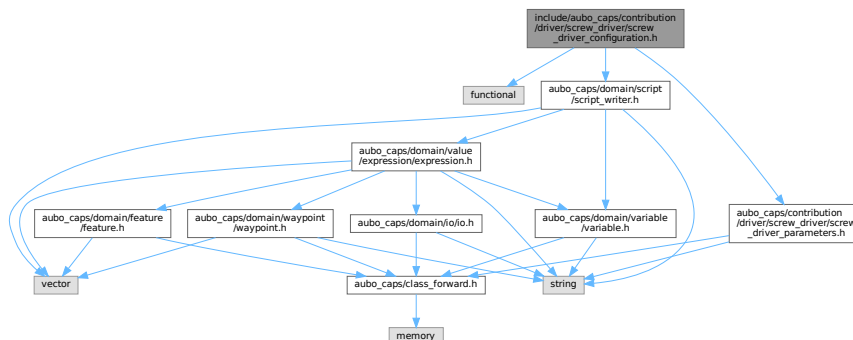
12.28 include/aubo_caps/contribution/driver/screw_driver/screw_driver_configuration.h File Reference

```

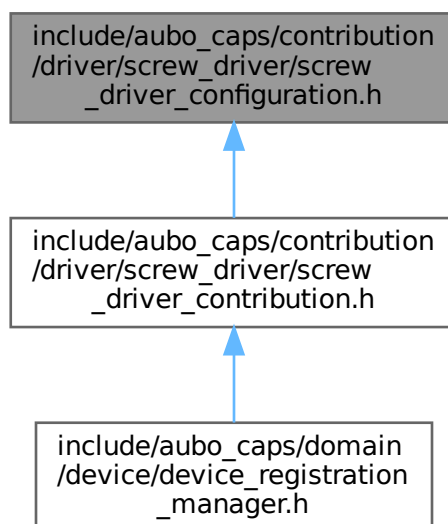
#include <functional>
#include <aubo_caps/domain/script/script_writer.h>

```

#include <aubo_caps/contribution/driver/screw_driver/screw_driver_parameters.h>
 Include dependency graph for screw_driver_configuration.h:



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



Classes

- class [arcs::aubo_scope::ProgramSelectionParameters](#)
- class [arcs::aubo_scope::PrepareToStartScrewdriverParameters](#)
- class [arcs::aubo_scope::FeedScrewParameters](#)
- class [arcs::aubo_scope::DriveScrewNotOKParameters](#)
- class [arcs::aubo_scope::DriveScrewOKParameters](#)
- class [arcs::aubo_scope::ScrewdriverReadyParameters](#)

This interface provides the parameters relevant for generating script for the Screwdriver Ready feedback operation capability (registered using `ScrewdriverFeedbackCapabilities#registerScrewdriverReadyCapability(ScriptCodeC`

- class [arcs::aubo_scope::ScrewdriverConfiguration](#)

This interface provides access to register or setup various properties and capabilities of a screwdriver.

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) (ProgramSelectionParameters)
- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) (PrepareToStartScrewdriverParameters)
- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) (FeedScrewParameters)
- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) (DriveScrewNotOKParameters)
- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) (DriveScrewOKParameters)
- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) (ScrewdriverReadyParameters)
- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) (ScrewdriverConfiguration)

12.29 [screw_driver_configuration.h](#)

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_SCREW_DRIVER_CONFIGURATION_H
00002 #define AUBO_SCOPE_SCREW_DRIVER_CONFIGURATION_H
00003
00004 #include <functional>
00005 #include <aubo_caps/domain/script/script_writer.h>
00006 #include <aubo_caps/contribution/driver/screw_driver/screw_driver_parameters.h>
00007
00008 namespace arcs {
00009 namespace aubo_scope {
00010
00011 ARCS_CLASS_FORWARD(ProgramSelectionParameters);
00012 /**
00013  * <p>
00014  * This interface provides parameters relevant for generating script code for
00015  * the Program Selection capability (registered using {@link
00016  * ScrewdriverCapabilities#registerProgramSelectionCapability(ScrewdriverProgramListProvider,
00017  * ScriptCodeGenerator)}).
00018  * </p>
00019  *
00020  * These parameters are passed when AuboScope calls the implementation of the
00021  * {@link ScriptCodeGenerator#generateScript(ScriptWriter, Object)} method
00022  * responsible for the generating the script code.
00023  */
00024 class ProgramSelectionParameters
00025 {
00026 public:
00027     virtual ~ProgramSelectionParameters() = default;
00028
00029     /**
00030      * @return The selected screwdriver program to use for the operation.
00031      * @throws UnsupportedOperationException if the screwdriver has not
00032      * registered this capability
00033      */
00034     virtual ScrewdriverProgram getScrewdriverProgram() = 0;
00035
00036     /**
00037      * @return The operation type selected by the end user.
00038      * @throws UnsupportedOperationException if the screwdriver has not
00039      * registered this capability
00040      */
00041     virtual ScrewdriverOperationType getScrewdriverOperationType() = 0;
00042 };
00043
00044 ARCS_CLASS_FORWARD(PrepareToStartScrewdriverParameters);
00045 /**
00046  * <p>
00047  * This interface provides parameters relevant for generating script code for
00048  * the Prepare To Start Screwdriver capability (registered using {@link
00049  * ScrewdriverCapabilities#registerPrepareToStartScrewdriverCapability(ScriptCodeGenerator)}).
00050  * </p>
00051  *
00052  * <p>
00053  * These parameters are passed when AuboScope calls the implementation of the
00054  * {@link ScriptCodeGenerator#generateScript(ScriptWriter, Object)} method

```



```

00055 * responsible for the generating the script code.
00056 * </p>
00057 *
00058 * The interface is intentionally empty, but could have parameters in future
00059 * releases.
00060 */
00061 class PrepareToStartScrewdriverParameters
00062 {
00063 public:
00064     virtual ~PrepareToStartScrewdriverParameters() = default;
00065 };
00066
00067 ARCS_CLASS_FORWARD(FeedScrewParameters);
00068 /**
00069 * <p>
00070 * This interface provides parameters relevant for generating script code for
00071 * the Feed Screw capability (registered using {@link
00072 * ScrewdriverCapabilities#registerFeedScrewCapability(ScriptCodeGenerator)}).
00073 * </p>
00074 *
00075 * <p>
00076 * These parameters are passed when AuboScope calls the implementation of the
00077 * {@link ScriptCodeGenerator#generateScript(ScriptWriter, Object)} method
00078 * responsible for the generating the script code.
00079 * </p>
00080 *
00081 * The interface is intentionally empty, but could have parameters in future
00082 * releases.
00083 */
00084 class FeedScrewParameters
00085 {
00086 public:
00087     virtual ~FeedScrewParameters() = default;
00088 };
00089
00090 ARCS_CLASS_FORWARD(DriveScrewNotOKParameters);
00091 /**
00092 * <p>
00093 * This interface provides parameters relevant for generating script code for
00094 * the Drive Screw Not OK feedback capability (registered using {@link
00095 * ScrewdriverFeedbackCapabilities#registerDriveScrewNotOKCapability(ScriptCodeGenerator)}).
00096 * </p>
00097 *
00098 * <p>
00099 * These parameters are passed when AuboScope calls the implementation of the
00100 * {@link ScriptCodeGenerator#generateScript(ScriptWriter, Object)} method
00101 * responsible for the generating the script code.
00102 * </p>
00103 *
00104 * The interface is intentionally empty, but could have parameters in future
00105 * releases.
00106 */
00107 class DriveScrewNotOKParameters
00108 {
00109 public:
00110     virtual ~DriveScrewNotOKParameters() = default;
00111 };
00112
00113 ARCS_CLASS_FORWARD(DriveScrewOKParameters);
00114 /**
00115 * <p>
00116 * This interface provides parameters relevant for generating script code for
00117 * the Drive Screw OK feedback capability (registered using {@link
00118 * ScrewdriverFeedbackCapabilities#registerDriveScrewOKCapability(ScriptCodeGenerator)}).
00119 * </p>
00120 *
00121 * <p>
00122 * These parameters are passed when AuboScope calls the implementation of the
00123 * {@link ScriptCodeGenerator#generateScript(ScriptWriter, Object)} method
00124 * responsible for the generating the script code.
00125 * </p>
00126 *
00127 * The interface is intentionally empty, but could have parameters in future
00128 * releases.
00129 */
00130 class DriveScrewOKParameters
00131 {
00132 public:
00133     virtual ~DriveScrewOKParameters() = default;
00134 };
00135
00136 ARCS_CLASS_FORWARD(ScrewdriverReadyParameters);
00137 /**
00138 * This interface provides the parameters relevant for generating script for the
00139 * Screwdriver Ready feedback operation capability (registered using {@link
00140 * ScrewdriverFeedbackCapabilities#registerScrewdriverReadyCapability(ScriptCodeGenerator)}).
00141 *

```

```

00142 * <p>
00143 * These parameters are passed when AuboScope calls the implementation of the
00144 * {@link ScriptCodeGenerator#generateScript(ScriptWriter, Object)} method
00145 * responsible for the generating the script code.
00146 * </p>
00147 *
00148 * The interface is intentionally empty, but could have parameters in future
00149 * releases.
00150 */
00151 class ScrewdriverReadyParameters
00152 {
00153 public:
00154     virtual ~ScrewdriverReadyParameters() = default;
00155 };
00156
00157 ARCS_CLASS_FORWARD(ScrewdriverConfiguration);
00158 /**
00159 * This interface provides access to register or setup various properties and
00160 * capabilities of a screwdriver. It is optional to register these capabilities
00161 * and properties.
00162 */
00163 class ScrewdriverConfiguration
00164 {
00165 public:
00166     virtual ~ScrewdriverConfiguration() = default;
00167
00168     /**
00169      * Add a screwdriver program to the list of programs.
00170      *
00171      * @param screwdriverProgram program to add, not <code>null</code>
00172      * @throws DuplicateScrewdriverProgramID if the ids of the screwdriver
00173      * programs (returned by the method {@link ScrewdriverProgram#getId()})
00174      * are not unique.
00175      * @throws NameOfScrewdriverProgramCannotBeNullOrEmpty if the display name
00176      * of a program (returned by the method {@link
00177      * ScrewdriverProgram#getDisplayName()}) is <code>null</code> or an empty
00178      * string.
00179      */
00180     virtual void addScrewdriverProgram(
00181         ScrewdriverProgramPtr screwdriverProgram) = 0;
00182
00183     /**
00184      * Add a list of screwdriver programs to the list of programs.
00185      *
00186      * @param screwdriverProgramList list of programs to add, not
00187      * <code>null</code>
00188      * @throws DuplicateScrewdriverProgramID if the ids of the screwdriver
00189      * programs (returned by the method
00190      * {@link ScrewdriverProgram#getId()})
00191      * are not unique.
00192      * @throws NameOfScrewdriverProgramCannotBeNullOrEmpty if the display name
00193      * of a program (returned by the method
00194      * {@link
00195      * ScrewdriverProgram#getDisplayName()}) is <code>null</code> or an empty
00196      * string.
00197      */
00198     virtual void addScrewdriverPrograms(
00199         std::vector<ScrewdriverProgramPtr> screwdriverProgramList) = 0;
00200
00201     /**
00202      * <p>
00203      * Register a program selection capability for a screwdriver that supports
00204      * selecting between a set of screwdriver programs, typically defined on an
00205      * external control box for the screwdriver.
00206      * </p>
00207      *
00208      * <p>
00209      * The provided implementation of the {@link ScriptCodeGenerator} interface
00210      * must generate the script code for selecting the specified screwdriver
00211      * program. The return value of the script code must be a boolean, i.e.
00212      * 'True' or 'False'. 'True' must be returned if selecting the program
00213      * succeeded and 'False' must be returned if the program selection failed.
00214      * </p>
00215      *
00216      * The generated script code is executed before the script code for starting
00217      * the screwdriver (could be from a different node), i.e the script code
00218      * generated by
00219      * {@link
00220      * ScrewdriverContribution#generateStartScrewdriverScript(ScriptWriter,
00221      * ScrewdriverParameters)}.
00222      *
00223      * @param programListProvider provider of the list of screwdriver programs,
00224      * not <code>null</code>
00225      * @param scriptCodeGenerator script code generator responsible for
00226      * generating the script code for selecting the specified screwdriver
00227      * program. When the script code needs to be generated, the
00228      * {@link

```

```

00229 * ScriptCodeGenerator#generateScript(ScriptWriter, Object)} method will be
00230 * called (by AuboScope). Return value for the generated script code must be
00231 * a boolean, i.e. 'True' or 'False'.
00232 * @throws CapabilityAlreadyRegistered if this capability has already been
00233 * registered
00234 * @throws CalledOutsideScrewdriverConfigurationPhase if this method is
00235 * called at the wrong time, i.e. outside the scope of the {@link
00236 * ScrewdriverContribution#configureScrewdriver(ScrewdriverConfiguration,
00237 * ScrewdriverApiProvider)} method.
00238 */
00239 virtual void registerProgramSelectionCapability(
00240     std::function<void(ScriptWriterPtr, ProgramSelectionParametersPtr)>
00241     scriptCodeGenerator) = 0;
00242
00243 /**
00244 * Register an operation type capability for a screwdriver that
00245 * supports/requires the parameter for the screwdriving operation type (i.e.
00246 * if the operation is Tighten or Loosen).
00247 *
00248 * @throws CapabilityAlreadyRegistered if this capability has already been
00249 * registered
00250 * @throws CalledOutsideScrewdriverConfigurationPhase if this method is
00251 * called at the wrong time, i.e. outside the scope of the {@link
00252 * ScrewdriverContribution#configureScrewdriver(ScrewdriverConfiguration,
00253 * ScrewdriverApiProvider)} method.
00254 */
00255 virtual void registerOperationTypeCapability() = 0;
00256
00257 /**
00258 * <p>
00259 * Register a capability/requirement for a screwdriver that needs to
00260 * generate script code for preparing the screwdriver for a screwdriving
00261 * operation (before starting the screw driver).
00262 * </p>
00263 *
00264 * <p>
00265 * The provided implementation of the {@link ScriptCodeGenerator} interface
00266 * must generate the script code for preparing the screwdriver. The return
00267 * value of the script code must be a boolean, i.e. 'True' or 'False'.
00268 * 'True' must be returned if the preparation of the screwdriver was
00269 * successful and 'False' must be returned if the preparation failed.
00270 * </p>
00271 *
00272 * The generated script code is executed before the script code for starting
00273 * the screwdriver (could be from a different node), i.e the script code
00274 * generated by
00275 * {@link
00276 * ScrewdriverContribution#generateStartScrewdriverScript(ScriptWriter,
00277 * ScrewdriverParameters)}.
00278 *
00279 * @param scriptCodeGenerator script code generator responsible for
00280 * generating the script code for preparing the screwdriver. When the script
00281 * code needs to be generated, the
00282 * {@link
00283 * ScriptCodeGenerator#generateScript(ScriptWriter, Object)} method will be
00284 * called (by AuboScope). Return value for the generated script code must be
00285 * a boolean, i.e. 'True' or 'False'.
00286 * @throws CapabilityAlreadyRegistered if this capability has already been
00287 * registered
00288 * @throws CalledOutsideScrewdriverConfigurationPhase if this method is
00289 * called at the wrong time, i.e. outside the scope of the {@link
00290 * ScrewdriverContribution#configureScrewdriver(ScrewdriverConfiguration,
00291 * ScrewdriverApiProvider)} method.
00292 */
00293 virtual void registerPrepareToStartScrewdriverCapability(
00294     std::function<void(ScriptWriterPtr,
00295         PrepareToStartScrewdriverParametersPtr)>
00296     scriptCodeGenerator) = 0;
00297
00298 /**
00299 * <p>
00300 * Register a capability for a screwdriver that supports automatic screw
00301 * feeding.
00302 * </p>
00303 *
00304 * <p>
00305 * The provided implementation of the {@link ScriptCodeGenerator} interface
00306 * must generate the script code for feeding a screw to the screwdriver. The
00307 * return value of the script code must be a boolean, i.e. 'True' or
00308 * 'False'. 'True' must be returned if the screw feed operation was
00309 * successful and 'False' must be returned if the operation failed.
00310 * </p>
00311 *
00312 * The generated script code is executed before the script code for
00313 * preparing the screwdriver, i.e the script code generated by {@link
00314 * #registerPrepareToStartScrewdriverCapability(ScriptCodeGenerator)}.
00315 */

```

```

00316 * @param scriptCodeGenerator script code generator responsible for
00317 * generating the script code for feeding a screw to the screwdriver. When
00318 * the script code needs to be generated, the
00319 * {@link
00320 * ScriptCodeGenerator#generateScript(ScriptWriter, Object)} method will be
00321 * called (by AuboScope). Return value for the generated script code must be
00322 * a boolean, i.e. 'True' or 'False'
00323 * @throws CapabilityAlreadyRegistered if this capability has already been
00324 * registered
00325 * @throws CalledOutsideScrewdriverConfigurationPhase if this method is
00326 * called at the wrong time, i.e. outside the scope of the {@link
00327 * ScrewdriverContribution#configureScrewdriver(ScrewdriverConfiguration,
00328 * ScrewdriverApiProvider)} method.
00329 */
00330 virtual void registerFeedScrewCapability(
00331     std::function<void(ScriptWriterPtr, FeedScrewParametersPtr)>
00332     scriptCodeGenerator) = 0;
00333
00334 /**
00335 * <p>
00336 * Register a feedback capability for a screwdriver that is capable of
00337 * providing information about whether or not an ongoing screwdriving
00338 * operation (driving the screw) ended successfully (ended in OK-state).
00339 * </p>
00340 *
00341 * <p>
00342 * <b>Note:</b> This capability must only be used to provide feedback about
00343 * the success of the screwing operation. It should not be used to provide
00344 * feedback about failures. To support feedback about when the operation
00345 * fails, register the Drive Screw Not Ok feedback capability the using the
00346 * {@link #registerDriveScrewNotOKCapability(ScriptCodeGenerator)} method.
00347 * </p>
00348 *
00349 * <p>
00350 * The provided implementation of the {@link ScriptCodeGenerator} interface
00351 * must generate the script code for determining if the screwing operation
00352 * succeeded. The return value of the script code must be a boolean, i.e.
00353 * 'True' or 'False'. 'True' must be returned if the screwing operation was
00354 * successful, otherwise 'False' must be returned.
00355 * </p>
00356 *
00357 * <p>
00358 * <b>Note:</b> A return value of 'False' must not be used to indicate that
00359 * the operation failed, but simply that a success was not (yet) detected.
00360 * If the screwdriver can provide feedback on whether or not the screwing
00361 * operation failed, the Drive Screw Not Ok feedback capability must be
00362 * used.
00363 * </p>
00364 *
00365 * <b>Example:</b> <br>
00366 * The digital tool input 0 will go high if the operation ended in OK-state.
00367 * The script code is: <pre> return get_tool_digital_in(0)
00368 * </pre>
00369 *
00370 * So the implementation of {@link
00371 * ScriptCodeGenerator#generateScript(ScriptWriter, Object)} could be: <pre>
00372 * scriptWriter.appendLine("return get_tool_digital_in(0)");
00373 * </pre>
00374 *
00375 * @param scriptCodeGenerator script code generator responsible for
00376 * generating the script code for determining if the screwing operation
00377 * succeeded. When the script code needs to be generated, the
00378 * {@link
00379 * ScriptCodeGenerator#generateScript(ScriptWriter, Object)} method will be
00380 * called (by AuboScope). Return value for the
00381 * generated script code must be a boolean, i.e. 'True' or 'False'.
00382 * @throws CapabilityAlreadyRegistered if this capability has already been
00383 * registered
00384 * @throws CalledOutsideScrewdriverConfigurationPhase if this method is
00385 * called at the wrong time, i.e. outside the scope of the {@link
00386 * ScrewdriverContribution#configureScrewdriver(ScrewdriverConfiguration,
00387 * ScrewdriverApiProvider)} method.
00388 */
00389 virtual void registerDriveScrewOKCapability(
00390     std::function<void(ScriptWriterPtr, DriveScrewOKParametersPtr)>
00391     scriptCodeGenerator) = 0;
00392
00393 /**
00394 * <p>
00395 * Register a feedback capability for a screwdriver that is capable of
00396 * providing information about whether or not an ongoing screwdriving
00397 * operation (driving the screw) failed, i.e. ended in Not Ok (NOK) state.
00398 * </p>
00399 *
00400 * <p>
00401 * <b>Note:</b> This capability must only be used to provide feedback about
00402 * the failure of the screwing operation. It should not be used to provide

```

```

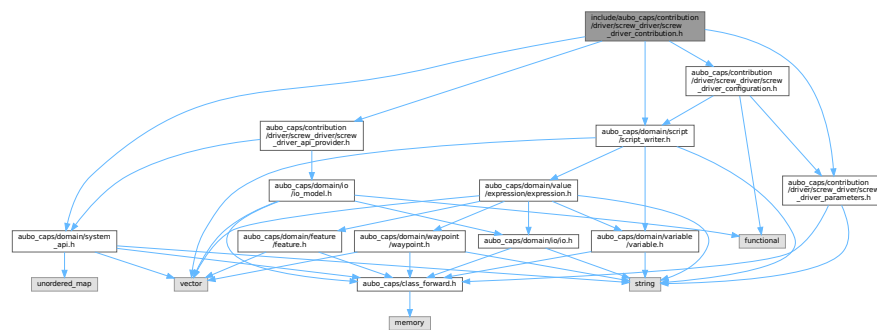
00403 * feedback about successes. To support feedback about when the operation
00404 * succeeds, register the Drive Screw Ok feedback capability the using the
00405 * {@link #registerDriveScrewOKCapability(ScriptCodeGenerator)} method.
00406 * </p>
00407 *
00408 * <p>
00409 * The provided implementation of the {@link ScriptCodeGenerator} interface
00410 * must generate the script code for determining if the screwing operation
00411 * ended in failure. The return value of the script code must be a boolean,
00412 * i.e. 'True' or 'False'. 'True' must be returned if the screwing operation
00413 * failed, otherwise 'False' must be returned.
00414 * </p>
00415 *
00416 * <p>
00417 * <b>Note:</b> A return value of 'False' must not be used to indicate that
00418 * the operation succeeded, but simply that a failure was not (yet)
00419 * detected. If the screwdriver can provide feedback on whether or not the
00420 * screwing operation succeeded, the Drive Screw Ok feedback capability
00421 * should be used.
00422 * </p>
00423 *
00424 * <b>Example:</b> <br>
00425 * The digital tool input 1 will go high if the operation ended in NOT
00426 * OK-state. The script code is: <pre> return get_tool_digital_in(1)
00427 * </pre>
00428 * So the implementation of {@link
00429 * ScriptCodeGenerator#generateScript(ScriptWriter, Object)} could be: <pre>
00430 *   scriptWriter.appendLine("return get_tool_digital_in(1)");
00431 * </pre>
00432 *
00433 *
00434 * @param scriptCodeGenerator script code generator responsible for
00435 * generating the script code for determining if the screwing operation
00436 * failed. When the script code needs to be generated, the {@link
00437 * ScriptCodeGenerator#generateScript(ScriptWriter, Object)} method will be
00438 * called (by AuboScope). Return value for the
00439 * generated script code must be a boolean, i.e. 'True' or 'False'.
00440 * @throws CapabilityAlreadyRegistered if this capability has already been
00441 * registered
00442 * @throws CalledOutsideScrewdriverConfigurationPhase if this method is
00443 * called at the wrong time, i.e. outside the scope of the {@link
00444 * ScrewdriverContribution#configureScrewdriver(ScrewdriverConfiguration,
00445 * ScrewdriverApiProvider)} method.
00446 */
00447 virtual void registerDriveScrewNotOKCapability(
00448     std::function<void(ScriptWriterPtr, DriveScrewNotOKParametersPtr)>
00449     scriptCodeGenerator) = 0;
00450
00451 /**
00452 * <p>
00453 * Register a feedback capability for a screwdriver that is capable of
00454 * providing information about whether or not the screwdriver device is
00455 * ready to operate (receive commands). This capability is used (by
00456 * AuboScope) in a program to ensure that the screwdriver is ready before
00457 * sending commands to it.
00458 * </p>
00459 *
00460 * <p>
00461 * The provided implementation of the {@link ScriptCodeGenerator} interface
00462 * must generate the script code for determining if the screwdriver is
00463 * ready. The return value of the script code must be a boolean, i.e. 'True'
00464 * or 'False'. 'True' must be returned if the screwdriver is ready,
00465 * otherwise 'False' must be returned.
00466 * </p>
00467 *
00468 * The generated script code is executed before the script code for the
00469 * program selected capability (if that capability has been registered).
00470 *
00471 * @param scriptCodeGenerator script code generator responsible for
00472 * generating the script code for determining if the screwdriver is ready.
00473 * When the script code needs to be generated, the
00474 * {@link
00475 * ScriptCodeGenerator#generateScript(ScriptWriter, Object)} method will be
00476 * called (by AuboScope). Return value for the
00477 * generated script code must be a boolean, i.e. 'True' or 'False'.
00478 * @throws CapabilityAlreadyRegistered if this capability has already been
00479 * registered
00480 * @throws CalledOutsideScrewdriverConfigurationPhase if this method is
00481 * called at the wrong time, i.e. outside the scope of the {@link
00482 * ScrewdriverContribution#configureScrewdriver(ScrewdriverConfiguration,
00483 * ScrewdriverApiProvider)} method.
00484 */
00485 virtual void registerScrewdriverReadyCapability(
00486     std::function<void(ScriptWriterPtr, ScrewdriverReadyParametersPtr)>
00487     scriptCodeGenerator) = 0;
00488 };
00489 } // namespace aubo_scope

```

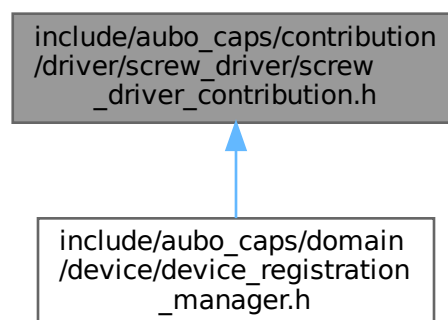
```
00490 } // namespace arcs
00491
00492 #endif
```

12.30 include/aubo_caps/contribution/driver/screw_driver/screw_driver_contribution.h File Reference

```
#include <aubo_caps/domain/system_api.h>
#include <aubo_caps/domain/script/script_writer.h>
#include <aubo_caps/contribution/driver/screw_driver/screw_driver_configuration.h>
#include <aubo_caps/contribution/driver/screw_driver/screw_driver_parameters.h>
#include <aubo_caps/contribution/driver/screw_driver/screw_driver_api_provider.h>
Include dependency graph for screw_driver_contribution.h:
```



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



Classes

- class [arcs::aubo_scope::ScrewdriverContribution](#)
[ScrewdriverContribution](#) This interface defines an API for adding functionality for screwdrivers to AuboScope.

Namespaces

- namespace `arcs`
- namespace `arcs::aubo_scope`

Functions

- `arcs::aubo_scope::ARCS_CLASS_FORWARD` (`ScrewdriverContribution`)

12.31 screw_driver_contribution.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_SCREW_DRIVER_CONTRIBUTION_H
00002 #define AUBO_SCOPE_SCREW_DRIVER_CONTRIBUTION_H
00003
00004 #include <aubo_caps/domain/system_api.h>
00005 #include <aubo_caps/domain/script/script_writer.h>
00006 #include <aubo_caps/contribution/driver/screw_driver/screw_driver_configuration.h>
00007 #include <aubo_caps/contribution/driver/screw_driver/screw_driver_parameters.h>
00008 #include <aubo_caps/contribution/driver/screw_driver/screw_driver_api_provider.h>
00009
00010 namespace arcs {
00011     namespace aubo_scope {
00012         ARCS_CLASS_FORWARD(ScrewdriverContribution);
00013     }
00014     /**
00015      * \chinese
00016      *
00017      *      AuboScope      API
00018      *
00019      *      " "      AuboScope
00020      *
00021      *
00022      * <b> </b>      AuboScope      CB3
00023      *      AuboCap      CB3      AuboCap
00024      *
00025      *
00026      *      " "
00027      *
00028      * {@link ScrewdriverContribution#configureScrewdriver(ScrewdriverConfiguration,
00029      * ScrewdriverApiProvider)}
00030      *
00031      *
00032      *      IP
00033      *      UI
00034      *      UI
00035      * {@link ScrewdriverContribution#configureInstallation(CustomUserInputConfiguration,
00036      * SystemConfiguration, TCPConfiguration, ScrewdriverApiProvider)}
00037      *
00038      * \endchinese
00039      * \english
00040      * ScrewdriverContribution
00041      * This interface defines an API for adding functionality for screwdrivers to
00042      * AuboScope.
00043      *
00044      * Implementing this interface will "integrate" (hook) this screwdriving
00045      * contribution into the builtin (native) AuboScope Screwdriving program node
00046      * and Screwdriving installation screen. This will expose the screwdriving
00047      * functionality supported by the screwdriver to the end user.
00048      *
00049      * <b>Note:</b> Both the builtin (native) AuboScope Screwdriving program node
00050      * and Screwdriver contributions are not supported on CB3 robots. If a AuboCap
00051      * registers a Screwdriver contribution on a CB3 robot, the AuboCap will fail to
00052      * be activated.
00053      *
00054      * The Screwdriving program node allows the end user to program screwdriving
00055      * operations. The configuration options available to the end user are based on
00056      * the properties and capabilities supported by the screwdriver. As a minimum,
00057      * all screwdrivers must support "default" screw (start screwdriver) and stop
00058      * (stop screwdriver) operations. All other capabilities are optional to support
00059      * and can be registered when the method
00060      * {@link ScrewdriverContribution#configureScrewdriver(ScrewdriverConfiguration,
00061      * ScrewdriverApiProvider)} is called.
00062      *
00063      * The integration in the Screwdriving installation screen allows the user to
00064      * select the screwdriver for programming screwing operations (using the

```



```

00065 * Screwdriving program node). It also offers the possibility to define any
00066 * custom inputs from the end user that are required to setup the screwdriver,
00067 * e.g. specifying an IP address. This functionality is optional, since such
00068 * configuration properties are specific for each screwdriver device. The
00069 * corresponding generated custom UI will be accessible in the screen when the
00070 * screwdriver is selected (by the end user) and replace the default
00071 * Screwdriving installation UI. Access to create a custom configuration based
00072 * on user inputs is provided when the method
00073 * {@link
00074 * ScrewdriverContribution#configureInstallation(CustomUserInputConfiguration,
00075 * SystemConfiguration, TCPConfiguration, ScrewdriverApiProvider)} is called.
00076 * \endenglish
00077 */
00078 class ScrewdriverContribution
00079 {
00080 public:
00081     virtual ~ScrewdriverContribution() = default;
00082
00083     /**
00084      * \chinese
00085      *
00086      * <ul>
00087      * <li> AuboScope </li>
00088      * <li> AuboScope
00089      * </li>
00090      * </ul>
00091      *
00092      *
00093      * @param locale AuboScope
00094      * @return <code>null</code>
00095      * \endchinese
00096      * \english
00097      * This method must return the title of the screwdriver contribution. The
00098      * title is displayed in: <ul> <li>The builtin (native) AuboScope
00099      * Screwdriving installation screen at the top where the screwdriver is
00100      * selected
00101      * </li>
00102      * <li>The screen for the builtin (native) AuboScope Screwdriving
00103      * program node (if this screwdriver has been selected in the Screwdriving
00104      * installation screen)
00105      * </li>
00106      * </ul>
00107      *
00108      * This method is called once upon startup.
00109      *
00110      * @param locale the current locale of AuboScope. Can be used for supporting
00111      * titles in several languages.
00112      * @return the title of this screw driver contribution, not
00113      * <code>null</code> nor an empty string.
00114      * \endenglish
00115      */
00116     virtual std::string getTitle(/*Locale locale*/) = 0;
00117
00118     /**
00119      * \chinese
00120      * <p> configuration </p>
00121      * <p>
00122      * <b> </b> " " </p>
00123      *
00124      * @param screwdriverConfiguration
00125      * @param apiProvider AuboScope
00126      * \endchinese
00127      * \english
00128      * <p>
00129      * When this method is called, use the configuration parameter to register
00130      * or setup optional properties and capabilities of the screwdriver.
00131      * </p>
00132      * ...
00133      * @param screwdriverConfiguration a configuration instance that can be used
00134      * for registering or setting up the properties and capabilities of the
00135      * screwdriver.
00136      * @param apiProvider provides access to functionality and services
00137      * available from within AuboScope which can be relevant for setting up the
00138      * screwdriver capabilities (e.g. the {@link SystemAPI} interface with
00139      * functionality for querying information about the robot).
00140      * \endenglish
00141      */
00142     virtual void configureScrewdriver(
00143         ScrewdriverConfiguration screwdriverConfiguration,
00144         ScrewdriverAPIProviderPtr apiProvider);
00145
00146     /**
00147      * \chinese
00148      * <p> </p>
00149      * / /
00150      *
00151      * @param scriptWriter

```



```

00152     * @param parameters    /
00153     * \endchinese
00154     * \english
00155     * <p>
00156     * When this method is called, the script code for driving the screw must be
00157     * generated.
00158     * </p>
00159     * The relevant parameters for the registered optional screwdriver
00160     * capabilities/requirements defined/configured by the end user are provided
00161     * as input to the script code generation.
00162     *
00163     * @param scriptWriter use this script writer instance to generate the
00164     * script code for driving the screw
00165     * @param parameters the parameters for the screwdriving operation
00166     * defined/configured by the end user
00167     * \endenglish
00168     */
00169 virtual void generateStartScrewdriverScript(
00170     ScriptWriterPtr scriptWriter, ScrewdriverParametersPtr parameters) = 0;
00171
00172 /**
00173     * \chinese
00174     * <p>
00175     * </p>
00176     * /
00177     *
00178     * @param scriptWriter
00179     * @param parameters    /
00180     * \endchinese
00181     * \english
00182     * <p>
00183     * When this method is called, the script code for stopping the screw driver
00184     * must be generated (i.e. the screwdriver must stop the current
00185     * screwdriving operation).
00186     * </p>
00187     * The relevant parameters for the registered optional screwdriver
00188     * capabilities/requirements defined/configured by the end user are provided
00189     * as input to the script code generation.
00190     *
00191     * @param scriptWriter use this script writer instance to generate the
00192     * script code for stopping the screwdriver
00193     * @param parameters the parameters for the screwdriving operation
00194     * defined/configured by the end user
00195     * \endenglish
00196     */
00197 virtual void generateStopScrewdriverScript(
00198     ScriptWriterPtr scriptWriter, ScrewdriverParametersPtr parameters) = 0;
00199 ;
00200 };
00201 } // namespace aubo_scope
00202 } // namespace arcs
00203
00204 #endif

```

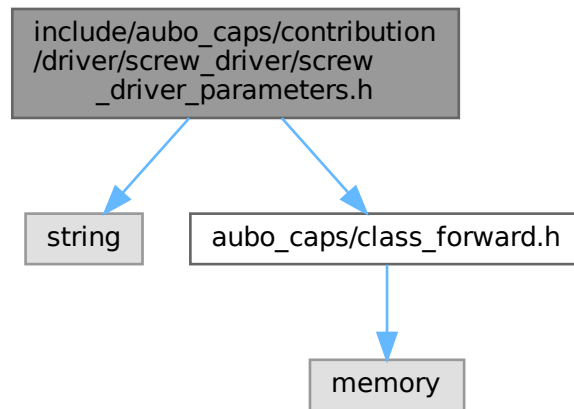
12.32 include/aubo_caps/contribution/driver/screw_driver/screw_driver_parameters.h File Reference

```

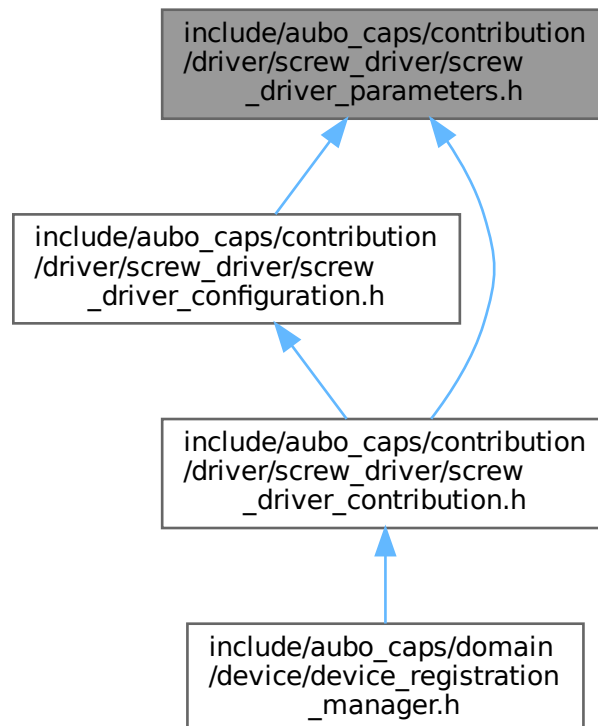
#include <string>
#include <aubo_caps/class_forward.h>

```

Include dependency graph for `screw_driver_parameters.h`:



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



Classes

- class [arcs::aubo_scope::ScrewdriverProgram](#)
This abstract class represents a screwdriver program, typically defined on an external control box for the screwdriver.
- class [arcs::aubo_scope::ScrewdriverParameters](#)
This interface represents parameters for a screwdriver operation defined/configured by the end user.

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Enumerations

- enum [arcs::aubo_scope::ScrewdriverOperationType](#) { [arcs::aubo_scope::TIGHTEN](#) , [arcs::aubo_scope::LOOSEN](#) }
- This interface represents the type of screwing operation performed by the screwdriver.

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([ScrewdriverProgram](#))
- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([ScrewdriverParameters](#))

12.33 screw_driver_parameters.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_SCREW_DIRVER_PARAMETERS_H
00002 #define AUBO_SCOPE_SCREW_DIRVER_PARAMETERS_H
00003
00004 #include <string>
00005 #include <aubo_caps/class_forward.h>
00006
00007 namespace arcs {
00008 namespace aubo_scope {
00009
00010 ARCS_CLASS_FORWARD(ScrewdriverProgram);
00011 /**
00012  * This abstract class represents a screwdriver program, typically defined on an
00013  * external control box for the screwdriver.
00014  */
00015 class ScrewdriverProgram
00016 {
00017 public:
00018     virtual ~ScrewdriverProgram() = 0;
00019
00020     /**
00021      * <p>
00022      * This method must return an identifier to uniquely identify this
00023      * screwdriver program. The identifier will be used for persistence and
00024      * resolving the correct screwdriver program. Therefore the identifier must
00025      * remain constant over time and versions of the screwdriver AuboCap.
00026      * </p>
00027      *
00028      * This method must be implemented.
00029      *
00030      * @return a unique identifier string which will remain constant over time
00031      * and versions of the screwdriver AuboCap.
00032      */
00033     virtual std::string getId() = 0;
00034
00035     /**
00036      * <p>
00037      * This method should return the name of the screwdriver program. The name

```

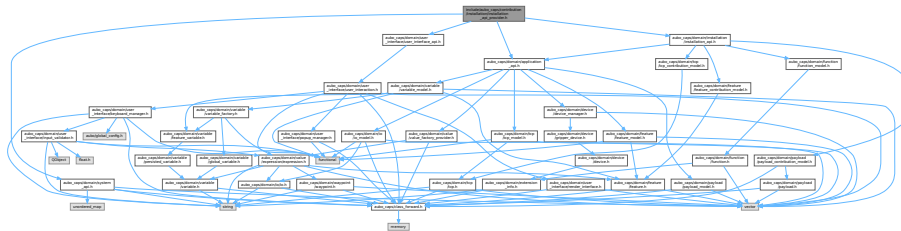
```

00038     * will be displayed in the UI, e.g. in a combo box. The name returned is
00039     * allowed to change over versions of the driver AuboCap as well as be
00040     * different depending on the current language selected in AuboScope, in
00041     * case translations are supported.
00042     * </p>
00043     *
00044     * This method must be implemented.
00045     *
00046     * @return the name of the screwdriver program to display in the UI.
00047     */
00048     virtual std::string getDisplayName() = 0;
00049 };
00050
00051 /**
00052  * This interface represents the type of screwing operation performed by the
00053  * screwdriver.
00054  */
00055 enum ScrewdriverOperationType
00056 {
00057     /**
00058      * The screwdriver operation will follow the screw in a tightening direction
00059      * (in).
00060      */
00061     TIGHTEN,
00062     /**
00063      * The screwdriver operation will follow the screw in a loosening direction
00064      * (out).
00065      */
00066     LOOSEN
00067 };
00068
00069 ARCS_CLASS_FORWARD(ScrewdriverParameters);
00070
00071 /**
00072  * This interface represents parameters for a screwdriver operation
00073  * defined/configured by the end user.
00074  */
00075 class ScrewdriverParameters
00076 {
00077 public:
00078     virtual ~ScrewdriverParameters() = default;
00079
00080     /**
00081      * <p>
00082      * Get the selected screwdriver program to be used for the screwdriver
00083      * operation.
00084      * </p>
00085      *
00086      * This is only valid, if the program selection capability has been
00087      * registered (using the method
00088      * {@link
00089      * ScrewdriverCapabilities#registerProgramSelectionCapability(ScrewdriverProgramListProvider,
00090      * ScriptCodeGenerator)}).
00091      *
00092      * @return The selected screwdriver program to use for the operation.
00093      * @throws UnsupportedOperationException if the screwdriver has not
00094      * registered the program selection capability
00095      */
00096     virtual ScrewdriverProgram getScrewdriverProgram() = 0;
00097
00098     /**
00099      * <p>
00100      * Get the selected screwdriver operation type.
00101      * </p>
00102      *
00103      * This is only valid, if the operation type capability has been registered
00104      * (using the method
00105      * {@link ScrewdriverCapabilities#registerOperationTypeCapability()}).
00106      *
00107      * @return The operation type selected by the user.
00108      * @throws UnsupportedOperationException if the screwdriver has not
00109      * registered the operation type capability
00110      */
00111     virtual ScrewdriverOperationType getScrewdriverOperationType() = 0;
00112 };
00113 // namespace aubo_scope
00114 // namespace arcs
00115 #endif

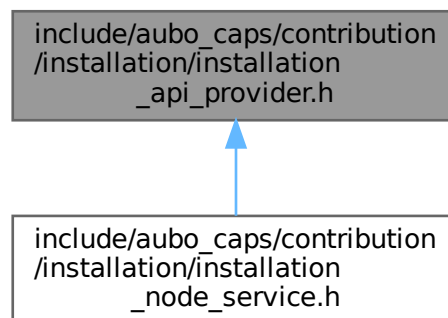
```

12.34 include/aubo_caps/contribution/installation/installation_api_provider.h File Reference

```
#include <aubo_caps/domain/application_api.h>
#include <aubo_caps/domain/system_api.h>
#include <aubo_caps/domain/user_interface/user_interface_api.h>
#include <aubo_caps/domain/installation/installation_api.h>
Include dependency graph for installation_api_provider.h:
```



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



Classes

- class [arcs::aubo_scope::InstallationApiProvider](#)
[InstallationApiProvider](#) Provides access to functionality and services relevant for installation nodes.

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([InstallationApiProvider](#))

12.35 installation_api_provider.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_INSTALLATION_API_PROVIDER_H
00002 #define AUBO_SCOPE_INSTALLATION_API_PROVIDER_H
00003
00004 #include <aubo_caps/domain/application_api.h>
00005 #include <aubo_caps/domain/system_api.h>
00006 #include <aubo_caps/domain/user_interface/user_interface_api.h>
00007 #include <aubo_caps/domain/installation/installation_api.h>
00008
00009 namespace arcs {
00010 namespace aubo_scope {
00011 ARCS_CLASS_FORWARD(InstallationApiProvider);
00012
00013 /**
00014  * \chinese
00015  *   API
00016  *   AuboScope
00017  * AuboScope
00018  * \endchinese
00019  * \english
00020  * InstallationApiProvider
00021  * Provides access to functionality and services relevant for installation
00022  * nodes. This includes what is available from within AuboScope, as well as
00023  * creating additional model elements to be used within AuboScope.
00024  * \endenglish
00025  */
00026 class InstallationApiProvider
00027 {
00028 public:
00029     InstallationApiProvider(InstallationApiProvider &f);
00030     InstallationApiProvider(InstallationApiProvider &&f);
00031     virtual ~InstallationApiProvider();
00032
00033     /**
00034      * \chinese
00035      *
00036      * @return SystemAPI
00037      * \endchinese
00038      * \english
00039      * Provides access to system related functionality
00040      *
00041      * @return an instance of SystemAPI
00042      * \endenglish
00043      */
00044     SystemApiPtr getSystemApi();
00045
00046     /**
00047      * \chinese
00048      *
00049      * @return UserInterfaceAPI
00050      * \endchinese
00051      * \english
00052      * Provides access to functionality related to user interface and end user
00053      * interaction
00054      *
00055      * @return an instance of UserInterfaceAPI
00056      * \endenglish
00057      */
00058     UserInterfaceApiPtr getUserInterfaceApi();
00059
00060     /**
00061      * \chinese
00062      *
00063      * @return InstallationAPI
00064      * \endchinese
00065      * \english
00066      * Provides access to functionality relevant related to the installation
00067      *
00068      * @return an instance of InstallationAPI
00069      * \endenglish
00070      */
00071     InstallationApiPtr getInstallationApi();
00072
00073 private:
00074     friend class DataSwitch;
00075     InstallationApiProvider();
00076     void *d_{ nullptr };
00077 };
00078
00079 } // namespace aubo_scope

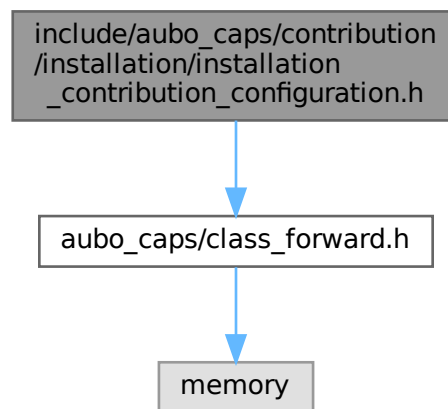
```

```
00083 } // namespace arcs
00084
00085 #endif // INSTALLATIONAPIPROVIDER_H
```

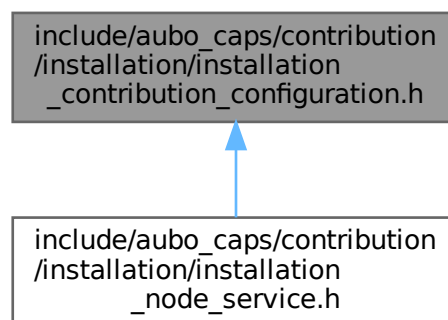
12.36 include/aubo_caps/contribution/installation/installation_contribution_configuration.h File Reference

#include <aubo_caps/class_forward.h>

Include dependency graph for installation_contribution_configuration.h:



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



Classes

- class [arcs::aubo_scope::InstallationContributionConfiguration](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([InstallationContributionConfiguration](#))

12.37 installation_contribution_configuration.h

[Go to the documentation of this file.](#)

```

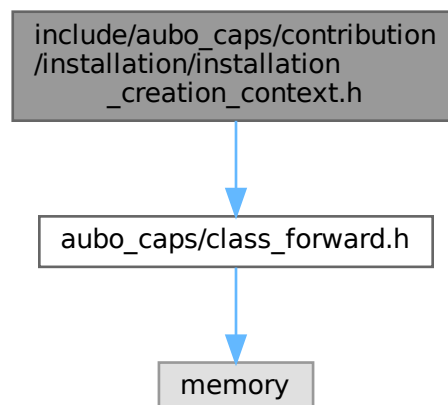
00001 #ifndef AUBO_SCOPE_INSTALLATION_CONTRIBUTION_CONFIGURATION_H
00002 #define AUBO_SCOPE_INSTALLATION_CONTRIBUTION_CONFIGURATION_H
00003
00004 #include <aubo_caps/class_forward.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008
00009 ARCS_CLASS_FORWARD(InstallationContributionConfiguration);
00010
00011 class InstallationContributionConfiguration
00012 {
00013 public:
00014     virtual ~InstallationContributionConfiguration() = default;
00015     // Intentionally left blank for future extension
00016 };
00017
00018 } // namespace aubo_scope
00019 } // namespace arcs
00020
00021 #endif // AUBO_SCOPE_INSTALLATION_CONTRIBUTION_CONFIGURATION_H

```

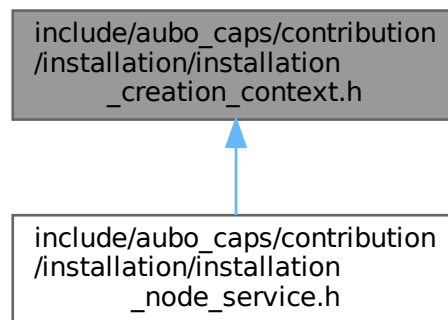
12.38 include/aubo_caps/contribution/installation/installation_↵
creation_context.h File Reference

```
#include <aubo_caps/class_forward.h>
```

Include dependency graph for installation_creation_context.h:



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



Classes

- class [arcs::aubo_scope::InstallationCreationContext](#)
[InstallationCreationContext](#) Provides information about the context in which an installation node is created.

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([InstallationCreationContext](#))

12.39 installation_creation_context.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_INSTALLATION_CREATION_CONTEXT_H
00002 #define AUBO_SCOPE_INSTALLATION_CREATION_CONTEXT_H
00003
00004 #include <aubo_caps/class_forward.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008
00009 ARCS_CLASS_FORWARD(InstallationCreationContext);
00010
00011 /**
00012  * \chinese
00013  *
00014  *
00015  * \endchinese
00016  * \english
00017  * InstallationCreationContext
00018  * Provides information about the context in which an installation node is
00019  * created.
00020  * \endenglish
  
```

```

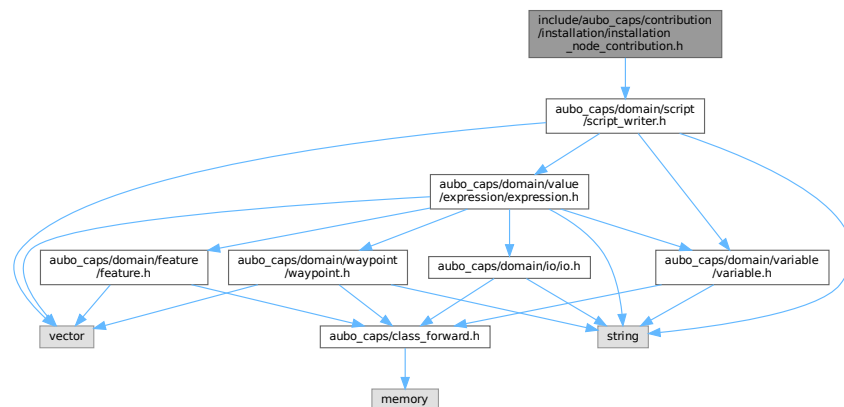
00021 */
00022 class InstallationCreationContext
00023 {
00024 public:
00025     enum NodeCreationType
00026     {
00027         NEW,
00028         LOAD
00029     };
00030
00031     InstallationCreationContext(InstallationCreationContext &f);
00032     InstallationCreationContext(InstallationCreationContext &&f);
00033     ~InstallationCreationContext();
00034
00035     NodeCreationType getNodeCreationType();
00036
00037     std::string getCreatedVersion();
00038     std::string getLastSavedVersion();
00039
00040 private:
00041     friend class DataSwitch;
00042     InstallationCreationContext();
00043     void *d_{ nullptr };
00044 };
00045
00046 } // namespace aubo_scope
00047 } // namespace arcs
00048
00049 #endif // AUBO_SCOPE_INSTALLATION_CREATION_CONTEXT_H

```

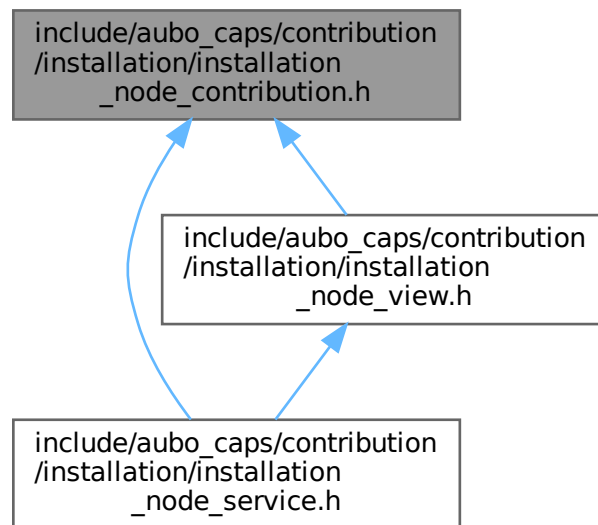
12.40 include/aubo_caps/contribution/installation/installation_node_↵ _contribution.h File Reference

#include <aubo_caps/domain/script/script_writer.h>

Include dependency graph for installation_node_contribution.h:



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



Classes

- class [arcs::aubo_scope::InstallationNodeContribution](#)
[InstallationNodeContribution](#) API for installation node contributions.

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([InstallationNodeContribution](#))

12.41 installation_node_contribution.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_INSTALLATION_NODE_CONTRIBUTION_H
00002 #define AUBO_SCOPE_INSTALLATION_NODE_CONTRIBUTION_H
00003
00004 #include <aubo_caps/domain/script/script_writer.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008   ARCS_CLASS_FORWARD(InstallationNodeContribution);
00009
00010 /**
00011  * \chinese
00012  *

```

```

00013 *      API
00014 * \endchinese
00015 * \english
00016 * InstallationNodeContribution
00017 * API for installation node contributions.
00018 * \endenglish
00019 */
00020 class InstallationNodeContribution
00021 {
00022 public:
00023     virtual ~InstallationNodeContribution() = default;
00024
00025     /**
00026     * \chinese
00027     *      AuboCap
00028     * \endchinese
00029     * \english
00030     * Called each time the end user opens this AuboCap contribution in the
00031     * Installation Tab.
00032     * \endenglish
00033     */
00034     virtual void openView() = 0;
00035
00036     /**
00037     * \chinese
00038     *      AuboCap
00039     * \endchinese
00040     * \english
00041     * Called each time the end user exits this AuboCap contribution in the
00042     * Installation Tab.
00043     * \endenglish
00044     */
00045     virtual void closeView() = 0;
00046
00047     /**
00048     * \chinese
00049     *
00050     * @param script_writer
00051     * \endchinese
00052     * \english
00053     * Defines script code that is added to the beginning of the script code
00054     * executed when a program is launched.
00055     *
00056     * @param script_writer serves to add script commands to program script code
00057     * preamble.
00058     * \endenglish
00059     */
00060
00061     virtual void generateScript(ScriptWriterPtr script_writer) = 0;
00062 };
00063
00064 } // namespace aubo_scope
00065 } // namespace arcs
00066
00067 #endif // AUBO_SCOPE_INSTALLATIONNODECONTRIBUTION_H

```

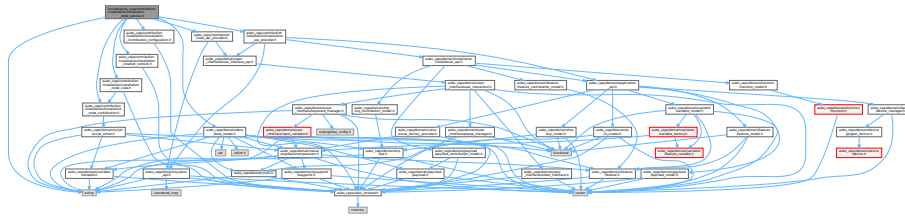
12.42 include/aubo_caps/contribution/installation/installation_node_↵ _service.h File Reference

```

#include <string>
#include <aubo_caps/domain/data/data_model.h>
#include <aubo_caps/contribution/view_api_provider.h>
#include <aubo_caps/contribution/installation/installation_node_view.h>
#include <aubo_caps/contribution/installation/installation_node_contribution.h>
#include <aubo_caps/contribution/installation/installation_creation_context.h>
#include <aubo_caps/contribution/installation/installation_contribution_configuration.h>
#include <aubo_caps/contribution/installation/installation_api_provider.h>

```

Include dependency graph for installation_node_service.h:



Classes

- class [arcs::aubo_scope::InstallationNodeService](#)
[InstallationNodeService](#) Defines an API required for defining and adding to AuboScope an installation node and corresponding screen where the user interface is Qt-based.

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([InstallationNodeService](#))

12.43 installation_node_service.h

[Go to the documentation of this file.](#)

```
00001 #ifndef AUBO_SCOPE_INSTALLATION_SERVICE_H
00002 #define AUBO_SCOPE_INSTALLATION_SERVICE_H
00003
00004 #include <string>
00005 #include <aubo_caps/domain/data/data_model.h>
00006 #include <aubo_caps/contribution/view_api_provider.h>
00007 #include <aubo_caps/contribution/installation/installation_node_view.h>
00008 #include <aubo_caps/contribution/installation/installation_node_contribution.h>
00009 #include <aubo_caps/contribution/installation/installation_creation_context.h>
00010 #include <aubo_caps/contribution/installation/installation_contribution_configuration.h>
00011 #include <aubo_caps/contribution/installation/installation_api_provider.h>
00012
00013 namespace arcs {
00014     namespace aubo_scope {
00015         ARCS_CLASS_FORWARD(InstallationNodeService);
00016
00017         /**
00018          * \chinese
00019          *
00020          * AuboScope Qt API
00021          * \endchinese
00022          * \english
00023          * InstallationNodeService
00024          * Defines an API required for defining and adding to AuboScope an installation
00025          * node and corresponding screen where the user interface is Qt-based.
00026          * \endenglish
00027          */
00028         class InstallationNodeService
00029         {
00030         public:
00031             virtual ~InstallationNodeService() = default;
00032
00033             /**
00034              * \chinese
```

```

00035 * @param locale AuboScope
00036 * @return
00037 *
00038 * \endchinese
00039 * \english
00040 * @param locale the current locale of AuboScope. Can be used for supporting
00041 * titles in several languages.
00042 * @return The text displayed for this installation contribution in the
00043 * left-hand side navigation of the Installation Tab as well as the title of
00044 * the corresponding installation node screen. Called once at start up.
00045 * \endenglish
00046 */
00047 virtual std::string getTitle() = 0;
00048
00049 /**
00050 * \chinese
00051 * @return
00052 * \endchinese
00053 * \english
00054 * @return The icon displayed for this installation contribution. Called once
00055 * at start up.
00056 * \endenglish
00057 */
00058 virtual std::string getIcon() = 0;
00059
00060 /**
00061 * \chinese
00062 * configuration
00063 *
00064 * <p>
00065 * ContributionConfiguration
00066 *
00067 * </p>
00068 *
00069 * @param configuration
00070 * \endchinese
00071 * \english
00072 * This method is called once after this service is registered. Modify the
00073 * configuration parameter to configure your contribution. The configuration
00074 * object will already have default values for its properties matching most
00075 * use cases. <p> The values of the ContributionConfiguration object will be
00076 * read once immediately after this method call. Changing values at a later
00077 * stage will have no effect, so do not store a reference to the
00078 * configuration object.
00079 * </p>
00080 * If the default values are appropriate, leave this method empty.
00081 *
00082 * @param configuration a modifiable ContributionConfiguration with default
00083 * values
00084 * \endenglish
00085 */
00086
00087 virtual void configureContribution(
00088     InstallationContributionConfigurationPtr configuration) = 0;
00089
00090 /**
00091 * \chinese
00092 * UI
00093 *
00094 * @param api_provider AuboScope
00095 *
00096 * @return
00097 * \endchinese
00098 * \english
00099 * Creates a new View instance which implements the UI for your installation
00100 * node screen. Called once when a new installation is loaded or created.
00101 *
00102 * @param api_provider Provides access to functionality and services
00103 * available from within AuboScope related to user interface and end user
00104 * interaction
00105 * @return the view
00106 * \endenglish
00107 */
00108 virtual InstallationNodeViewPtr createView(
00109     ViewApiProviderPtr api_provider) = 0;
00110
00111 /**
00112 * \chinese
00113 * <p> </p>
00114 * <p>
00115 * AuboCap </p>
00116 *
00117 * @param api_provider AuboScope
00118 * @param view createView
00119 * @param model
00120 * @param context

```

```

00122 * @return
00123 * \endchinese
00124 * \english
00125 * <p>
00126 * Creates a new installation node instance.
00127 * </p>
00128 * <p>
00129 * The returned node must use the supplied data model object to retrieve and
00130 * store the data contained in it. All modifications to the supplied data
00131 * model from the installation node constructor are ignored when an existing
00132 * installation is loaded. The data model object is shared between all
00133 * installation nodes contributed by the same AuboCap.
00134 * </p>
00135 *
00136 * @param api_provider Provides access to functionality and services
00137 * available from within AuboScope relevant for the installation node
00138 * @param view the View created by {@link #createView(ViewAPIProvider)}
00139 * @param model object where all configuration data of the new installation
00140 * node instance is to be stored in and retrieved from
00141 * @param context the context in which this node is created
00142 * @return the newly constructed installation node contribution instance
00143 * \endenglish
00144 */
00145 virtual InstallationNodeContributionPtr createInstallationNode(
00146     InstallationApiProviderPtr api_provider, InstallationNodeViewPtr view,
00147     DataModelPtr model, InstallationCreationContextPtr context) = 0;
00148 };
00149
00150 } // namespace aubo_scope
00151 } // namespace arcs
00152
00153 #endif

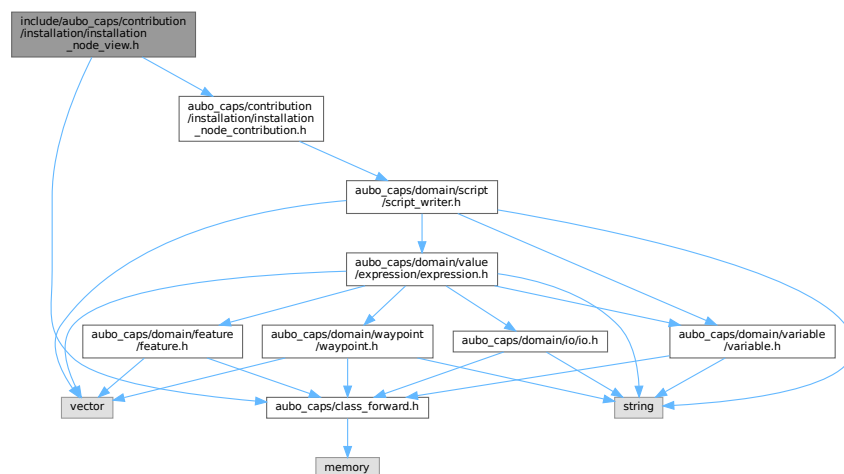
```

12.44 include/aubo_caps/contribution/installation/installation_node_view.h File Reference

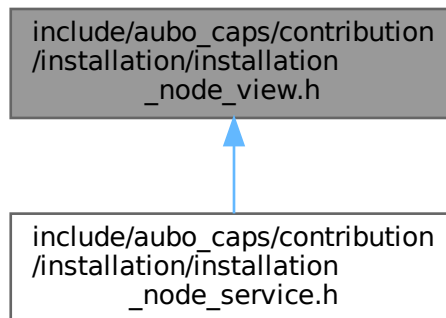
#include <aubo_caps/class_forward.h>

#include <aubo_caps/contribution/installation/installation_node_contribution.h>

Include dependency graph for installation_node_view.h:



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



Classes

- class [arcs::aubo_scope::InstallationNodeView](#)
[InstallationNodeView](#) API for an installation node View which defines the UI for the corresponding [InstallationNodeContribution](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Typedefs

- typedef long [WidgetHandle](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([InstallationNodeView](#))

12.44.1 Typedef Documentation

12.44.1.1 WidgetHandle

typedef long [WidgetHandle](#)

Definition at line 7 of file [installation_node_view.h](#).

12.45 installation_node_view.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_INSTALLATION_NODE_VIEW_H
00002 #define AUBO_SCOPE_INSTALLATION_NODE_VIEW_H
00003
00004 #include <aubo_caps/class_forward.h>
00005 #include <aubo_caps/contribution/installation/installation_node_contribution.h>
00006
00007 typedef long WidgetHandle;
00008
00009 namespace arcs {
00010 namespace aubo_scope {
00011 ARCS_CLASS_FORWARD(InstallationNodeView);
00012
00013 /**
00014  * \chinese
00015  *
00016  *      API      {@link InstallationNodeContribution}  UI
00017  *
00018  *      @param <C>      UI      {@link InstallationNodeContribution}
00019  *
00020  * \endchinese
00021  * \english
00022  * InstallationNodeView
00023  * API for an installation node View which defines the UI for the corresponding
00024  * {@link InstallationNodeContribution}
00025  *
00026  * @param <C> the (generic) type parameter for the interface representing the
00027  * type of the
00028  *      {@link InstallationNodeContribution} for which this view
00029  * implements the UI
00030  * \endenglish
00031  */
00032 class InstallationNodeView
00033 {
00034 public:
00035     virtual ~InstallationNodeView() = default;
00036
00037     /**
00038      * \chinese
00039      *      InstallationNodeContribution  UI
00040      *
00041      *
00042      * @param panel  UI
00043      * @param contribution
00044      * \endchinese
00045      * \english
00046      * Build the UI for the corresponding {@link InstallationNodeContribution}
00047      * on the provided panel. Use the contribution for your logic and data model
00048      * manipulations.
00049      *
00050      * @param panel the panel to build the UI on
00051      * @param contribution the corresponding installation node contribution
00052      * \endenglish
00053      */
00054     virtual void buildUi(WidgetHandle panel,
00055                         InstallationNodeContributionPtr contribution) = 0;
00056 };
00057
00058 } // namespace aubo_scope
00059 } // namespace arcs
00060
00061 #endif // AUBO_SCOPE_INSTALLATION_NODE_VIEW_H

```

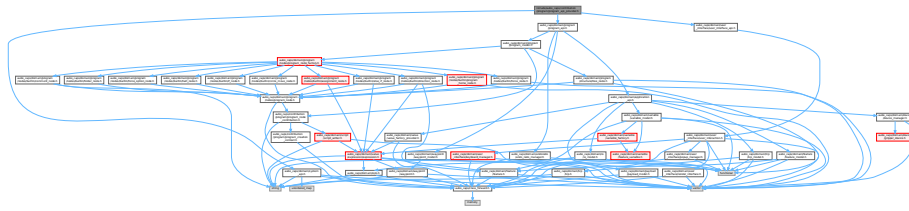
12.46 include/aubo_caps/contribution/program/program_api_provider.h File Reference

```

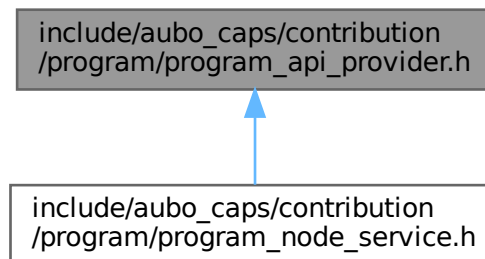
#include <aubo_caps/domain/system_api.h>
#include <aubo_caps/domain/program/program_api.h>
#include <aubo_caps/domain/user_interface/user_interface_api.h>

```

Include dependency graph for `program_api_provider.h`:



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



Classes

- class `arcs::aubo_scope::ProgramApiProvider`
`ProgramApiProvider` Provides access to functionality and services relevant for program nodes.

Namespaces

- namespace `arcs`
- namespace `arcs::aubo_scope`

Functions

- `arcs::aubo_scope::ARCS_CLASS_FORWARD` (`ProgramApiProvider`)

12.47 program_api_provider.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_PROGRAM_API_PROVIDER_H
00002 #define AUBO_SCOPE_PROGRAM_API_PROVIDER_H
00003
00004 #include <aubo_caps/domain/system_api.h>
00005 #include <aubo_caps/domain/program/program_api.h>
00006 #include <aubo_caps/domain/user_interface/user_interface_api.h>
00007

```

```

00008 namespace arcs {
00009 namespace aubo_scope {
00010
00011 ARCS_CLASS_FORWARD(ProgramApiProvider);
00012
00013 /**
00014  * \chinese
00015  *   API
00016  *
00017  *   AuboScope
00018  * \endchinese
00019  * \english
00020  * ProgramApiProvider
00021  * Provides access to functionality and services relevant for program nodes.
00022  * This includes what is available from within AuboScope, as well as creating
00023  * additional model elements to be used within AuboScope.
00024  * \endenglish
00025  */
00026 class ProgramApiProvider
00027 {
00028 public:
00029     ProgramApiProvider(ProgramApiProvider &f);
00030     ProgramApiProvider(ProgramApiProvider &&f);
00031     virtual ~ProgramApiProvider();
00032
00033     /**
00034      * \chinese
00035      *
00036      *
00037      * @return SystemAPI
00038      * \endchinese
00039      * \english
00040      * Provides access to system related functionality
00041      *
00042      * @return an instance of SystemAPI
00043      * \endenglish
00044      */
00045     SystemApiPtr getSystemApi();
00046
00047     /**
00048      * \chinese
00049      *
00050      *
00051      * @return UserInterfaceAPI
00052      * \endchinese
00053      * \english
00054      * Provides access to functionality related to user interface and end user
00055      * interaction
00056      *
00057      * @return an instance of UserInterfaceAPI
00058      * \endenglish
00059      */
00060     UserInterfaceApiPtr getUserInterfaceApi();
00061
00062     /**
00063      * \chinese
00064      *
00065      *
00066      * @return ProgramAPI
00067      * \endchinese
00068      * \english
00069      * Provides access to functionality relevant related to programs
00070      *
00071      * @return an instance of ProgramAPI
00072      * \endenglish
00073      */
00074     ProgramApiPtr getProgramApi();
00075
00076 private:
00077     ProgramApiProvider();
00078     friend class DataSwitch;
00079     void *d_{ nullptr };
00080 };
00081
00082 } // namespace aubo_scope
00083 } // namespace arcs
00084
00085 #endif // AUBO_SCOPE_PROGRAM_API_PROVIDER_H

```


Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([ProgramContributionConfiguration](#))

12.49 program_contribution_configuration.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_CONTRIBUTION_CONFIGURATION_H
00002 #define AUBO_SCOPE_CONTRIBUTION_CONFIGURATION_H
00003
00004 #include <any>
00005 #include <vector>
00006 #include <functional>
00007
00008 #include <aubo_caps/domain/program/structure/tree_node.h>
00009
00010 namespace arcs {
00011 namespace aubo_scope {
00012
00013 ARCS_CLASS_FORWARD(ProgramContributionConfiguration);
00014
00015 /**
00016  * \chinese
00017  *
00018  *
00019  * \endchinese
00020  * \english
00021  * ProgramContributionConfiguration
00022  * Provides access to configure the properties of an installation node
00023  * contribution.
00024  * \endenglish
00025  */
00026 class ProgramContributionConfiguration
00027 {
00028 public:
00029     ProgramContributionConfiguration(ProgramContributionConfiguration &f);
00030     ProgramContributionConfiguration(ProgramContributionConfiguration &&f);
00031     virtual ~ProgramContributionConfiguration();
00032
00033     /**
00034      * If set to <code>true</code> it is possible for the program node to have
00035      * child nodes.
00036      *
00037      * <p>
00038      * The default value is <code>false</code>.
00039      * </p>
00040      *
00041      * @param childrenAllowed determines whether this node can have child nodes
00042      */
00043     void setChildrenAllowed(bool childrenAllowed);
00044     void setChildrenAllowedOnly(const std::vector<std::string> &childrenList);
00045
00046     /**
00047      * If set to <code>false</code> this program node will not appear in the
00048      * Program Tab in AuboScope and can not be inserted in the program tree by
00049      * the end user. It can only be inserted programmatically using the
00050      * {@link ProgramNodeFactory}. This
00051      * also means, it can not be inserted using copy/paste of that node
00052      * directly. It can however be inserted by copy/paste of an ancestor of such
00053      * a node.
00054      *
00055      * <p>
00056      * The default value is <code>true</code>.
00057      * </p>
00058      *
00059      * @param userInsertable determines whether the end user can insert this
00060      * node
00061      */
00062     void setUserInsertable(bool userInsertable);
00063
00064     /**

```

```

00065 * If set to <code>true</code> it is not possible for the end user to create
00066 * new program nodes of this type. Loading of existing programs will however
00067 * still be supported for this program node.
00068 *
00069 * <p>
00070 * The default value is <code>false</code>.
00071 * </p>
00072 *
00073 * @param deprecated determines whether this node is deprecated
00074 */
00075 void setDeprecated(bool deprecated);
00076
00077 /**
00078 * Set the display order id for this type of program node contribution. This
00079 * makes it possible to specify how all the program node contributions from
00080 * a AuboCap should be ordered/sorted in AuboScope.
00081 *
00082 * <p>
00083 * All program node contributions from a AuboCap will be grouped together
00084 * and sorted (in ascending order) according to their display order id.
00085 * </p>
00086 *
00087 * <p>
00088 * The default value is 0.0. If the display order id for all program node
00089 * contributions in a AuboCap have the default value, the standard built-in
00090 * sorting in AuboScope will be used.
00091 * </p>
00092 *
00093 * @param displayOrderId Determines the display order of the AuboCap program
00094 * node type
00095 */
00096 void setDisplayOrderId(double displayOrderId);
00097
00098 /**
00099 * <p>
00100 * Configure whether or not the end user is allowed to set a breakpoint on
00101 * or single step the AuboCap program node.
00102 * </p>
00103 *
00104 * <p>
00105 * The default value is <code>true</code>.
00106 * </p>
00107 *
00108 * @param isBreakpointAllowed <code>true</code> if it should be allowed to
00109 * set a breakpoint on or single step this AuboCap program node. If it
00110 * should be disallowed, the value must be <code>false</code>.
00111 */
00112 void setAllowBreakpointOnNode(bool isBreakpointAllowed);
00113
00114 /**
00115 * <p>
00116 * Configure whether or not the end user is allowed to set a breakpoint on
00117 * or single step any of the child nodes in the sub-tree under the AuboCap
00118 * program node. Note that it will only be possible to set a breakpoint on a
00119 * child node if the child node itself supports breakpoints.
00120 * </p>
00121 *
00122 * <p>
00123 * The default value is <code>false</code>.
00124 * </p>
00125 *
00126 * @param isBreakpointOnChildNodesAllowed <code>true</code> if it should be
00127 * allowed to set a breakpoint on or single step any of the child nodes in
00128 * the sub-tree under this AuboCap program node. If it should be disallowed,
00129 * the value must be <code>false</code>.
00130 */
00131 void setAllowBreakpointOnChildNodesInSubtree(
00132     bool isBreakpointOnChildNodesAllowed);
00133
00134 /**
00135 * <p>
00136 * Configure whether or not the end user can start the program from the
00137 * AuboCap program node when it is selected in the program tree.
00138 * </p>
00139 *
00140 * <p>
00141 * The default value is <code>true</code>.
00142 * </p>
00143 *
00144 * @param isStartFromNodeAllowed <code>true</code> if it should be allowed
00145 * to start the program directly from this AuboCap program node. If it
00146 * should be disallowed, the value must be <code>false</code>.
00147 */
00148 */
00149 void setAllowStartFromNode(bool isStartFromNodeAllowed);
00150
00151 /**

```

```

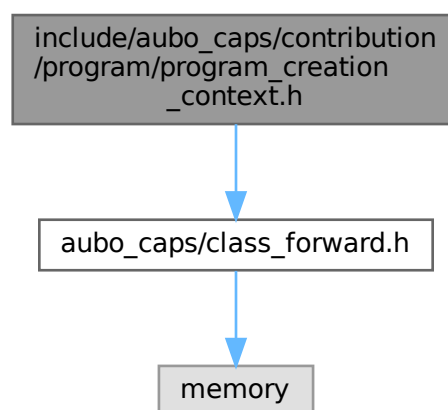
00152 * <p>
00153 * Configure whether or not the end user can start the program directly from
00154 * any selected child node in the sub-tree under the AuboCap program node.
00155 * Note that it will only be possible to start from a selected child node if
00156 * the child node itself supports starting from the child node.
00157 * </p>
00158 *
00159 * <p>
00160 * The default value is <code>false</code>.
00161 * </p>
00162 *
00163 * @param isStartFromChildNodesAllowed <code>true</code> if it should be
00164 * allowed to start the program directly from any child program node. If it
00165 * should be disallowed, the value must be <code>false</code>..
00166 */
00167 void setAllowStartFromChildNodesInSubtree(
00168     bool isStartFromChildNodesAllowed);
00169
00170 /**
00171 * Configure parent node of this node. End user cannot insert this node to
00172 *
00173 * The default value is <code>""</code>.
00174 * @param parentNodeName
00175 */
00176 void setAllowInsertOnlyIntoNodes(
00177     const std::vector<std::string> &parentNodeNames);
00178
00179 private:
00180     ProgramContributionConfiguration();
00181     friend class DataSwitch;
00182     void *d_{ nullptr };
00183 };
00184
00185 } // namespace aubo_scope
00186 } // namespace arcs
00187
00188 #endif // AUBO_SCOPE_CONTRIBUTION_CONFIGURATION_H

```

12.50 include/aubo_caps/contribution/program/program_creation_context.h File Reference ↗

#include <aubo_caps/class_forward.h>

Include dependency graph for program_creation_context.h:



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



Classes

- class `arcs::aubo_scope::ProgramCreationContext`
`ProgramCreationContext` Provides information about the context in which an installation node is created.

Namespaces

- namespace `arcs`
- namespace `arcs::aubo_scope`

Functions

- `arcs::aubo_scope::ARCS_CLASS_FORWARD` (`ProgramCreationContext`)

12.51 `program_creation_context.h`

[Go to the documentation of this file.](#)

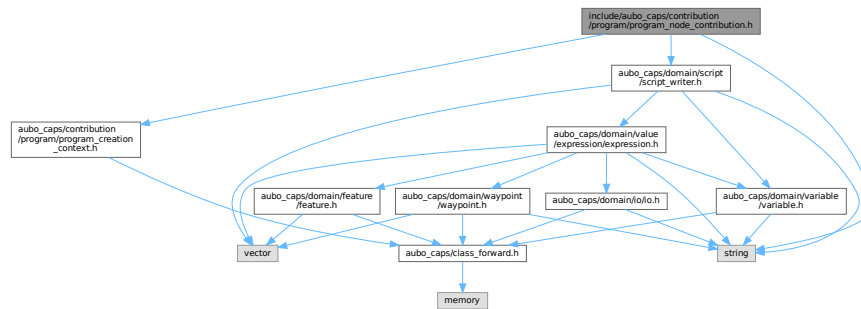
```

00001 #ifndef AUBO_SCOPE_PROGRAM_CREATION_CONTEXT_H
00002 #define AUBO_SCOPE_PROGRAM_CREATION_CONTEXT_H
00003
00004 #include <aubo_caps/class_forward.h>
00005
00006 namespace arcs {
00007     namespace aubo_scope {
00008         ARCS_CLASS_FORWARD(ProgramCreationContext);
00009
00010         /**
00011          * \chinese
00012          *
00013          *
00014          * \endchinese
00015          * \english
00016          * ProgramCreationContext
00017          * Provides information about the context in which an installation node is
00018          * created.
00019          * \endenglish
00020          */
00021         class ProgramCreationContext
00022         {
00023         public:
00024             enum NodeCreationType
00025             {
00026                 NEW,
00027                 LOAD
00028             };
00029
00030             ProgramCreationContext(ProgramCreationContext &f);
00031             ProgramCreationContext(ProgramCreationContext &&f);
00032             ~ProgramCreationContext();
00033
00034             NodeCreationType getNodeCreationType();
00035
00036             std::string getCreatedVersion();
00037             std::string getLastSavedVersion();
00038
00039         private:
00040             friend class DataSwitch;
00041             ProgramCreationContext();
00042             void *d_{ nullptr };
00043         };
00044
00045     } // namespace aubo_scope
00046 } // namespace arcs
00047
00048 #endif // AUBO_SCOPE_PROGRAM_CREATION_CONTEXT_H

```


12.52 `include/aubo_caps/contribution/program/program_node_↵`
`contribution.h` File Reference

```
#include <string>
#include <aubo_caps/contribution/program/program_creation_context.h>
#include <aubo_caps/domain/script/script_writer.h>
Include dependency graph for program_node_contribution.h:
```



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



Classes

- class [arcs::aubo_scope::ProgramNodeContribution](#)
[ProgramNodeContribution](#) Defines an API required for specifying a AuboCap Program Node within AuboScope.

Namespaces

- namespace `arcs`
- namespace `arcs::aubo_scope`

Functions

- `arcs::aubo_scope::ARCS_CLASS_FORWARD` (`ProgramNodeContribution`)

12.53 program_node_contribution.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_PROGRAM_NODE_CONTRIBUTION_H
00002 #define AUBO_SCOPE_PROGRAM_NODE_CONTRIBUTION_H
00003
00004 #include <string>
00005 #include <aubo_caps/contribution/program/program_creation_context.h>
00006 #include <aubo_caps/domain/script/script_writer.h>
00007
00008 namespace arcs {
00009 namespace aubo_scope {
00010 ARCS_CLASS_FORWARD(ProgramNodeContribution);
00011
00012 /**
00013  * \chinese
00014  *
00015  *   AuboScope   AuboCap   API
00016  * \endchinese
00017  * \english
00018  * ProgramNodeContribution
00019  * Defines an API required for specifying a AuboCap Program Node within
00020  * AuboScope.
00021  * \endenglish
00022  */
00023 class ProgramNodeContribution
00024 {
00025 public:
00026     virtual ~ProgramNodeContribution() = default;
00027
00028     /**
00029      * \chinese
00030      *
00031      * \endchinese
00032      * \english
00033      * Called when this node is selected in the program tree.
00034      * \endenglish
00035      */
00036     virtual void openView() = 0;
00037
00038     /**
00039      * \chinese
00040      *
00041      * \endchinese
00042      * \english
00043      * Called when this node is unselected in the program tree or when
00044      * navigating to another view.
00045      * \endenglish
00046      */
00047     virtual void closeView() = 0;
00048
00049     /**
00050      * \chinese
00051      * @return
00052      * @throws IllegalStateException
00053      * \endchinese
00054      * \english
00055      * @return The text displayed in the Program Tree for the program node.
00056      * @throws IllegalStateException if the data model or child program
00057      * sub-tree is modified
00058      * \endenglish
00059      */
00060     virtual std::string getTitle() = 0;
00061
00062     /**
00063      * \chinese
00064      * <p>
00065      *     <code>true</code>    <code>>false</code>
00066      * </p>
00067      *
00068      * <p>
00069      *
00070      *
00071      * </p>
00072      *
00073      *   DataModel
00074      * PersistedVariable   TCP
00075      *
00076      * @return    <code>true</code>
00077      * <code>>false</code>
00078      * @throws IllegalStateException
00079      * \endchinese
00080      * \english
00081      * <p>
00082      * Should return <code>true</code> if this program node is complete, and

```

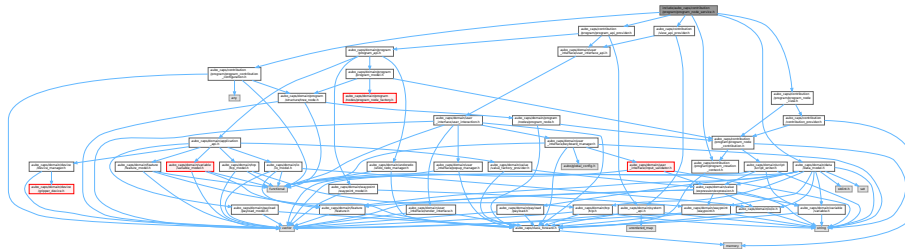
```

00083 * <code>false</code> otherwise.
00084 * </p>
00085 *
00086 * <p>
00087 * Undefined program nodes will be marked with yellow in the program
00088 * tree. The user cannot start the program as long as it contains one or
00089 * more undefined program nodes.
00090 * </p>
00091 *
00092 * If this program node stores an unresolvable entity (e.g. {@link
00093 * variable/PersistedVariable} or {@link tcp/TCP}) in its DataModel, the
00094 * program node will be treated as undefined.
00095 *
00096 * @return <code>true</code> if this program node is complete and otherwise
00097 * <code>false</code>.
00098 * @throws IllegalStateException if the data model or child program
00099 * sub-tree is modified.
00100 * \endenglish
00101 */
00102 virtual bool isDefined() = 0;
00103
00104 /**
00105 * \chinese
00106 *
00107 *
00108 * @param script_writer
00109 * \endchinese
00110 * \english
00111 * Defines the script code to be generated by this program node within
00112 * the program.
00113 *
00114 * @param script_writer serves to add script commands that define the behaviour
00115 * of this program node.
00116 * \endenglish
00117 */
00118 virtual void generateScript(ScriptWriterPtr script_writer) = 0;
00119
00120 /**
00121 * \chinese
00122 * @brief postAction HTTP POST
00123 * @param pattern URL
00124 * @param body
00125 * @return
00126 * \endchinese
00127 * \english
00128 * @brief post
00129 * @param pattern
00130 * @param body
00131 * @return
00132 * \endenglish
00133 */
00134 virtual std::string postAction(const char *pattern, const std::string &body)
00135 {
00136     return "";
00137 }
00138
00139 /**
00140 * \chinese
00141 * @brief getAction HTTP GET
00142 * @param pattern URL
00143 * @param body
00144 * @return
00145 * \endchinese
00146 * \english
00147 * @brief GET
00148 * @param pattern
00149 * @param body
00150 * @return
00151 * \endenglish
00152 */
00153 virtual std::string getAction(const char *pattern, const std::string &body)
00154 {
00155     return "";
00156 }
00157 };
00158
00159 } // namespace aubo_scope
00160 } // namespace arcs
00161
00162 #endif // AUBO_SCOPE_PROGRAM_NODE_CONTRIBUTION_H

```

12.54 include/aubo_caps/contribution/program/program_node_service.h File Reference

```
#include <aubo_caps/contribution/view_api_provider.h>
#include <aubo_caps/contribution/program/program_node_view.h>
#include <aubo_caps/contribution/program/program_creation_context.h>
#include <aubo_caps/contribution/program/program_api_provider.h>
#include <aubo_caps/contribution/program/program_contribution_configuration.h>
#include <aubo_caps/domain/data/data_model.h>
Include dependency graph for program_node_service.h:
```



Classes

- class [arcs::aubo_scope::ProgramNodeService](#)
[ProgramNodeService](#) Defines an API required for adding a new type of program node to AuboScope.

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([ProgramNodeService](#))

12.55 program_node_service.h

[Go to the documentation of this file.](#)

```
00001 #ifndef AUBO_SCOPE_PROGRAM_NODE_SERVICE_H
00002 #define AUBO_SCOPE_PROGRAM_NODE_SERVICE_H
00003
00004 #include <aubo_caps/contribution/view_api_provider.h>
00005 #include <aubo_caps/contribution/program/program_node_view.h>
00006 #include <aubo_caps/contribution/program/program_creation_context.h>
00007 #include <aubo_caps/contribution/program/program_api_provider.h>
00008 #include <aubo_caps/contribution/program/program_contribution_configuration.h>
00009 #include <aubo_caps/domain/data/data_model.h>
00010
00011 namespace arcs {
00012 namespace aubo_scope {
00013 ARCS_CLASS_FORWARD(ProgramNodeService);
00014
00015 /**
00016  * \chinese
00017  *
00018  * AuboScope API
```

```

00019 * \endchinese
00020 * \english
00021 * ProgramNodeService
00022 * Defines an API required for adding a new type of program node to AuboScope.
00023 * \endenglish
00024 */
00025 class ProgramNodeService
00026 {
00027 public:
00028     virtual ~ProgramNodeService() = default;
00029
00030     /**
00031      * \chinese
00032      *
00033      * @return
00034      * \endchinese
00035      * \english
00036      * Get the unique identifier for this kind of program node. The identifier
00037      * will be used when storing programs that use the program node.
00038      *
00039      * @return the unique identifier.
00040      * \endenglish
00041      */
00042     virtual std::string getId() = 0;
00043
00044     /**
00045      * \chinese
00046      * @return
00047      * \endchinese
00048      * \english
00049      * @return The text displayed in the Structure Tab for program nodes created
00050      * by this factory.
00051      * \endenglish
00052      */
00053     virtual std::string getTitle() = 0;
00054
00055     /**
00056      * \chinese
00057      * @return
00058      * \endchinese
00059      * \english
00060      * @return The icon displayed for program nodes created by this factory.
00061      * \endenglish
00062      */
00063     virtual std::string getIcon() = 0;
00064
00065     /**
00066      * \chinese
00067      *
00068      * @param configuration
00069      * \endchinese
00070      * \english
00071      * configureContribution
00072      * @param configuration
00073      * \endenglish
00074      */
00075     virtual void configureContribution(
00076         ProgramContributionConfigurationPtr configuration) = 0;
00077
00078     /**
00079      * \chinese
00080      * UI
00081      *
00082      * @param api_provider AuboScope
00083      *
00084      * @return
00085      * \endchinese
00086      * \english
00087      * Creates a new View instance which implements the UI for your program node
00088      * screen. Called once when a new program is loaded or created.
00089      *
00090      * @param api_provider Provides access to functionality and services
00091      * available from within AuboScope related to user interface and end user
00092      * interaction
00093      * @return the view
00094      * \endenglish
00095      */
00096     virtual ProgramNodeViewPtr createView(ViewApiProviderPtr api_provider) = 0;
00097
00098     /**
00099      * \chinese
00100      * <p>
00101      *
00102      * </p>
00103      *
00104      * <p>
00105
```

```

00106 *
00107 * /
00108 *
00109 * /
00110 * </p>
00111 *
00112 * @param api_provider AuboScope
00113 * @param view
00114 * @param model
00115 * @param context
00116 * @return
00117 * \endchinese
00118 * \english
00119 * <p>
00120 * Creates a new program node contribution instance.
00121 * </p>
00122 *
00123 * <p>
00124 * The returned node must use the supplied data model object to retrieve and
00125 * store the data contained in it. All modifications to the supplied data
00126 * model from the program node constructor are ignored when an existing
00127 * program node is loaded or copy/pasted. Every change to the model object
00128 * is registered as a separate undo/redo event in the program tree.
00129 * </p>
00130 *
00131 * @param api_provider the AuboCap API object with access to domain data
00132 * @param view the View created by createView
00133 * @param model object where all configuration data of the new program node
00134 * instance is to be stored in and retrieved from
00135 * @param context the creation context
00136 * @return the newly constructed program node contribution instance
00137 * \endenglish
00138 */
00139 virtual ProgramNodeContributionPtr createNode(
00140     ProgramApiProviderPtr api_provider, ProgramNodeViewPtr view,
00141     DataModelPtr model, ProgramCreationContextPtr context) = 0;
00142 };
00143
00144 } // namespace aubo_scope
00145 } // namespace arcs
00146
00147 #endif // AUBO_SCOPE_PROGRAM_NODE_SERVICE_H

```

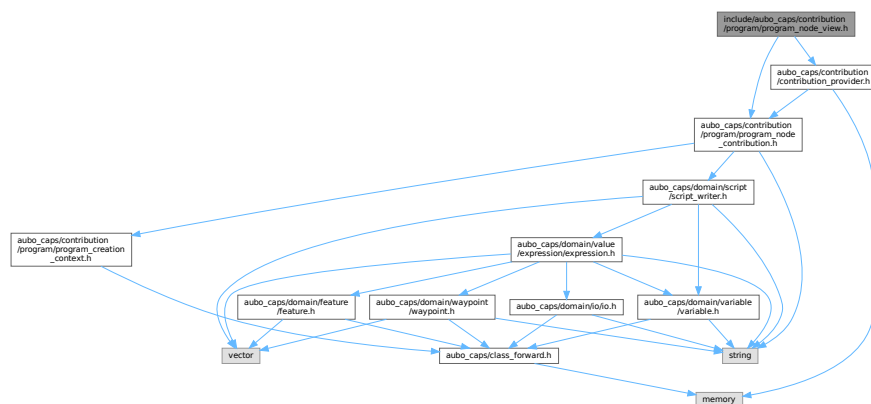
12.56 include/aubo_caps/contribution/program/program_node_view.h

File Reference

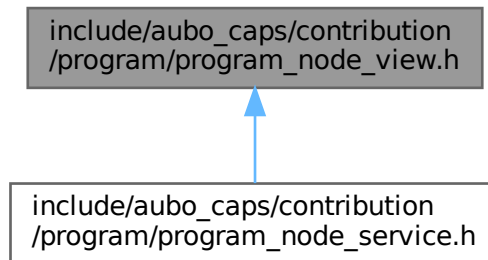
#include <aubo_caps/contribution/program/program_node_contribution.h>

#include <aubo_caps/contribution/contribution_provider.h>

Include dependency graph for program_node_view.h:



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



Classes

- class [arcs::aubo_scope::ProgramNodeView](#)
[ProgramNodeView](#) API for a program node View which defines the UI for the corresponding [ProgramNodeContribution](#).

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Typedefs

- typedef long [arcs::aubo_scope::WidgetHandle](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([ProgramNodeView](#))

12.57 program_node_view.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_PROGRAM_NODE_VIEW_H
00002 #define AUBO_SCOPE_PROGRAM_NODE_VIEW_H
00003
00004 #include <aubo_caps/contribution/program/program_node_contribution.h>
00005 #include <aubo_caps/contribution/contribution_provider.h>
00006
00007 namespace arcs {
00008     namespace aubo_scope {
00009         typedef long WidgetHandle;
00010         ARCS_CLASS_FORWARD(ProgramNodeView);
00011
00012         /**
00013          * \chinese
00014          *
00015          * API    {@link ProgramNodeContribution} UI
00016          * {@link ProgramNodeView}
00017         */
00018     }
00019 }
  
```

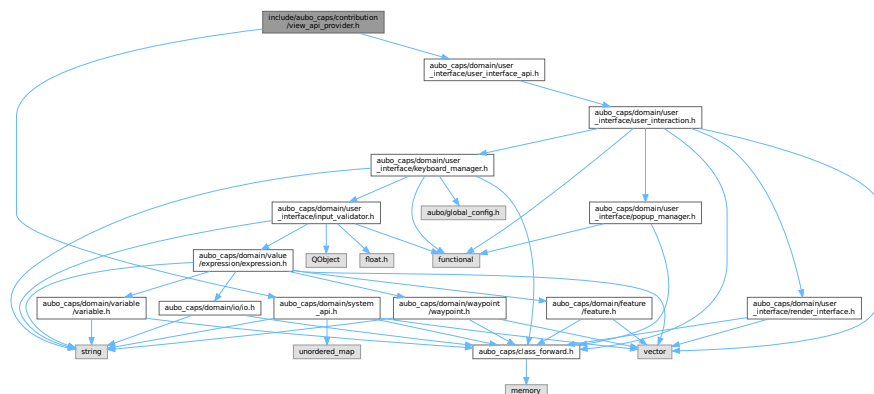
```

00017 * \endchinese
00018 * \english
00019 * ProgramNodeView
00020 * API for a program node View which defines the UI for the corresponding {@link
00021 * ProgramNodeContribution}. The instance of {@link ProgramNodeView} is a
00022 * shared entity between several contributions.
00023 * \endenglish
00024 */
00025 class ProgramNodeView
00026 {
00027 public:
00028     virtual ~ProgramNodeView() = default;
00029
00030     /**
00031      * \chinese
00032      *     ProgramNodeContribution    UI
00033      *
00034      *
00035      * @param panel    UI
00036      * @param provider
00037      * \endchinese
00038      * \english
00039      * Build the UI for the corresponding {@link ProgramNodeContribution} on the
00040      * provided panel. Use the contribution for your logic and data model
00041      * manipulations.
00042      *
00043      * @param panel the panel to build the UI on
00044      * @param provider the provider for corresponding program node contribution
00045      * currently selected (in the program tree)
00046      * \endenglish
00047      */
00048     virtual void buildUi(
00049         WidgetHandle panel,
00050         ContributionProviderPtr<ProgramNodeContribution> provider) = 0;
00051 };
00052
00053 } // namespace aubo_scope
00054 } // namespace arcs
00055
00056 #endif // AUBO_SCOPE_PROGRAM_NODE_VIEW_H

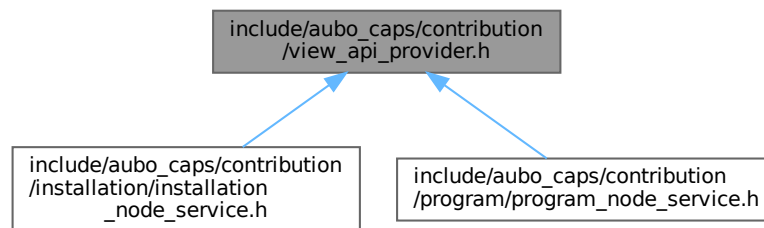
```

12.58 include/aubo_caps/contribution/view_api_provider.h File Reference

#include <aubo_caps/domain/system_api.h>
 #include <aubo_caps/domain/user_interface/user_interface_api.h>
 Include dependency graph for view_api_provider.h:



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



Classes

- class [arcs::aubo_scope::ViewApiProvider](#)
[ViewApiProvider](#) Provides access to functionality relevant for the View (UI) of an installation or a program node contribution.

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([ViewApiProvider](#))

12.59 view_api_provider.h

[Go to the documentation of this file.](#)

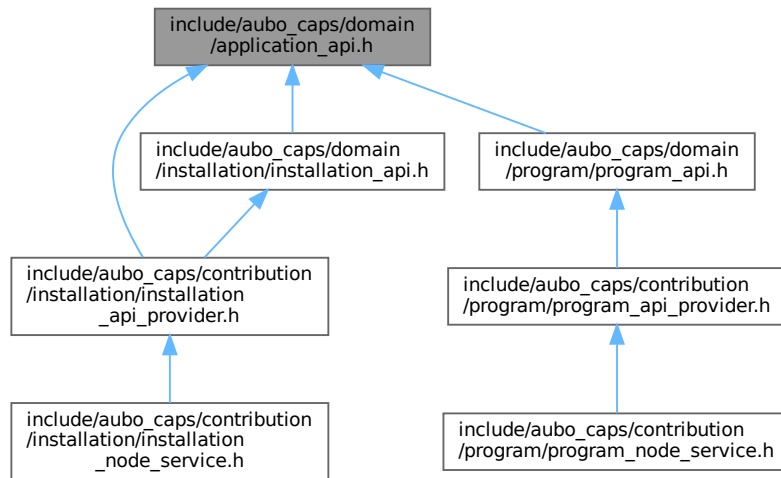
```

00001 #ifndef AUBO_SCOPE_VIEW_API_PROVIDER_H
00002 #define AUBO_SCOPE_VIEW_API_PROVIDER_H
00003
00004 #include <aubo_caps/domain/system_api.h>
00005 #include <aubo_caps/domain/user_interface/user_interface_api.h>
00006
00007 namespace arcs {
00008 namespace aubo_scope {
00009 ARCS_CLASS_FORWARD(ViewApiProvider);
00010
00011 /**
00012  * \chinese
00013  *   API
00014  *   UI
00015  *
00016  * <b> </b>      HTML      AuboCap
00017  * \endchinese
00018  * \english
00019  * ViewApiProvider
00020  * Provides access to functionality relevant for the View (UI) of an
00021  * installation or a program node contribution.
00022  *
00023  * <b>NOTE:</b> This interface is not relevant for AuboCap program and
00024  * installation nodes with a HTML-based user interface.
00025  * \endenglish
00026  */
00027 class ViewApiProvider
  
```

12.60 include/aubo_caps/domain/application_api.h File Reference

[illegible]

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



Classes

- class [arcs::aubo_scope::ApplicationApi](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_sdk](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_sdk::ARCS_CLASS_FORWARD](#) (RobotProxy)
- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) (ApplicationApi)
- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) (InstallationNodeContribution)

12.61 application_api.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_APPLICATION_API_H
00002 #define AUBO_SCOPE_APPLICATION_API_H
00003
00004 #include <aubo_caps/domain/io/io_model.h>
00005 #include <aubo_caps/domain/device/device_manager.h>
00006 #include <aubo_caps/domain/feature/feature_model.h>
00007 #include <aubo_caps/domain/payload/payload_model.h>
00008 #include <aubo_caps/domain/tcp/tcp_model.h>
00009 #include <aubo_caps/domain/variable/variable_model.h>
00010 #include <aubo_caps/domain/value/value_factory_provider.h>
00011
00012 namespace arcs {
00013
00014 namespace aubo_sdk {
```

```

00015 ARCS_CLASS_FORWARD(RobotProxy);
00016 }
00017
00018 namespace aubo_scope {
00019 ARCS_CLASS_FORWARD(ApplicationApi);
00020 ARCS_CLASS_FORWARD(InstallationNodeContribution);
00021
00022 /**
00023  * \chinese
00024  * @defgroup ApplicationApi ApplicationApi( )
00025  * @ingroup ApplicationApi
00026  *
00027  *          AuboScope
00028  *          AuboScope
00029  * \endchinese
00030  * \english
00031  * @defgroup ApplicationApi ApplicationApi
00032  * @ingroup ApplicationApi
00033  * ApplicationApi
00034  * Provides access to functionality and services which are relevant for various
00035  * different applications and contexts. This includes what is available from
00036  * within AuboScope, as well as creating additional model elements to be used
00037  * within AuboScope.
00038  * \endenglish
00039  */
00040 */
00041 class ARCS_ABI_EXPORT ApplicationApi
00042 {
00043 public:
00044     ApplicationApi(ApplicationApi &f);
00045     ApplicationApi(ApplicationApi &&f);
00046     virtual ~ApplicationApi();
00047
00048     /**
00049      * @ingroup ApplicationApi
00050      * @ref IoModel
00051      * \chinese
00052      * IO
00053      * @return
00054      * \endchinese
00055      * \english
00056      * Returns an interface for working with the inputs and outputs of the robot.
00057      * @return An interface for working with the inputs and outputs of the robot.
00058      * \endenglish
00059      */
00060     IoModelPtr getIoModel();
00061
00062     /**
00063      * @ingroup ApplicationApi
00064      * @ref FeatureModel
00065      * \chinese
00066      *
00067      * @return
00068      * \endchinese
00069      * \english
00070      * Returns an interface for working with the features of the current
00071      * installation.
00072      * @return An interface for working with the features of the current
00073      * installation.
00074      * \endenglish
00075      */
00076     FeatureModelPtr getFeatureModel();
00077
00078     /**
00079      * @ingroup ApplicationApi
00080      * @ref TcpModel
00081      * \chinese
00082      * TCP
00083      * @return TCP
00084      * \endchinese
00085      * \english
00086      * Returns an interface for accessing the TCPs of the current installation.
00087      * @return An interface for accessing the TCPs of the current installation.
00088      * \endenglish
00089      */
00090     TcpModelPtr getTcpModel();
00091
00092     /**
00093      * @ingroup ApplicationApi
00094      * @ref PayloadModel
00095      * \chinese
00096      * AuboScope
00097      *
00098      * @return AuboScope
00099      * \endchinese
00100      * \english
00101      * Gets an interface which provides access the payloads in the current

```

```

00102     * AudoScope installation.
00103     *
00104     * @return An interface for accessing the payloads in the current AudoScope
00105     * installation.
00106     * \endenglish
00107     */
00108 PayloadModelPtr getPayloadModel();
00109
00110 /**
00111  * @ingroup ApplicationApi
00112  * @ref VariableModel
00113  * \chinese
00114  *
00115  * @return
00116  * \endchinese
00117  * \english
00118  * Returns an interface for working with variables.
00119  * @return An interface for working with variables.
00120  * \endenglish
00121  */
00122 VariableModelPtr getVariableModel();
00123
00124 /**
00125  * @ingroup ApplicationApi
00126  * @ref ValueFactory
00127  * \chinese
00128  *
00129  * @return
00130  * \endchinese
00131  * \english
00132  * Returns an interface for accessing various factories capable of creating
00133  * value objects.
00134  * @return An interface for accessing various factories capable of creating
00135  * value objects.
00136  * \endenglish
00137  */
00138 ValueFactoryPtr getValueFactory();
00139
00140 // /**
00141 //  * @return An interface with access to various queryable-only system
00142 //  resources
00143 //  */
00144 // ResourceModel getResourceModel();
00145
00146 /**
00147  * \chinese
00148  * { @link InstallationNodeContribution }
00149  *
00150  * @param installationType
00151  * @param <T> { @link InstallationNodeContribution }
00152  * @return
00153  * \endchinese
00154  * \english
00155  * This method can be used to get a specific { @link
00156  * InstallationNodeContribution } instance.
00157  *
00158  * @param installationType The class of the installation node
00159  * contribution to return.
00160  * @param <T> The generic for specifying the { @link
00161  * InstallationNodeContribution }.
00162  * @return The installation node instance.
00163  * \endenglish
00164  */
00165 template <typename T>
00166 std::shared_ptr<T> getInstallationNode()
00167 {
00168     return std::dynamic_pointer_cast<T>(
00169         getInstallationNode(typeid(T).name()));
00170 }
00171 InstallationNodeContributionPtr getInstallationNode(
00172     const char *typeidOfClass);
00173
00174 /**
00175  * @ingroup ApplicationApi
00176  * @ref DeviceManager
00177  * \chinese
00178  * { @link DeviceManager }
00179  *
00180  * @param deviceManagerClass
00181  * @param <T>
00182  * @return
00183  * \endchinese
00184  * \english
00185  * This method can be used to get a specific { @link DeviceManager }
00186  * instance.
00187  *
00188  * @param deviceManagerClass The class for the device manager to return.

```

```

00189  * @param <T> The generic type of class to return.
00190  * @return An interface for registering and working with a specific
00191  * device.
00192  * \endenglish
00193  */
00194  DeviceManagerPtr getDeviceManager();
00195
00196  /**
00197  * \chinese
00198  *      SDK
00199  * @return
00200  * \endchinese
00201  * \english
00202  * Returns the robot proxy, providing an interaction interface with the
00203  * underlying robot SDK.
00204  * @return The robot proxy pointer.
00205  * \endenglish
00206  */
00207  aubo_sdk::RobotProxyPtr getRobotProxy();
00208
00209 protected:
00210  ApplicationApi();
00211
00212 private:
00213  friend class DataSwitch;
00214  void *d_{ nullptr };
00215 };
00216
00217 // namespace aubo_scope
00218 } // namespace arcs
00219
00220 #endif // AUBO_SCOPE_APPLICATION_API_H

```

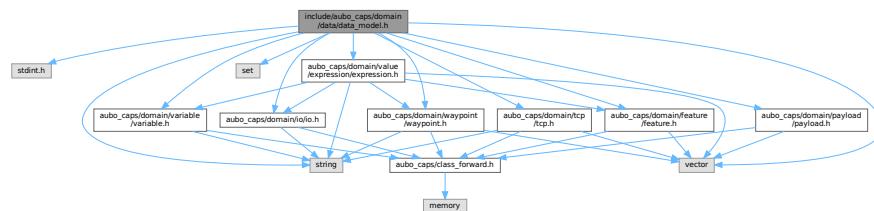
12.62 include/auco_caps/domain/data/data_model.h File Reference

```

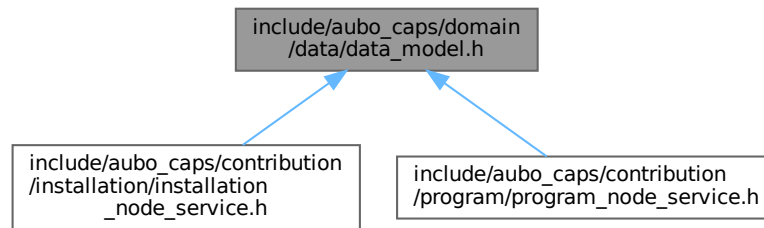
#include <stdint.h>
#include <string>
#include <vector>
#include <set>
#include <auco_caps/domain/feature/feature.h>
#include <auco_caps/domain/variable/variable.h>
#include <auco_caps/domain/tcp/tcp.h>
#include <auco_caps/domain/io/io.h>
#include <auco_caps/domain/waypoint/waypoint.h>
#include <auco_caps/domain/payload/payload.h>
#include <auco_caps/domain/value/expression/expression.h>

```

Include dependency graph for data_model.h:



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



Classes

- class [arcs::aubo_scope::DataModel](#)
[DataModel](#) This interface is used for storing and managing data that represents the current configuration of, e.g.

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([DataModel](#))

12.63 data_model.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_DATA_MODEL_H
00002 #define AUBO_SCOPE_DATA_MODEL_H
00003
00004 #include <stdint.h>
00005 #include <string>
00006 #include <vector>
00007 #include <set>
00008
00009 #include <aubo_caps/domain/feature/feature.h>
00010 #include <aubo_caps/domain/variable/variable.h>
00011 #include <aubo_caps/domain/tcp/tcp.h>
00012 #include <aubo_caps/domain/io/io.h>
00013 #include <aubo_caps/domain/waypoint/waypoint.h>
00014 #include <aubo_caps/domain/payload/payload.h>
00015 #include <aubo_caps/domain/value/expression/expression.h>
00016
00017 namespace arcs {
00018 namespace aubo_scope {
00019 ARCS_CLASS_FORWARD(DataModel);
00020
00021 /**
00022  * \chinese
00023  *
00024  * { @link ProgramNodeContribution }
00025  * { @link InstallationNodeContribution }
00026  *
00027  */
  
```

```

00028 *
00029 * <i>myBool</i>      <code>true</code>
00030 * <code>Angle</code>      <code>Angle</code>
00031 * <code>true</code>
00032 *
00033 *   AuboCap      <code>DataModel</code>
00034 *   AuboScope
00035 *
00036 *           <code>DataModel</code>
00037 *           HTML
00038 * <code>DataModel</code>      /      Qt   AuboStudio
00039 *   {@link UndoRedoManager}      /
00040 *
00041 *
00042 * <i>myAngle</i>      <code>TCP</code>      <i>myAngle</i>
00043 *   {@link #get(std::string key, Angle default_value)}
00044 *           <code>default_value</code>
00045 * \endchinese
00046 * \english
00047 * DataModel
00048 * This interface is used for storing and managing data that represents the
00049 * current configuration of, e.g. a
00050 * {@link ProgramNodeContribution} or {@link InstallationNodeContribution}.
00051 * Methods for adding, removing, retrieving and changing values in a dictionary
00052 * are provided.
00053 *
00054 * Setting a new value for a key already in use, will overwrite the current
00055 * value with the new value. This happens regardless of the type of value (e.g.
00056 * storing the value <code>true</code> under the key <i>myBool</i>
00057 * and afterwards storing an <code>Angle</code> object under the same key will
00058 * overwrite the value <code>true</code> with the provided <code>Angle</code>
00059 * object).
00060 *
00061 * A auboCap installation screen has an underlying <code>DataModel</code>
00062 * object. That object is saved and loaded along with each AuboScope
00063 * installation.
00064 *
00065 * Similarly, each contributed program node instance has an underlying
00066 * <code>DataModel</code> object. That object is saved and loaded along with the
00067 * program where the node occurs. Undo/redo actions are supported for all
00068 * modifications to the <code>DataModel</code> object in HTML-based program node
00069 * contributions. Qt-based aubo_studio plugins must use the {@link
00070 * UndoRedoManager} to record the changes on the undo/redo stack.
00071 *
00072 * When retrieving an object, both key and object type must match what was
00073 * previously stored. This means that if a
00074 * <code>TCP</code> object was stored using the key <i>myAngle</i>, then
00075 * attempting to retrieve it using
00076 * {@link #get(std::string key, Angle default_value)} with the key
00077 * <i>myAngle</i> will not return the stored value, since the types do not
00078 * match. Instead the provided <code>default_value</code> will be returned.
00079 * \endenglish
00080 *
00081 */
00082 class ARCS_ABI_EXPORT DataModel
00083 {
00084 public:
00085     DataModel(DataModel &f);
00086     DataModel(DataModel &&f);
00087     ~DataModel();
00088
00089     /**
00090     * \chinese
00091     * <code>bool</code>
00092     *
00093     * @param key      <code>null</code> <code>std::string</code>
00094     * @param value
00095     *
00096     * @throws IllegalArgumentException <code>null</code> <code>std::string</code>
00097     * @throws IllegalStateException   Qt-based AuboCap
00098     * {@link UndoableChanges}   {@link UndoRedoManager}
00099     * \endchinese
00100     * \english
00101     * Assign a <code>bool</code> value to the specified key.
00102     *
00103     * @param key key in the data model (not <code>null</code> and not an empty
00104     * <code>std::string</code>).
00105     * @param value value assigned to key.
00106     *
00107     * @throws IllegalArgumentException if the key is <code>null</code> or an
00108     * empty <code>std::string</code>.
00109     * @throws IllegalStateException if called from a Qt-based AuboCap
00110     * program node outside of an {@link UndoableChanges} scope (see also {@link
00111     * UndoRedoManager}).
00112     * \endenglish
00113     */
00114     void set(const std::string &key, bool value);

```



```

00115
00116 /**
00117  * \chinese
00118  *      <code>bool</code>
00119  *
00120  * @param key
00121  * @param default_value
00122  * @return      default_value
00123  * \endchinese
00124  * \english
00125  * Get the <code>bool</code> value assigned to the specified key.
00126  *
00127  * @param key key in the data model.
00128  * @param default_value value to be returned, if key does not exist.
00129  * @return the value assigned to the key. If not exist, default_value is
00130  * returned.
00131  * \endenglish
00132  */
00133 bool get(const std::string &key, bool default_value) const;
00134
00135 /**
00136  * \chinese
00137  *      <code>int</code>
00138  *
00139  * @param key
00140  * @param value
00141  *
00142  * @throws IllegalArgumentException <code>null</code>
00143  * @throws IllegalStateException { @link UndoableChanges }
00144  * \endchinese
00145  * \english
00146  * Assign an <code>int</code> value to the specified key.
00147  *
00148  * @param key key in the data model (not <code>null</code> and not an empty
00149  * <code>std::string</code>).
00150  * @param value value assigned to key.
00151  *
00152  * @throws IllegalArgumentException if the key is <code>null</code> or an
00153  * empty <code>std::string</code>.
00154  * @throws IllegalStateException if called from a Qt-based AuboCap
00155  * program node outside of an { @link UndoableChanges } scope (see also { @link
00156  * UndoRedoManager }).
00157  * \endenglish
00158  */
00159 void set(const std::string &key, int value);
00160
00161 /**
00162  * \chinese
00163  *      <code>int</code>
00164  *
00165  * @param key
00166  * @param default_value
00167  * @return
00168  * \endchinese
00169  * \english
00170  * Get the <code>int</code> value assigned to the specified key.
00171  *
00172  * @param key key in the data model.
00173  * @param default_value value to be returned, if key does not exist.
00174  * @return the value assigned to the key. If not exist, default_value is
00175  * returned.
00176  * \endenglish
00177  */
00178 int get(const std::string &key, int default_value) const;
00179
00180 /**
00181  * \chinese
00182  *      <code>long</code>
00183  * \endchinese
00184  * \english
00185  * Assign a <code>long</code> value to the specified key.
00186  *
00187  * @param key key in the data model (not <code>null</code> and not an empty
00188  * <code>std::string</code>).
00189  * @param value value assigned to key.
00190  *
00191  * @throws IllegalArgumentException if the key is <code>null</code> or an
00192  * empty <code>std::string</code>.
00193  * @throws IllegalStateException if called from a Qt-based AuboCap
00194  * program node outside of an { @link UndoableChanges } scope (see also { @link
00195  * UndoRedoManager }).
00196  * \endenglish
00197  */
00198 void set(const std::string &key, long value);
00199 void set(const std::string &key, uint64_t value);
00200
00201 /**

```

```

00202 * \chinese
00203 * <code>long</code>
00204 * \endchinese
00205 * \english
00206 * Get the <code>long</code> value assigned to the specified key.
00207 *
00208 * @param key key in the data model.
00209 * @param default_value value to be returned, if key does not exist.
00210 * @return the value assigned to the key. If not exist, default_value is
00211 * returned.
00212 * \endenglish
00213 */
00214 long get(const std::string &key, long default_value) const;
00215 uint64_t get(const std::string &key, uint64_t default_value) const;
00216
00217 /**
00218 * \chinese
00219 * <code>float</code>
00220 * \endchinese
00221 * \english
00222 * Assign a <code>float</code> value to the specified key.
00223 *
00224 * @param key key in the data model (not <code>null</code> and not an empty
00225 * <code>std::string</code>).
00226 * @param value value assigned to key.
00227 *
00228 * @throws IllegalArgumentException if the key is <code>null</code> or an
00229 * empty <code>std::string</code>.
00230 * @throws IllegalStateException if called from a Qt-based AuboCap
00231 * program node outside of an {@link UndoableChanges} scope (see also {@link
00232 * UndoRedoManager}).
00233 * \endenglish
00234 */
00235 void set(const std::string &key, float value);
00236
00237 /**
00238 * \chinese
00239 * <code>float</code>
00240 * \endchinese
00241 * \english
00242 * Get the <code>float</code> value assigned to the specified key.
00243 *
00244 * @param key key in the data model.
00245 * @param default_value value to be returned, if key does not exist.
00246 * @return the value assigned to the key. If not exist, default_value is
00247 * returned.
00248 * \endenglish
00249 */
00250 float get(const std::string &key, float default_value) const;
00251
00252 /**
00253 * \chinese
00254 * <code>double</code>
00255 * \endchinese
00256 * \english
00257 * Assign a <code>double</code> value to the specified key.
00258 *
00259 * @param key key in the data model (not <code>null</code> and not an empty
00260 * <code>std::string</code>).
00261 * @param value value assigned to key.
00262 *
00263 * @throws IllegalArgumentException if the key is <code>null</code> or an
00264 * empty <code>std::string</code>.
00265 * @throws IllegalStateException if called from a Qt-based AuboCap
00266 * program node outside of an {@link UndoableChanges} scope (see also {@link
00267 * UndoRedoManager}).
00268 * \endenglish
00269 */
00270 void set(const std::string &key, double value);
00271
00272 /**
00273 * \chinese
00274 * <code>double</code>
00275 * \endchinese
00276 * \english
00277 * Get the <code>double</code> value assigned to the specified key.
00278 *
00279 * @param key key in the data model.
00280 * @param default_value value to be returned, if key does not exist.
00281 * @return the value assigned to the key. If not exist, default_value is
00282 * returned.
00283 * \endenglish
00284 */
00285 double get(const std::string &key, double default_value) const;
00286
00287 /**
00288 * \chinese

```

```

00289 * <code>std::string</code>
00290 * \endchinese
00291 * \english
00292 * Assign a <code>std::string</code> value to the specified key.
00293 *
00294 * @param key key in the data model (not <code>null</code> and not an empty
00295 * <code>std::string</code>).
00296 * @param value value assigned to key.
00297 *
00298 * @throws IllegalArgumentException if the key is <code>null</code> or an
00299 * empty <code>std::string</code>.
00300 * @throws IllegalStateException if called from a Qt-based AuboCap
00301 * program node outside of an {@link UndoableChanges} scope (see also {@link
00302 * UndoRedoManager}).
00303 * \endenglish
00304 */
00305 void set(const std::string &key, const std::string &value);
00306
00307 /**
00308 * \chinese
00309 * <code>std::string</code>
00310 * \endchinese
00311 * \english
00312 * Get the <code>std::string</code> value assigned to the specified key.
00313 *
00314 * @param key key in the data model.
00315 * @param default_value value to be returned, if key does not exist.
00316 * @return the value assigned to the key. If not exist, default_value is
00317 * returned.
00318 * \endenglish
00319 */
00320 std::string get(const std::string &key,
00321                const std::string &default_value) const;
00322 /**
00323 * \chinese
00324 * <code>Variable</code>
00325 * \endchinese
00326 * \english
00327 * Assign a <code>Variable</code> value to the specified key.
00328 *
00329 * @param key key in the data model (not <code>null</code> and not an empty
00330 * <code>std::string</code>).
00331 * @param value value assigned to key.
00332 *
00333 * @throws IllegalArgumentException if the key is <code>null</code> or an
00334 * empty <code>std::string</code>.
00335 * @throws IllegalStateException if called from a Qt-based AuboCap
00336 * program node outside of an {@link UndoableChanges} scope (see also {@link
00337 * UndoRedoManager}).
00338 * \endenglish
00339 */
00340 void set(const std::string &key, VariablePtr value);
00341 void set(const std::string &key, ExpressionPtr value);
00342 void set(const std::string &key, PayloadPtr value);
00343 void set(const std::string &key, IoPtr value);
00344 void set(const std::string &key, WaypointPtr value);
00345
00346 /**
00347 * \chinese
00348 * <code>Variable</code>
00349 * @return
00350 * \endchinese
00351 * \english
00352 * Get the <code>Variable</code> value assigned to the specified key.
00353 *
00354 * @param key key in the data model.
00355 * @param default_value value to be returned, if key does not exist.
00356 * @return the value assigned to the key. If not exist, default_value is
00357 * returned.
00358 * \endenglish
00359 */
00360 VariablePtr get(const std::string &key, VariablePtr default_value) const;
00361 ExpressionPtr get(const std::string &key,
00362                  ExpressionPtr default_value) const;
00363 PayloadPtr get(const std::string &key, PayloadPtr default_value) const;
00364 IoPtr get(const std::string &key, IoPtr default_value) const;
00365 WaypointPtr get(const std::string &key, WaypointPtr default_value) const;
00366
00367 /**
00368 * \chinese
00369 * <code>bool[]</code>
00370 * \endchinese
00371 * \english
00372 * Assign a <code>bool[]</code> as value to the specified key.
00373 * @param key key in the data model (not <code>null</code>) and not an empty
00374 * <code>std::string</code>).
00375 * @param value value assigned to key.

```

```

00376 *
00377 * @throws IllegalArgumentException if the key is <code>null</code> or an
00378 * empty <code>std::string</code>.
00379 * @throws IllegalStateException if called from a Qt-based AuboCap
00380 * program node outside of an {@link UndoableChanges} scope (see also {@link
00381 * UndoRedoManager}).
00382 * \endenglish
00383 */
00384 void set(const std::string &key, const std::vector<bool> &value);
00385
00386 /**
00387 * \chinese
00388 * <code>bool[]</code>
00389 * \endchinese
00390 * \english
00391 * Get the <code>bool[]</code> as value assigned to the specified key.
00392 *
00393 * @param key key in the data model.
00394 * @param default_value value to be returned, if key does not exist.
00395 * @return the value assigned to the key. If not exist, default_value is
00396 * returned.
00397 * \endenglish
00398 */
00399 std::vector<bool> get(const std::string &key,
00400                     const std::vector<bool> &default_value) const;
00401
00402 /**
00403 * \chinese
00404 * <code>int[]</code>
00405 * \endchinese
00406 * \english
00407 * Assign a <code>int[]</code> as value to the specified key.
00408 *
00409 * @param key key in the data model (not <code>null</code> and not an empty
00410 * <code>std::string</code>).
00411 * @param value value assigned to key.
00412 *
00413 * @throws IllegalArgumentException if the key is <code>null</code> or an
00414 * empty <code>std::string</code>.
00415 * @throws IllegalStateException if called from a Qt-based AuboCap
00416 * program node outside of an {@link UndoableChanges} scope (see also {@link
00417 * UndoRedoManager}).
00418 * \endenglish
00419 */
00420 void set(const std::string &key, const std::vector<int> &value);
00421
00422 /**
00423 * \chinese
00424 * <code>int[]</code>
00425 * \endchinese
00426 * \english
00427 * Get the <code>int[]</code> as value assigned to the specified key.
00428 *
00429 * @param key key in the data model.
00430 * @param default_value value to be returned, if key does not exist.
00431 * @return the value assigned to the key. If not exist, default_value is
00432 * returned.
00433 * \endenglish
00434 */
00435 std::vector<int> get(const std::string &key,
00436                     const std::vector<int> &default_value) const;
00437
00438 /**
00439 * \chinese
00440 * <code>long[]</code>
00441 * \endchinese
00442 * \english
00443 * Assign a <code>long[]</code> as value to the specified key.
00444 *
00445 * @param key key in the data model (not <code>null</code> and not an empty
00446 * <code>std::string</code>).
00447 * @param value value assigned to key.
00448 *
00449 * @throws IllegalArgumentException if the key is <code>null</code> or an empty
00450 * <code>std::string</code>.
00451 * @throws IllegalStateException if called from a Qt-based AuboCap
00452 * program node outside of an {@link UndoableChanges} scope (see also {@link
00453 * UndoRedoManager}).
00454 * \endenglish
00455 */
00456 void set(const std::string &key, const std::vector<long> &value);
00457 void set(const std::string &key, const std::vector<uint64_t> &value);
00458
00459 /**
00460 * \chinese
00461 * <code>long[]</code>
00462 * \endchinese

```

```

00463 * \english
00464 * Get the <code>long[]</code> as value assigned to the specified key.
00465 *
00466 * @param key key in the data model.
00467 * @param default_value value to be returned, if key does not exist.
00468 * @return the value assigned to the key. If not exist, default_value is
00469 * returned.
00470 * \endenglish
00471 */
00472 std::vector<long> get(const std::string &key,
00473                     const std::vector<long> &default_value) const;
00474 std::vector<uint64_t> get(const std::string &key,
00475                          const std::vector<uint64_t> &default_value) const;
00476
00477 /**
00478 * \chinese
00479 * <code>float[]</code>
00480 * \endchinese
00481 * \english
00482 * Assign a <code>float[]</code> as value to the specified key.
00483 *
00484 * @param key key in the data model (not <code>null</code> and not an empty
00485 * <code>std::string</code>).
00486 * @param value value assigned to key.
00487 *
00488 * @throws IllegalArgumentException if the key is <code>null</code> or an
00489 * empty <code>std::string</code>.
00490 * @throws IllegalStateException if called from a Qt-based AboCap
00491 * program node outside of an {@link UndoableChanges} scope (see also {@link
00492 * UndoRedoManager}).
00493 * \endenglish
00494 */
00495 void set(const std::string &key, const std::vector<float> &value);
00496
00497 /**
00498 * \chinese
00499 * <code>float[]</code>
00500 * \endchinese
00501 * \english
00502 * Get the <code>float[]</code> as value assigned to the specified key.
00503 *
00504 * @param key key in the data model.
00505 * @param default_value value to be returned, if key does not exist.
00506 * @return the value assigned to the key. If not exist, default_value is
00507 * returned.
00508 * \endenglish
00509 */
00510 std::vector<float> get(const std::string &key,
00511                      const std::vector<float> &default_value) const;
00512
00513 /**
00514 * \chinese
00515 * <code>double[]</code>
00516 * \endchinese
00517 * \english
00518 * Assign a <code>double[]</code> as value to the specified key.
00519 *
00520 * @param key key in the data model (not <code>null</code> and not an empty
00521 * <code>std::string</code>).
00522 * @param value value assigned to key.
00523 *
00524 * @throws IllegalArgumentException if the key is <code>null</code> or an
00525 * empty <code>std::string</code>.
00526 * @throws IllegalStateException if called from a Qt-based AboCap
00527 * program node outside of an {@link UndoableChanges} scope (see also {@link
00528 * UndoRedoManager}).
00529 * \endenglish
00530 */
00531 void set(const std::string &key, const std::vector<double> &value);
00532
00533 /**
00534 * \chinese
00535 * <code>double[]</code>
00536 * \endchinese
00537 * \english
00538 * Get the <code>double[]</code> as value assigned to the specified key.
00539 *
00540 * @param key key in the data model.
00541 * @param default_value value to be returned, if key does not exist.
00542 * @return the value assigned to the key. If not exist, default_value is
00543 * returned.
00544 * \endenglish
00545 */
00546 std::vector<double> get(const std::string &key,
00547                       const std::vector<double> &default_value) const;
00548
00549 /**

```

```

00550 * \chinese
00551 * <code>std::string[]</code>
00552 * \endchinese
00553 * \english
00554 * Assign a <code>std::string[]</code> as value to the specified key.
00555 *
00556 * @param key key in the data model (not <code>null</code> and not an empty
00557 * <code>std::string</code>).
00558 * @param value value assigned to key.
00559 *
00560 * @throws IllegalArgumentException if the key is <code>null</code> or an
00561 * empty <code>std::string</code>.
00562 * @throws IllegalStateException if called from a Qt-based AuboCap
00563 * program node outside of an {@link UndoableChanges} scope (see also {@link
00564 * UndoRedoManager}).
00565 * \endenglish
00566 */
00567 void set(const std::string &key, const std::vector<std::string> &value);
00568
00569 /**
00570 * \chinese
00571 * <code>std::string[]</code>
00572 * \endchinese
00573 * \english
00574 * Get the <code>std::string[]</code> as value assigned to the specified
00575 * key.
00576 *
00577 * @param key key in the data model.
00578 * @param default_value value to be returned, if key does not exist.
00579 * @return the value assigned to the key. If not exist, default_value is
00580 * returned.
00581 * \endenglish
00582 */
00583 std::vector<std::string> get(
00584     const std::string &key,
00585     const std::vector<std::string> &default_value) const;
00586
00587 /**
00588 * \chinese
00589 * <code>TCP</code>
00590 * \endchinese
00591 * \english
00592 * Assign a <code>TCP</code> value to the specified key.
00593 *
00594 * @param key key in the data model (not <code>null</code> and not an empty
00595 * <code>std::string</code>).
00596 * @param value value assigned to key.
00597 *
00598 * @throws IllegalArgumentException if the key is <code>null</code> or an
00599 * empty <code>std::string</code>.
00600 * @throws IllegalStateException if called from a Qt-based AuboCap
00601 * program node outside of an {@link UndoableChanges} scope (see also {@link
00602 * UndoRedoManager}).
00603 * \endenglish
00604 */
00605 void set(const std::string &key, TCPPtr value);
00606
00607 /**
00608 * \chinese
00609 * <code>TCP</code>
00610 * \endchinese
00611 * \english
00612 * Get the <code>TCP</code> value assigned to the specified key.
00613 *
00614 * @param key key in the data model.
00615 * @param default_value value to be returned, if key does not exist.
00616 * @return the value assigned to the key. If not exist, default_value is
00617 * returned.
00618 * \endenglish
00619 */
00620 TCPPtr get(const std::string &key, TCPPtr default_value) const;
00621
00622 /**
00623 * \chinese
00624 * <code>Feature</code>
00625 * \endchinese
00626 * \english
00627 * Assign a <code>Feature</code> value to the specified key.
00628 *
00629 * @param key key in the data model (not <code>null</code> and not an empty
00630 * <code>std::string</code>).
00631 * @param value value assigned to key.
00632 *
00633 * @throws IllegalArgumentException if the key is <code>null</code> or an
00634 * empty <code>std::string</code>.
00635 * @throws IllegalStateException if called from a Qt-based AuboCap
00636 * program node outside of an {@link UndoableChanges} scope (see also {@link

```

```

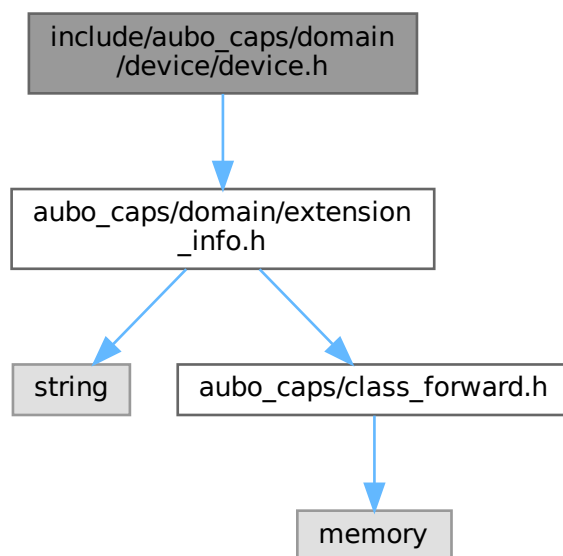
00637     * UndoRedoManager}).
00638     * \endenglish
00639     */
00640 void set(const std::string &key, FeaturePtr value);
00641
00642 /**
00643  * \chinese
00644  * <code>Feature</code>
00645  * \endchinese
00646  * \english
00647  * Get the <code>Feature</code> value assigned to the specified key.
00648  *
00649  * @param key key in the data model.
00650  * @param default_value value to be returned, if key does not exist.
00651  * @return the value assigned to the key. If not exist, default_value is
00652  * returned.
00653  * \endenglish
00654  */
00655 FeaturePtr get(const std::string &key, FeaturePtr default_value) const;
00656
00657 /**
00658  * \chinese
00659  *
00660  * @return
00661  * \endchinese
00662  * \english
00663  * Get a set of all the keys in the data model.
00664  *
00665  * @return A Set of keys.
00666  * \endenglish
00667  */
00668 std::set<std::string> getKeys() const;
00669
00670 /**
00671  * \chinese
00672  *
00673  * @param key
00674  * @return <code>true</code> <code>false</code>
00675  * \endchinese
00676  * \english
00677  * Check if a key is present in the data model.
00678  *
00679  * @param key key in the data model (not <code>null</code> and not an empty
00680  * <code>std::string</code>).
00681  * @return <code>true</code>, if key exist, otherwise <code>false</code>.
00682  * \endenglish
00683  */
00684 bool isSet(const std::string &key) const;
00685
00686 /**
00687  * \chinese
00688  *
00689  * @param key
00690  * @return <code>true</code> <code>false</code>
00691  * @throws IllegalStateException Qt-based AuboCap
00692  * {@link UndoableChanges}
00693  * \endchinese
00694  * \english
00695  * Remove a key-value pair from the data model.
00696  *
00697  * @param key key in the data model (not <code>null</code> and not an empty
00698  * <code>std::string</code>).
00699  * @return <code>true</code>, if succeed, otherwise <code>false</code>.
00700  * @throws IllegalStateException if called from a Qt-based AuboCap
00701  * program node outside of an {@link UndoableChanges} scope (see also {@link
00702  * UndoRedoManager}).
00703  * \endenglish
00704  */
00705 bool remove(const std::string &key);
00706
00707 private:
00708 friend class DataSwitch;
00709 DataModel();
00710 void *d_{ nullptr };
00711 };
00712
00713 } // namespace aubo_scope
00714 } // namespace arcs
00715
00716 #endif

```

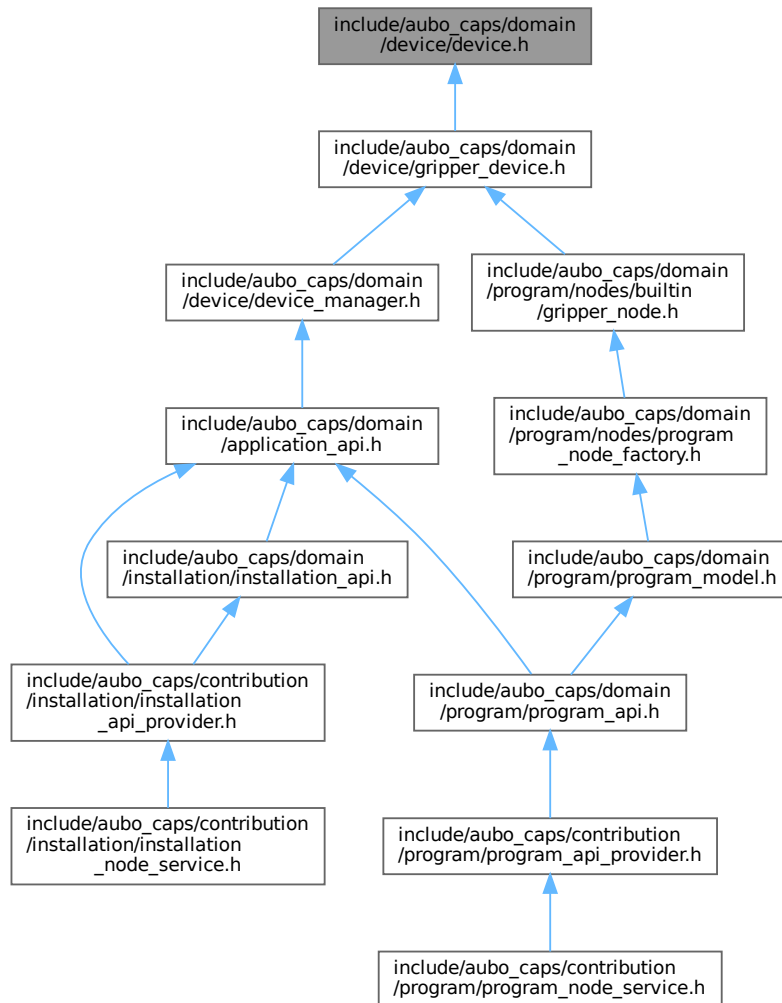
12.64 include/aubo_caps/domain/device/device.h File Reference

`#include <aubo_caps/domain/extension_info.h>`

Include dependency graph for device.h:



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



Classes

- class [arcs::aubo_scope::Device](#)
[Device](#) This base interface represents a device supported by AuboScope, e.g.

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(Device\)](#)

12.65 device.h

[Go to the documentation of this file.](#)

```

00001 #include <aubo_caps/domain/extension_info.h>
00002
00003 namespace arcs {
00004 namespace aubo_scope {
00005 ARCS_CLASS_FORWARD(Device);
00006
00007 /**
00008  * \chinese
00009  *
00010  *     AuboScope
00011  * \endchinese
00012  * \english
00013  * Device
00014  * This base interface represents a device supported by AuboScope, e.g. a
00015  * gripping device.
00016  * \endenglish
00017  */
00018 class ARCS_ABI_EXPORT Device
00019 {
00020 public:
00021     virtual ~Device() = 0;
00022     /**
00023      * \chinese
00024      *     / AuboScope AuboCap
00025      * @return AuboCap
00026      * \endchinese
00027      * \english
00028      * Get information about the AuboCap that added/contributed this gripper
00029      * device to AuboScope.
00030      *
00031      * @return AuboCap information for this gripper device.
00032      * \endenglish
00033      */
00034     virtual AuboCapInfoPtr getContributorInfo() = 0;
00035
00036     /**
00037      * \chinese
00038      * <p>
00039      *     id
00040      * </p>
00041      * id " "
00042      * @return
00043      * \endchinese
00044      * \english
00045      * <p>
00046      * Get the unique identifier (id) for this gripper. This can be used to
00047      * identify this specific gripper among the set of individual grippers
00048      * available in a specific multi-gripper device.
00049      * </p>
00050      *
00051      * Note that this id is not "globally" unique, but only unique for the
00052      * individual grippers belonging to a specific multi-gripper device.
00053      *
00054      * @return The unique identifier of this gripper.
00055      * \endenglish
00056      */
00057     virtual std::string getId() = 0;
00058
00059     /**
00060      * \chinese
00061      * <p>
00062      *     AuboScope UI
00063      * </p>
00064      * <b> </b>
00065      *     AuboScope AuboCap
00066      * @return AuboScope
00067      * \endchinese
00068      * \english
00069      * <p>
00070      * The name of the gripper displayed in the AuboScope UI, e.g. in the
00071      * Gripper toolbar and program node.
00072      * </p>
00073      *
00074      * <b>Note:</b> This name must not be used to uniquely identify a specific
00075      * individual gripper, since the name might be translated (depending on the
00076      * selected language in AuboScope) or be changed between versions of the
00077      * AuboCap that added the gripper.
00078      *
00079      * @return The name of this gripper displayed in AuboScope.
00080      * \endenglish
00081      */
00082     virtual std::string getDisplayName() = 0;

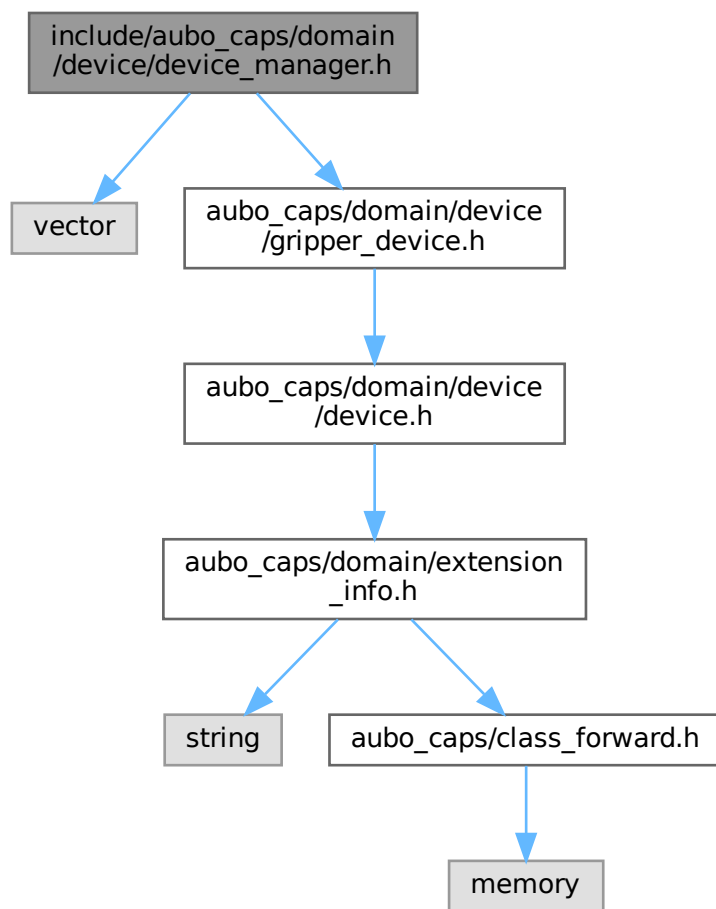
```

```
00083
00084 /**
00085  * \chinese
00086  *      AuboScope      AuboCap
00087  *
00088  * @return      AuboScope  <code>true</code>
00089  * <code>false</code>
00090  * \endchinese
00091  * \english
00092  * A device cannot be guaranteed to be present in AuboScope. A device will
00093  * not be present if the AuboCap that added the device is not installed.
00094  * This method can be used to determine, if the device is present.
00095  *
00096  * @return <code>true</code> if this gripper device is present in AuboScope,
00097  * otherwise <code>false</code>.
00098  * \endenglish
00099  */
00100 virtual bool isResolvable() = 0;
00101 };
00102 } // namespace aubo_scope
00103 } // namespace arcs
```

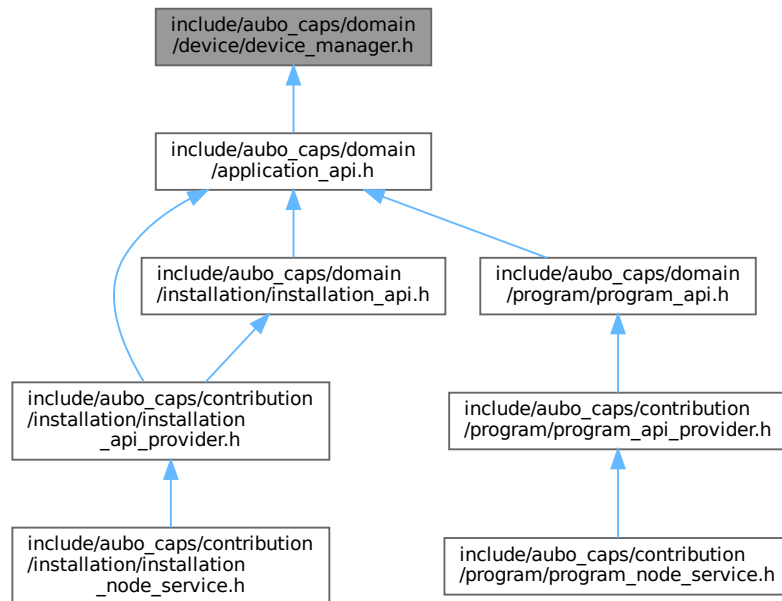
12.66 include/aubo_caps/domain/device/device_manager.h File Reference

```
#include <vector>
#include <aubo_caps/domain/device/gripper_device.h>
```

Include dependency graph for device_manager.h:



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



Classes

- class [arcs::aubo_scope::DeviceManager](#)
[DeviceManager](#) Base interface for device managers.

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([DeviceManager](#))

12.67 device_manager.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_DEVICE_MANAGER_H
00002 #define AUBO_SCOPE_DEVICE_MANAGER_H
00003
00004 #include <vector>
00005 #include <aubo_caps/domain/device/gripper_device.h>
00006
00007 namespace arcs {
00008 namespace aubo_scope {
00009 ARCS_CLASS_FORWARD(DeviceManager);
00010
00011 /**

```


Classes

- class `arcs::aubo_scope::DeviceRegistrationManager`
`DeviceRegistrationManager` Base interface for device registration managers.

Namespaces

- namespace `arcs`
- namespace `arcs::aubo_scope`

Functions

- `arcs::aubo_scope::ARCS_CLASS_FORWARD` (`DeviceRegistrationManager`)

12.69 device_registration_manager.h

[Go to the documentation of this file.](#)

```

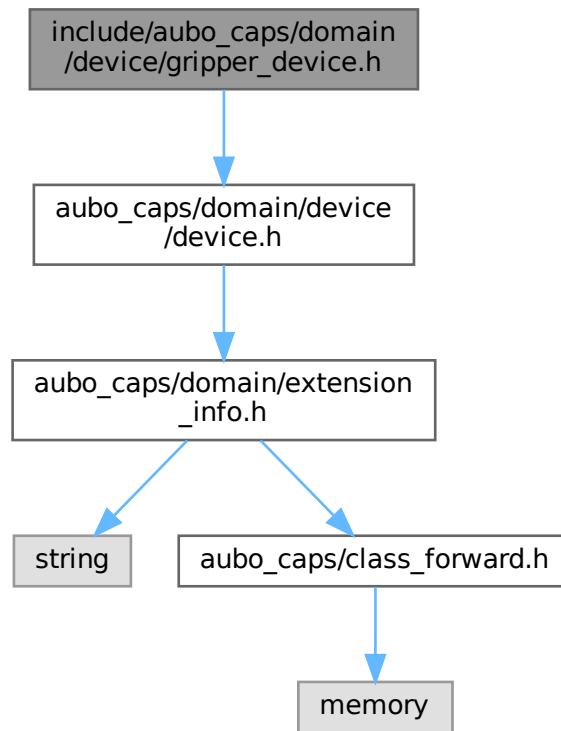
00001 #ifndef AUBO_SCOPE_DEVICE_REGISTRATION_MANAGER_H
00002 #define AUBO_SCOPE_DEVICE_REGISTRATION_MANAGER_H
00003
00004 #include <aubo_caps/contribution/driver/gripper/gripper_contribution.h>
00005 #include <aubo_caps/contribution/driver/screw_driver/screw_driver_contribution.h>
00006
00007 namespace arcs {
00008 namespace aubo_scope {
00009 ARCS_CLASS_FORWARD(DeviceRegistrationManager);
00010
00011 /**
00012  * \chinese
00013  *
00014  *
00015  *
00016  * / AuboScope
00017  * \endchinese
00018  * \english
00019  * DeviceRegistrationManager
00020  * Base interface for device registration managers.
00021  *
00022  * A device registration manager is used for registering a program/installation
00023  * node contribution/service as conformant with a specific device type supported
00024  * by AuboScope.
00025  * \endenglish
00026  */
00027 class ARCS_ABI_EXPORT DeviceRegistrationManager
00028 {
00029 public:
00030     virtual ~DeviceRegistrationManager() = default;
00031
00032     /**
00033      * \chinese
00034      *
00035      * @param contribution
00036      * \endchinese
00037      * \english
00038      * Register a gripper contribution.
00039      * @param contribution the gripper contribution to register.
00040      * \endenglish
00041      */
00042     virtual void registerGripperContribution(
00043         GripperContributionPtr contribution) = 0;
00044
00045     /**
00046      * \chinese
00047      *
00048      * @param contribution
00049      * \endchinese
00050      * \english
00051      * Register a screwdriver contribution.
00052      * @param contribution the screwdriver contribution to register.
00053      * \endenglish
00054      */
00055     virtual void registerScrewDriverContribution(
00056         ScrewdriverContributionPtr contribution) = 0;
00057 };
00058 } // namespace aubo_scope
00059 } // namespace arcs
00060
00061 #endif

```

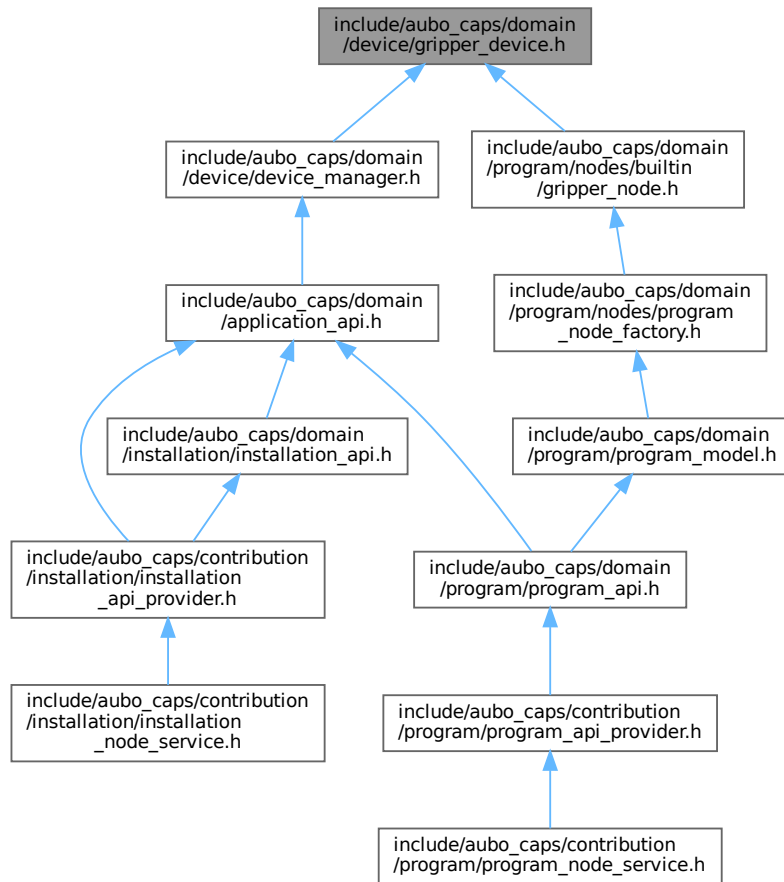
12.70 include/aubo_caps/domain/device/gripper_device.h File Reference

#include <aubo_caps/domain/device/device.h>

Include dependency graph for gripper_device.h:



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



Classes

- class [arcs::aubo_scope::GripperDevice](#)
[GripperDevice](#) This interface represents a gripper device in AuboScope.

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(GripperDevice\)](#)

12.71 gripper_device.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_GRIPPER_DEVICE_H
00002 #define AUBO_SCOPE_GRIPPER_DEVICE_H
00003
00004 #include <aubo_caps/domain/device/device.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(GripperDevice);
00009
00010 /**
00011  * \chinese
00012  *
00013  *   AuboScope
00014  * \endchinese
00015  * \english
00016  * GripperDevice
00017  * This interface represents a gripper device in AuboScope.
00018  * \endenglish
00019  */
00020 class ARCS_ABI_EXPORT GripperDevice : public Device
00021 {
00022 public:
00023     virtual ~GripperDevice() = default;
00024 };
00025 } // namespace aubo_scope
00026 } // namespace arcs
00027
00028 #endif

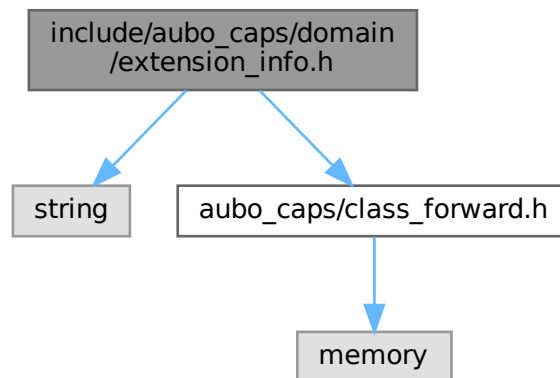
```

12.72 include/aubo_caps/domain/extension_info.h File Reference

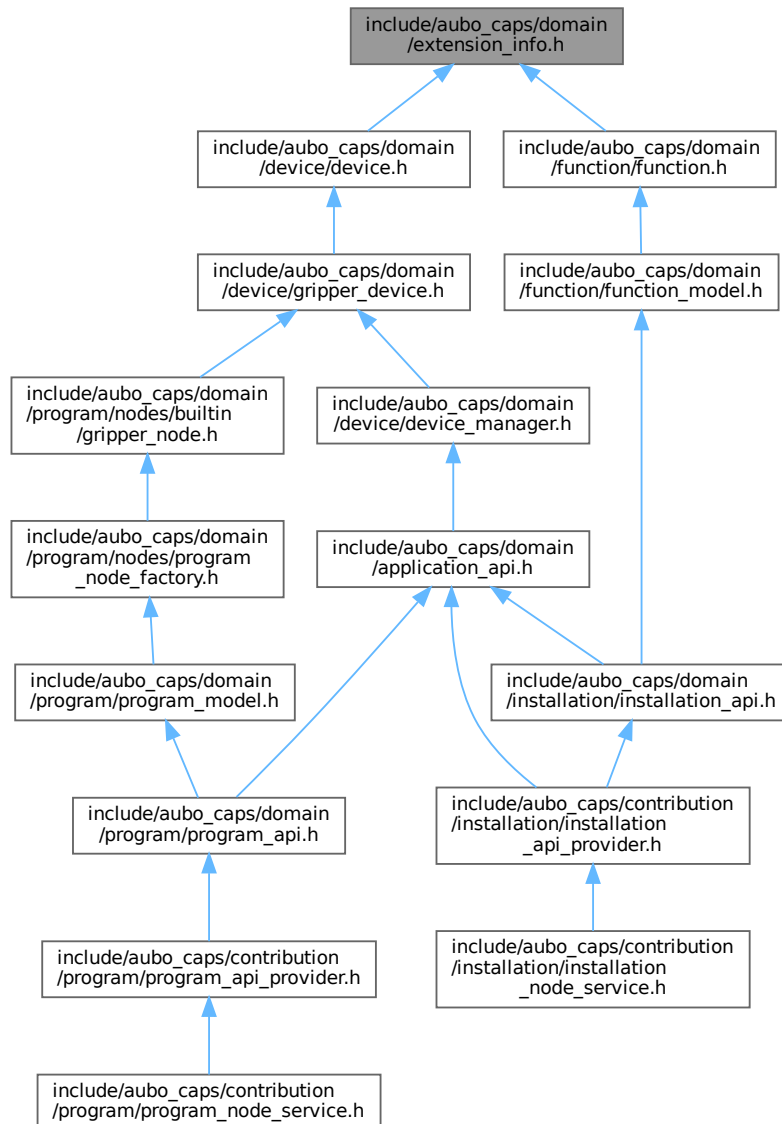
```
#include <string>
```

```
#include <aubo_caps/class_forward.h>
```

Include dependency graph for extension_info.h:



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



Classes

- class [arcs::aubo_scope::AuboCapInfo](#)

This interface provides various metadata information about an installed AuboCap, such as the name of the AuboCap.

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(AuboCapInfo\)](#)

12.73 extension_info.h

[Go to the documentation of this file.](#)

```

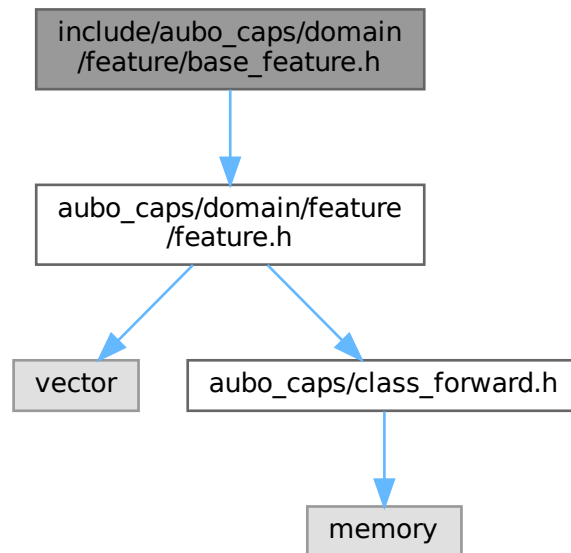
00001 #ifndef AUBO_SCOPE_EXTENSION_INFO_H
00002 #define AUBO_SCOPE_EXTENSION_INFO_H
00003
00004 #include <string>
00005 #include <aubo_caps/class_forward.h>
00006
00007 namespace arcs {
00008 namespace aubo_scope {
00009 ARCS_CLASS_FORWARD(AuboCapInfo);
00010
00011 /**
00012  * This interface provides various metadata information about an installed
00013  * AuboCap, such as the name of the AuboCap.
00014  *
00015  * @since 1.1.0
00016  */
00017 class ARCS_ABI_EXPORT AuboCapInfo
00018 {
00019 public:
00020     virtual ~AuboCapInfo() = default;
00021
00022     /**
00023      * @retAubon The symbolic name (id) of the AuboCap.
00024      */
00025     virtual std::string getSymbolicName() = 0;
00026
00027     /**
00028      * @retAubon The name of the AuboCap.
00029      */
00030     virtual std::string getName() = 0;
00031
00032     /**
00033      * @retAubon The vendor of the AuboCap.
00034      */
00035     virtual std::string getVendor() = 0;
00036 };
00037 } // namespace aubo_scope
00038 } // namespace arcs
00039
00040 #endif

```

12.74 include/aubo_caps/domain/feature/base_feature.h File Reference

#include <aubo_caps/domain/feature/feature.h>

Include dependency graph for base_feature.h:



Classes

- class [arcs::aubo_scope::BaseFeature](#)
This interface represents the Base feature in AuboScope.

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([BaseFeature](#))

12.75 base__feature.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_BASE_FEATURE_H
00002 #define AUBO_SCOPE_BASE_FEATURE_H
00003
00004 #include <aubo_caps/domain/feature/feature.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(BaseFeature);
00009
00010 /**
00011  * This interface represents the Base feature in AuboScope.
00012  */
00013 class ARCS_ABI_EXPORT BaseFeature : public Feature
00014 {
00015 public:
00016     BaseFeature(BaseFeature &f);
00017     BaseFeature(BaseFeature &&f);
00018     virtual ~BaseFeature();
00019
00020 private:
00021     friend class DataSwitch;
00022     BaseFeature();
00023     void *d_{ nullptr };
00024 };
00025
00026 } // namespace aubo_scope
00027 } // namespace arcs
00028
00029 #endif // AUBO_SCOPE_BASE_FEATURE_H

```

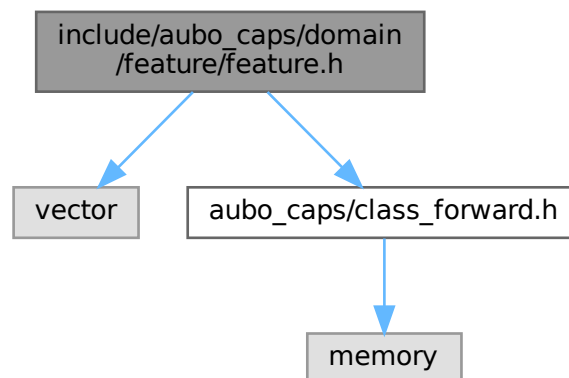
12.76 include/aubo_caps/domain/feature/feature.h File Reference

```

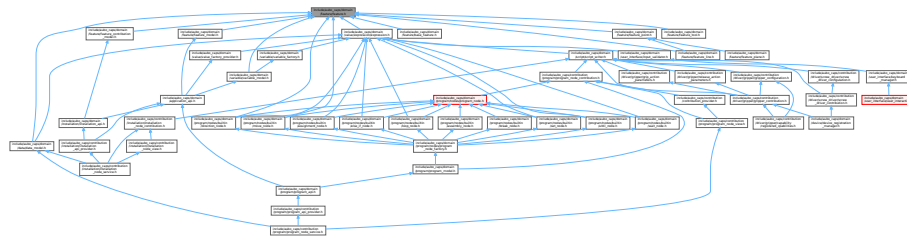
#include <vector>
#include <aubo_caps/class_forward.h>

```

Include dependency graph for feature.h:



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



Classes

- class [arcs::aubo_scope::Feature](#)
[Feature](#) This base interface represents all types of features available in AuboScope.
- struct [arcs::aubo_scope::FeatureDefinedFilter](#)
- struct [arcs::aubo_scope::FeatureJoggableFilter](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([Feature](#))

12.77 feature.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_FEATURE_H
00002 #define AUBO_SCOPE_FEATURE_H
00003
00004 #include <vector>
00005 #include <aubo_caps/class_forward.h>
00006
00007 namespace arcs {
00008 namespace aubo_scope {
00009
00010 ARCS_CLASS_FORWARD(Feature);
00011
00012 /**
00013  * \chinese
00014  *
00015  *     AuboScope
00016  * \endchinese
00017  * \english
00018  * Feature
00019  * This base interface represents all types of features available in AuboScope.
00020  * \endenglish
00021  */
00022 class ARCS_ABI_EXPORT Feature
00023 {
00024 public:
00025     Feature(Feature &f);
00026     Feature(Feature &&f);
00027     virtual ~Feature();
00028
00029     /**
00030      * \chinese
00031      *
00032      * @return

```

```

00033     * \endchinese
00034     * \english
00035     * Note: The feature can be renamed at any time.
00036     * @return The name of the feature.
00037     * \endenglish
00038     */
00039     std::string getDisplayName();
00040
00041     /**
00042     * \chinese
00043     * /
00044     * @return <code>true</code> <code>>false</code>
00045     * \endchinese
00046     * \english
00047     * Returns whether the position of the feature has been fully
00048     * defined/specified.
00049     *
00050     * @return <code>true</code> if the feature is defined, <code>>false</code>
00051     * otherwise.
00052     * \endenglish
00053     */
00054     bool isDefined();
00055
00056     /**
00057     * \chinese
00058     * /
00059     * @return <code>true</code> <code>>false</code>
00060     * \endchinese
00061     * \english
00062     * Returns whether the feature can be used as a variable in the program.
00063     *
00064     * @return <code>true</code> if the feature is variable, <code>>false</code>
00065     * otherwise.
00066     * \endenglish
00067     */
00068     bool isVariable();
00069     bool isJoggable();
00070
00071     /**
00072     * \chinese
00073     *     AuboScope
00074     *     AuboCap
00075     * @return AuboScope <code>true</code>
00076     * <code>>false</code>
00077     * \endchinese
00078     * \english
00079     * A feature cannot be guaranteed to be present in AuboScope. The end user
00080     * can load a different installation (not containing the feature) or the
00081     * feature can be removed by the end user or the AuboCap that added the
00082     * feature. This method can be used to determine, if the feature is present.
00083     *
00084     * @return <code>true</code> if this feature is present in AuboScope,
00085     * otherwise <code>>false</code>.
00086     * \endenglish
00087     */
00088     bool isResolvable();
00089
00090     /**
00091     * \chinese
00092     * /
00093     * @return
00094     * \endchinese
00095     * \english
00096     * Returns the Cartesian location of the feature.
00097     * @return A pose representing the Cartesian location of the feature.
00098     * \endenglish
00099     */
00100     std::vector<double> getPose();
00101
00102     size_t hashCode();
00103
00104 protected:
00105     Feature();
00106
00107 private:
00108     friend class DataSwitch;
00109     void *d_{ nullptr };
00110 };
00111
00112 struct FeatureDefinedFilter
00113 {
00114     bool operator()(FeaturePtr feature)
00115     {
00116         return feature && feature->isDefined();
00117     }
00118 };
00119

```

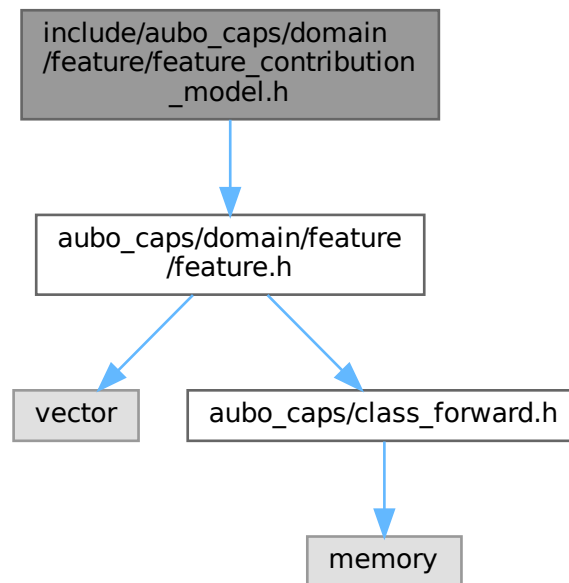


```
00120 struct FeatureJoggableFilter
00121 {
00122     bool operator()(FeaturePtr feature)
00123     {
00124         return feature && feature->isJoggable() && feature->isDefined();
00125     }
00126 };
00127
00128 } // namespace aubo_scope
00129 } // namespace arcs
00130 #endif
```

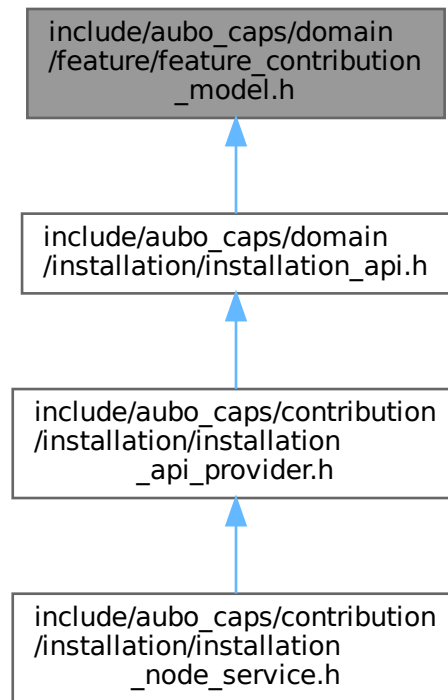
12.78 include/aubo_caps/domain/feature/feature_contribution_model.h File Reference

#include <aubo_caps/domain/feature/feature.h>

Include dependency graph for feature_contribution_model.h:



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



Classes

- class [arcs::aubo_scope::FeatureContributionModel](#)
[FeatureContributionModel](#) This interface provides functionality for adding, updating and removing features in AuboScope.

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([FeatureContributionModel](#))

12.79 feature_contribution_model.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_FEATURE_CONTRIBUTION_MODEL_H
00002 #define AUBO_SCOPE_FEATURE_CONTRIBUTION_MODEL_H
00003
00004 #include <aubo_caps/domain/feature/feature.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008
00009 ARCS_CLASS_FORWARD(FeatureContributionModel);
00010
00011 /**
00012  * @ingroup InstallationApi
00013  * \chinese
00014  *
00015  *     AuboScope
00016  *
00017  * <b> </b>
00018  * \endchinese
00019  * \english
00020  * FeatureContributionModel
00021  * This interface provides functionality for adding, updating and removing
00022  * features in AuboScope.
00023  *
00024  * <b>Note:</b> This interface must only be used in an installation
00025  * contribution.
00026  * \endenglish
00027  */
00028 class ARCS_ABI_EXPORT FeatureContributionModel
00029 {
00030 public:
00031     FeatureContributionModel(FeatureContributionModel &f);
00032     FeatureContributionModel(FeatureContributionModel &&f);
00033     ~FeatureContributionModel();
00034
00035     /**
00036      * \chinese
00037      *     AuboScope
00038      *
00039      * @param idKey          AuboCap
00040      * @param suggestedName
00041      * [a-zA-Z][a-zA-Z0-9_]{0,14}
00042      * @param pose
00043      * @return AuboScope
00044      * @throws FeatureAlreadyAddedException idKey
00045      * @throws IllegalFeatureNameException
00046      * \endchinese
00047      * \english
00048      * Add a feature to the current AuboScope installation. This makes it
00049      * selectable by the end user. The feature is not modifiable by the end
00050      * user.
00051      *
00052      * @param idKey The key to identify this feature by.
00053      * @param suggestedName Suggested name for the feature.
00054      * @param pose The pose of the feature with respect to the robot base.
00055      * @return The feature created and registered in AuboScope.
00056      * @throws FeatureAlreadyAddedException If a feature has previously been
00057      * added the same <code>idKey</code> identifier.
00058      * @throws IllegalFeatureNameException If the suggested name does not match
00059      * required regex.
00060      * \endenglish
00061      */
00062     FeaturePtr addFeature(const std::string &idKey,
00063                          const std::string &suggestedName,
00064                          const std::vector<double> &pose);
00065
00066     /**
00067      * \chinese
00068      *     AuboCap idKey
00069      * @param idKey
00070      * @return <code>null</code>
00071      * \endchinese
00072      * \english
00073      * Returns the feature previously added by this AuboCap using the same
00074      * <code>idKey</code> identifier. Use this to verify if the feature is
00075      * present in the current installation.
00076      *
00077      * @param idKey The key to identify this feature by.
00078      * @return The feature previously added by this AuboCap.
00079      * Returns <code>null</code> if no feature exists in the current
00080      * installation.
00081      * \endenglish
00082      */
00082     FeaturePtr getFeature(const std::string &idKey);

```

```

00083
00084 /**
00085  * \chinese
00086  *   AuboCap
00087  *   @param idKey
00088  *   @param newPose
00089  *   @throws FeatureNotFoundException idKey
00090  * \endchinese
00091  * \english
00092  * Update the pose of an existing feature added by this AuboCap.
00093  *
00094  * @param idKey The key to identify the feature.
00095  * @param newPose The new pose to set for the feature.
00096  * @throws FeatureNotFoundException If no feature exists with the provided
00097  * <code>idKey</code>.
00098  * \endenglish
00099  */
00100 void updateFeature(const std::string &idKey,
00101                  const std::vector<double> &newPose);
00102 void renameFeature(const std::string &idKey, const std::string &newName);
00103
00104 /**
00105  * \chinese
00106  *   AuboCap
00107  *   @param idKey
00108  *   @throws FeatureNotFoundException idKey
00109  * \endchinese
00110  * \english
00111  * Remove a feature added by this AuboCap from AuboScope. Program nodes
00112  * using the feature will be become undefined because the feature is no
00113  * longer resolvable.
00114  *
00115  * @param idKey The key used to add the feature with.
00116  * @throws FeatureNotFoundException If no feature exists with the provided
00117  * <code>idKey</code>.
00118  * \endenglish
00119  */
00120 void removeFeature(const std::string &idKey);
00121
00122 private:
00123   friend class DataSwitch;
00124   FeatureContributionModel();
00125   void *d_{ nullptr };
00126 };
00127
00128 } // namespace aubo_scope
00129 } // namespace arcs
00130
00131 #endif // AUBO_SCOPE_FEATURE_CONTRIBUTION_MODEL_H

```

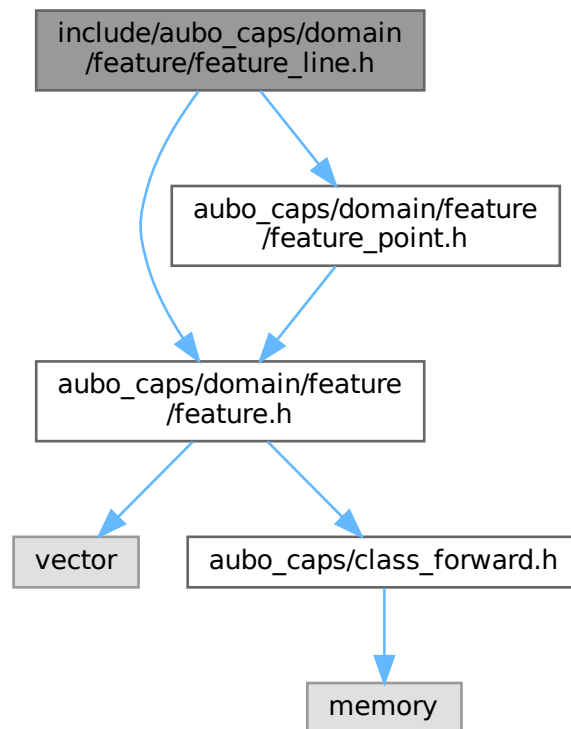
12.80 include/aubo_caps/domain/feature/feature_line.h File Reference

```

#include <aubo_caps/domain/feature/feature.h>
#include <aubo_caps/domain/feature/feature_point.h>

```

Include dependency graph for feature_line.h:



Classes

- class `arcs::aubo_scope::FeatureLine`

Namespaces

- namespace `arcs`
- namespace `arcs::aubo_scope`

Functions

- `arcs::aubo_scope::ARCS_CLASS_FORWARD` (`FeatureLine`)

12.81 feature_line.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_FEATURE_LINE_H
00002 #define AUBO_SCOPE_FEATURE_LINE_H
00003
00004 #include <aubo_caps/domain/feature/feature.h>

```

```

00005 #include <aubo_caps/domain/feature/feature_point.h>
00006
00007 namespace arcs {
00008 namespace aubo_scope {
00009
00010 ARCS_CLASS_FORWARD(FeatureLine);
00011
00012 class ARCS_ABI_EXPORT FeatureLine : public Feature
00013 {
00014 public:
00015     FeatureLine(FeatureLine &f);
00016     FeatureLine(FeatureLine &&f);
00017     virtual ~FeatureLine();
00018
00019 private:
00020     friend class DataSwitch;
00021     FeatureLine();
00022     void *d_{ nullptr };
00023 };
00024
00025 } // namespace aubo_scope
00026 } // namespace arcs
00027
00028 #endif // AUBO_SCOPE_FEATURE_LINE_H

```

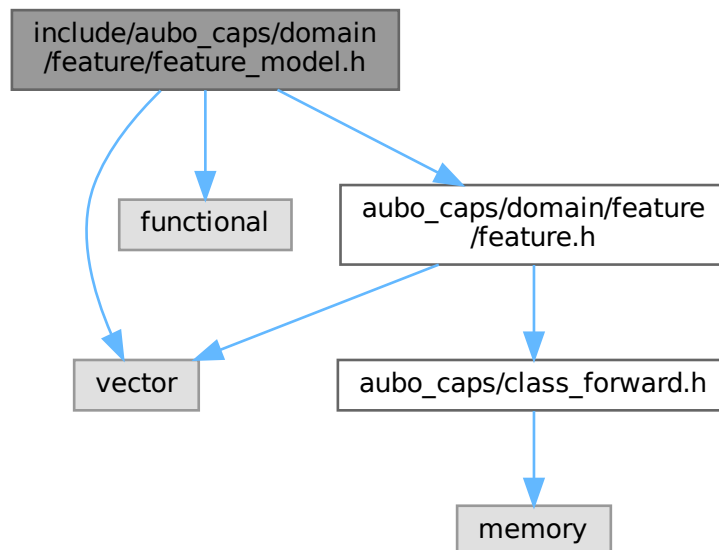
12.82 include/aubo_caps/domain/feature/feature_model.h File Reference

```

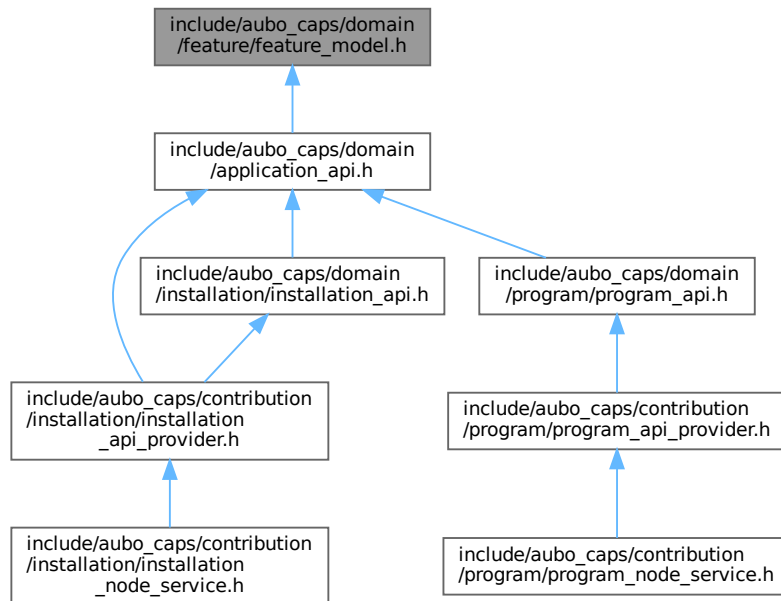
#include <vector>
#include <functional>
#include <aubo_caps/domain/feature/feature.h>

```

Include dependency graph for feature_model.h:



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



Classes

- class `arcs::aubo_scope::FeatureModel`
`FeatureModel` Provides methods that returns features from the current robot installation.

Namespaces

- namespace `arcs`
- namespace `arcs::aubo_scope`

Functions

- `arcs::aubo_scope::ARCS_CLASS_FORWARD` (`FeatureModel`)

12.83 feature_model.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_FEATURE_MODEL_H
00002 #define AUBO_SCOPE_FEATURE_MODEL_H
00003
00004 #include <vector>
00005 #include <functional>
00006 #include <aubo_caps/domain/feature/feature.h>
00007
00008 namespace arcs {
00009 namespace aubo_scope {
00010
00011 ARCS_CLASS_FORWARD(FeatureModel);

```

```

00012
00013 /**
00014  * @ingroup ApplicationApi
00015  * \chinese
00016  *
00017  *
00018  * \endchinese
00019  * \english
00020  * FeatureModel
00021  * Provides methods that returns features from the current robot installation.
00022  * \endenglish
00023  */
00024 class ARCS_ABI_EXPORT FeatureModel
00025 {
00026 public:
00027     FeatureModel(FeatureModel &f);
00028     FeatureModel(FeatureModel &&f);
00029     ~FeatureModel();
00030
00031     std::vector<FeaturePtr> getAll();
00032
00033     /**
00034     * @ingroup FeatureModel
00035     * \chinese
00036     *
00037     * @return
00038     * \endchinese
00039     * \english
00040     * Returns the geometric features of the current installation.
00041     * @return the geometric features of the current installation.
00042     * \endenglish
00043     */
00044     template <typename T>
00045     std::vector<FeaturePtr> getGeomFeture();
00046
00047     /**
00048     * @ingroup FeatureModel
00049     * \chinese
00050     *
00051     * @param name
00052     * @return
00053     * \endchinese
00054     * \english
00055     * Returns a feature by name.
00056     * @param name the name of the feature.
00057     * @return the feature with the given name.
00058     * \endenglish
00059     */
00060     FeaturePtr getByName(const std::string &name);
00061
00062     /**
00063     * @ingroup FeatureModel
00064     * \chinese
00065     *
00066     * @return
00067     * \endchinese
00068     * \english
00069     * Returns the base feature of the current installation.
00070     * @return the base feature of the current installation.
00071     * \endenglish
00072     */
00073     FeaturePtr getBaseFeature();
00074
00075     /**
00076     * @ingroup FeatureModel
00077     * \chinese
00078     *
00079     * @return
00080     * \endchinese
00081     * \english
00082     * Returns the tool feature of the current installation.
00083     * @return the tool feature of the current installation.
00084     * \endenglish
00085     */
00086     FeaturePtr getToolFeature();
00087
00088     /**
00089     * @ingroup FeatureModel
00090     * \chinese
00091     *
00092     * @param filter
00093     * @return
00094     * \endchinese
00095     * \english
00096     * Returns features matching the given filter.
00097     * @param filter the filter to apply.
00098     * @return a collection of features that match the filter.

```



```

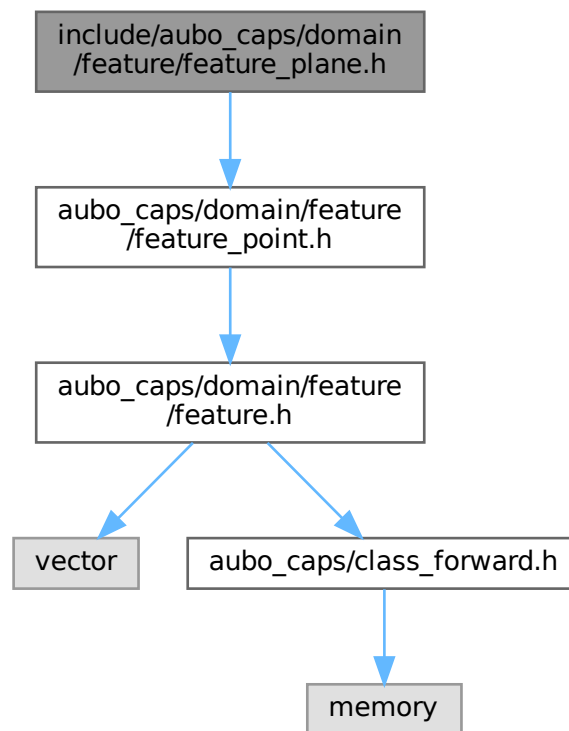
00099      * \endenglish
00100      */
00101      std::vector<FeaturePtr> getFeatures(std::function<bool(FeaturePtr)> filter);
00102
00103  private:
00104      friend class DataSwitch;
00105      FeatureModel();
00106      void *d_{ nullptr };
00107  };
00108
00109  } // namespace aubo_scope
00110  } // namespace arcs
00111
00112  #endif // AUBO_SCOPE_FEATURE_MODEL_H

```

12.84 include/aubo_caps/domain/feature/feature_plane.h File Reference

#include <aubo_caps/domain/feature/feature_point.h>

Include dependency graph for feature_plane.h:



Classes

- class `arcs::aubo_scope::FeaturePlane`

Namespaces

- namespace `arcs`
- namespace `arcs::aubo_scope`

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(FeaturePlane\)](#)

12.85 feature_plane.h

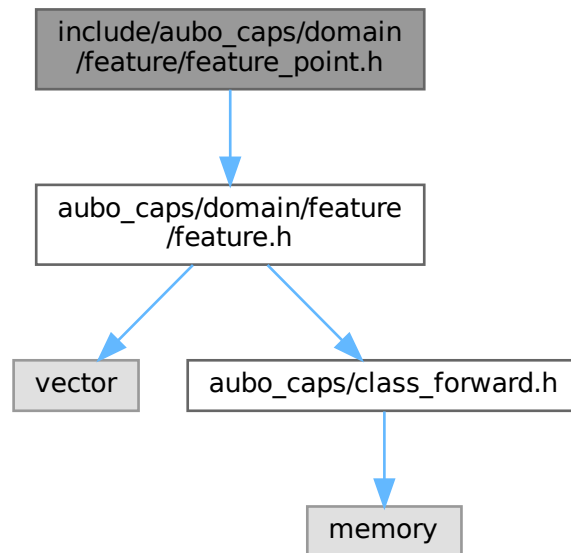
[Go to the documentation of this file.](#)

```
00001 #ifndef AUBO_SCOPE_FEATURE_PLANE_H
00002 #define AUBO_SCOPE_FEATURE_PLANE_H
00003
00004 #include <aubo_caps/domain/feature/feature_point.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008   ARCS_CLASS_FORWARD(FeaturePlane);
00009
00010   class ARCS_ABI_EXPORT FeaturePlane : public Feature
00011   {
00012   public:
00013     FeaturePlane(FeaturePlane &f);
00014     FeaturePlane(FeaturePlane &&f);
00015     virtual ~FeaturePlane();
00016
00017   private:
00018     friend class DataSwitch;
00019     FeaturePlane();
00020     void *d_{ nullptr };
00021   };
00022
00023 } // namespace aubo_scope
00024 } // namespace arcs
00025
00026 #endif // AUBO_SCOPE_FEATURE_PLANE_H
```

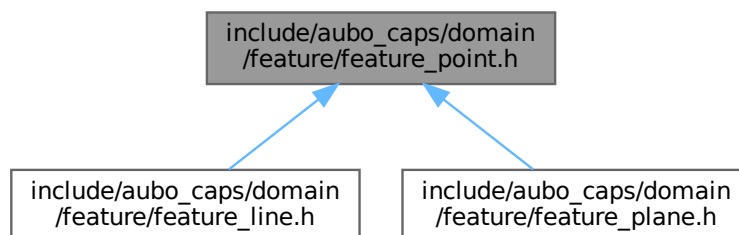
12.86 include/aubo_caps/domain/feature/feature_point.h File Reference

```
#include <aubo_caps/domain/feature/feature.h>
```

Include dependency graph for feature_point.h:



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



Classes

- class [arcs::aubo_scope::FeaturePoint](#)

This interface represents a Point feature in AuboScope.

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([FeaturePoint](#))

12.87 [feature_point.h](#)

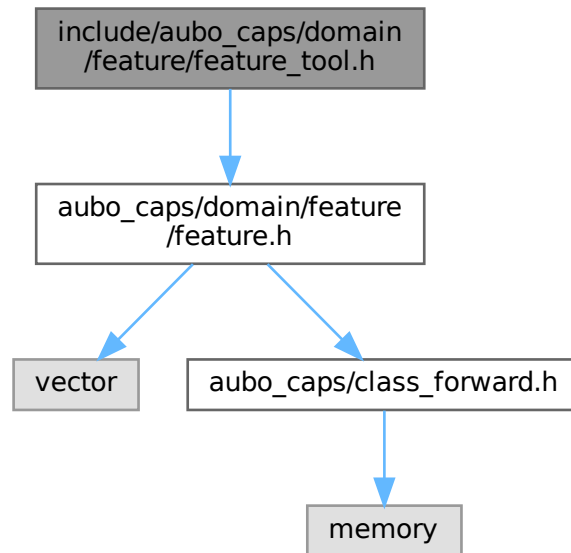
[Go to the documentation of this file.](#)

```
00001 #ifndef AUBO_SCOPE_FEATURE_POINT_H
00002 #define AUBO_SCOPE_FEATURE_POINT_H
00003
00004 #include <aubo_caps/domain/feature/feature.h>
00005
00006 namespace arcs {
00007     namespace aubo_scope {
00008         ARCS_CLASS_FORWARD(FeaturePoint);
00009
00010         /**
00011          * This interface represents a Point feature in AuboScope
00012          */
00013         class ARCS_ABI_EXPORT FeaturePoint : public Feature
00014         {
00015         public:
00016             FeaturePoint(FeaturePoint &f);
00017             FeaturePoint(FeaturePoint &&f);
00018             virtual ~FeaturePoint();
00019
00020         private:
00021             friend class DataSwitch;
00022             FeaturePoint();
00023             void *d_{ nullptr };
00024         };
00025
00026     } // namespace aubo_scope
00027 } // namespace arcs
00028
00029 #endif // AUBO_SCOPE_FEATURE_POINT_H
```

12.88 include/aubo_caps/domain/feature/feature_tool.h File Reference

`#include <aubo_caps/domain/feature/feature.h>`

Include dependency graph for feature_tool.h:



Classes

- class [arcs::aubo_scope::FeatureTool](#)
This interface represents the Tool feature in AuboScope.

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([FeatureTool](#))

12.89 feature__tool.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_FEATURE_TOOL_H
00002 #define AUBO_SCOPE_FEATURE_TOOL_H
00003
00004 #include <aubo_caps/domain/feature/feature.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(FeatureTool);
00009
00010 /**
00011  * This interface represents the Tool feature in AuboScope.
00012  */
00013 class ARCS_ABI_EXPORT FeatureTool : public Feature
00014 {
00015 public:
00016     FeatureTool(FeatureTool &f);
00017     FeatureTool(FeatureTool &&f);
00018     virtual ~FeatureTool();
00019
00020 private:
00021     friend class DataSwitch;
00022     FeatureTool();
00023     void *d_{ nullptr };
00024 };
00025
00026 } // namespace aubo_scope
00027 } // namespace arcs
00028
00029 #endif // AUBO_SCOPE_FEATURE_TOOL_H

```

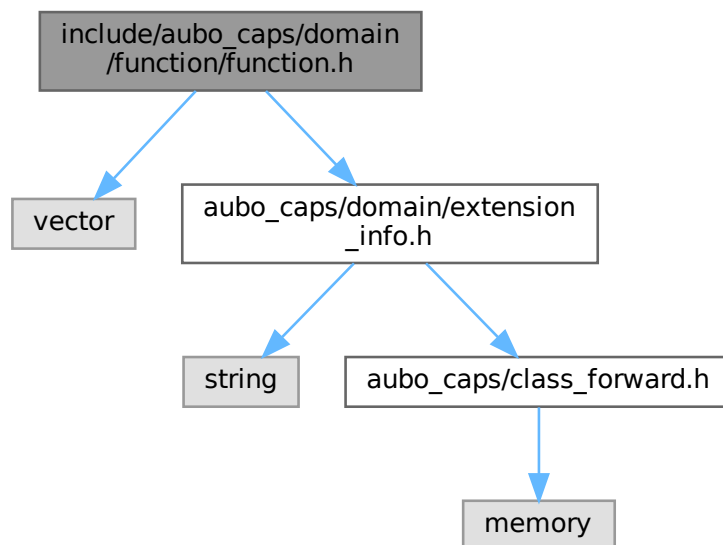
12.90 include/aubo_caps/domain/function/function.h File Reference

```

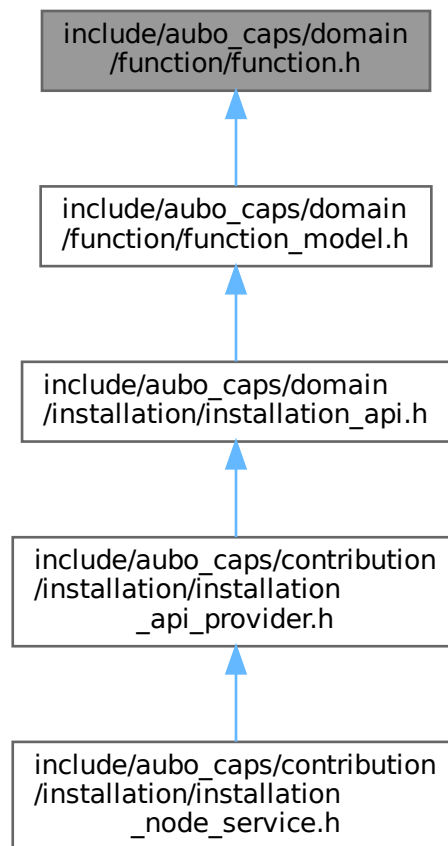
#include <vector>
#include <aubo_caps/domain/extension_info.h>

```

Include dependency graph for function.h:



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



Classes

- class [arcs::aubo_scope::Function](#)
[Function](#) Represents a function shown in the [Expression](#) Editor.

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([Function](#))

12.91 function.h

[Go to the documentation of this file.](#)

```

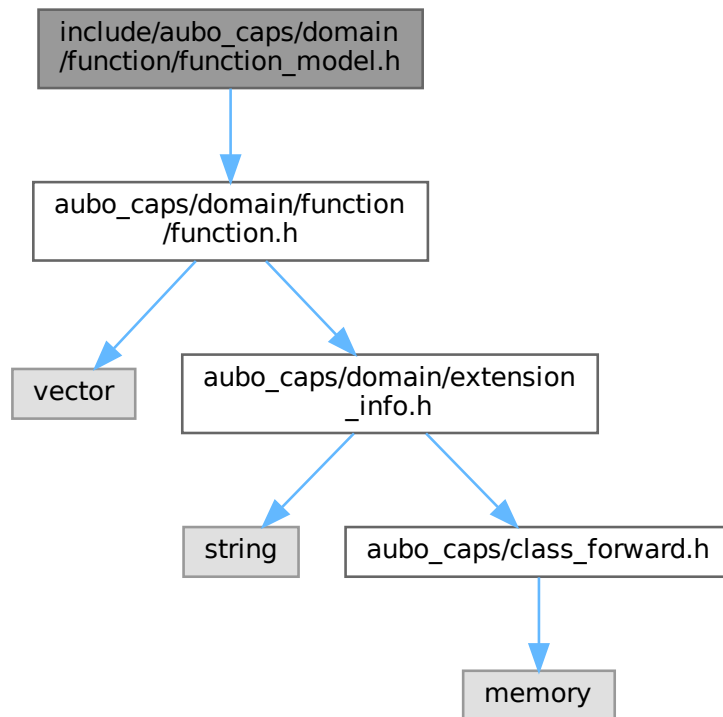
00001 #ifndef AUBO_SCOPE_FUNCTION_H
00002 #define AUBO_SCOPE_FUNCTION_H
00003
00004 #include <vector>
00005 #include <aubo_caps/domain/extension_info.h>
00006
00007 namespace arcs {
00008 namespace aubo_scope {
00009 ARCS_CLASS_FORWARD(Function);
00010
00011 /**
00012  * \chinese
00013  *
00014  *
00015  *
00016  * {@link FunctionModel}
00017  * \endchinese
00018  * \english
00019  * Function
00020  * Represents a function shown in the Expression Editor.
00021  *
00022  * See also {@link FunctionModel}.
00023  * \endenglish
00024  */
00025 class ARCS_ABI_EXPORT Function
00026 {
00027 public:
00028     Function(Function &f);
00029     Function(Function &&f);
00030     ~Function();
00031
00032     /**
00033      * \chinese
00034      *
00035      * @return
00036      * \endchinese
00037      * \english
00038      * Returns the name of the function.
00039      * @return The name of the function.
00040      * \endenglish
00041      */
00042     std::string getName();
00043
00044     /**
00045      * \chinese
00046      *
00047      * @return
00048      * \endchinese
00049      * \english
00050      * Returns a list of the arguments of the function.
00051      * @return A list of the arguments of the function.
00052      * \endenglish
00053      */
00054     std::vector<std::string> getArguments();
00055
00056     /**
00057      * \chinese
00058      * / AuboScope AuboCap
00059      * @return AuboCap
00060      * \endchinese
00061      * \english
00062      * Get information about the AuboCap that added/contributed this function
00063      * to AuboScope.
00064      *
00065      * @return AuboCap information for this function.
00066      * \endenglish
00067      */
00068     AuboCapInfoPtr getContributorInfo();
00069
00070 private:
00071     friend class DataSwitch;
00072     Function();
00073     void *d_{ nullptr };
00074 };
00075
00076 } // namespace aubo_scope
00077 } // namespace arcs
00078
00079 #endif // AUBO_SCOPE_FUNCTION_H

```

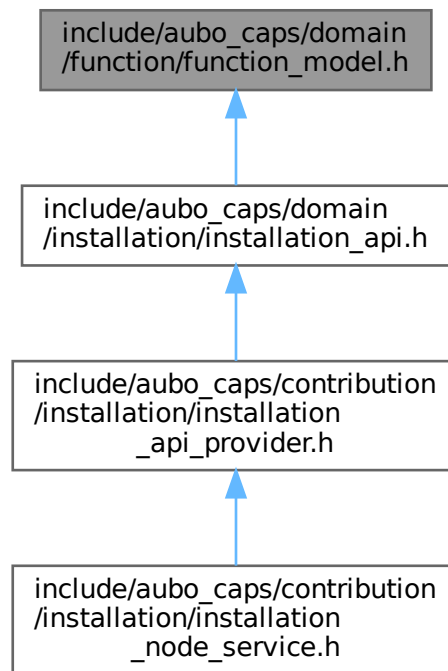

12.92 include/aubo_caps/domain/function/function_model.h File Reference

`#include <aubo_caps/domain/function/function.h>`

Include dependency graph for `function_model.h`:



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



Classes

- class [arcs::aubo_scope::FunctionModel](#)
[FunctionModel](#) This interface provides methods to add and remove script functions in the [Expression Editor](#) accessible from Assignment and If program nodes among others.

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([FunctionModel](#))

12.93 function_model.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_FUNCTION_MODEL_H
00002 #define AUBO_SCOPE_FUNCTION_MODEL_H
00003
00004 #include <aubo_caps/domain/function/function.h>

```

```

00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(FunctionModel);
00009
00010 /**
00011 * @ingroup InstallationApi
00012 * \chinese
00013 *
00014 *           If
00015 *
00016 *
00017 *
00018 *   {@link InstallationNodeContribution#generateScript(ScriptWriter)}
00019 *
00020 *
00021 *
00022 *
00023 *
00024 *   {@link Function}
00025 * \endchinese
00026 * \english
00027 * FunctionModel
00028 * This interface provides methods to add and remove script functions in the
00029 * Expression Editor accessible from Assignment and If program nodes among
00030 * others.
00031 *
00032 * Added functions merely serve as an easy way for the end customer to use the
00033 * contributed functions. The implementation of a given script function must be
00034 * added in the preamble of program through the use of
00035 * {@link InstallationNodeContribution#generateScript(ScriptWriter)}.
00036 *
00037 * The full set of functions which can be contributed must be added in the
00038 * preamble at all times, regardless of whether they are added to the Expression
00039 * Editor or not. Otherwise old programs that rely on a script function
00040 * contribution might stop working.
00041 *
00042 * See also {@link Function}.
00043 * \endenglish
00044 */
00045 class ARCS_ABI_EXPORT FunctionModel
00046 {
00047 public:
00048     FunctionModel(FunctionModel &f);
00049     FunctionModel(FunctionModel &&f);
00050     ~FunctionModel();
00051
00052     /**
00053     * \chinese
00054     *
00055     * @param name          [a-zA-Z][a-zA-Z0-9_]{0,24}
00056     * 25   AUBO Robot
00057     * @param argumentNames
00058     *
00059     * @return              null
00060     * @throws FunctionException
00061     * \endchinese
00062     * \english
00063     * Add a script function to the Expression Editor.
00064     * @param name Name of the function. Safe names must match regex
00065     * [a-zA-Z][a-zA-Z0-9_]{0,24} for a total of 25 characters.
00066     * @param argumentNames Is a list of hints for the function arguments.
00067     * @return If the script function was successfully added, an object
00068     * representing the Function will be returned. Null otherwise.
00069     * @throws FunctionException if the name is illegal or more than five
00070     * arguments are specified.
00071     * \endenglish
00072     */
00073     FunctionPtr addFunction(const std::string &name,
00074                             const std::vector<std::string> &argumentNames);
00075
00076     /**
00077     * \chinese
00078     *   AuboCap
00079     * @param function
00080     * @return  <code>true</code>   <code>false</code>
00081     * \endchinese
00082     * \english
00083     * Remove a function. Only functions added by this AuboCap can be removed.
00084     * @param function Function to be removed.
00085     * @return <code>true</code>, if the function was successfully removed.
00086     * \endenglish
00087     */
00088     bool removeFunction(FunctionPtr func);
00089
00090     /**
00091     * \chinese

```

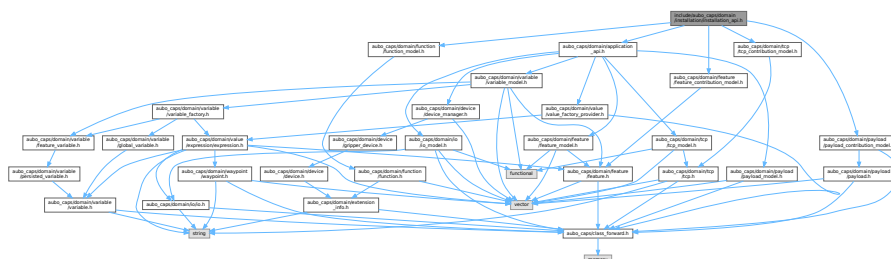
```

00092 *
00093 * @param name
00094 * @return null
00095 * \endchinese
00096 * \english
00097 * Get a function by name.
00098 * @param name The name of the function.
00099 * @return The function if found, otherwise null.
00100 * \endenglish
00101 */
00102 FunctionPtr getFunction(const std::string &name);
00103
00104 /**
00105 * \chinese
00106 *
00107 * @return
00108 * \endchinese
00109 * \english
00110 * Get all added functions.
00111 * @return A collection of added functions.
00112 * \endenglish
00113 */
00114 std::vector<FunctionPtr> getFunctions();
00115
00116 private:
00117 friend class DataSwitch;
00118 FunctionModel();
00119 void *d_{ nullptr };
00120 };
00121
00122 } // namespace aubo_scope
00123 } // namespace arcs
00124
00125 #endif // AUBO_SCOPE_FUNCTION_MODEL_H

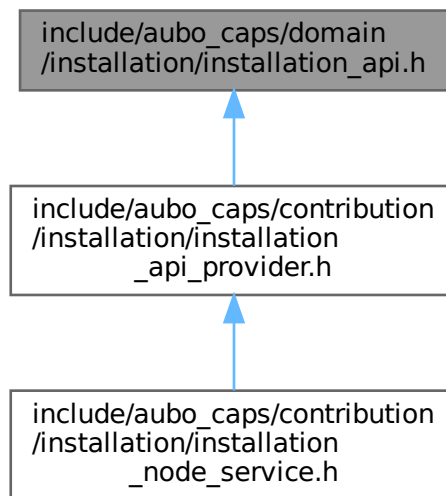
```

12.94 include/aubo_caps/domain/installation/installation_api.h File Reference

```
#include <aubo_caps/domain/application_api.h>
#include <aubo_caps/domain/function/function_model.h>
#include <aubo_caps/domain/tcp/tcp_contribution_model.h>
#include <aubo_caps/domain/feature/feature_contribution_model.h>
#include <aubo_caps/domain/payload/payload_contribution_model.h>
Include dependency graph for installation_api.h:
```



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



Classes

- class [arcs::aubo_scope::InstallationApi](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([InstallationApi](#))

12.95 installation_api.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_INSTALLATION_API_H
00002 #define AUBO_SCOPE_INSTALLATION_API_H
00003
00004 #include <aubo_caps/domain/application_api.h>
00005 #include <aubo_caps/domain/function/function_model.h>
00006 #include <aubo_caps/domain/tcp/tcp_contribution_model.h>
00007 #include <aubo_caps/domain/feature/feature_contribution_model.h>
00008 #include <aubo_caps/domain/payload/payload_contribution_model.h>
00009
00010 namespace arcs {
00011     namespace aubo_scope {
00012         ARCS_CLASS_FORWARD(InstallationApi);
00013     }
00014 } /**
00015  * \chinese
  
```

```

00016 * @defgroup InstallationApi InstallationApi( )
00017 * @ingroup InstallationApi
00018 *
00019 *     AuboScope
00020 * AuboScope
00021 * \endchinese
00022 * \english
00023 * @defgroup InstallationApi InstallationApi
00024 * @ingroup InstallationApi
00025 * InstallationApi
00026 * Provides access to functionality and services related to the installation.
00027 * This includes what is available from within AuboScope, as well as creating
00028 * additional model elements to be used within AuboScope.
00029 * \endenglish
00030 */
00031 class ARCS_ABI_EXPORT InstallationApi : public ApplicationApi
00032 {
00033 public:
00034     InstallationApi(InstallationApi &f);
00035     InstallationApi(InstallationApi &&f);
00036     virtual ~InstallationApi();
00037
00038     /**
00039     * @ingroup InstallationApi
00040     * \chinese
00041     *     AuboScope
00042     * @return
00043     * \endchinese
00044     * \english
00045     * Returns an interface for adding script functions to AuboScope.
00046     * @return An interface for adding script functions to AuboScope.
00047     * \endenglish
00048     */
00049     FunctionModelPtr getFunctionModel();
00050
00051     /**
00052     * @ingroup InstallationApi
00053     * \chinese
00054     *     AuboScope TCP
00055     * @return TCP
00056     * \endchinese
00057     * \english
00058     * Returns an interface for contributing TCPs to AuboScope.
00059     * @return An interface for contributing TCPs to AuboScope.
00060     * \endenglish
00061     */
00062     TcpContributionModelPtr getTcpContributionModel();
00063
00064     /**
00065     * @ingroup InstallationApi
00066     * \chinese
00067     *     AuboScope
00068     * @return
00069     * \endchinese
00070     * \english
00071     * Returns an interface for contributing features to AuboScope.
00072     * @return An interface for contributing features to AuboScope.
00073     * \endenglish
00074     */
00075     FeatureContributionModelPtr getFeatureContributionModel();
00076
00077     /**
00078     * @ingroup InstallationApi
00079     * \chinese
00080     * <p>
00081     *     AuboScope
00082     * </p>
00083     *
00084     * @return
00085     * \endchinese
00086     * \english
00087     * <p>
00088     * Gets an interface which provides functionality for contributing payloads
00089     * to the AuboScope installation.
00090     * </p>
00091     *
00092     * @return An interface for contributing payloads to AuboScope.
00093     * \endenglish
00094     */
00095     PayloadContributionModelPtr getPayloadContributionModel();
00096
00097 private:
00098     friend class DataSwitch;
00099     InstallationApi();
00100     void *d_{ nullptr };
00101 };
00102

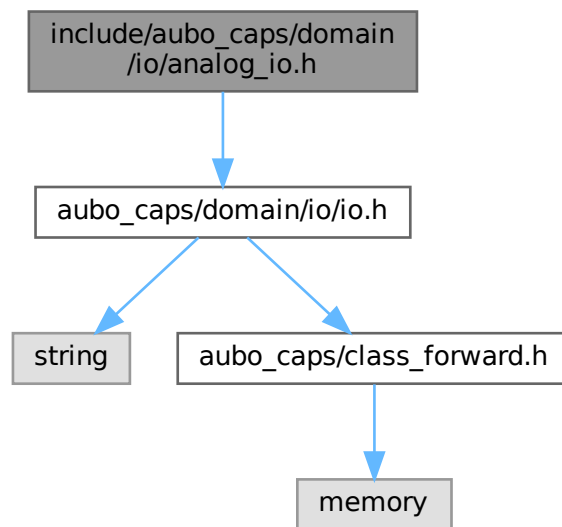
```

```
00103 } // namespace aubo_scope
00104 } // namespace arcs
00105
00106 #endif // AUBO_SCOPE_INSTALLATION_API_H
```

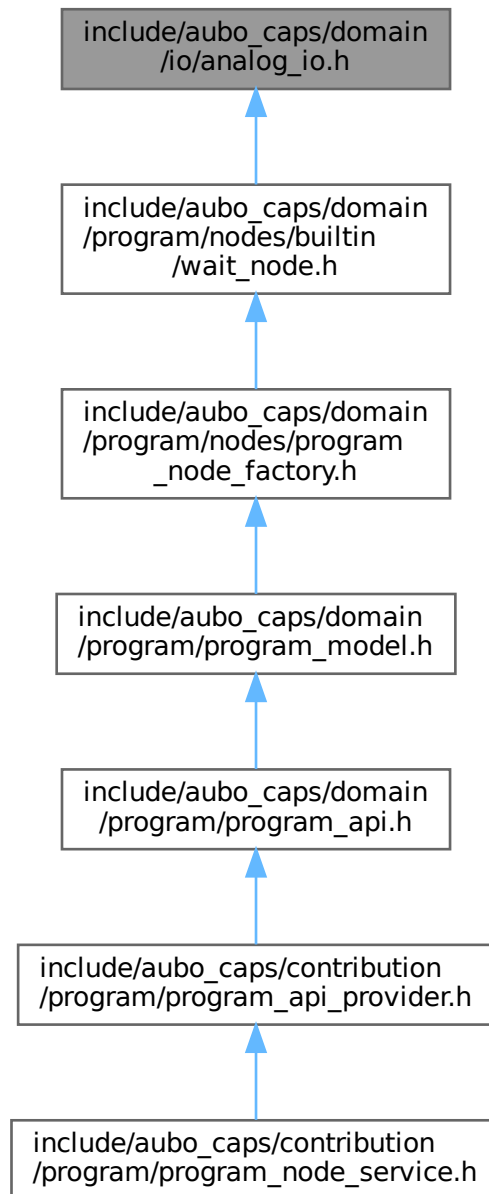
12.96 include/aubo_caps/domain/io/analog_io.h File Reference

#include <aubo_caps/domain/io/io.h>

Include dependency graph for analog_io.h:



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



Classes

- class [arcs::aubo_scope::AnalogIo](#)
[AnalogIo](#) This interface provides support for analog I/Os.
- struct [arcs::aubo_scope::AnalogIoFilter](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(AnalogIo\)](#)

12.97 analog_io.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_ANALOG_IO_H
00002 #define AUBO_SCOPE_ANALOG_IO_H
00003
00004 #include <aubo_caps/domain/io/io.h>
00005
00006 namespace arcs {
00007     namespace aubo_scope {
00008         ARCS_CLASS_FORWARD(AnalogIo);
00009
00010         /**
00011          * \chinese
00012          *   IO
00013          *   I/O
00014          * \endchinese
00015          * \english
00016          * AnalogIo
00017          * This interface provides support for analog I/Os.
00018          * \endenglish
00019          */
00020         class ARCS_ABI_EXPORT AnalogIo : public Io
00021         {
00022         public:
00023             AnalogIo(AnalogIo &f);
00024             AnalogIo(AnalogIo &&f);
00025             ~AnalogIo();
00026
00027             double getMinRangeValue();
00028             double getMaxRangeValue();
00029
00030             bool isCurrent();
00031             bool isVoltage();
00032
00033             /**
00034              * \chinese
00035              *   I/O   I/O
00036              * @param value
00037              * @return      <code>true</code>   <code>>false</code>
00038              * \endchinese
00039              * \english
00040              * Set analog I/O output signal. Notice the I/O must be an output I/O.
00041              *
00042              * @param value The analog value to be set.
00043              * @return <code>true</code> if the analog signal was set,
00044              *        <code>>false</code> if the signal was not set, e.g. because the controller
00045              *        was not running.
00046              * \endenglish
00047              */
00048             bool setValue(double value);
00049
00050             /**
00051              * \chinese
00052              *   I/O
00053              * @return I/O
00054              * \endchinese
00055              * \english
00056              * Returns the last reading of the I/O.
00057              * @return Returns the last reading of the I/O.
00058              * \endenglish
00059              */
00060             double getValue();
00061
00062         private:
00063             friend class DataSwitch;
00064             AnalogIo();
00065             void *d_{ nullptr };
00066         };
00067
00068         struct AnalogIoFilter
00069         {
00070             AnalogIoFilter(Io::InterfaceType interface, bool is_input)
00071                 : interface_(interface), is_input_(is_input)
00072             {
00073             }
00074         };

```

```

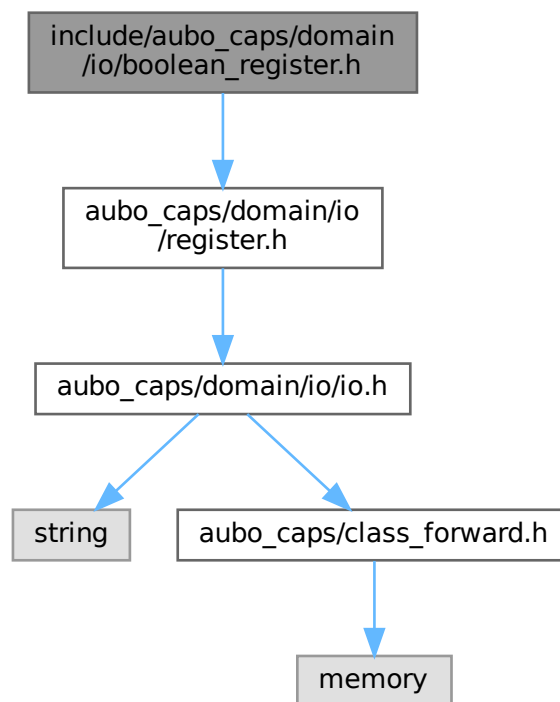
00074     bool operator()(IoPtr io)
00075     {
00076         return std::dynamic_pointer_cast<AnalogIo>(io) != nullptr &&
00077             io->getInterfaceType() == interface_ &&
00078             io->isInput() == is_input_;
00079     }
00080 private:
00081     Io::InterfaceType interface_;
00082     bool is_input_;
00083 };
00084 // namespace aubo_scope
00085 // namespace arcs
00086 #endif // AUBO_SCOPE_ANALOG_IO_H

```

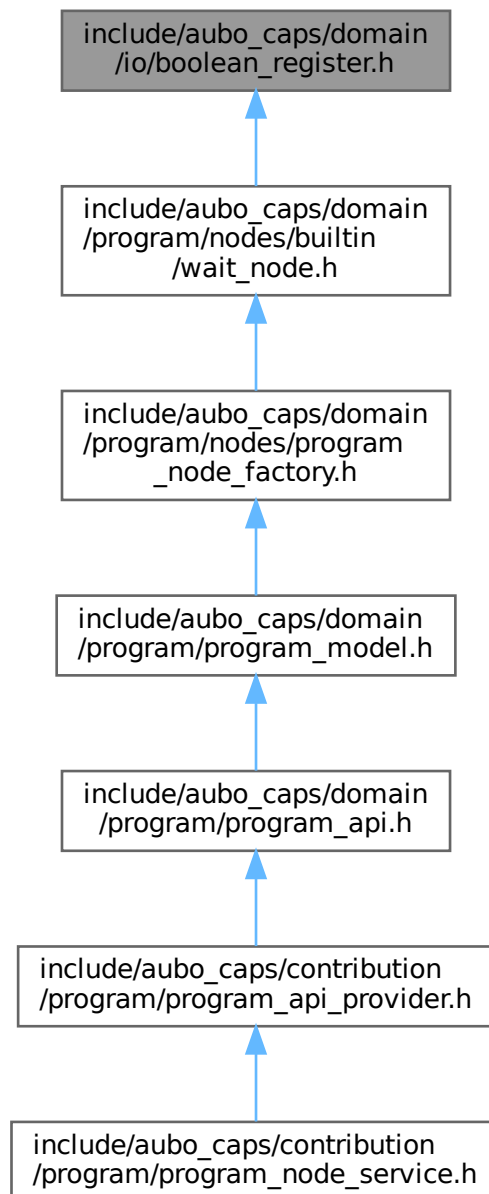
12.98 include/aubo_caps/domain/io/boolean_register.h File Reference

#include <aubo_caps/domain/io/register.h>

Include dependency graph for boolean_register.h:



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



Classes

- class [arcs::aubo_scope::BoolRegister](#)
This interface provides support for bool registers.

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(BoolRegister\)](#)

12.99 boolean_register.h

[Go to the documentation of this file.](#)

```

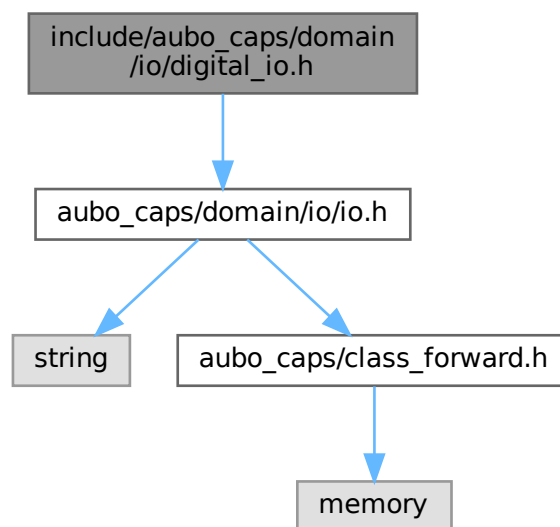
00001 #ifndef AUBO_SCOPE_BOOLEAN_REGISTER_H
00002 #define AUBO_SCOPE_BOOLEAN_REGISTER_H
00003
00004 #include <aubo_caps/domain/io/register.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008   ARCS_CLASS_FORWARD(BoolRegister);
00009
00010 /**
00011  * This interface provides support for bool registers.
00012  */
00013   class ARCS_ABI_EXPORT BoolRegister : public Register
00014   {
00015   public:
00016     virtual ~BoolRegister() = default;
00017   };
00018 } // namespace aubo_scope
00019 } // namespace arcs
00020
00021 #endif // AUBO_SCOPE_BOOLEAN_REGISTER_H

```

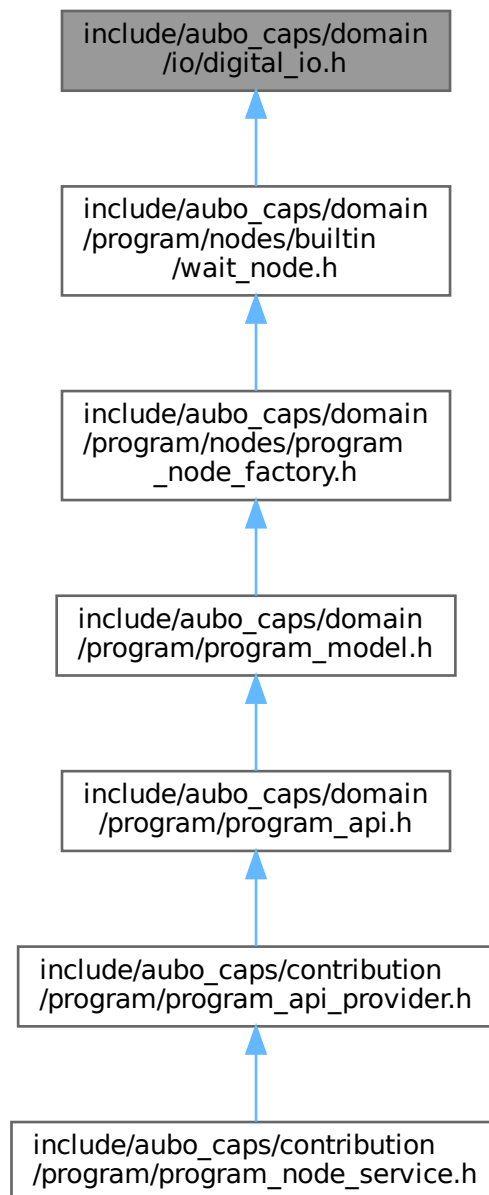
12.100 include/aubo_caps/domain/io/digital_io.h File Reference

```
#include <aubo_caps/domain/io/io.h>
```

Include dependency graph for digital_io.h:



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



Classes

- class [arcs::aubo_scope::DigitalIo](#)
[DigitalIo](#) This interface provides support for digital I/Os.
- struct [arcs::aubo_scope::DigitalIoFilter](#)
- struct [arcs::aubo_scope::InputIoFilter](#)
- struct [arcs::aubo_scope::OutputIoFilter](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([DigitalIo](#))

12.101 digital_io.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_DIGITAL_IO_H
00002 #define AUBO_SCOPE_DIGITAL_IO_H
00003
00004 #include <aubo_caps/domain/io/io.h>
00005
00006 namespace arcs {
00007     namespace aubo_scope {
00008         ARCS_CLASS_FORWARD(DigitalIo);
00009
00010         /**
00011          * \chinese
00012          * IO
00013          * I/O
00014          * \endchinese
00015          * \english
00016          * DigitalIo
00017          * This interface provides support for digital I/Os.
00018          * \endenglish
00019          */
00020         class ARCS_ABI_EXPORT DigitalIo : public Io
00021         {
00022         public:
00023             DigitalIo(DigitalIo &f);
00024             DigitalIo(DigitalIo &&f);
00025             ~DigitalIo();
00026
00027             /**
00028              * \chinese
00029              * I/O I/O
00030              * @param high
00031              * @return <code>true</code> <code>>false</code>
00032              * \endchinese
00033              * \english
00034              * Set digital I/O output signal. Note that the I/O must be an output I/O.
00035              *
00036              * @param high the digital value to be set.
00037              * @return <code>true</code> if the digital signal was set,
00038              * <code>false</code> if the signal was not set, e.g. because the controller
00039              * was not running.
00040              * \endenglish
00041              */
00042             bool setValue(bool high);
00043
00044             /**
00045              * \chinese
00046              * I/O
00047              * @return I/O
00048              * \endchinese
00049              * \english
00050              * Returns the last reading of the I/O.
00051              * @return Returns the last reading of the I/O.
00052              * \endenglish
00053              */
00054             bool getValue();
00055
00056         private:
00057             friend class DataSwitch;
00058             DigitalIo();
00059             void *d_{ nullptr };
00060         };
00061
00062         struct DigitalIoFilter
00063         {
00064             DigitalIoFilter(Io::InterfaceType interface, bool is_input)

```

```

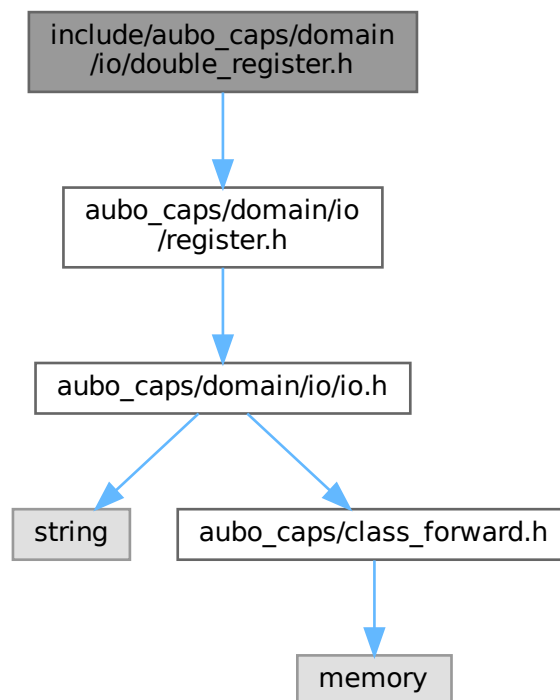
00065     : interface_(interface), is_input_(is_input)
00066     {
00067     }
00068     bool operator()(IoPtr io)
00069     {
00070         return std::dynamic_pointer_cast<DigitalIo>(io) != nullptr &&
00071             io->getInterfaceType() == interface_ &&
00072             io->isInput() == is_input_;
00073     }
00074
00075 private:
00076     Io::InterfaceType interface_;
00077     bool is_input_;
00078 };
00079
00080 struct InputIoFilter
00081 {
00082     bool operator()(IoPtr io) { return io && io->isInput(); }
00083 };
00084
00085 struct OutputIoFilter
00086 {
00087     bool operator()(IoPtr io) { return io && !io->isInput(); }
00088 };
00089
00090 } // namespace aubo_scope
00091 } // namespace arcs
00092
00093 #endif // AUBO_SCOPE_DIGITAL_IO_H

```

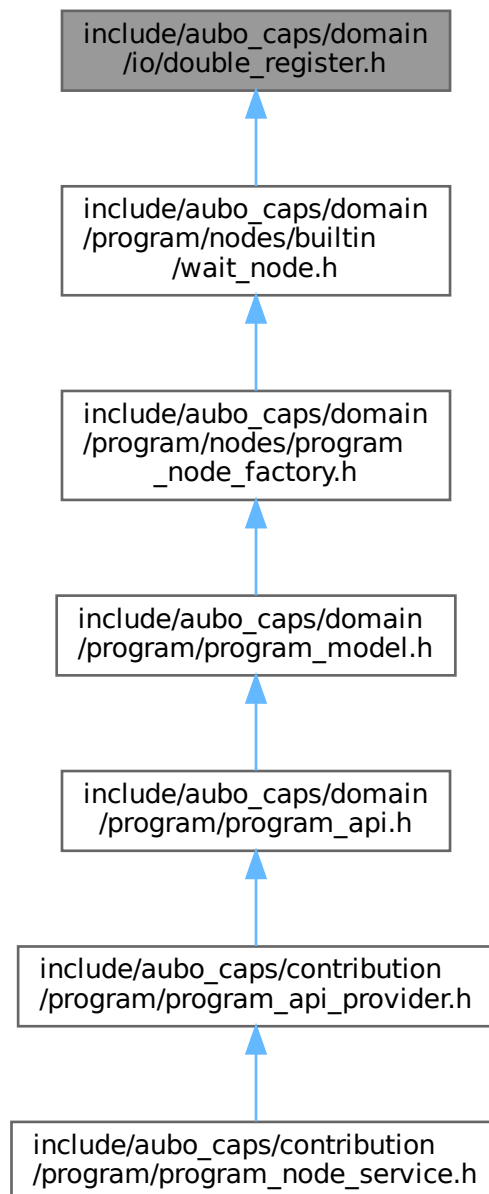
12.102 include/aubo_caps/domain/io/double_register.h File Reference

#include <aubo_caps/domain/io/register.h>

Include dependency graph for double_register.h:



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



Classes

- class [arcs::aubo_scope::DoubleRegister](#)
This interface provides support for float registers.

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([DoubleRegister](#))

12.103 double_register.h

[Go to the documentation of this file.](#)

```

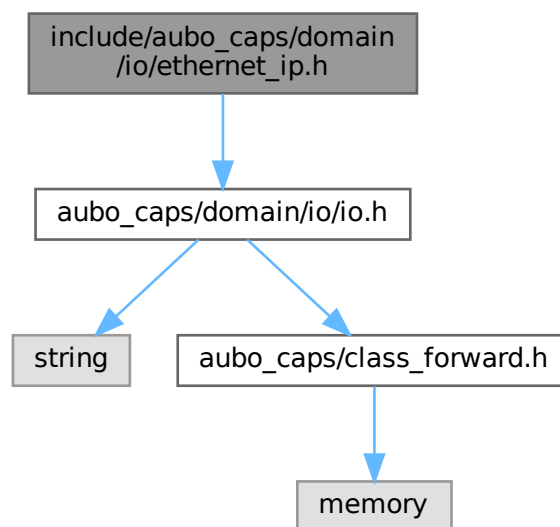
00001 #ifndef AUBO_SCOPE_DOUBLE_REGISTER_H
00002 #define AUBO_SCOPE_DOUBLE_REGISTER_H
00003
00004 #include <aubo_caps/domain/io/register.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008     ARCS_CLASS_FORWARD(DoubleRegister);
00009
00010 /**
00011  * This interface provides support for float registers.
00012  */
00013     class ARCS_ABI_EXPORT DoubleRegister : public Register
00014     {
00015     public:
00016         virtual ~DoubleRegister() = default;
00017     };
00018 } // namespace aubo_scope
00019 } // namespace arcs
00020
00021 #endif // AUBO_SCOPE_DOUBLE_REGISTER_H

```

12.104 include/aubo_caps/domain/io/ethernet_ip.h File Reference

```
#include <aubo_caps/domain/io/io.h>
```

Include dependency graph for ethernet_ip.h:



Classes

- class [arcs::aubo_scope::EtherNetIP](#)
[EtherNetIP](#) This interface provides support for EtherNet/IP I/Os.
- struct [arcs::aubo_scope::EtherNetIPFilter](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([EtherNetIP](#))

12.105 ethernet_ip.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_ETHERNET_IP_H
00002 #define AUBO_SCOPE_ETHERNET_IP_H
00003
00004 #include <aubo_caps/domain/io/io.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(EtherNetIP);
00009
00010 /**
00011  * \chinese
00012  * EtherNet/IP IO
00013  * EtherNet/IP I/O
00014  * \endchinese
00015  * \english
00016  * EtherNetIP
00017  * This interface provides support for EtherNet/IP I/Os.
00018  * \endenglish
00019  */
00020 class ARCS_ABI_EXPORT EtherNetIP : public Io
00021 {
00022 public:
00023 EtherNetIP(EtherNetIP &f);
00024 EtherNetIP(EtherNetIP &&f);
00025 ~EtherNetIP();
00026
00027 /**
00028  * \chinese
00029  * EtherNet/IP I/O
00030  * @return
00031  * \endchinese
00032  * \english
00033  * Returns the byte offset for this EtherNet/IP I/O.
00034  * @return Get the byte offset for this EtherNet/IP I/O.
00035  * \endenglish
00036  */
00037 int getByteOffset();
00038
00039 /**
00040  * \chinese
00041  * EtherNet/IP I/O
00042  * @return
00043  * \endchinese
00044  * \english
00045  * Returns the bit offset for this EtherNet/IP I/O.
00046  * @return Get the bit offset for this EtherNet/IP I/O.
00047  * \endenglish
00048  */
00049 int getBitOffset();
00050
00051 /**
00052  * \chinese
00053  * EtherNet/IP I/O

```

```

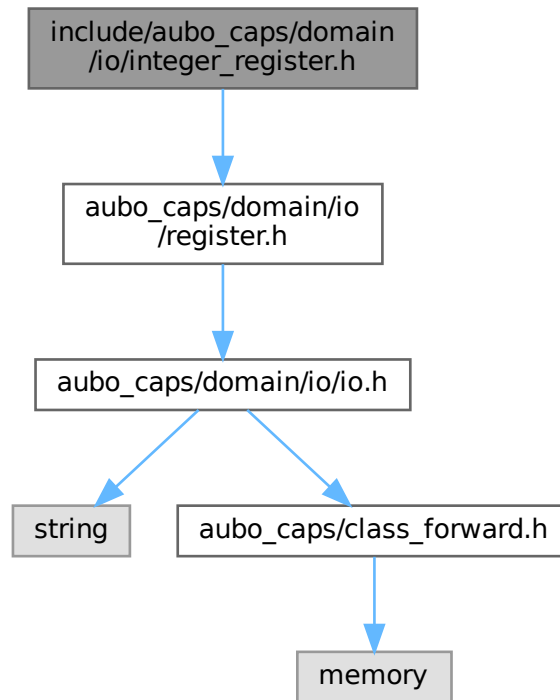
00054     * @return
00055     * \endchinese
00056     * \english
00057     * Returns whether this EtherNet/IP I/O is a slave.
00058     * @return Get the is slave for this EtherNet/IP I/O.
00059     * \endenglish
00060     */
00061     bool isSlave();
00062
00063     /**
00064     * \chinese
00065     * EtherNet/IP
00066     * @param value
00067     * @return <code>true</code> <code>false</code>
00068     * \endchinese
00069     * \english
00070     * Set EtherNet/IP signal to an integer value.
00071     * @param value Set EtherNet/IP signal to an integer value.
00072     * @return <code>true</code> if the EtherNet/IP signal was sent down to the
00073     * controller, and <code>false</code> if the signal was not sent down.
00074     * \endenglish
00075     */
00076     bool setValue(int value);
00077
00078     /**
00079     * \chinese
00080     * EtherNet/IP
00081     * @return EtherNet/IP
00082     * \endchinese
00083     * \english
00084     * Returns the integer value of the EtherNet/IP signal.
00085     * @return the integer value of the EtherNet/IP signal.
00086     * \endenglish
00087     */
00088     int getValue();
00089
00090 private:
00091     friend class DataSwitch;
00092     EtherNetIP();
00093     void *d_{ nullptr };
00094 };
00095
00096 struct EtherNetIPFilter
00097 {
00098     EtherNetIPFilter(Io::IoType type, bool is_input)
00099         : type_(type), is_input_(is_input)
00100     {
00101     }
00102     bool operator()(IoPtr io)
00103     {
00104         return std::dynamic_pointer_cast<EtherNetIP>(io) != nullptr &&
00105             io->getType() == type_ && io->isInput() == is_input_;
00106     }
00107
00108 private:
00109     Io::IoType type_;
00110     bool is_input_;
00111 };
00112
00113 } // namespace aubo_scope
00114 } // namespace arcs
00115
00116 #endif // EtherNet/IPIO_H

```

12.106 include/aubo_caps/domain/io/integer_register.h File Reference

```
#include <aubo_caps/domain/io/register.h>
```

Include dependency graph for integer_register.h:



Classes

- class [arcs::aubo_scope::IntegerRegister](#)
This interface provides support for integer registers.

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([IntegerRegister](#))

12.107 integer_register.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_INTEGER_REGISTER_H
00002 #define AUBO_SCOPE_INTEGER_REGISTER_H
00003
00004 #include <aubo_caps/domain/io/register.h>
00005
00006 namespace arcs {
00007     namespace aubo_scope {
00008         ARCS_CLASS_FORWARD(IntegerRegister);
00009     }
00010     /**
00011      * This interface provides support for integer registers.
00012      */
00013 }
00014 class ARCS_ABI_EXPORT IntegerRegister : public Register
00015 {
00016 public:
00017     virtual ~IntegerRegister() = default;
00018 };
00019
00020 } // namespace aubo_scope
00021 } // namespace arcs
00022
00023 #endif // AUBO_SCOPE_INTEGER_REGISTER_H

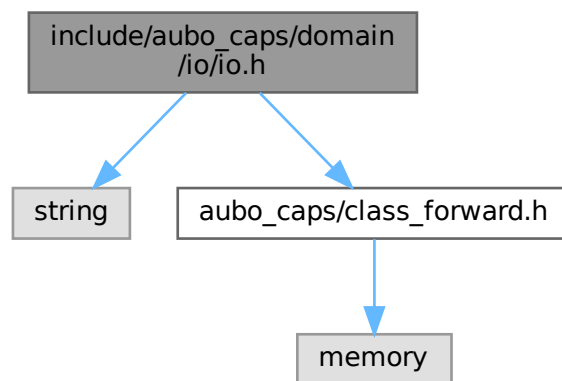
```

12.108 include/aubo_caps/domain/io/io.h File Reference

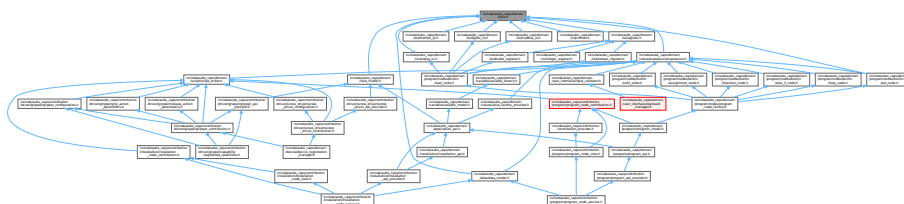
```
#include <string>
```

```
#include <aubo_caps/class_forward.h>
```

Include dependency graph for io.h:



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



Classes

- class [arcs::aubo_scope::Io](#)
[Io](#) This is the base interface representing all types of I/Os available in AuboScope.

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([Io](#))

12.109 io.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_IO_H
00002 #define AUBO_SCOPE_IO_H
00003
00004 #include <string>
00005 #include <aubo_caps/class_forward.h>
00006
00007 namespace arcs {
00008 namespace aubo_scope {
00009 ARCS_CLASS_FORWARD(Io);
00010
00011 /**
00012  * \chinese
00013  * IO
00014  *   AuboScope   I/O
00015  * \endchinese
00016  * \english
00017  * Io
00018  * This is the base interface representing all types of I/Os available in
00019  * AuboScope.
00020  * \endenglish
00021  */
00022 class ARCS_ABI_EXPORT Io
00023 {
00024 public:
00025     enum IoType : int
00026     {
00027         DIGITAL,
00028         ANALOG,
00029         INTEGER,
00030         BOOLEAN,
00031         DOUBLE
00032     };
00033     enum InterfaceType : int
00034     {
00035         STANDARD,
00036         TOOL,
00037         CONFIGURABLE,
00038         LINKIO,
00039         MODBUS,
00040         GENERAL_PURPOSE,
00041         PROFINET,
00042         ETHERNET_IP,
00043     };
00044
00045 public:
00046     Io(Io &f);
00047     Io(Io &&f);
00048     virtual ~Io();
00049
00050     /**
00051     * \chinese
00052     *   I/O
00053     * @return I/O
00054     * \endchinese

```

```

00055     * \english
00056     * Returns the user defined name of the IO in the currently selected
00057     * installation.
00058     * @return the user defined name of the IO.
00059     * \endenglish
00060     */
00061     std::string getDisplayName() const;
00062
00063     /**
00064     * \chinese
00065     * I/O
00066     * @return I/O
00067     * \endchinese
00068     * \english
00069     * Returns the robot default name of the I/O.
00070     * @return the robot default name of the I/O.
00071     * \endenglish
00072     */
00073     std::string getDefaultName() const;
00074
00075     /**
00076     * \chinese
00077     * I/O
00078     * @return I/O
00079     * \endchinese
00080     * \english
00081     * Returns the current reading of the I/O as a string.
00082     * @return get current reading of the I/O as a string.
00083     * \endenglish
00084     */
00085     std::string getValueStr() const;
00086
00087     /**
00088     * \chinese
00089     * I/O
00090     * @return I/O
00091     * \endchinese
00092     * \english
00093     * Returns the I/O type.
00094     * @return the I/O type.
00095     * \endenglish
00096     */
00097     IoType getType() const;
00098
00099     /**
00100     * \chinese
00101     * I/O
00102     * @return
00103     * \endchinese
00104     * \english
00105     * Returns the interface type this I/O belongs to.
00106     * @return the interface type this I/O belongs to.
00107     * \endenglish
00108     */
00109     InterfaceType getInterfaceType() const;
00110
00111     /**
00112     * \chinese
00113     * I/O
00114     * @return I/O
00115     * \endchinese
00116     * \english
00117     * Returns the action number of the I/O.
00118     * @return the action number of the I/O.
00119     * \endenglish
00120     */
00121     int getAction() const;
00122
00123     /**
00124     * \chinese
00125     * I/O
00126     * @return I/O <code>true</code> <code>>false</code>
00127     * \endchinese
00128     * \english
00129     * Returns <code>true</code> if I/O supports reading of values and
00130     * <code>>false</code> otherwise.
00131     * @return <code>true</code> if I/O supports reading of values.
00132     * \endenglish
00133     */
00134     bool isInput() const;
00135
00136     /**
00137     * \chinese
00138     * I/O MODBUS I/O
00139     * I/O
00140     * @return I/O <code>true</code> <code>>false</code>
00141     * \endchinese

```

```

00142     * \english
00143     * Some I/Os, such as MODBUS I/Os and analog Tool inputs, are in some
00144     * situations not present in AuboScope or available for use. This method can
00145     * be used to determine if the I/O is available.
00146     * @return <code>true</code>, if I/O can be resolved.
00147     *         <code>false</code> otherwise.
00148     * \endenglish
00149     */
00150     bool isResolvable() const;
00151
00152     size_t hashCode();
00153
00154 protected:
00155     Io();
00156
00157 private:
00158     friend class DataSwitch;
00159     void *d_{ nullptr };
00160 };
00161
00162 } // namespace aubo_scope
00163 } // namespace arcs
00164
00165 #endif // AUBO_SCOPE_IO_H

```

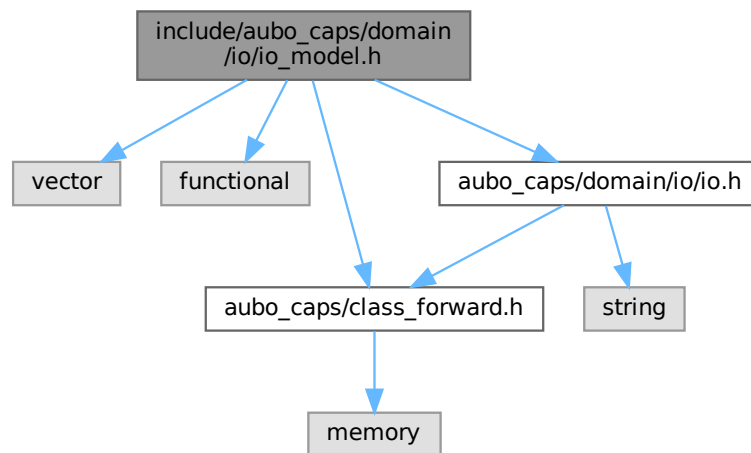
12.110 include/aubo_caps/domain/io/io_model.h File Reference

```

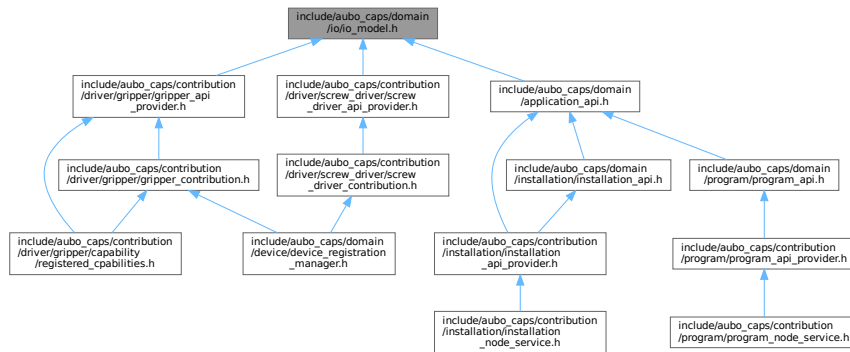
#include <vector>
#include <functional>
#include <aubo_caps/class_forward.h>
#include <aubo_caps/domain/io/io.h>

```

Include dependency graph for io_model.h:



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



Classes

- class [arcs::aubo_scope::IoModel](#)
[IoModel](#) Provides methods that returns I/Os from the current robot installation.

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(IoModel\)](#)

12.111 io_model.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_IO_MODEL_H
00002 #define AUBO_SCOPE_IO_MODEL_H
00003
00004 #include <vector>
00005 #include <functional>
00006 #include <aubo_caps/class_forward.h>
00007 #include <aubo_caps/domain/io/io.h>
00008
00009 namespace arcs {
00010 namespace aubo_scope {
00011 ARCS_CLASS_FORWARD(IoModel);
00012
00013 /**
00014 * @ingroup ApplicationApi
00015 * \chinese
00016 * IO
00017 * I/O
00018 * \endchinese
00019 * \english
00020 * IoModel
00021 * Provides methods that returns I/Os from the current robot installation.
00022 * \endenglish
00023 */
00024 class ARCS_ABI_EXPORT IoModel
00025 {
00026 public:

```

```

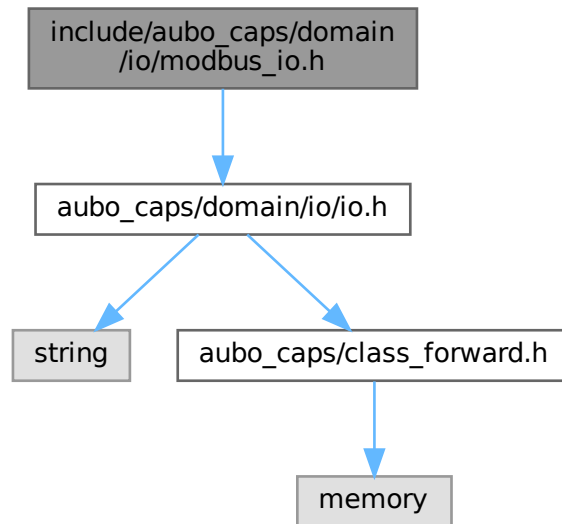
00027 IoModel(IoModel &f);
00028 IoModel(IoModel &&f);
00029 ~IoModel();
00030
00031 /**
00032  * @ingroup IoModel
00033  * \chinese
00034  * I/O
00035  * @return I/O
00036  * \endchinese
00037  * \english
00038  * Returns the collection of available I/Os, including Modbus, EuroMap67 and
00039  * named General Purpose Registers.
00040  * @return the collection of available I/Os.
00041  * \endenglish
00042  */
00043 std::vector<IoPtr> getAll();
00044
00045 /**
00046  * @ingroup IoModel
00047  * @ref Io
00048  * \chinese
00049  * I/O
00050  * @param name I/O
00051  * @return I/O
00052  * \endchinese
00053  * \english
00054  * Get an I/O by name.
00055  * @param name the name of the I/O.
00056  * @return the I/O with the given name.
00057  * \endenglish
00058  */
00059 IoPtr getIo(const std::string &name);
00060
00061 /**
00062  * @ingroup IoModel
00063  * \chinese
00064  * I/O
00065  * @param filter
00066  * @return I/O
00067  * \endchinese
00068  * \english
00069  * Get a subset of I/Os using a filter.
00070  * @param filter see {@link IOFilterFactory}.
00071  * @return the collection of IOs that are accepted by the filter.
00072  * \endenglish
00073  */
00074 std::vector<IoPtr> getIos(std::function<bool(IoPtr)> filter);
00075
00076 private:
00077 friend class DataSwitch;
00078 IoModel();
00079 void *d_{ nullptr };
00080 };
00081
00082 } // namespace aubo_scope
00083 } // namespace arcs
00084 #endif

```

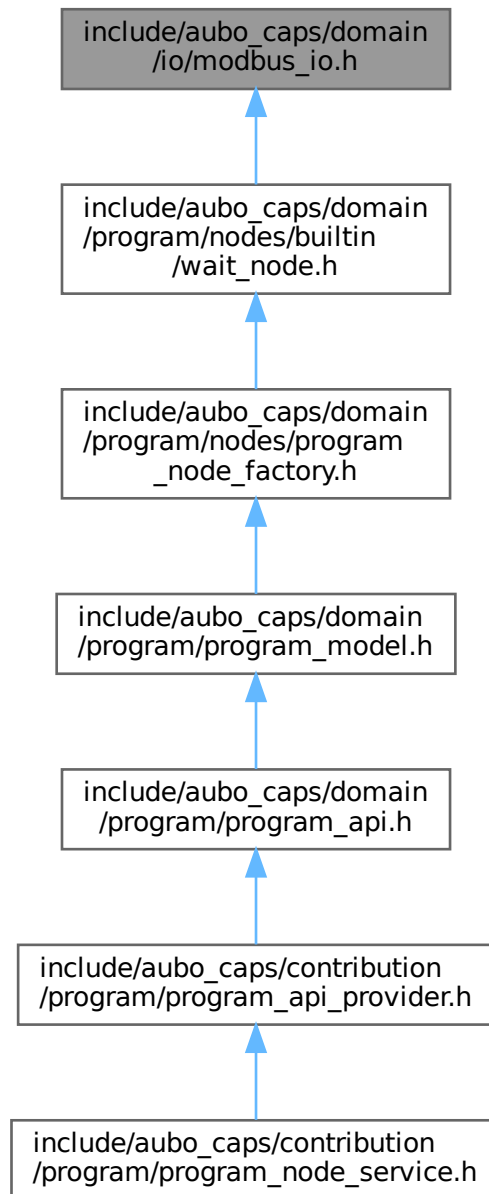
12.112 include/aubo_caps/domain/io/modbus_io.h File Reference

```
#include <aubo_caps/domain/io/io.h>
```

Include dependency graph for modbus_io.h:



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



Classes

- class [arcs::aubo_scope::ModbusIo](#)
[ModbusIo](#) This interface provides support for MODBUS I/Os.
- struct [arcs::aubo_scope::ModbusIoFilter](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(ModbusIo\)](#)

12.113 modbus_io.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_MODBUS_IO_H
00002 #define AUBO_SCOPE_MODBUS_IO_H
00003
00004 #include <aubo_caps/domain/io/io.h>
00005
00006 namespace arcs {
00007     namespace aubo_scope {
00008         ARCS_CLASS_FORWARD(ModbusIo);
00009
00010         /**
00011          * @ingroup ApplicationApi
00012          * \chinese
00013          * Modbus IO
00014          *     MODBUS I/O
00015          * \endchinese
00016          * \english
00017          * ModbusIo
00018          * This interface provides support for MODBUS I/Os.
00019          * \endenglish
00020          */
00021         class ARCS_ABI_EXPORT ModbusIo : public Io
00022         {
00023         public:
00024             ModbusIo(ModbusIo &f);
00025             ModbusIo(ModbusIo &&f);
00026             ~ModbusIo();
00027
00028             /**
00029              * @ingroup ModbusIo
00030              * \chinese
00031              * MODBUS I/O IP
00032              * @return IP
00033              * \endchinese
00034              * \english
00035              * Returns the IP address for this MODBUS I/O.
00036              * @return Get the IP address for this MODBUS I/O.
00037              * \endenglish
00038              */
00039             std::string getIpAddress();
00040
00041             /**
00042              * @ingroup ModbusIo
00043              * \chinese
00044              * MODBUS I/O
00045              * @return
00046              * \endchinese
00047              * \english
00048              * Returns the signal address for this MODBUS I/O.
00049              * @return the signal address for this MODBUS I/O.
00050              * \endenglish
00051              */
00052             std::string getSignalAddress();
00053
00054             /**
00055              * @ingroup ModbusIo
00056              * \chinese
00057              * MODBUS I/O
00058              * @return
00059              * \endchinese
00060              * \english
00061              * Returns whether this MODBUS I/O is a slave.
00062              * @return Get the is slave for this MODBUS I/O.
00063              * \endenglish
00064              */
00065             bool isSlave();
00066
00067             /**
00068              * @ingroup ModbusIo
00069              * \chinese
00070              * MODBUS
00071              * @param value
00072              * @return <code>true</code> <code>false</code>
00073              * \endchinese

```

```

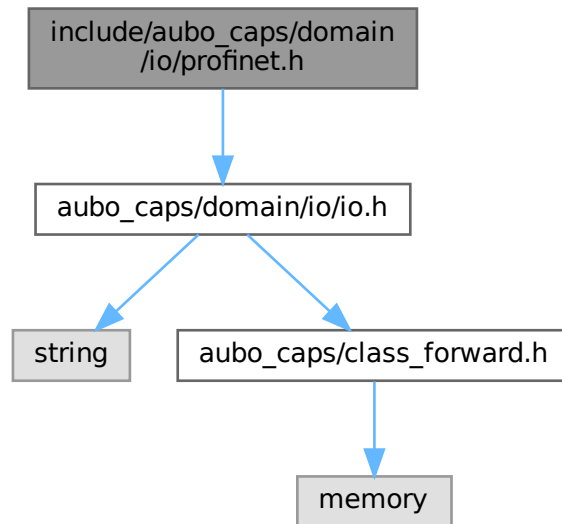
00074     * \english
00075     * Set MODBUS signal to an integer value.
00076     * @param value Set MODBUS signal to an integer value.
00077     * @return <code>true</code> if the MODBUS signal was sent down to the
00078     * controller, and <code>false</code> if the signal was not sent down.
00079     * \endenglish
00080     */
00081     bool setValue(int value);
00082
00083     /**
00084     * @ingroup ModbusIo
00085     * \chinese
00086     * MODBUS
00087     * @return MODBUS
00088     * \endchinese
00089     * \english
00090     * Returns the integer value of the MODBUS signal.
00091     * @return the integer value of the MODBUS signal.
00092     * \endenglish
00093     */
00094     int getValue();
00095
00096 private:
00097     friend class DataSwitch;
00098     ModbusIo();
00099     void *d_{ nullptr };
00100 };
00101
00102 struct ModbusIoFilter
00103 {
00104     ModbusIoFilter(Io::IoType type, bool is_input)
00105         : type_(type), is_input_(is_input)
00106     {
00107     }
00108     bool operator()(IoPtr io)
00109     {
00110         return std::dynamic_pointer_cast<ModbusIo>(io) != nullptr &&
00111             io->getType() == type_ && io->isInput() == is_input_;
00112     }
00113
00114 private:
00115     Io::IoType type_;
00116     bool is_input_;
00117 };
00118
00119 } // namespace aubo_scope
00120 } // namespace arcs
00121
00122 #endif // MODBUSIO_H

```

12.114 include/aubo_caps/domain/io/profinet.h File Reference

```
#include <aubo_caps/domain/io/io.h>
```

Include dependency graph for profinet.h:



Classes

- class [arcs::aubo_scope::Profinet](#)
[Profinet](#) This interface provides support for PROFINET I/Os.
- struct [arcs::aubo_scope::ProfinetFilter](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([Profinet](#))

12.115 profinet.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_PROFINET_H
00002 #define AUBO_SCOPE_PROFINET_H
00003
00004 #include <aubo_caps/domain/io/io.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(Profinet);
00009
00010 /**
00011  * @ingroup ApplicationApi
00012  * \chinese
00013  * PROFINET IO
00014  * PROFINET I/O
00015  * \endchinese
00016  * \english
00017  * Profinet
00018  * This interface provides support for PROFINET I/Os.
00019  * \endenglish
00020  */
00021 class ARCS_ABI_EXPORT Profinet : public Io
00022 {
00023 public:
00024     Profinet(Profinet &f);
00025     Profinet(Profinet &&f);
00026     ~Profinet();
00027
00028     /**
00029     * @ingroup Profinet
00030     * \chinese
00031     * PROFINET I/O
00032     * @return
00033     * \endchinese
00034     * \english
00035     * Returns the byte offset for this Profinet I/O.
00036     * @return Get the byte offset for this Profinet I/O.
00037     * \endenglish
00038     */
00039     int getByteOffset();
00040
00041     /**
00042     * @ingroup Profinet
00043     * \chinese
00044     * PROFINET I/O
00045     * @return
00046     * \endchinese
00047     * \english
00048     * Returns the bit offset for this Profinet I/O.
00049     * @return Get the bit offset for this Profinet I/O.
00050     * \endenglish
00051     */
00052     int getBitOffset();
00053
00054     /**
00055     * @ingroup Profinet
00056     * \chinese
00057     * PROFINET I/O
00058     * @return
00059     * \endchinese
00060     * \english
00061     * Returns whether this Profinet I/O is a slave.
00062     * @return <code>true</code> if this is a slave.
00063     * \endenglish
00064     */
00065     bool isSlave();
00066
00067     /**
00068     * @ingroup Profinet
00069     * \chinese
00070     * PROFINET PROFINET <code>>false</code> 0
00071     * <code>true</code> 0
00072     *
00073     * @param value
00074     * @return PROFINET <code>true</code>
00075     * <code>>false</code> PROFINET
00076     * Aubo PROFINET
00077     * \endchinese
00078     * \english
00079     * Set PROFINET signal to an integer value. If the PROFINET signal is
00080     * digital, then <code>>false</code> is represented by 0 and
00081     * <code>true</code> are represented by all numbers different from 0.
00082     */

```



```

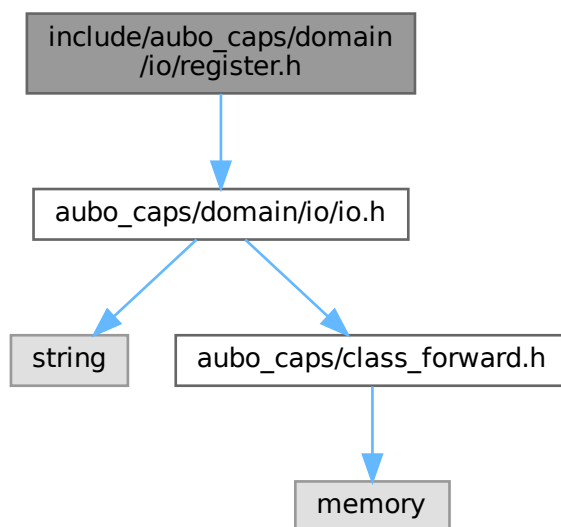
00083     * @param value the integer value to set.
00084     * @return <code>true</code> if the PROFINET signal was sent down to the
00085     * controller, and <code>false</code> if the signal was not sent down, e.g.
00086     * because the controller was not running. Notice that the PROFINET signal
00087     * must support writes, consult the Aubo PROFINET protocol.
00088     * \endenglish
00089     */
00090     bool setValue(int value);
00091
00092     /**
00093     * @ingroup Profinet
00094     * \chinese
00095     *   PROFINET   PROFINET   <code>false</code> 0
00096     *   <code>true</code> 1
00097     * @return PROFINET
00098     * \endchinese
00099     * \english
00100     * Returns the integer value of the PROFINET signal. If the PROFINET signal
00101     * is digital then <code>false</code> is represented by 0 and
00102     * <code>true</code> is represented by the value 1.
00103     * @return the integer value of the PROFINET signal.
00104     * \endenglish
00105     */
00106     int getValue();
00107
00108 private:
00109     friend class DataSwitch;
00110     Profinet();
00111     void *d_{ nullptr };
00112 };
00113
00114 struct ProfinetFilter
00115 {
00116     ProfinetFilter(Io::IoType type, bool is_input)
00117         : type_(type), is_input_(is_input)
00118     {
00119     }
00120     bool operator()(IoPtr io)
00121     {
00122         return std::dynamic_pointer_cast<Profinet>(io) != nullptr &&
00123             io->getType() == type_ && io->isInput() == is_input_;
00124     }
00125
00126 private:
00127     Io::IoType type_;
00128     bool is_input_;
00129 };
00130
00131 } // namespace aubo_scope
00132 } // namespace arcs
00133
00134 #endif // PROFINETIO_H

```

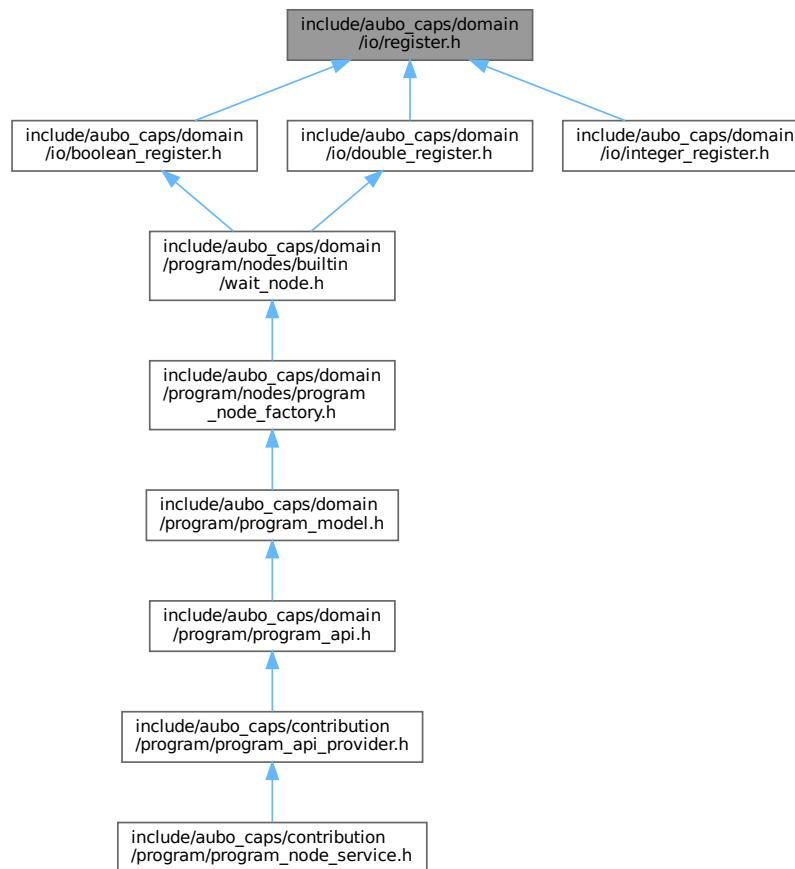
12.116 include/aubo_caps/domain/io/register.h File Reference

#include <aubo_caps/domain/io/io.h>

Include dependency graph for register.h:



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



Classes

- class [arcs::aubo_scope::Register](#)
This interface provides support for registers.

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(Register\)](#)

12.117 register.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_REGISTER_H
00002 #define AUBO_SCOPE_REGISTER_H
00003
00004 #include <aubo_caps/domain/io/io.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(Register);
00009
00010 /**
00011  * This interface provides support for registers.
00012  *
00013  */
00014 class ARCS_ABI_EXPORT Register : public Io
00015 {
00016 public:
00017     virtual ~Register() = default;
00018 };
00019
00020 } // namespace aubo_scope
00021 } // namespace arcs
00022
00023 #endif // REGISTER_H

```

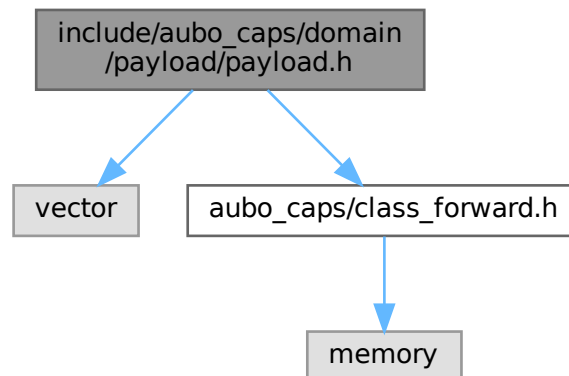
12.118 include/aubo_caps/domain/payload/payload.h File Reference

```

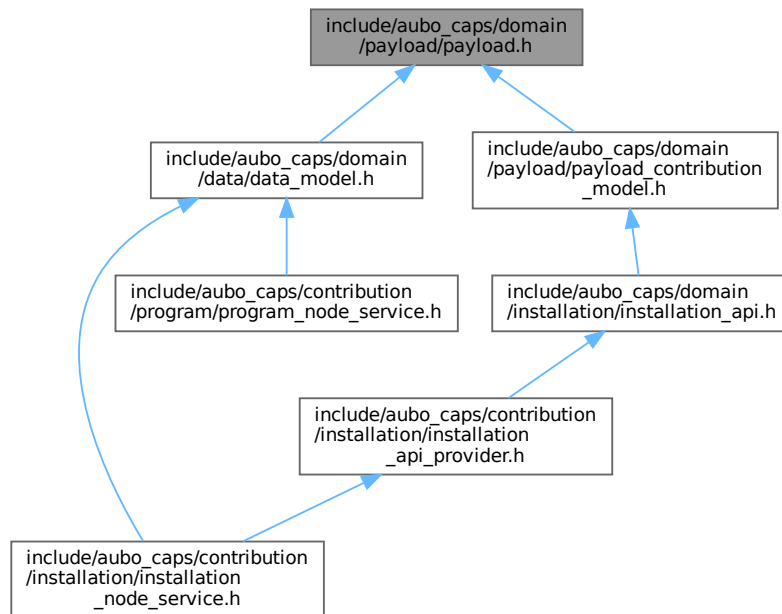
#include <vector>
#include <aubo_caps/class_forward.h>

```

Include dependency graph for payload.h:



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



Classes

- class `arcs::aubo_scope::Payload`
API interface for payload parameters.

Namespaces

- namespace `arcs`
- namespace `arcs::aubo_scope`

Functions

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (Payload)`

12.119 payload.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_PAYLOAD_H
00002 #define AUBO_SCOPE_PAYLOAD_H
00003
00004 #include <vector>
00005 #include <aubo_caps/class_forward.h>
00006
00007 namespace arcs {
00008 namespace aubo_scope {
00009 ARCS_CLASS_FORWARD(Payload);
00010
00011 /**

```

```

00012 * @ingroup InstallationApi
00013 * \~chinese API
00014 * \~english API interface for payload parameters
00015 */
00016 class ARCS_ABI_EXPORT Payload
00017 {
00018 public:
00019     Payload(Payload &f);
00020     Payload(Payload &&f);
00021     ~Payload();
00022
00023     /**
00024      * <p>
00025      * This enum represents the different types of warnings that the parameters
00026      * of a payload can have.
00027      * </p>
00028      *
00029      * A payload can be usable, but may have a value for a payload parameter or
00030      * may have a combination of (some of the) payload parameters, that could
00031      * affect performance of the robot, if the payload is used (i.e., if the
00032      * parameters are applied).
00033      */
00034     enum ParametersWarning
00035     {
00036
00037         /**
00038          * None of the parameters of the payload or the combination of payload
00039          * parameters has any warnings.
00040          */
00041         NO_WARNING,
00042
00043         /**
00044          * The payload has a mass of 0 kg which is most likely not
00045          * intended/valid. Note that the value for the payload mass should be
00046          * the total mass of the payload attached to the tool output flange of
00047          * the robot.
00048          */
00049         MASS_0KG,
00050
00051         /**
00052          * The mass of the payload exceeds the rated maximum supported payload
00053          * mass of the robot. Using the payload could affect performance of the
00054          * robot.
00055          */
00056         RATED_MASS_EXCEEDED,
00057
00058         /**
00059          * The combination of the mass and the center of gravity of could affect
00060          * the performance of the robot, if the payload is used.
00061          */
00062         MASS_AND_CENTER_OF_GRAVITY_EXCEEDED,
00063     };
00064
00065     /**
00066      * @ingroup Payload
00067      * @return The total mass of the payload (attached to the tool output flange
00068      * of the robot)
00069      */
00070     double getMass() const;
00071
00072     /**
00073      * Gets the payload's center of gravity (CoG), also referred to as center of
00074      * mass. It is defined as the offset between the center of the tool output
00075      * flange and the center of gravity of the attached payload.
00076      *
00077      * @return The center of gravity (CoG) of the payload relative to the center
00078      * of the tool output flange
00079      */
00080     std::vector<double> getCenterOfGravity() const;
00081
00082     /**
00083      * @ingroup Payload
00084      * <p>
00085      * Gets the inertia matrix of the payload. The inertia matrix is defined in
00086      * a coordinate system with center at the payload's center of gravity (CoG)
00087      * and the axes aligned with the tool output flange coordinate system
00088      * (axes).
00089      * </p>
00090      *
00091      * <b>Note:</b> The inertia matrix specified by the end user can be invalid.
00092      * The {@link #isUsable} method can be used determine, if the payload can be
00093      * used (i.e., has a valid inertia matrix).
00094      *
00095      * @return The inertia matrix of the payload
00096      */
00097     std::vector<double> getInertiaMatrix() const;
00098
00099

```

```

00099  /**
00100  * @ingroup Payload
00101  * <p>
00102  * Gets the name of the payload displayed in the AuboScope UI.
00103  * </p>
00104  *
00105  * <b>Note:</b> The payload can change name at any time, and thus the name
00106  * must not be used to uniquely identify a specific payload. The payload
00107  * name can change in the following situations: <ul> <li> The payload is
00108  * renamed by the end user </li> <li> The name gets translated (depending on
00109  * the selected language in AuboScope) </li> <li> The name is changed
00110  * between versions of the AuboCap that added the payload</li>
00111  * </ul>
00112  *
00113  * @return The name of the payload displayed in AuboScope
00114  */
00115  std::string getDisplayName() const;
00116
00117  /**
00118  * @ingroup Payload
00119  * <p>
00120  * A payload cannot be guaranteed to be present in AuboScope. This method
00121  * can be used to determine, if the payload is present.
00122  * </p>
00123  *
00124  * <p>
00125  * The payload will not be present, if the end user loads a different
00126  * installation which does not contain the payload, or if the payload is
00127  * removed by the end user or the AuboCap that added the payload.
00128  * </p>
00129  *
00130  * <b>Note:</b> Storing an undefined payload in a program node
00131  * contribution's (a {@link ProgramNodeContribution} instance) data model
00132  * ({@link DataModel} interface) will automatically make the program node
00133  * (and the program) undefined.
00134  *
00135  * @return <code>true</code> if this payload is present in AuboScope,
00136  * otherwise <code>false</code>.
00137  */
00138  bool isResolvable();
00139
00140  /**
00141  * @ingroup Payload
00142  * <p>
00143  * Use this method to determine if the payload can be used. An unresolvable
00144  * payload (see the {@link #isResolvable} method) will always be unusable.
00145  * </p>
00146  *
00147  * <p>
00148  * A payload can have an invalid inertia matrix, where all the individual
00149  * values of the matrix are within the valid range, but the combination of
00150  * values is invalid.
00151  * </p>
00152  *
00153  * <b>Note:</b> Storing an unusable payload in a program node contribution's
00154  * (a {@link ProgramNodeContribution} instance) data model ({@link
00155  * DataModel} interface) will automatically make the program node (and the
00156  * program) undefined.
00157  *
00158  * @return <code>true</code>, if the payload is usable, otherwise
00159  * <code>false</code>. <code>false</code> will always be returned when the
00160  * payload is unresolvable (see the {@link #isResolvable} method).
00161  */
00162  bool isUsable();
00163
00164  /**
00165  * @ingroup Payload
00166  * <p>
00167  * Use this method to check if the parameters of this payload have any
00168  * warnings.
00169  * </p>
00170  *
00171  * <p>
00172  * A payload can be usable, but may have a value for a payload parameter or
00173  * may have a combination of (some of the) payload parameters, that could
00174  * affect performance of the robot, if the payload is used (i.e., if the
00175  * parameters are applied).
00176  * </p>
00177  *
00178  * An unresolvable payload (see {@link #isResolvable()}) will have no
00179  * warning for its parameters (i.e.
00180  * {@link ParametersWarning#NO_WARNING} is returned).
00181  *
00182  * @return The warning (if any) for the parameters of this payload.
00183  */
00184  ParametersWarning getParametersWarning();
00185

```

```

00186     size_t hashCode();
00187
00188 private:
00189     friend class DataSwitch;
00190     Payload();
00191     void *d_{ nullptr };
00192 };
00193 } // namespace aubo_scope
00194 } // namespace arcs
00195
00196 #endif // AUBO_SCOPE_PAYLOAD_H

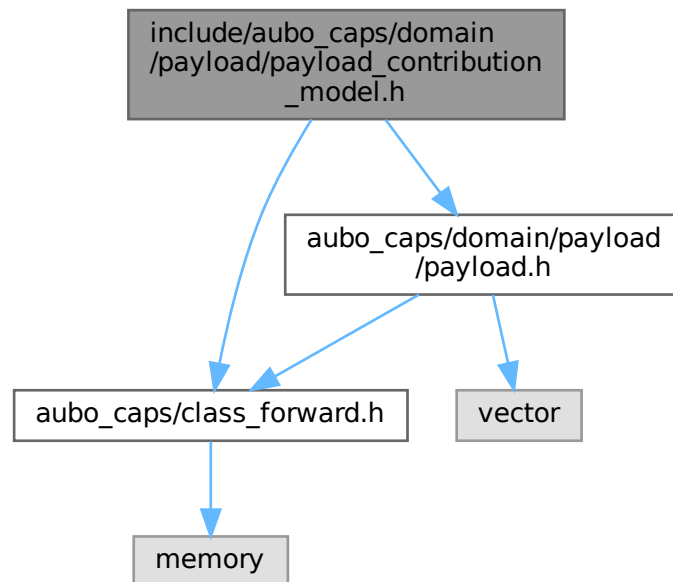
```

12.120 include/auco_caps/domain/payload/payload_contribution_↔ model.h File Reference

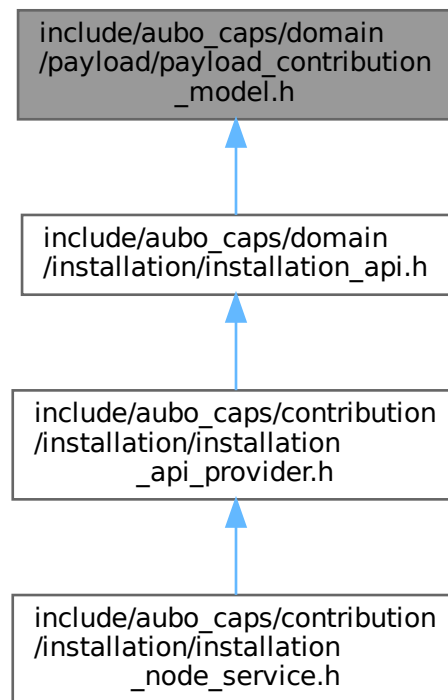
#include <auco_caps/class_forward.h>

#include <auco_caps/domain/payload/payload.h>

Include dependency graph for payload_contribution_model.h:



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



Classes

- class [arcs::aubo_scope::PayloadContributionModel](#)
API interface for dynamic payload management.

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([PayloadContributionModel](#))

12.121 payload_contribution_model.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_PAYLOAD_CONTRIBUTION_MODEL_H
00002 #define AUBO_SCOPE_PAYLOAD_CONTRIBUTION_MODEL_H
00003
00004 #include <aubo_caps/class_forward.h>

```

```

00005 #include <aubo_caps/domain/payload/payload.h>
00006
00007 namespace arcs {
00008 namespace aubo_scope {
00009 ARCS_CLASS_FORWARD(PayloadContributionModel);
00010
00011 /**
00012  * @ingroup InstallationApi
00013  * \~chinese
00014  * \~english API interface for dynamic payload management
00015  */
00016
00017 class ARCS_ABI_EXPORT PayloadContributionModel
00018 {
00019 public:
00020     PayloadContributionModel(PayloadContributionModel &f);
00021     PayloadContributionModel(PayloadContributionModel &&f);
00022     ~PayloadContributionModel();
00023
00024     /**
00025      * @ingroup PayloadContributionModel
00026      * @ref Payload
00027      * \chinese
00028      * <p>
00029      *   AuboScope
00030      * </p>
00031      *
00032      * <p>
00033      *   AuboCap    {@link PayloadModel}
00034      *   aubo_studio
00035      * </p>
00036      *
00037      * <p>
00038      *   {@link #addPayload(String, String, Mass, Position,
00039      *   InertiaMatrix)}
00040      * </p>
00041      *
00042      * <p>
00043      * <b> </b>
00044      * </p>
00045      *
00046      *   CoG    {@link RobotLimits}
00047      *   {@link RobotModel#getRobotLimits()}
00048      *
00049      * @param idKey    <code>null</code>
00050      * @param AuboCap    AuboCap    aubo_studio " "
00051      * @param suggestedName    <code>null</code>
00052      *   [a-zA-Z][a-zA-Z0-9_]{0,14} 15
00053      *
00054      * @param mass    <code>null</code>
00055      *   {@link SimpleValueFactory#createMass(double, Mass.Unit)}
00056      * @param centerOfGravity    CoG
00057      *   <code>null</code> {@link PositionFactory}
00058      *
00059      * @return AuboScope
00060      * @throws PayloadAlreadyAddedException    <code>idKey</code>
00061      *   {@link #getPayload(String)}
00062      *   {@link #updatePayload(String, Mass, Position)}
00063      *   {@link #updatePayload(String, Mass, Position, InertiaMatrix)}
00064      * @throws IllegalPayloadNameException
00065      * @throws IllegalMassException    AuboScope
00066      * @throws IllegalCenterOfGravityException    AuboScope
00067      *
00068      * \endchinese
00069      * \english
00070      * <p>
00071      *   Add a payload to the current installation in AuboScope.
00072      * </p>
00073      *
00074      * <p>
00075      *   This makes the payload selectable by the end user as well as available to
00076      *   other AuboCap contributions (through the
00077      *   {@link PayloadModel} interface). The payload is not modifiable by the end
00078      *   user or other aubo_studio plugins.
00079      * </p>
00080      *
00081      * <p>
00082      *   A default guess for the inertia matrix for the payload will be computed
00083      *   based on the specified payload mass and center of gravity. If data for
00084      *   the payload's inertia matrix is available, the method
00085      *   {@link #addPayload(String, String, Mass, Position, InertiaMatrix)} can
00086      *   be used instead (this might give better performance of the robot arm).
00087      * </p>
00088      *
00089      * <p>
00090      * <b>Note:</b> The specified payload mass must be the total mass of the
00091

```

```

00092 * payload attached to the tool output flange of the robot.
00093 * </p>
00094 *
00095 * The valid range for the payload mass and the coordinates of the center of
00096 * gravity (CoG) are available through the
00097 * {@link RobotLimits} interface which can be accessed with {@link
00098 * RobotModel#getRobotLimits()} (an instance of the
00099 * {@link RobotModel} interface can be retrieved through {@link
00100 * SystemAPI#getRobotModel()}).
00101 *
00102 * @param idKey The key to identify this payload by, not <code>null</code>
00103 * nor an empty string. The key is for this AuboCap only, i.e. it only has
00104 * to be unique for this AuboCap and not "globally" for other aubo_studio
00105 * plugins.
00106 * @param suggestedName Suggested name for the payload, not
00107 * <code>null</code> nor an empty string. Valid names must match regex
00108 * [a-zA-Z][a-zA-Z0-9_]{0,14} for a total of 15 characters. The final name
00109 * can be retrieved from the returned payload instance.
00110 * @param mass The total mass of the payload attached to the tool output
00111 * flange of the robot, not <code>null</code>. Create it using {@link
00112 * SimpleValueFactory#createMass(double, Mass.Unit)}.
00113 * @param centerOfGravity The center of gravity (CoG), also referred to as
00114 * center of mass, for the payload. It is defined as the offset between the
00115 * center of the tool output flange and the center of gravity of the
00116 * attached payload. Cannot be <code>null</code>. Create it using the {@link
00117 * PositionFactory} interface.
00118 * @return The payload created and registered in AuboScope.
00119 * @throws PayloadAlreadyAddedException If a payload has previously been
00120 * added using the same <code>idKey</code> identifier. Use {@link
00121 * #getPayload(String)} to check if the payload has already been added to
00122 * the current installation. Use {@link #updatePayload(String, Mass,
00123 * Position)} or
00124 * {@link #updatePayload(String, Mass,
00125 * Position, InertiaMatrix)} to update the payload if the payload parameters
00126 * have changed.
00127 * @throws IllegalPayloadNameException If the suggested name does not match
00128 * required regex.
00129 * @throws IllegalMassException If the specified mass is not inside the
00130 * valid range as defined by AuboScope.
00131 * @throws IllegalCenterOfGravityException If any of the values of the
00132 * specified center of gravity are not inside the valid range as defined by
00133 * AuboScope.
00134 * \endenglish
00135 */
00136 PayloadPtr addPayload(const std::string &id_key,
00137                      const std::string &suggested_name, double mass,
00138                      const std::vector<double> &center_of_gravity);
00139
00140 /**
00141 * @ingroup PayloadContributionModel
00142 * @ref Payload
00143 * \chinese
00144 * <p>
00145 * AuboScope
00146 * </p>
00147 *
00148 * <p>
00149 * AuboCap    {@link PayloadModel}
00150 * aubo_studio
00151 * </p>
00152 *
00153 * {@link #addPayload(String, String, Mass,
00154 * Position, InertiaMatrix)}
00155 *
00156 * <p>
00157 * <b> </b>
00158 * </p>
00159 *
00160 * CoG    {@link RobotLimits}
00161 * {@link RobotModel#getRobotLimits()}
00162 *
00163 * @param idKey    <code>null</code>
00164 * AuboCap
00165 * @param suggestedName    <code>null</code>
00166 * @param mass    <code>null</code>
00167 * @param centerOfGravity    CoG    <code>null</code>
00168 * @param inertiaMatrix
00169 * <code>null</code>
00170 * @return AuboScope
00171 * @throws PayloadAlreadyAddedException    <code>idKey</code>
00172 *
00173 * @throws IllegalPayloadNameException
00174 * @throws IllegalMassException
00175 * @throws IllegalCenterOfGravityException
00176 * @throws IllegalInertiaMatrixException
00177 * \endchinese
00178 * \english

```

```

00179 * <p>
00180 * Add a payload to the current installation in AuboScope.
00181 * </p>
00182 *
00183 * <p>
00184 * This makes the payload selectable by the end user as well as available to
00185 * other AuboCap contributions (through the
00186 * {@link PayloadModel} interface). The payload is not modifiable by the end
00187 * user or other aubo_studio plugins.
00188 * </p>
00189 *
00190 * Use the {@link #addPayload(String, String, Mass, Position,
00191 * InertiaMatrix)} method instead if data for the payload's inertia matrix
00192 * is not available.
00193 *
00194 * <p>
00195 * <b>Note:</b> The specified payload mass must be the total mass of the
00196 * payload attached to the tool output flange of the robot.
00197 * </p>
00198 *
00199 * The valid range for the payload mass and the coordinates of the center of
00200 * gravity (CoG) are available through the
00201 * {@link RobotLimits} interface which can be accessed with {@link
00202 * RobotModel#getRobotLimits()} (an instance of the
00203 * {@link RobotModel} interface can be retrieved through {@link
00204 * SystemAPI#getRobotModel()}).
00205 *
00206 * @param idKey The key to identify this payload by, not <code>null</code>
00207 * nor an empty string. The key is for this AuboCap only, i.e. it only has
00208 * to be unique for this AuboCap and not "globally" for other aubo_studio
00209 * plugins.
00210 * @param suggestedName Suggested name for the payload, not
00211 * <code>null</code> nor an empty string. Valid names must match regex
00212 * [a-zA-Z][a-zA-Z0-9_]{0,14} for a total of 15 characters. The final name
00213 * can be retrieved from the returned payload instance.
00214 * @param mass The total mass of the payload attached to the tool output
00215 * flange of the robot, not <code>null</code>. Create it using {@link
00216 * SimpleValueFactory#createMass(double, Mass.Unit)}.
00217 * @param centerOfGravity The center of gravity (CoG), also referred to as
00218 * center of mass, for the payload. It is defined as the offset between the
00219 * center of the tool output flange and the center of gravity of the
00220 * attached payload. Cannot be <code>null</code>. Create it using the {@link
00221 * PositionFactory} interface.
00222 * @param inertiaMatrix The inertia matrix of the payload, defined in a
00223 * coordinate system with center at the payload's center of gravity and the
00224 * axes aligned with the tool output flange coordinate system (axes). Cannot
00225 * be <code>null</code>. Create it using
00226 * {@link #createInertiaMatrix(double, double, double,
00227 * double, double, double, MomentOfInertia.Unit)}.
00228 * @return The payload created and registered in AuboScope.
00229 * @throws PayloadAlreadyAddedException If a payload has previously been
00230 * added using the same <code>idKey</code> identifier. Use {@link
00231 * #getPayload(String)} to check if the payload has already been added to
00232 * the current installation.
00233 * Use {@link #updatePayload(String, Mass,
00234 * Position)} or
00235 * {@link #updatePayload(String, Mass,
00236 * Position, InertiaMatrix)} to update the payload if the payload parameters
00237 * have changed.
00238 * @throws IllegalPayloadNameException If the suggested name does not match
00239 * required regex.
00240 * @throws IllegalMassException If the specified mass is not inside the
00241 * valid range as defined by AuboScope.
00242 * @throws IllegalCenterOfGravityException If any of the values of the
00243 * specified center of gravity are not inside the valid range as defined by
00244 * AuboScope.
00245 * @throws IllegalInertiaMatrixException If any of the values of the
00246 * specified inertia matrix are not inside the valid range as defined by
00247 * AuboScope.
00248 * \endenglish
00249 */
00250 PayloadPtr addPayload(const std::string &id_key,
00251                      const std::string &suggested_name, double mass,
00252                      const std::vector<double> &center_of_gravity,
00253                      const std::vector<double> &inertiaMatrix);
00254
00255 /**
00256 * @ingroup PayloadContributionModel
00257 * @ref Payload
00258 * \chinese
00259 * AuboCap <code>idKey</code>
00260 *
00261 *
00262 * @param idKey <code>null</code>
00263 * @return AuboCap <code>idKey</code>
00264 * <code>null</code>
00265 * \endchinese

```

```

00266 * \english
00267 * Returns the payload previously added by this AuboCap using the same
00268 * <code>idKey</code> identifier. Use this to verify if the payload is
00269 * present in the current installation.
00270 *
00271 * @param idKey The key to identify this payload by, not <code>null</code>
00272 * nor an empty string.
00273 * @return The payload previously added by this AuboCap.
00274 * Returns <code>null</code> if no payload exists in current
00275 * installation (with the specified <code>idKey</code> identifier).
00276 * \endenglish
00277 */
00278 PayloadPtr getPayload(const std::string &id_key);
00279
00280 /**
00281 * @ingroup PayloadContributionModel
00282 * @ref Payload
00283 * \chinese
00284 * <p>
00285 * AuboCap CoG
00286 * </p>
00287 *
00288 * <b> </b>
00289 *
00290 * { @link #updatePayload(String, Mass, Position, InertiaMatrix)}
00291 *
00292 * @param idKey <code>null</code>
00293 *
00294 * @param mass <code>null</code>
00295 * { @link SimpleValueFactory#createMass(double, Mass.Unit)}
00296 * @param centerOfGravity <code>null</code>
00297 * { @link PositionFactory}
00298 * @throws PayloadNotFoundException <code>idKey</code>
00299 *
00300 * @throws IllegalMassException AuboScope
00301 * @throws IllegalCenterOfGravityException
00302 * \endchinese
00303 * \english
00304 * <p>
00305 * Update the mass and center of gravity (CoG) of an existing payload added
00306 * by this AuboCap.
00307 * </p>
00308 *
00309 * <b>Note:</b> If an inertia matrix has been specified for the payload,
00310 * calling this method will replace (overwrite) it with a computed default
00311 * guess based the new specified payload mass and center of gravity. To avoid
00312 * overwriting an existing inertia matrix (or to update an existing inertia
00313 * matrix), instead use the method
00314 * { @link #updatePayload(String, Mass, Position, InertiaMatrix)}.
00315 *
00316 * @param idKey The key to identify the payload, not <code>null</code> nor
00317 * an empty string. A payload must have been added using the same key prior
00318 * to calling this method.
00319 * @param mass The total mass of the payload attached to the tool output
00320 * flange of the robot, not <code>null</code>. Create it using { @link
00321 * SimpleValueFactory#createMass(double, Mass.Unit)}.
00322 * @param centerOfGravity The center of gravity of the payload, not { @code
00323 * null}. Create it using the
00324 * { @link PositionFactory} interface.
00325 * @throws PayloadNotFoundException If no payload exists with the provided
00326 * <code>idKey</code> identifier in the current installation.
00327 * @throws IllegalMassException If the specified mass is not inside the
00328 * valid range as defined by AuboScope.
00329 * @throws IllegalCenterOfGravityException If any of the values of the
00330 * specified center of gravity are not inside the valid range as defined by
00331 * AuboScope.
00332 * \endenglish
00333 */
00334 void updatePayload(const std::string &id_key, double mass,
00335 const std::vector<double> &center_of_gravity);
00336
00337 /**
00338 * @ingroup PayloadContributionModel
00339 * \chinese
00340 * AuboCap CoG
00341 *
00342 * @param idKey <code>null</code>
00343 *
00344 * @param mass <code>null</code>
00345 * { @link SimpleValueFactory#createMass(double, Mass.Unit)}
00346 * @param centerOfGravity <code>null</code>
00347 * { @link PositionFactory}
00348 * @param inertiaMatrix <code>null</code>
00349 * @throws PayloadNotFoundException <code>idKey</code>
00350 *
00351 * @throws IllegalMassException
00352 * @throws IllegalCenterOfGravityException

```

```

00353 * @throws IllegalInertiaMatrixException
00354 * \endchinese
00355 * \english
00356 * Update the mass, center of gravity (CoG) and inertia matrix of an
00357 * existing payload added by this AuboCap.
00358 *
00359 * @param idKey The key to identify the payload, not <code>null</code> nor
00360 * an empty string. A payload must have been added using the same key prior
00361 * to calling this method.
00362 * @param mass The total mass of the payload attached to the tool output
00363 * flange of the robot, not <code>null</code>. Create it using {@link
00364 * SimpleValueFactory#createMass(double, Mass.Unit)}.
00365 * @param centerOfGravity The center of gravity of the payload, not {@code
00366 * null}. Create it using the
00367 * {@link PositionFactory} interface.
00368 * @param inertiaMatrix The inertia matrix of the payload, not
00369 * <code>null</code>. Create it using
00370 * {@link #createInertiaMatrix(double, double, double,
00371 * double, double, double, MomentOfInertia.Unit)}
00372 * @throws PayloadNotFoundException If no payload exists with the provided
00373 * <code>idKey</code> identifier in the current installation.
00374 * @throws IllegalMassException If the specified mass is not inside the
00375 * valid range as defined by AuboScope.
00376 * @throws IllegalCenterOfGravityException If any of the values of the
00377 * specified center of gravity are not inside the valid range as defined by
00378 * AuboScope.
00379 * @throws IllegalInertiaMatrixException If any of the values of the
00380 * specified inertia matrix are not inside the valid range as defined by
00381 * AuboScope.
00382 * \endenglish
00383 */
00384 void updatePayload(const std::string id_key, double mass,
00385                  const std::vector<double> &center_of_gravity,
00386                  const std::vector<double> &inertia_matrix);
00387
00388 /**
00389 * @ingroup PayloadContributionModel
00390 * \chinese
00391 * AuboCap
00392 *
00393 *
00394 * @param idKey <code>null</code>
00395 * @throws PayloadNotFoundException <code>idKey</code>
00396 *
00397 * \endchinese
00398 * \english
00399 * Remove a payload added by this AuboCap from AuboScope. Program nodes
00400 * using the payload will be become undefined because the payload is no
00401 * longer resolvable.
00402 *
00403 * @param idKey The identifier key used to add the payload with, not {@code
00404 * null} nor an empty string.
00405 * @throws PayloadNotFoundException If no payload exists with the provided
00406 * <code>idKey</code> identifier in the current installation.
00407 * \endenglish
00408 */
00409 void removePayload(const std::string &id_key);
00410
00411 private:
00412 friend class DataSwitch;
00413 PayloadContributionModel();
00414 void *d_{ nullptr };
00415 };
00416 } // namespace aubo_scope
00417 } // namespace arcs
00418
00419 #endif // AUBO_SCOPE_PAYLOAD_CONTRIBUTION_MODEL_H

```

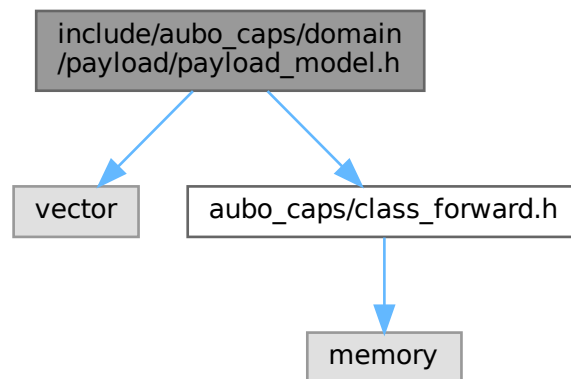
12.122 include/aubo_caps/domain/payload/payload_model.h File Reference

```

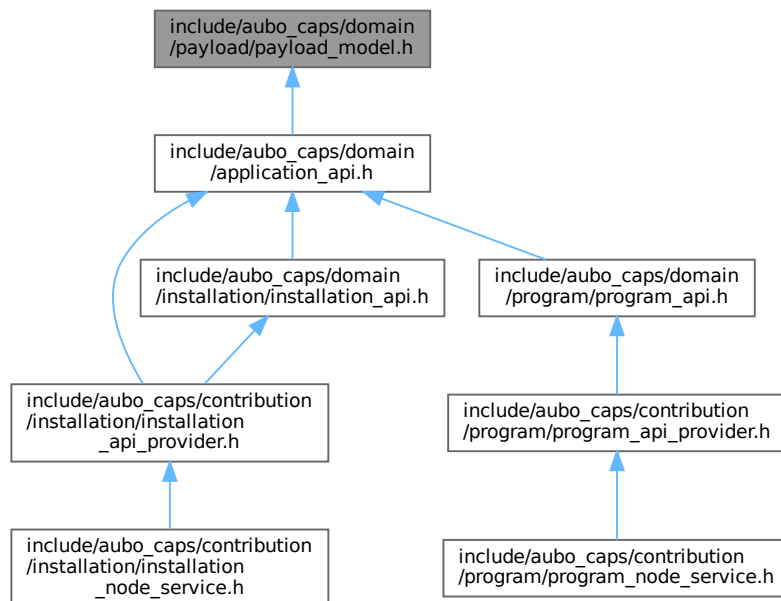
#include <vector>
#include <aubo_caps/class_forward.h>

```

Include dependency graph for payload_model.h:



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



Classes

- class `arcs::aubo_scope::PayloadModel`
API interface for payload parameters.

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([PayloadModel](#))
- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([Payload](#))

12.123 payload_model.h

[Go to the documentation of this file.](#)

```

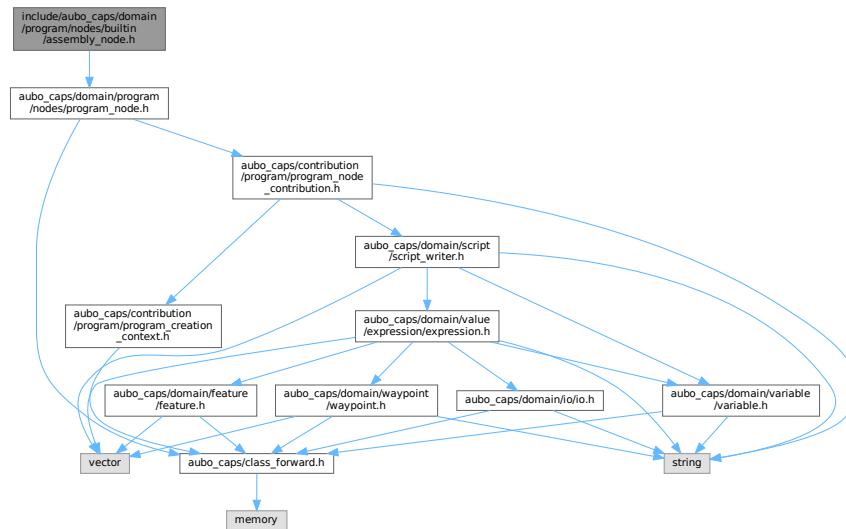
00001 #ifndef AUBO_SCOPE_PAYLOAD_MODEL_H
00002 #define AUBO_SCOPE_PAYLOAD_MODEL_H
00003
00004 #include <vector>
00005 #include <aubo_caps/class_forward.h>
00006
00007 namespace arcs {
00008     namespace aubo_scope {
00009         ARCS_CLASS_FORWARD(PayloadModel);
00010         ARCS_CLASS_FORWARD(Payload);
00011
00012         /**
00013          * @ingroup ApplicationApi
00014          * \~chinese API
00015          * \~english API interface for payload parameters
00016          */
00017
00018         class ARCS_ABI_EXPORT PayloadModel
00019         {
00020         public:
00021             PayloadModel(PayloadModel &f);
00022             PayloadModel(PayloadModel &&f);
00023             virtual ~PayloadModel();
00024
00025             /**
00026              * @ingroup PayloadModel
00027              * @brief getPayloads
00028              * @return A collection of all the payloads from the current
00029              *         installation in AuboScope, including user-defined payloads and
00030              *         payloads added by aubo_studio plugins.
00031              */
00032             std::vector<PayloadPtr> getPayloads() const;
00033
00034             /**
00035              * @ingroup PayloadModel
00036              * @ref Payload
00037              * @brief getPayload
00038              * @param name
00039              * @return
00040              */
00041             PayloadPtr getPayload(const std::string &name) const;
00042
00043             /**
00044              * @ingroup PayloadModel
00045              * @ref Payload
00046              * @brief getDefaultPayload
00047              * @return
00048              */
00049             PayloadPtr getDefaultPayload() const;
00050
00051             /**
00052              * @ingroup PayloadModel
00053              * @ref Payload
00054              * @brief getCurrentPayload
00055              * @return
00056              */
00057             PayloadPtr getCurrentPayload() const;
00058
00059         private:
00060             friend class DataSwitch;
00061             PayloadModel();
00062             void *d_{ nullptr };
00063         };
00064     } // namespace aubo_scope
00065 } // namespace arcs
00066
00067 #endif // PAYLOADMODEL_H

```

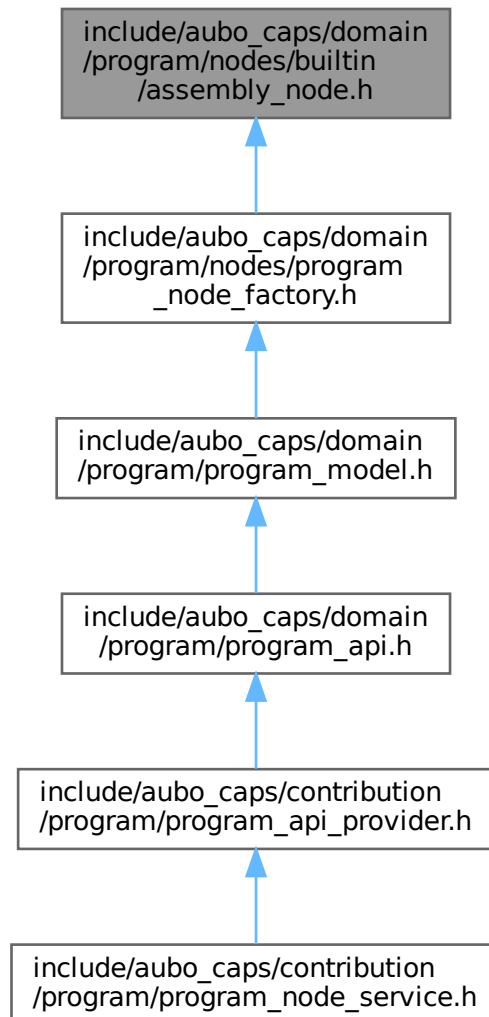

12.124 include/aubo_caps/domain/program/nodes/builtin/assembly_node.h File Reference

#include <aubo_caps/domain/program/nodes/program_node.h>

Include dependency graph for assembly_node.h:



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



Classes

- class [arcs::aubo_scope::AssemblyNode](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(AssemblyNode\)](#)

12.125 assembly_node.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_ASSEMBLY_NODE_H
00002 #define AUBO_SCOPE_ASSEMBLY_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(AssemblyNode);
00009
00010 class ARCS_ABI_EXPORT AssemblyNode : public ProgramNode
00011 {
00012 public:
00013     enum ConfigType
00014     {
00015         NONE,
00016         HOLE_SEARCH,
00017         COMPLIANT_INSERTION
00018     };
00019
00020     AssemblyNode(AssemblyNode &f);
00021     AssemblyNode(AssemblyNode &&f);
00022     virtual ~AssemblyNode();
00023
00024     void addAssemblyUntilNode();
00025
00026 private:
00027     friend class DataSwitch;
00028     AssemblyNode();
00029     void *d_{ nullptr };
00030 };
00031
00032 } // namespace aubo_scope
00033 } // namespace arcs
00034 #endif // AUBO_SCOPE_ASSEMBLY_NODE_H

```

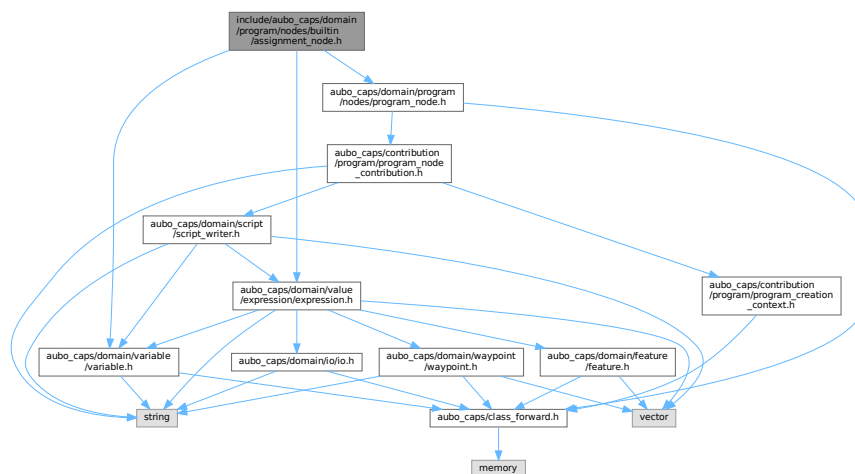
12.126 include/aubo_caps/domain/program/nodes/builtin/assignment_node.h File Reference ↩

```

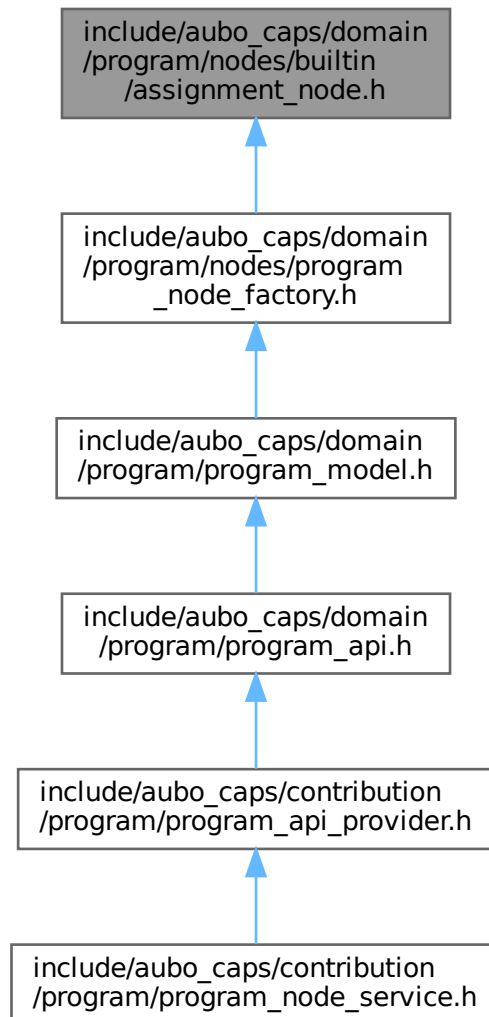
#include <aubo_caps/domain/program/nodes/program_node.h>
#include <aubo_caps/domain/variable/variable.h>
#include <aubo_caps/domain/value/expression/expression.h>

```

Include dependency graph for assignment_node.h:



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



Classes

- class [arcs::aubo_scope::AssignmentNode](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([AssignmentNode](#))

12.127 assignment_node.h

[Go to the documentation of this file.](#)

```

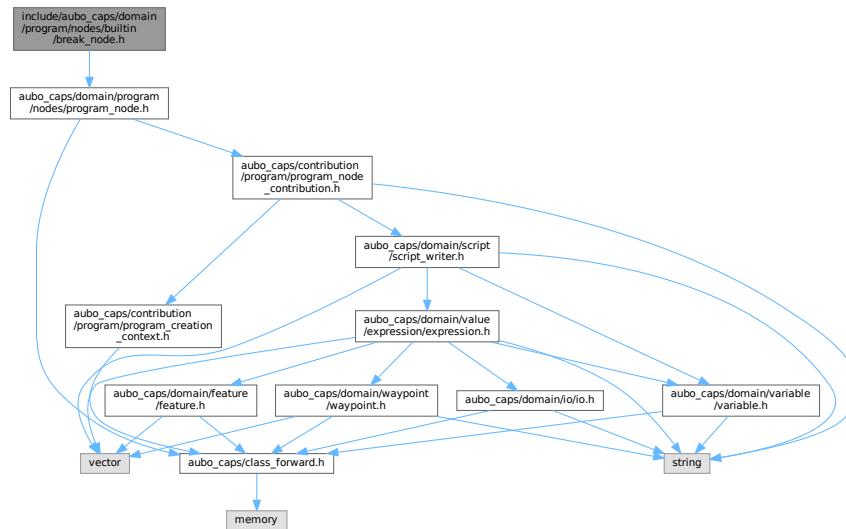
00001 #ifndef AUBO_SCOPE_ASSIGNMENT_NODE_H
00002 #define AUBO_SCOPE_ASSIGNMENT_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005 #include <aubo_caps/domain/variable/variable.h>
00006 #include <aubo_caps/domain/value/expression/expression.h>
00007
00008 namespace arcs {
00009 namespace aubo_scope {
00010 ARCS_CLASS_FORWARD(AssignmentNode);
00011
00012 class ARCS_ABI_EXPORT AssignmentNode : public ProgramNode
00013 {
00014 public:
00015     /**
00016      * The configuration type used to determine which type of configuration this
00017      * instance is.
00018      */
00019     enum ConfigType
00020     {
00021         /**
00022          * <p>
00023          * Expression assignment has been selected.
00024          * </p>
00025          *
00026          * The config instance can be cast to {@link
00027          * ExpressionAssignmentNodeConfig}.
00028          */
00029         EXPRESSION,
00030
00031         /**
00032          * Operator assignment has been selected. This type has no further
00033          * information. Setting this type of config will be ignored and will
00034          * have no effect on the node.
00035          */
00036         OPERATOR
00037     };
00038
00039     enum InputMode
00040     {
00041         INPUT_BOOL,
00042         INPUT_INTEGER,
00043         INPUT_FLOAT,
00044         INPUT_STRING,
00045     };
00046
00047     AssignmentNode(AssignmentNode &f);
00048     AssignmentNode(AssignmentNode &&f);
00049     virtual ~AssignmentNode();
00050
00051     /**
00052      * <p>
00053      *
00054      * </p>
00055      * @return
00056      */
00057     ConfigType getConfigType();
00058     void setConfigType(ConfigType type);
00059
00060     /**
00061      *
00062      * @return the variable assigned to the expression evaluation.
00063      */
00064     VariablePtr getVariable();
00065     void setVariable(VariablePtr var);
00066
00067     /**
00068      *
00069      * @return the expression whose evaluation is assigned to the variable.
00070      */
00071     ExpressionPtr getExpression();
00072     void setExpression(ExpressionPtr exp);
00073
00074 private:
00075     friend class DataSwitch;
00076     AssignmentNode();
00077     void *d_{ nullptr };
00078 };
00079
00080 } // namespace aubo_scope
00081 } // namespace arcs
00082 #endif // AUBO_SCOPE_ASSIGNMENT_NODE_H

```

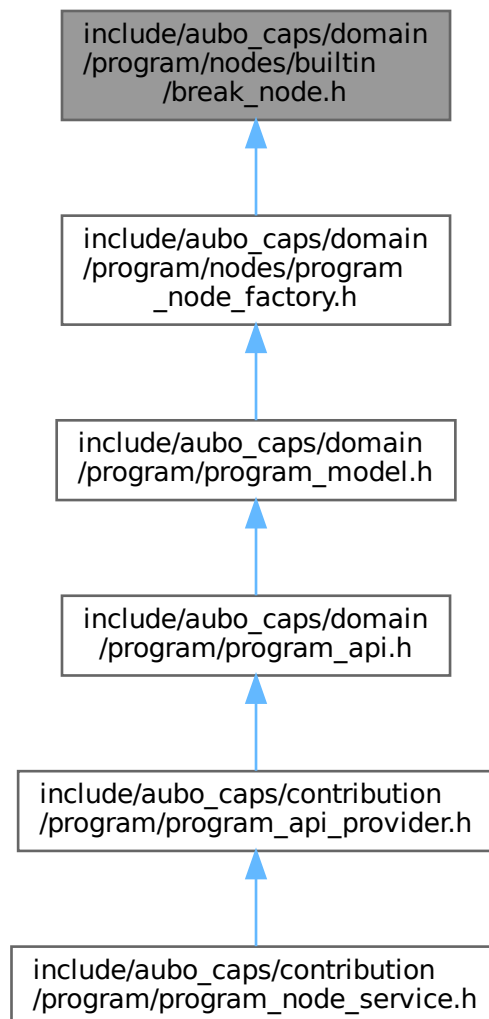
12.128 include/aubo_caps/domain/program/nodes/builtin/break_↵ node.h File Reference

#include <aubo_caps/domain/program/nodes/program_node.h>

Include dependency graph for break_node.h:



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



Classes

- class [arcs::aubo_scope::BreakNode](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([BreakNode](#))

12.129 break_node.h

[Go to the documentation of this file.](#)

```

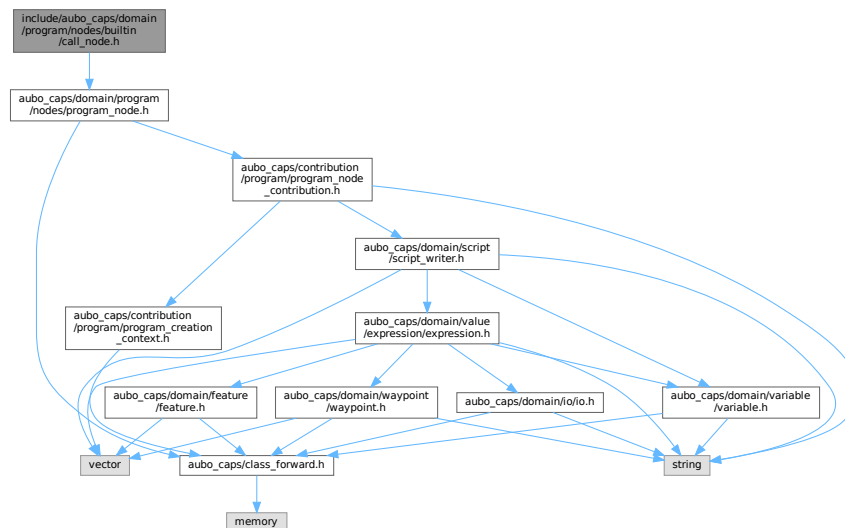
00001 #ifndef AUBO_SCOPE_BREAK_NODE_H
00002 #define AUBO_SCOPE_BREAK_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(BreakNode);
00009
00010 class ARCS_ABI_EXPORT BreakNode : public ProgramNode
00011 {
00012 public:
00013     BreakNode(BreakNode &f);
00014     BreakNode(BreakNode &&f);
00015     virtual ~BreakNode();
00016
00017 private:
00018     friend class DataSwitch;
00019     BreakNode();
00020     void *d_{ nullptr };
00021 };
00022
00023 } // namespace aubo_scope
00024 } // namespace arcs
00025 #endif // AUBO_SCOPE_BREAK_NODE_H

```

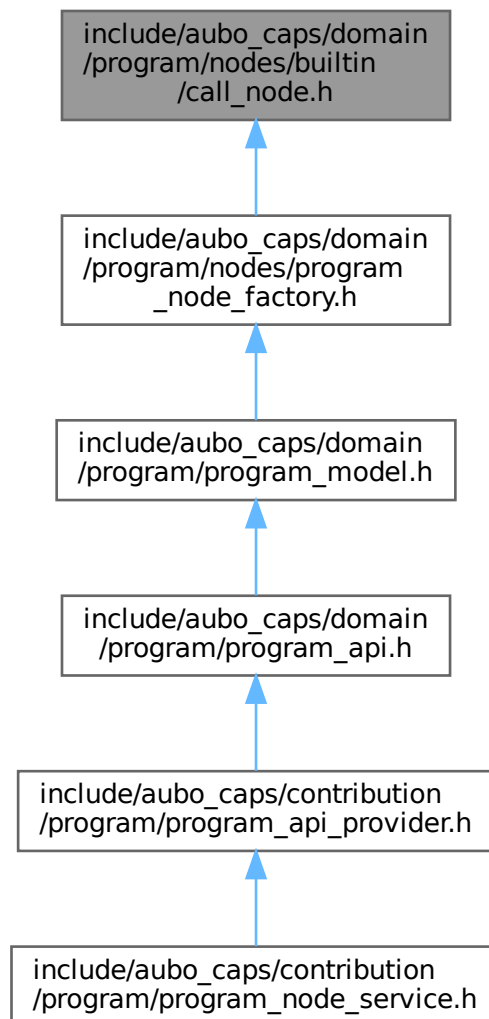
12.130 include/aubo_caps/domain/program/nodes/builtin/call_node.h

File Reference

#include <aubo_caps/domain/program/nodes/program_node.h>
 Include dependency graph for call_node.h:



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



Classes

- class [arcs::aubo_scope::CallNode](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([CallNode](#))

12.131 call_node.h

[Go to the documentation of this file.](#)

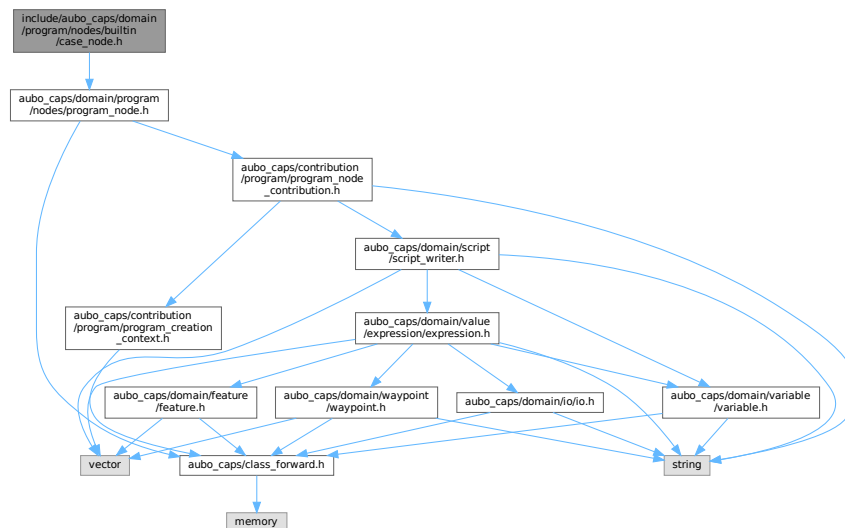
```

00001 #ifndef AUBO_SCOPE_CALL_NODE_H
00002 #define AUBO_SCOPE_CALL_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(CallNode);
00009
00010 class ARCS_ABI_EXPORT CallNode : public ProgramNode
00011 {
00012 public:
00013     CallNode(CallNode &f);
00014     CallNode(CallNode &&f);
00015     virtual ~CallNode();
00016
00017 private:
00018     friend class DataSwitch;
00019     CallNode();
00020     void *d_{ nullptr };
00021 };
00022
00023 } // namespace aubo_scope
00024 } // namespace arcs
00025 #endif // AUBO_SCOPE_CALL_NODE_H

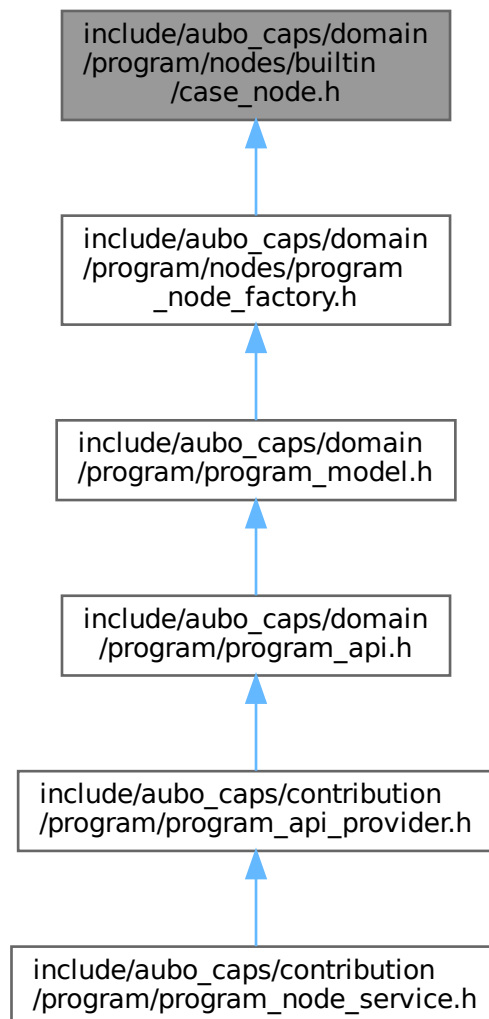
```

12.132 include/aubo_caps/domain/program/nodes/builtin/case_node.h File Reference

#include <aubo_caps/domain/program/nodes/program_node.h>
 Include dependency graph for case_node.h:



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



Classes

- class [arcs::aubo_scope::CaseNode](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([CaseNode](#))

12.133 case_node.h

[Go to the documentation of this file.](#)

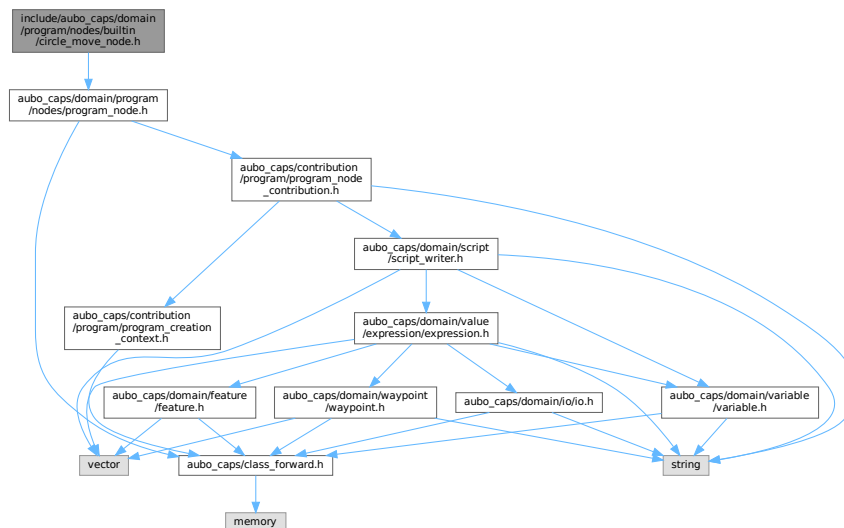
```

00001 #ifndef AUBO_SCOPE_CASE_NODE_H
00002 #define AUBO_SCOPE_CASE_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(CaseNode);
00009
00010 class ARCS_ABI_EXPORT CaseNode : public ProgramNode
00011 {
00012 public:
00013     CaseNode(CaseNode &f);
00014     CaseNode(CaseNode &&f);
00015     virtual ~CaseNode();
00016
00017 private:
00018     friend class DataSwitch;
00019     CaseNode();
00020     void *d_{ nullptr };
00021 };
00022
00023 } // namespace aubo_scope
00024 } // namespace arcs
00025 #endif // AUBO_SCOPE_CASE_NODE_H

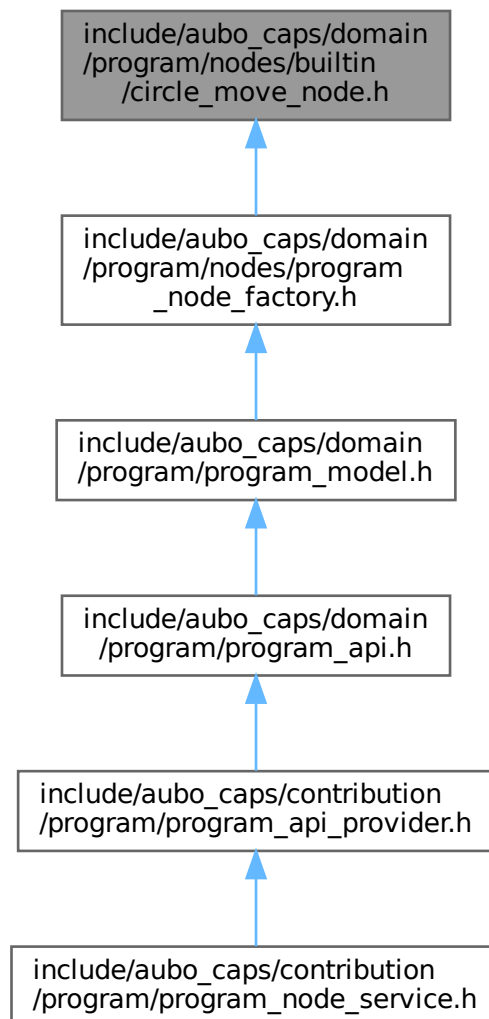
```

12.134 include/aubo_caps/domain/program/nodes/builtin/circle_↵ move_node.h File Reference

#include <aubo_caps/domain/program/nodes/program_node.h>
Include dependency graph for circle_move_node.h:



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



Classes

- class [arcs::aubo_scope::CircleMoveNode](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([CircleMoveNode](#))

12.135 circle_move_node.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_CIRCLE_MOVE_NODE_H
00002 #define AUBO_SCOPE_CIRCLE_MOVE_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(CircleMoveNode);
00009
00010 class ARCS_ABI_EXPORT CircleMoveNode : public ProgramNode
00011 {
00012 public:
00013     CircleMoveNode(CircleMoveNode &f);
00014     CircleMoveNode(CircleMoveNode &&f);
00015     virtual ~CircleMoveNode();
00016
00017 private:
00018     friend class DataSwitch;
00019     CircleMoveNode();
00020     void *d_{ nullptr };
00021 };
00022
00023 } // namespace aubo_scope
00024 } // namespace arcs
00025 #endif // AUBO_SCOPE_CIRCLE_MOVE_NODE_H

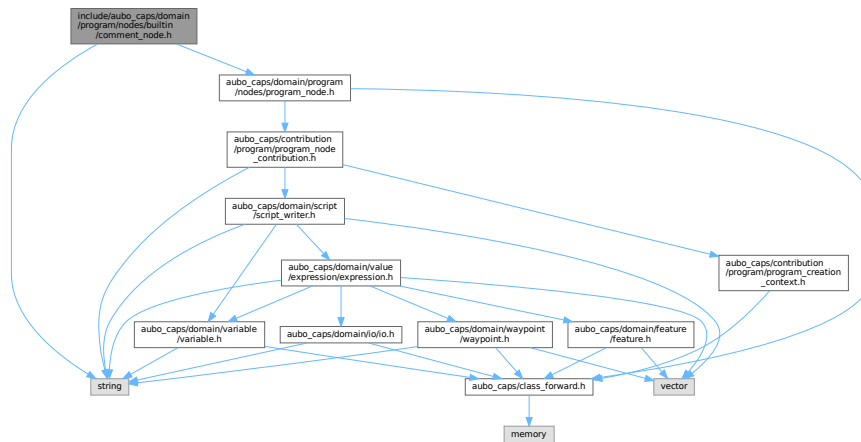
```

12.136 include/aubo_caps/domain/program/nodes/builtin/comment_node.h File Reference

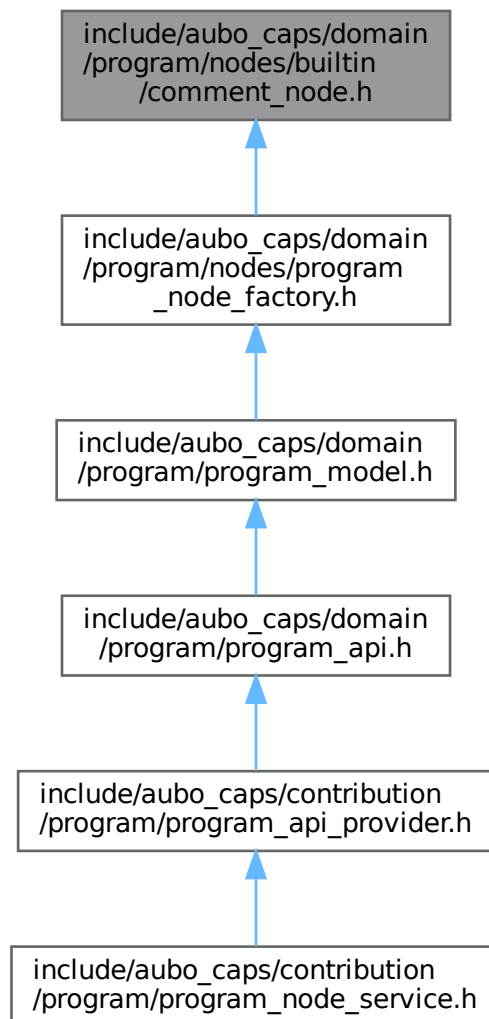
```
#include <string>
```

```
#include <aubo_caps/domain/program/nodes/program_node.h>
```

Include dependency graph for comment_node.h:



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



Classes

- class [arcs::aubo_scope::CommentNode](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([CommentNode](#))

12.137 comment_node.h

[Go to the documentation of this file.](#)

```

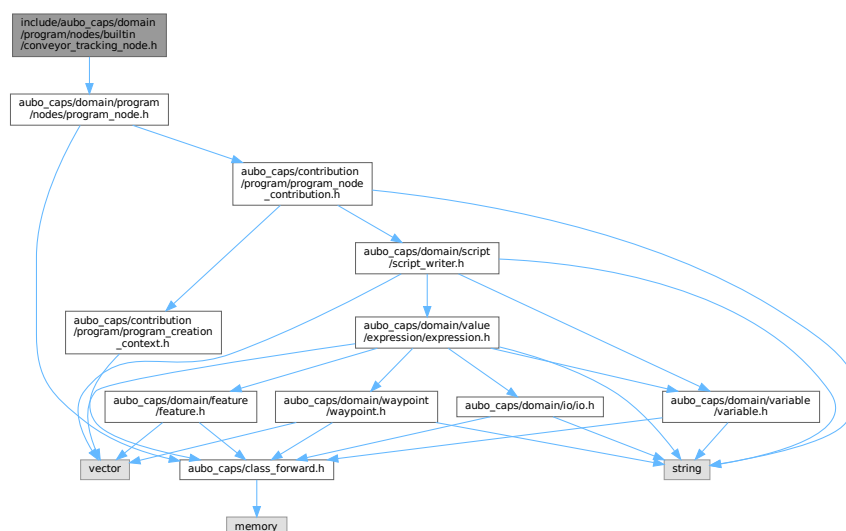
00001 #ifndef AUBO_SCOPE_COMMENT_NODE_H
00002 #define AUBO_SCOPE_COMMENT_NODE_H
00003
00004 #include <string>
00005 #include <aubo_caps/domain/program/nodes/program_node.h>
00006
00007 namespace arcs {
00008 namespace aubo_scope {
00009 ARCS_CLASS_FORWARD(CommentNode);
00010
00011 class ARCS_ABI_EXPORT CommentNode : public ProgramNode
00012 {
00013 public:
00014     CommentNode(CommentNode &f);
00015     CommentNode(CommentNode &&f);
00016     virtual ~CommentNode();
00017
00018     /**
00019     *
00020     * @return The text of the Comment node.
00021     */
00022     std::string getComment() const;
00023
00024     /**
00025     * * Set the text for the Comment node.
00026     * @param comment The Comment for the node.
00027     * @return This node.
00028     */
00029     void setComment(const std::string &param_string);
00030
00031 private:
00032     friend class DataSwitch;
00033     CommentNode();
00034     void *d_{ nullptr };
00035 };
00036
00037 } // namespace aubo_scope
00038 } // namespace arcs
00039 #endif // AUBO_SCOPE_COMMENT_NODE_H

```

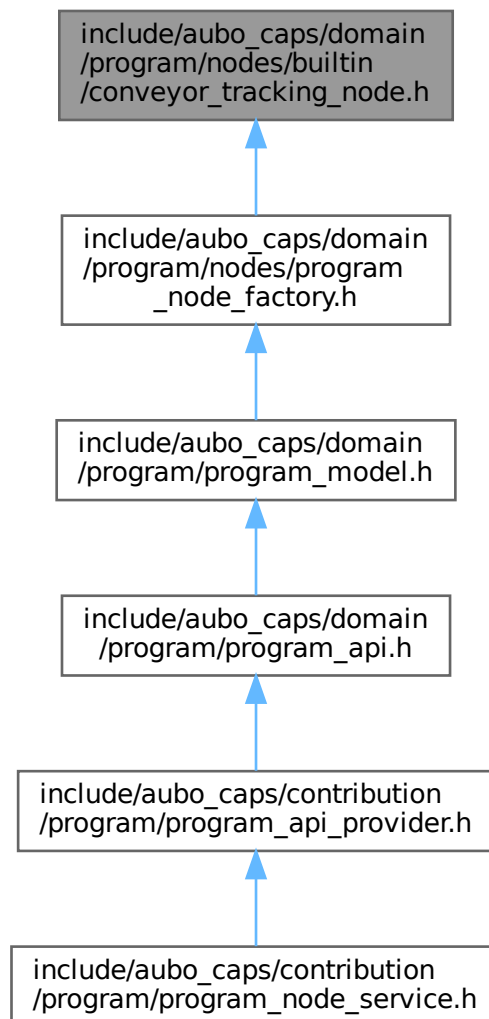
12.138 include/aubo_caps/domain/program/nodes/builtin/conveyor_tracking_node.h File Reference

#include <aubo_caps/domain/program/nodes/program_node.h>

Include dependency graph for conveyor_tracking_node.h:



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



Classes

- class [arcs::aubo_scope::ConveyorTrackingNode](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([ConveyorTrackingNode](#))

12.139 conveyor_tracking_node.h

[Go to the documentation of this file.](#)

```

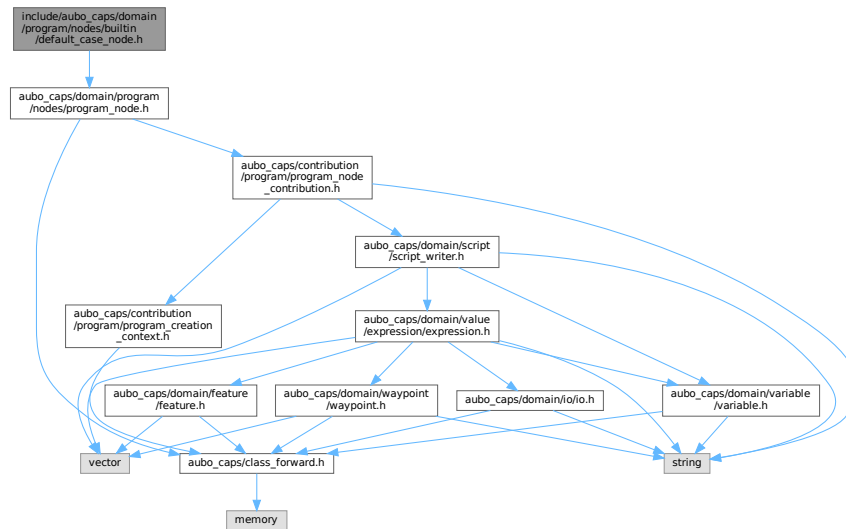
00001 #ifndef AUBO_SCOPE_CONVEYOR_TRACKING_NODE_H
00002 #define AUBO_SCOPE_CONVEYOR_TRACKING_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(ConveyorTrackingNode);
00009
00010 class ARCS_ABI_EXPORT ConveyorTrackingNode : public ProgramNode
00011 {
00012 public:
00013     enum ConveyorTrackingTargetType
00014     {
00015         NEW_WORKPIECE,
00016         PREVIOUS_WORKPIECE,
00017     };
00018
00019     enum ConveyorStartActionQueueType
00020     {
00021         CLEAR_QUEUE,
00022         RESERVE_QUEUE,
00023         ASK
00024     };
00025
00026     enum ConveyorStartActionType
00027     {
00028         CONTINUE_TRACKING,
00029         ABANDON_TRACKING
00030     };
00031
00032     ConveyorTrackingNode(ConveyorTrackingNode &f);
00033     ConveyorTrackingNode(ConveyorTrackingNode &&f);
00034     virtual ~ConveyorTrackingNode();
00035
00036 private:
00037     friend class DataSwitch;
00038     ConveyorTrackingNode();
00039     void *d_{ nullptr };
00040 };
00041
00042 } // namespace aubo_scope
00043 } // namespace arcs
00044 #endif // AUBO_SCOPE_CONVEYOR_TRACKING_NODE_H

```

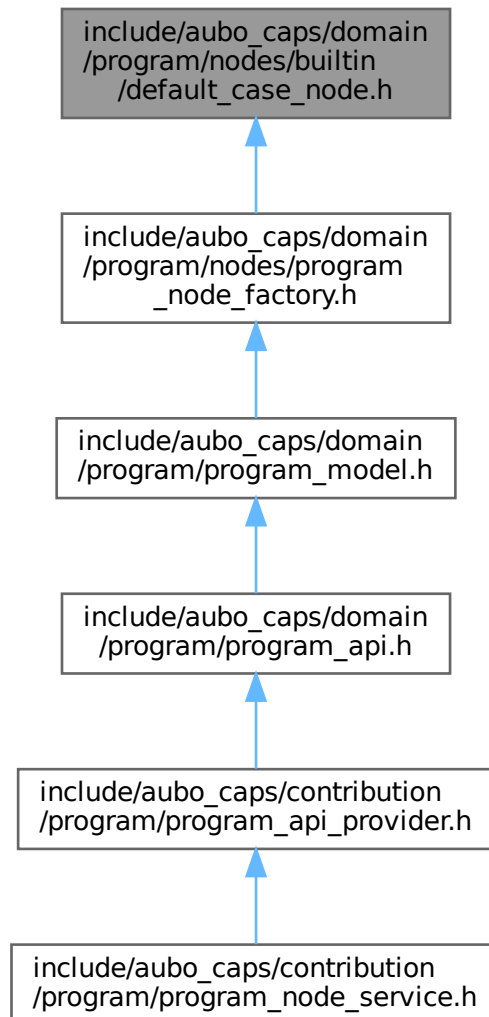
12.140 include/aubo_caps/domain/program/nodes/builtin/default_case_node.h File Reference

#include <aubo_caps/domain/program/nodes/program_node.h>

Include dependency graph for default_case_node.h:



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



Classes

- class [arcs::aubo_scope::DefaultCaseNode](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([DefaultCaseNode](#))

12.141 default_case_node.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_DEFAULT_CASE_NODE_H
00002 #define AUBO_SCOPE_DEFAULT_CASE_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(DefaultCaseNode);
00009
00010 class ARCS_ABI_EXPORT DefaultCaseNode : public ProgramNode
00011 {
00012 public:
00013     DefaultCaseNode(DefaultCaseNode &f);
00014     DefaultCaseNode(DefaultCaseNode &&f);
00015     virtual ~DefaultCaseNode();
00016
00017 private:
00018     friend class DataSwitch;
00019     DefaultCaseNode();
00020     void *d_{ nullptr };
00021 };
00022
00023 } // namespace aubo_scope
00024 } // namespace arcs
00025 #endif // AUBO_SCOPE_DEFAULT_CASE_NODE_H

```

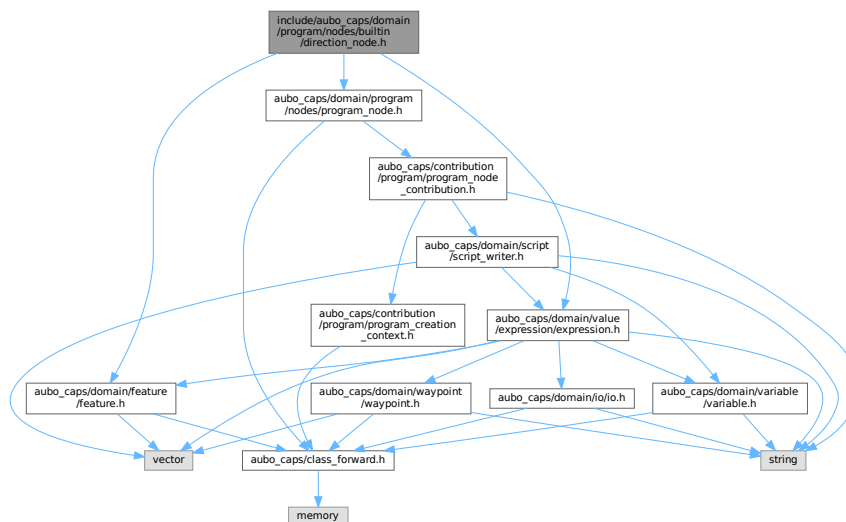
12.142 include/aubo_caps/domain/program/nodes/builtin/direction_node.h File Reference

```

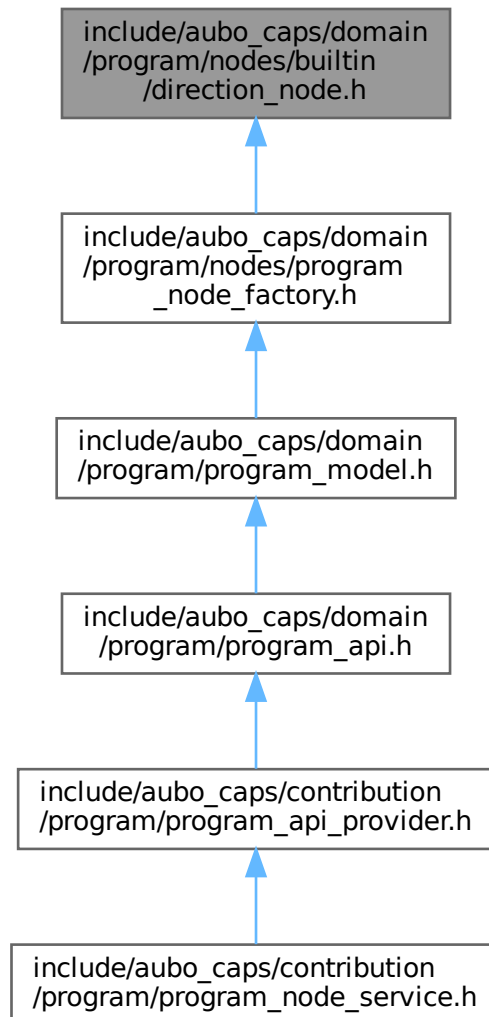
#include <aubo_caps/domain/program/nodes/program_node.h>
#include <aubo_caps/domain/value/expression/expression.h>
#include <aubo_caps/domain/feature/feature.h>

```

Include dependency graph for direction_node.h:



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



Classes

- class [arcs::aubo_scope::DirectionNode](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([DirectionNode](#))

12.143 direction_node.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_DIRECTION_NODE_H
00002 #define AUBO_SCOPE_DIRECTION_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005 #include <aubo_caps/domain/value/expression/expression.h>
00006 #include <aubo_caps/domain/feature/feature.h>
00007
00008 namespace arcs {
00009 namespace aubo_scope {
00010 ARCS_CLASS_FORWARD(DirectionNode);
00011
00012 class ARCS_ABI_EXPORT DirectionNode : public ProgramNode
00013 {
00014 public:
00015     enum ParameterType : int
00016     {
00017         /**
00018          * The tool motion parameters are inherited from the MoveL or
00019          * MoveP
00020          * parent.
00021          */
00022         SHARED_MOTION,
00023
00024         /**
00025          * The tool motion parameters are selected. <br>
00026          *
00027          * The Selection instance can be cast to {@link
00028          * ToolMotionParameters}.
00029          */
00030         TOOL_MOTION
00031     };
00032
00033     enum Axis : int
00034     {
00035         /**
00036          * Move in the positive X direction relative to the feature.
00037          */
00038         X_PLUS,
00039
00040         /**
00041          * Move in the negative X direction relative to the feature.
00042          */
00043         X_MINUS,
00044
00045         /**
00046          * Move in the positive Y direction relative to the feature.
00047          */
00048         Y_PLUS,
00049
00050         /**
00051          * Move in the negative Y direction relative to the feature.
00052          */
00053         Y_MINUS,
00054
00055         /**
00056          * Move in the positive Z direction relative to the feature.
00057          */
00058         Z_PLUS,
00059
00060         /**
00061          * Move in the negative Z direction relative to the feature.
00062          */
00063         Z_MINUS
00064     };
00065
00066     enum DirectionSelectionType : int
00067     {
00068         /**
00069          * A direction vector is specified using an Expression.
00070          * The Selection instance can be cast to {@link
00071          * DirectionVectorExpression}
00072          */
00073         EXPRESSION,
00074
00075         /**
00076          * Direction is specified using an Axis.
00077          * The Selection instance can be cast to {@link DirectionAxis}
00078          */
00079         AXIS
00080     };
00081
00082     enum FeatureSelectionType : int

```

```

00083 {
00084     /**
00085      * The selected feature is no longer available (e.g. because the user
00086      * has removed the feature in the Installation or the needed
00087      * Installation is not currently loaded)
00088      */
00089     UNRESOLVED_FEATURE,
00090
00091     /**
00092      * Feature is inherited from the MoveL or MoveP parent.
00093      */
00094     SHARED,
00095
00096     /**
00097      * Feature is selected. <br>
00098      * The Selection instance can be cast to {@link DirectionNodeFeature}
00099      */
00100     FEATURE
00101 };
00102
00103 DirectionNode(DirectionNode &f);
00104 DirectionNode(DirectionNode &&f);
00105 virtual ~DirectionNode();
00106
00107 ParameterType getType();
00108 void setType(ParameterType type);
00109
00110 /**
00111  * @return the Axis used to define the direction.
00112  */
00113 Axis getAxis();
00114 void setAxis(Axis axis);
00115
00116 /**
00117  *
00118  * @return the feature selected for this Direction node.
00119  */
00120 FeaturePtr getFeature();
00121 void setFeature(FeaturePtr feature);
00122
00123 DirectionSelectionType getDirectionSelectionType();
00124 void setDirectionSelectionType(DirectionSelectionType selection);
00125 /**
00126  *
00127  * @return the expression used to define the direction vector.
00128  */
00129 ExpressionPtr getExpression();
00130 void setExpression(ExpressionPtr expression);
00131
00132 FeatureSelectionType getFeatureSelectionType();
00133 void setFeatureSelectionType(FeatureSelectionType selection);
00134 /**
00135  * @return the tool speed to be achieved.
00136  */
00137 double getToolSpeed();
00138 void setToolSpeed(double speed);
00139
00140 /**
00141  * @return the tool acceleration to use for the movement
00142  */
00143 double getToolAcceleration();
00144 void setToolAcceleration(double acc);
00145
00146 private:
00147     friend class DataSwitch;
00148     DirectionNode();
00149     void *d_{ nullptr };
00150 };
00151
00152 } // namespace aubo_scope
00153 } // namespace arcs
00154 #endif // AUBO_SCOPE_DIRECTION_NODE_H

```

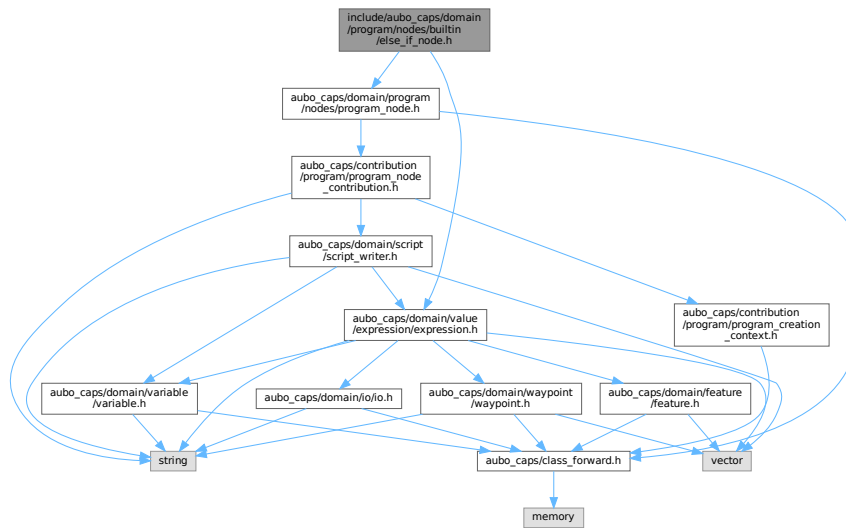
12.144 include/aubo_caps/domain/program/nodes/builtin/else_if_↵ node.h File Reference

```

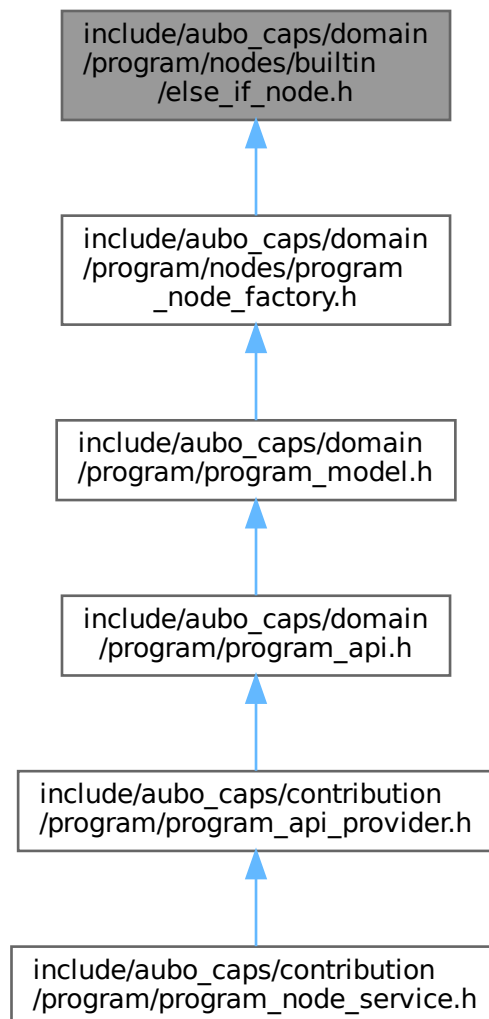
#include <aubo_caps/domain/program/nodes/program_node.h>
#include <aubo_caps/domain/value/expression/expression.h>

```


Include dependency graph for else_if_node.h:



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



Classes

- class [arcs::aubo_scope::ElseIfNode](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD\(ElseIfNode\)](#)

12.145 else_if_node.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_ELSE_IF_NODE_H
00002 #define AUBO_SCOPE_ELSE_IF_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005 #include <aubo_caps/domain/value/expression/expression.h>
00006
00007 namespace arcs {
00008     namespace aubo_scope {
00009         ARCS_CLASS_FORWARD(ElseIfNode);
00010
00011         class ARCS_ABI_EXPORT ElseIfNode : public ProgramNode
00012         {
00013         public:
00014             ElseIfNode(ElseIfNode &f);
00015             ElseIfNode(ElseIfNode &&f);
00016             virtual ~ElseIfNode();
00017
00018             /**
00019              * @param expression The expression that will be evaluated in this
00020              * ElseIfNode.
00021              * @return The same ElseIfNode.
00022              */
00023             void setExpression(ExpressionPtr expression);
00024
00025             /**
00026              * @return the Expression to be evaluated in this ElseIf node.
00027              */
00028             ExpressionPtr getExpression();
00029
00030         private:
00031             friend class DataSwitch;
00032             ElseIfNode();
00033             void *d_{ nullptr };
00034         };
00035
00036     } // namespace aubo_scope
00037 } // namespace arcs
00038 #endif // AUBO_SCOPE_ELSE_IF_NODE_H

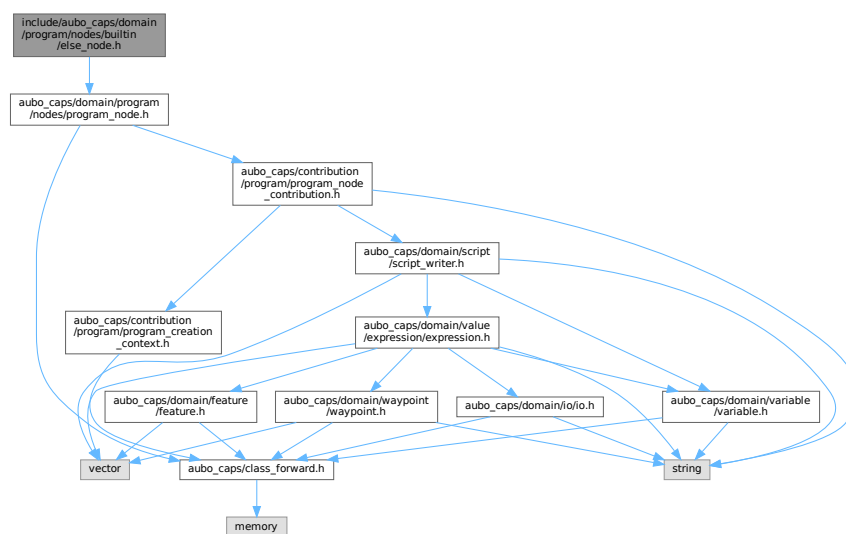
```

12.146 include/aubo_caps/domain/program/nodes/builtin/else_node.h

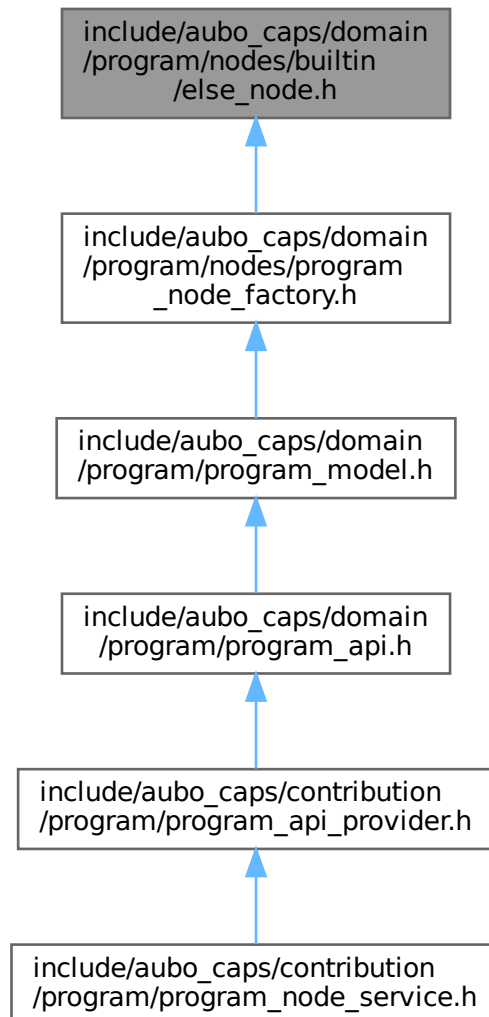
File Reference

#include <aubo_caps/domain/program/nodes/program_node.h>

Include dependency graph for else_node.h:



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



Classes

- class [arcs::aubo_scope::ElseNode](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD\(ElseNode\)](#)

12.147 else_node.h

[Go to the documentation of this file.](#)

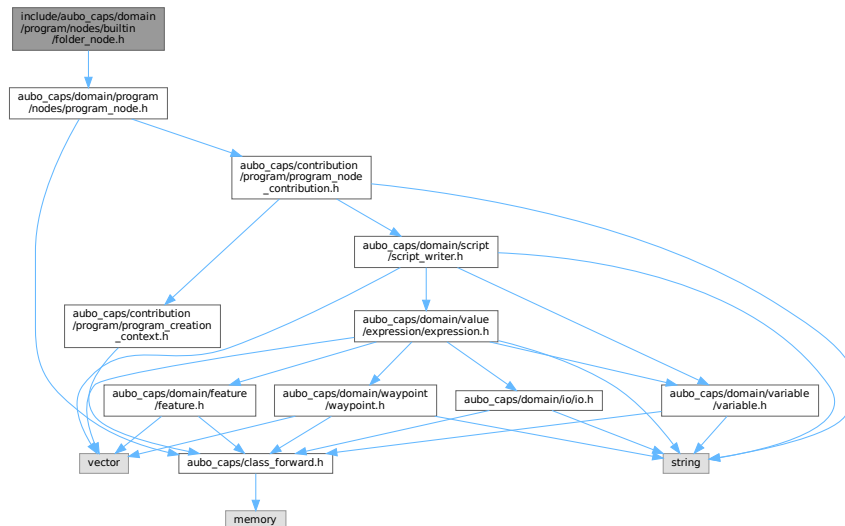
```

00001 #ifndef AUBO_SCOPE_ELSE_NODE_H
00002 #define AUBO_SCOPE_ELSE_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(ElseNode);
00009
00010 class ARCS_ABI_EXPORT ElseNode : public ProgramNode
00011 {
00012 public:
00013     ElseNode(ElseNode &f);
00014     ElseNode(ElseNode &&f);
00015     virtual ~ElseNode();
00016
00017 private:
00018     friend class DataSwitch;
00019     ElseNode();
00020     void *d_{ nullptr };
00021 };
00022
00023 } // namespace aubo_scope
00024 } // namespace arcs
00025 #endif // AUBO_SCOPE_ELSE_NODE_H

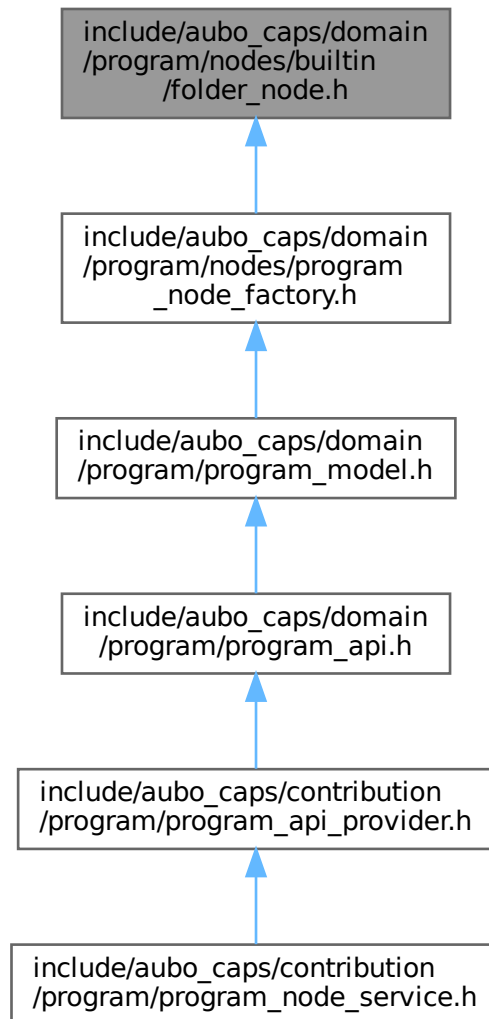
```

12.148 include/aubo_caps/domain/program/nodes/builtin/folder_node.h File Reference

#include <aubo_caps/domain/program/nodes/program_node.h>
 Include dependency graph for folder_node.h:



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



Classes

- class [arcs::aubo_scope::FolderNode](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) (FolderNode)

12.149 folder__node.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_FOLDER_NODE_H
00002 #define AUBO_SCOPE_FOLDER_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(FolderNode);
00009
00010 class ARCS_ABI_EXPORT FolderNode : public ProgramNode
00011 {
00012 public:
00013     FolderNode(FolderNode &f);
00014     FolderNode(FolderNode &&f);
00015     virtual ~FolderNode();
00016
00017     /**
00018      * * Set the name for the Folder node.
00019      * @param set The name for the node.
00020      * @return This node.
00021      */
00022     void setFolderName(const std::string &folder_name);
00023
00024 private:
00025     friend class DataSwitch;
00026     FolderNode();
00027     void *d_{ nullptr };
00028 };
00029
00030 } // namespace aubo_scope
00031 } // namespace arcs
00032 #endif // AUBO_SCOPE_FOLDER_NODE_H

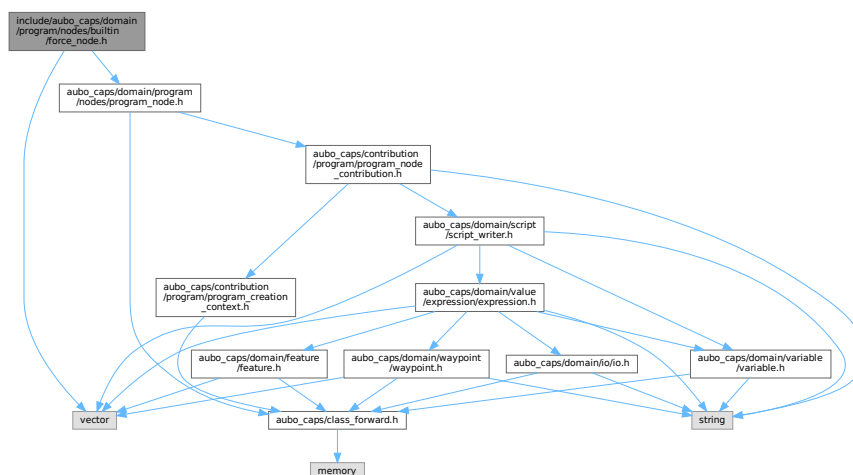
```

12.150 include/aubo_caps/domain/program/nodes/builtin/force__node.h File Reference

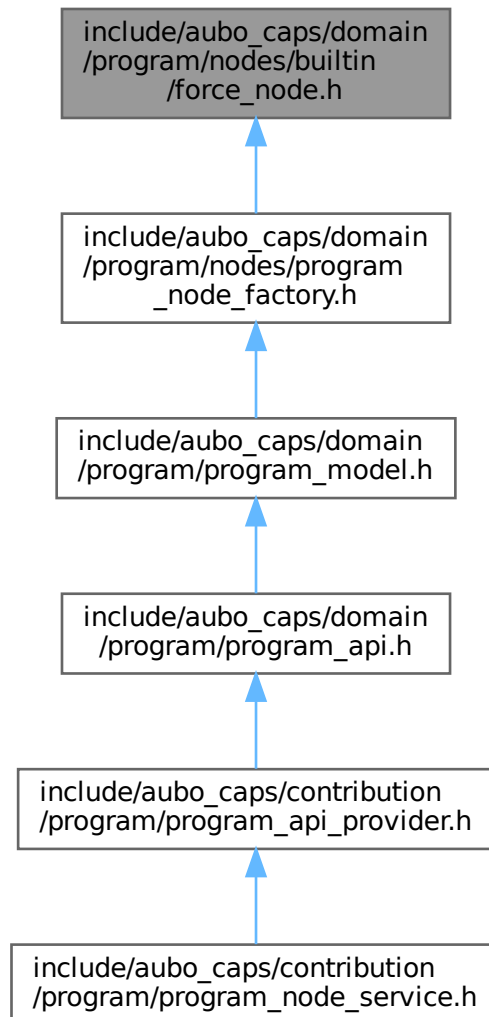
```
#include <vector>
```

```
#include <aubo_caps/domain/program/nodes/program_node.h>
```

Include dependency graph for force__node.h:



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



Classes

- class [arcs::aubo_scope::ForceNode](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(ForceNode\)](#)

12.151 force_node.h

[Go to the documentation of this file.](#)

```

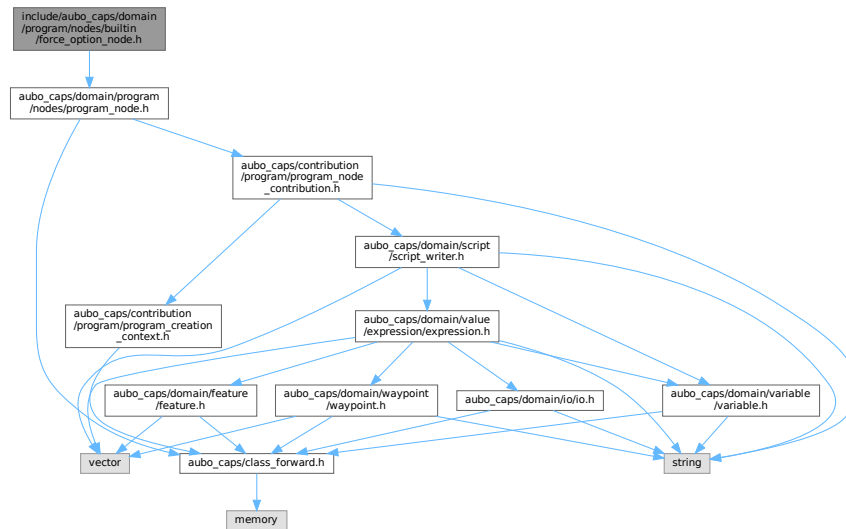
00001 #ifndef AUBO_SCOPE_FORCE_NODE_H
00002 #define AUBO_SCOPE_FORCE_NODE_H
00003
00004 #include <vector>
00005 #include <aubo_caps/domain/program/nodes/program_node.h>
00006
00007 namespace arcs {
00008 namespace aubo_scope {
00009 ARCS_CLASS_FORWARD(ForceNode);
00010
00011 class ARCS_ABI_EXPORT ForceNode : public ProgramNode
00012 {
00013 public:
00014 /**
00015  * The configuration type used to determine which type of configuration this
00016  * instance is.
00017  */
00018 enum ConfigType
00019 {
00020
00021 /**
00022  * <p>
00023  * Simple type is selected. This type has no further information.
00024  * </p>
00025  *
00026  * Setting this type of config will be ignored and will have no effect
00027  * on the node.
00028  */
00029 SIMPLE,
00030
00031 /**
00032  * <p>
00033  * Frame type is selected.
00034  * </p>
00035  *
00036  * The config instance can be cast to {@link FrameForceNodeConfig}.
00037  */
00038 FRAME,
00039
00040 /**
00041  * <p>
00042  * Point type is selected. This type has no further information.
00043  * </p>
00044  *
00045  * Setting this type of config will be ignored and will have no effect
00046  * on the node.
00047  */
00048 POINT,
00049
00050 /**
00051  * <p>
00052  * Motion type is selected. This type has no further information.
00053  * </p>
00054  *
00055  * Setting this type of config will be ignored and will have no effect
00056  * on the node.
00057  */
00058 MOTION
00059 };
00060
00061 ForceNode(ForceNode &f);
00062 ForceNode(ForceNode &&f);
00063 virtual ~ForceNode();
00064
00065 private:
00066 friend class DataSwitch;
00067 ForceNode();
00068 void *d_{ nullptr };
00069 };
00070
00071 } // namespace aubo_scope
00072 } // namespace arcs
00073 #endif // AUBO_SCOPE_FORCE_NODE_H

```

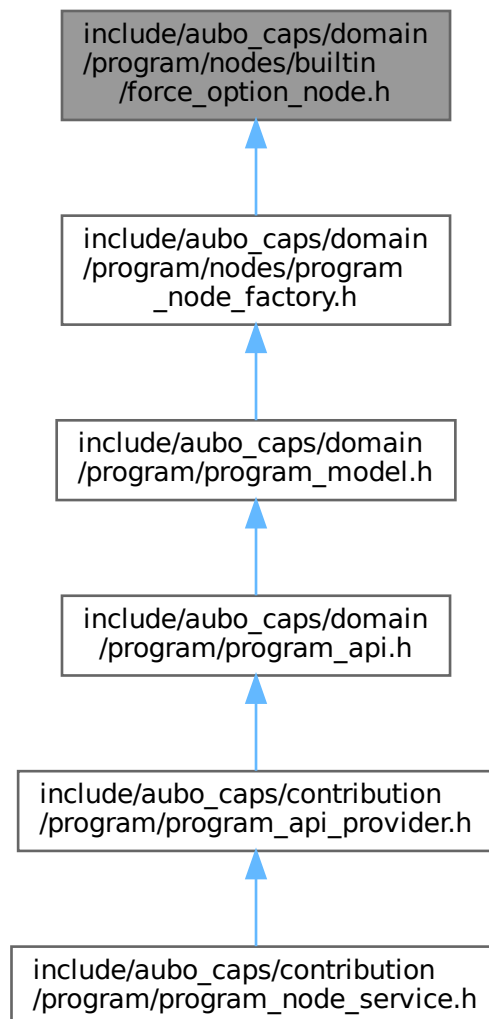
12.152 include/aubo_caps/domain/program/nodes/builtin/force_option_node.h File Reference

#include <aubo_caps/domain/program/nodes/program_node.h>

Include dependency graph for force_option_node.h:



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



Classes

- class [arcs::aubo_scope::ForceOptionNode](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) (ForceOptionNode)

12.153 force_option_node.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_FORCE_OPTION_NODE_H
00002 #define AUBO_SCOPE_FORCE_OPTION_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(ForceOptionNode);
00009
00010 class ARCS_ABI_EXPORT ForceOptionNode : public ProgramNode
00011 {
00012 public:
00013     ForceOptionNode(ForceOptionNode &f);
00014     ForceOptionNode(ForceOptionNode &&f);
00015     virtual ~ForceOptionNode();
00016
00017 private:
00018     friend class DataSwitch;
00019     ForceOptionNode();
00020     void *d_{ nullptr };
00021 };
00022
00023 } // namespace aubo_scope
00024 } // namespace arcs
00025 #endif // AUBO_SCOPE_FORCE_OPTION__NODE_H

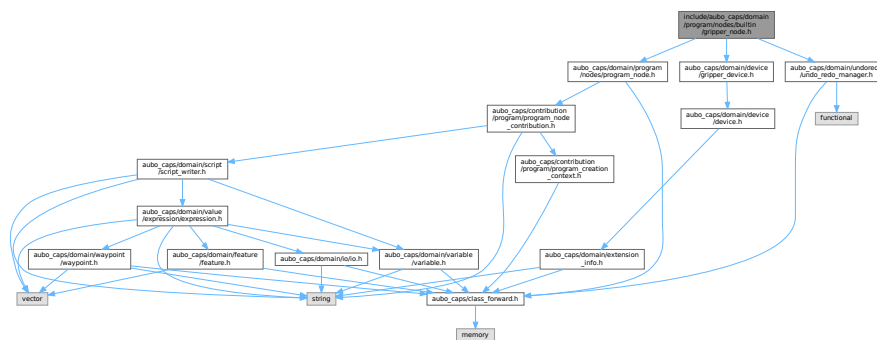
```

12.154 include/aubo_caps/domain/program/nodes/builtin/gripper_node.h File Reference

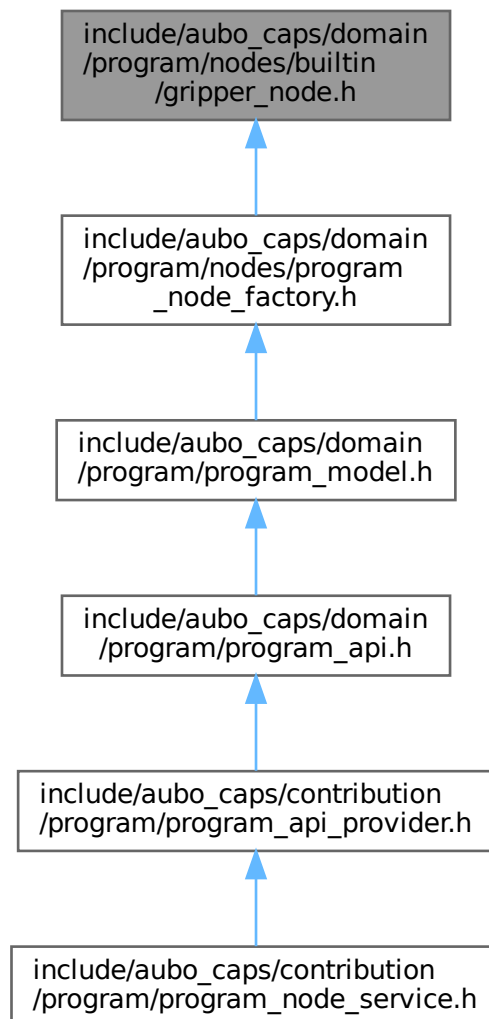
```

#include <aubo_caps/domain/device/gripper_device.h>
#include <aubo_caps/domain/program/nodes/program_node.h>
#include <aubo_caps/domain/undoredo/undo_redo_manager.h>
Include dependency graph for gripper_node.h:

```



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



Classes

- class [arcs::aubo_scope::GripperNode](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(GripperNode\)](#)

12.155 gripper_node.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_GRIPPER_NODE_H
00002 #define AUBO_SCOPE_GRIPPER_NODE_H
00003
00004 #include <aubo_caps/domain/device/gripper_device.h>
00005 #include <aubo_caps/domain/program/nodes/program_node.h>
00006 #include <aubo_caps/domain/undoredo/undo_redo_manager.h>
00007
00008 namespace arcs {
00009 namespace aubo_scope {
00010 ARCS_CLASS_FORWARD(GripperNode);
00011
00012 /**
00013  * \chinese
00014  * <p>
00015  * <p> GripperNodeFactory </p>
00016  * \endchinese
00017  * \english
00018  * <p>This interface represents a program node which can be used for programming
00019  * grip and release actions with a selected gripper device.</p>
00020  * <p>An instance of this interface can be built using the {@link
00021  * GripperNodeFactory} interface and is returned as a result when {@link
00022  * GripperNodeFactory#createGripperNode(GripperDevice)} is called.</p>
00023  * \endenglish
00024  */
00025 class ARCS_ABI_EXPORT GripperNode : public ProgramNode
00026 {
00027 public:
00028 /**
00029  * The configuration type used to determine which type of configuration this
00030  * instance is.
00031  */
00032 enum ConfigType
00033 {
00034
00035 /**
00036  * <p>
00037  * A Gripper action (i.e. grip or release) has not been selected
00038  * </p>
00039  *
00040  * The config instance can be cast to {@link UndefinedActionConfig}.
00041  * This is only relevant if the Gripper node uses a multi-gripper
00042  * device, i.e. a gripper device supporting the multi-gripper capability
00043  * (see the
00044  * {@link MultiGripperSupport} interface). The instance will provide
00045  * access to the individual gripper selection.
00046  */
00047 UNDEFINED,
00048
00049 /**
00050  * <p>
00051  * The gripper is configured for a grip action
00052  * </p>
00053  *
00054  * The config instance can be cast to {@link GripActionConfig}.
00055  */
00056 GRIP_ACTION,
00057
00058 /**
00059  * <p>
00060  * The gripper is configured for a release action
00061  * </p>
00062  *
00063  * The config instance can be cast to {@link ReleaseActionConfig}.
00064  */
00065 RELEASE_ACTION
00066 };
00067
00068 GripperNode(GripperNode &f);
00069 GripperNode(GripperNode &&f);
00070 virtual ~GripperNode();
00071
00072 /**
00073  * \chinese
00074  * @return getter
00075  * \endchinese
00076  * \english
00077  * This method returns the type of configuration. Cast this instance
00078  * appropriately to have access to specific getters.
00079  * @return The type of this config.
00080  * \endenglish
00081  */
00082

```

```

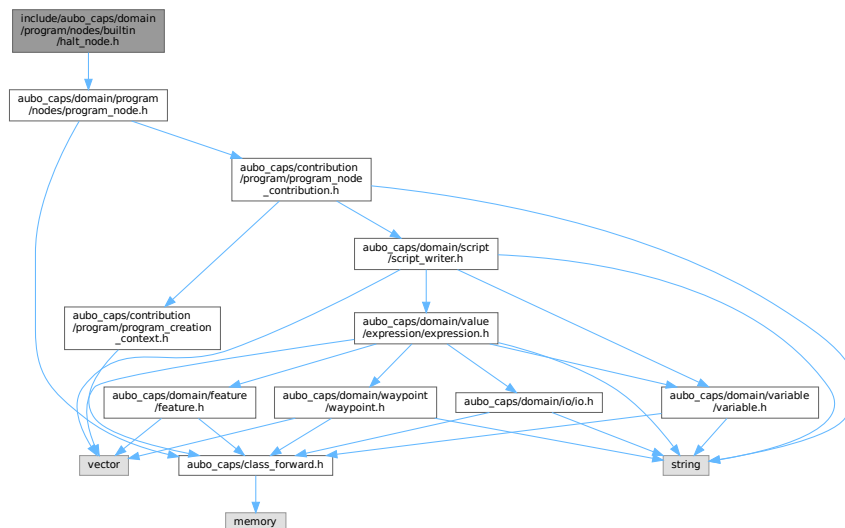
00083 ConfigType getConfigType();
00084
00085 void setConfigType(ConfigType type);
00086
00087 /**
00088  * \chinese
00089  * @return
00090  * \endchinese
00091  * \english
00092  * @return The gripper device used by this Gripper node
00093  * \endenglish
00094  */
00095 GripperDevicePtr getGripperDevice();
00096
00097 /**
00098  * The setting type used to determine which type of setting this instance
00099  * is.
00100  */
00101 enum SettingType
00102 {
00103
00104     /**
00105      * <p>
00106      * No selection has been made.
00107      * </p>
00108      */
00109     UNDEFINED0,
00110
00111     /**
00112      * <p>
00113      * A payload (from the installation) is selected.
00114      * </p>
00115      *
00116      * The setting instance can be cast to {@link PayloadSelection}.
00117      */
00118     PAYLOAD_SELECTION,
00119
00120     /**
00121      * <p>
00122      * Custom payload parameters are specified.
00123      * </p>
00124      *
00125      * The setting instance can be cast to {@link CustomPayloadParameters}.
00126      */
00127     CUSTOM_PARAMETERS
00128 };
00129
00130 /**
00131  * \chinese
00132  * @return      getter
00133  * @return
00134  * \endchinese
00135  * \english
00136  * This method returns the type of setting. Cast this instance appropriately
00137  * to have access to specific getters.
00138  * @return The type of this payload setting.
00139  * \endenglish
00140  */
00141 SettingType getSettingType();
00142
00143 private:
00144     friend class DataSwitch;
00145     GripperNode();
00146     void *d_{ nullptr };
00147 };
00148 } // namespace aubo_scope
00149 } // namespace arcs
00150
00151 #endif

```

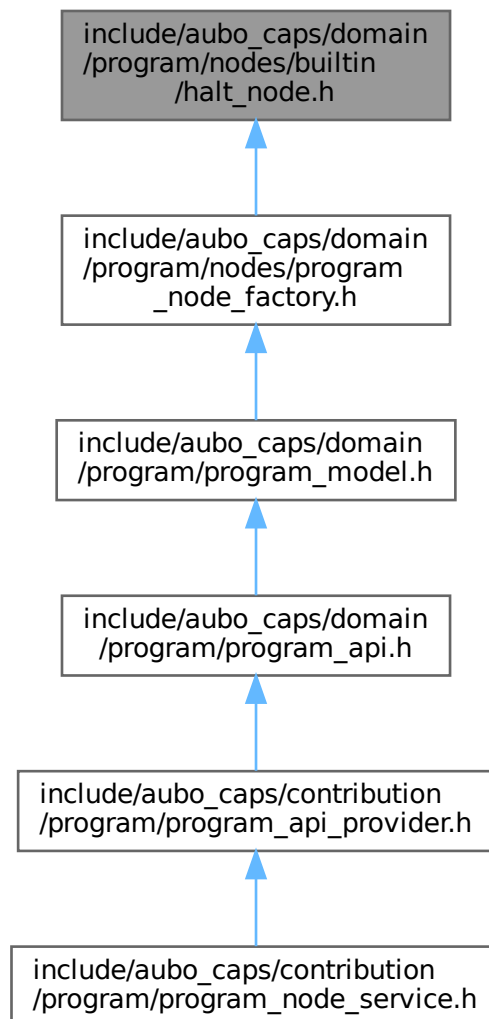
12.156 include/aubo_caps/domain/program/nodes/builtin/halt_node.h File Reference

```
#include <aubo_caps/domain/program/nodes/program_node.h>
```

Include dependency graph for halt_node.h:



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



Classes

- class [arcs::aubo_scope::HaltNode](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([HaltNode](#))

12.157 halt_node.h

[Go to the documentation of this file.](#)

```

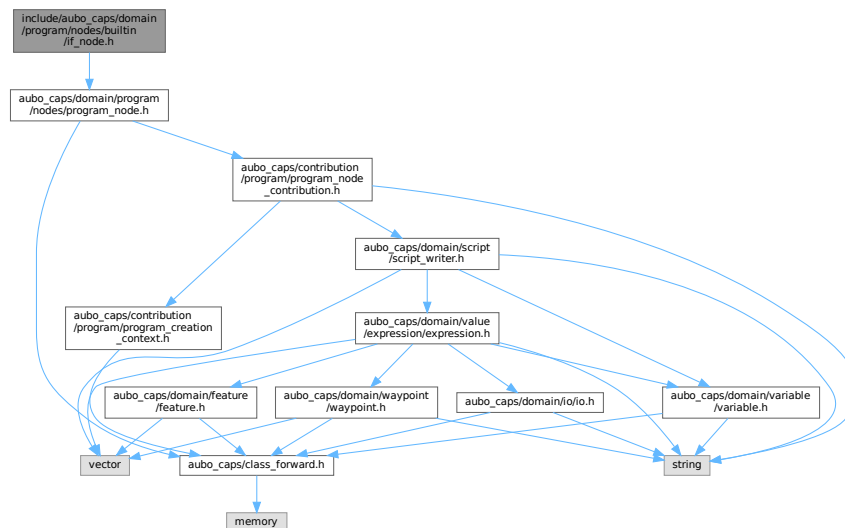
00001 #ifndef AUBO_SCOPE_HALT_NODE_H
00002 #define AUBO_SCOPE_HALT_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(HaltNode);
00009
00010 class ARCS_ABI_EXPORT HaltNode : public ProgramNode
00011 {
00012 public:
00013     HaltNode(HaltNode &f);
00014     HaltNode(HaltNode &&f);
00015     virtual ~HaltNode();
00016
00017 private:
00018     friend class DataSwitch;
00019     HaltNode();
00020     void *d_{ nullptr };
00021 };
00022
00023 } // namespace aubo_scope
00024 } // namespace arcs
00025 #endif // AUBO_SCOPE_HALT_NODE_H

```

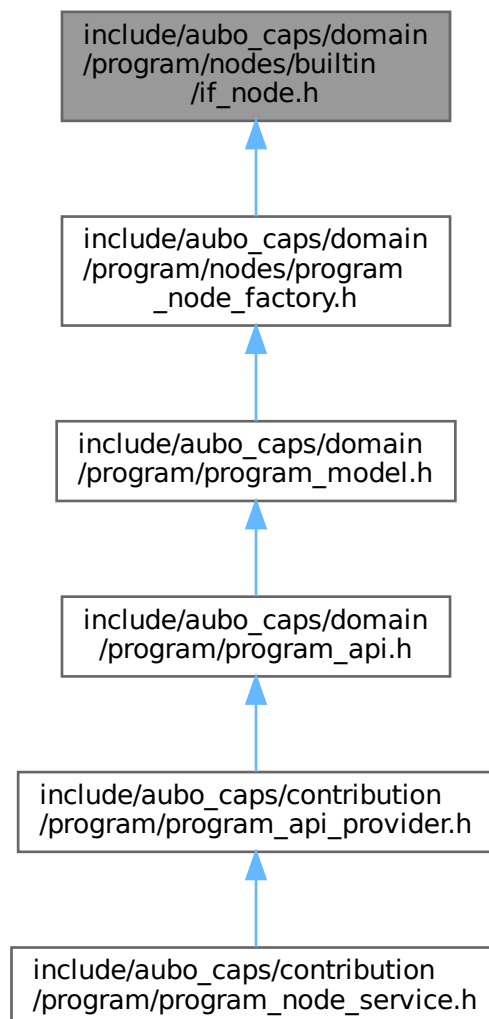
12.158 include/aubo_caps/domain/program/nodes/builtin/if_node.h

File Reference

#include <aubo_caps/domain/program/nodes/program_node.h>
 Include dependency graph for if_node.h:



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



Classes

- class [arcs::aubo_scope::IfNode](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([Expression](#))
- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([IfNode](#))

12.159 if_node.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_IF_NODE_H
00002 #define AUBO_SCOPE_IF_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(Expression);
00009 ARCS_CLASS_FORWARD(IfNode);
00010
00011 class ARCS_ABI_EXPORT IfNode : public ProgramNode
00012 {
00013 public:
00014     IfNode(IfNode &f);
00015     IfNode(IfNode &&f);
00016     virtual ~IfNode();
00017
00018     /**
00019      * \chinese
00020      *
00021      * @param expression If
00022      * @throws IllegalStateException UndoableChanges
00023      * \endchinese
00024      * \english
00025      * Sets the expression used by the node.
00026      * @param expression The expression that will be evaluated in this If Node.
00027      * @throws IllegalStateException if called from a Qt-based AuboCap
00028      * outside of an {@link UndoableChanges} scope.
00029      * \endenglish
00030      */
00031     void setExpression(ExpressionPtr expression);
00032
00033     /**
00034      * \chinese
00035      * @return If
00036      * \endchinese
00037      * \english
00038      * @return the Expression to be evaluated in this If node.
00039      * \endenglish
00040      */
00041     ExpressionPtr getExpression();
00042
00043 private:
00044     friend class DataSwitch;
00045     IfNode();
00046     void *d_{ nullptr };
00047 };
00048
00049 } // namespace aubo_scope
00050 } // namespace arcs
00051 #endif // AUBO_SCOPE_IF_NODE_H

```

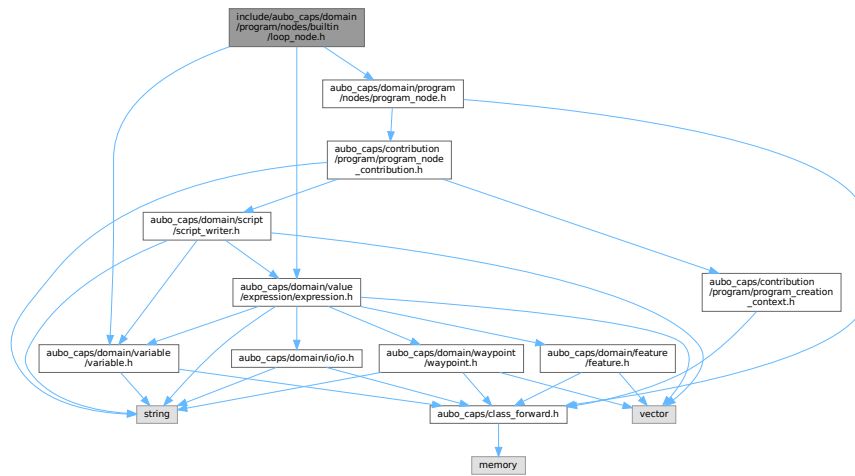
12.160 include/aubo_caps/domain/program/nodes/builtin/loop_node.h File Reference

```

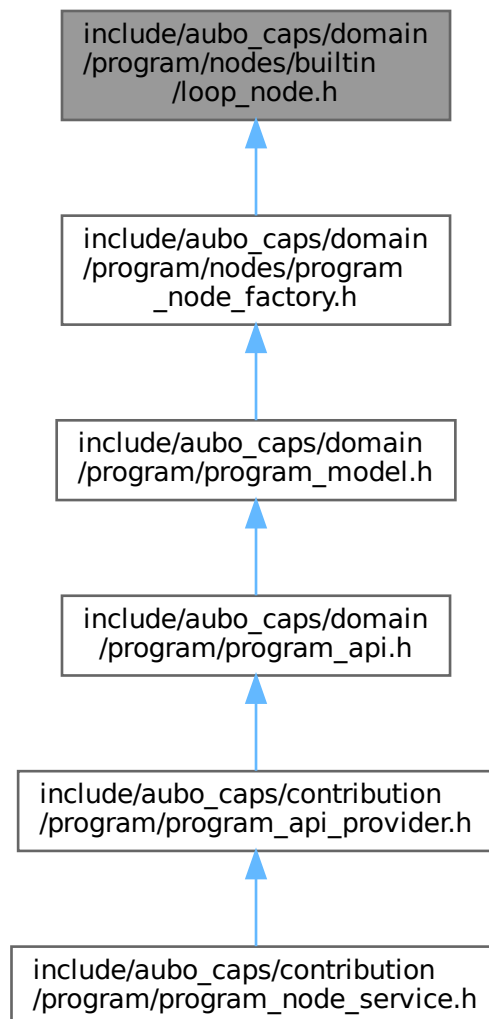
#include <aubo_caps/domain/program/nodes/program_node.h>
#include <aubo_caps/domain/value/expression/expression.h>
#include <aubo_caps/domain/variable/variable.h>

```

Include dependency graph for loop_node.h:



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



Classes

- class [arcs::aubo_scope::LoopNode](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([LoopNode](#))

12.161 loop_node.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_LOOP_NODE_H
00002 #define AUBO_SCOPE_LOOP_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005 #include <aubo_caps/domain/value/expression/expression.h>
00006 #include <aubo_caps/domain/variable/variable.h>
00007
00008 namespace arcs {
00009 namespace aubo_scope {
00010 ARCS_CLASS_FORWARD(LoopNode);
00011
00012 class ARCS_ABI_EXPORT LoopNode : public ProgramNode
00013 {
00014 public:
00015     /**
00016      * The configuration type used to determine which type of configuration this
00017      * instance is.
00018      */
00019     enum ConfigType
00020     {
00021
00022         /**
00023          * <p>
00024          * Looping infinitely has been selected.
00025          * </p>
00026          *
00027          * The config instance can be cast to {@link AlwaysLoopNodeConfig}
00028          */
00029         ALWAYS,
00030
00031         /**
00032          * <p>
00033          * Looping a certain number of times has been selected.
00034          * </p>
00035          *
00036          * The config instance can be cast to {@link CounterLoopNodeConfig}
00037          */
00038         COUNTER,
00039
00040         /**
00041          * <p>
00042          * Looping as long as the evaluation of an expression is true has been
00043          * selected.
00044          * </p>
00045          *
00046          * The config instance can be cast to {@link ExpressionLoopNodeConfig}
00047          */
00048         EXPRESSION
00049     };
00050
00051     LoopNode(LoopNode &f);
00052     LoopNode(LoopNode &&f);
00053     virtual ~LoopNode();
00054
00055     /**
00056      * This method returns the type of configuration. Cast this instance
00057      * appropriately to have access to specific getters.
00058      *
00059      * @return the type of this config.
00060      */
00061     ConfigType getConfigType();
00062     void setConfigType(ConfigType type);
00063
00064     /**
00065      * Creates a loop counter configuration.
00066      *
00067      * @param loopCount the number of iterations through the loop node.
00068      * @param errorHandler error handler for handling validation. If using
00069      * {@link ErrorHandler#AUTO_CORRECT} this will clamp the value to the
00070      * nearest valid loop count value.
00071      * @return the configuration.
00072      */
00073     void setLoopCount(int loop_count);
00074     int getLoopCount();
00075
00076     /**
00077      * Create a loop expression configuration.
00078      *
00079      * @param expression the expression whose evaluation must be true to
00080      * continue looping, not <code>null</code>.
00081      * @param evaluateContinuously if <code>true</code> the expression will be
00082      * evaluated continuously during the looping (i.e. the loop can be

```

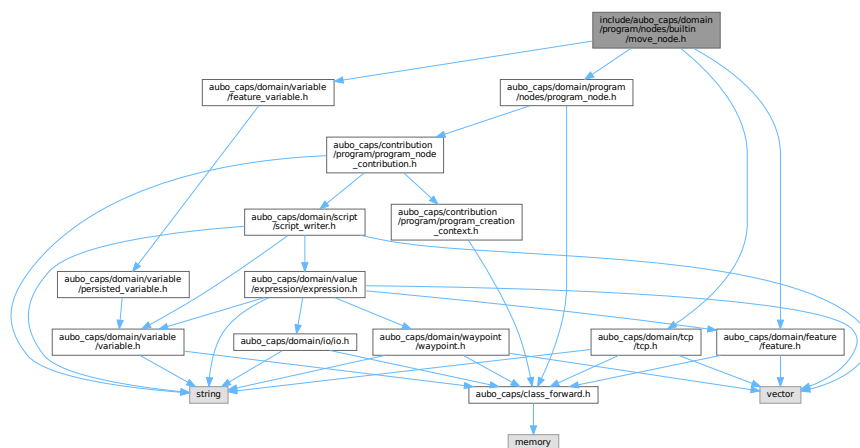
```

00083     * interrupted anytime during its execution), otherwise <code>false</code>
00084     * and the expression will be evaluated once per loop iteration.
00085     * @return the configuration.
00086     */
00087 void setExpression(ExpressionPtr expression);
00088 ExpressionPtr getExpression();
00089 /**
00090  *
00091  * @return <code>true</code> if the expression will be evaluated
00092  * continuously during the looping (i.e. the loop can be interrupted anytime
00093  * during its execution), otherwise <code>false</code> and the expression
00094  * will be evaluated once per loop iteration.
00095  */
00096 bool isEvaluateContinously();
00097 void setEvaluateContinously(bool cont);
00098 /**
00099  *
00100  * @return the variable for storing the loop count
00101  */
00102 void setLoopCountVariable(VariablePtr var);
00103 VariablePtr getLoopCountVariable();
00104 private:
00105 friend class DataSwitch;
00106 LoopNode();
00107 void *d_{ nullptr };
00108 };
00109 // namespace aubo_scope
00110 // namespace arcs
00111 #endif // AUBO_SCOPE_LOOP_NODE_H

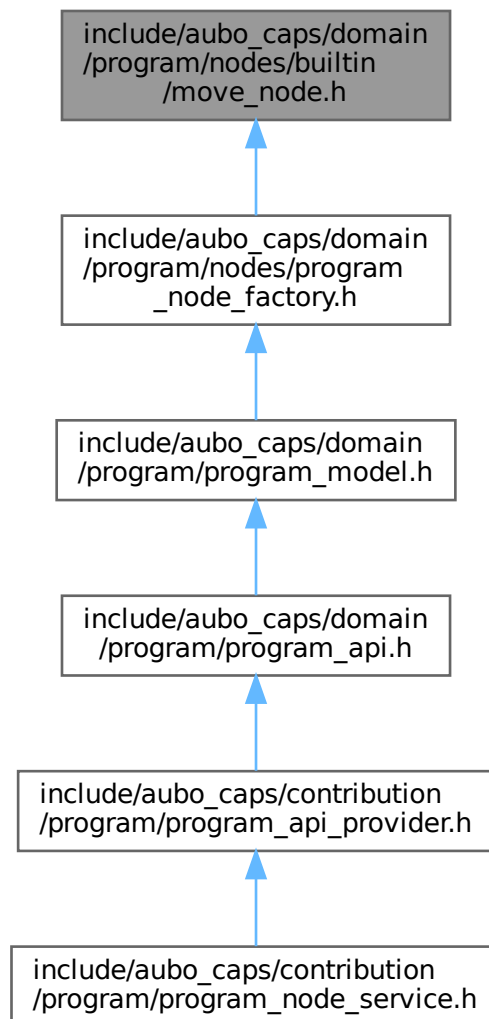
```

12.162 include/auco_caps/domain/program/nodes/builtin/move_node.h File Reference

#include <auco_caps/domain/program/nodes/program_node.h>
#include <auco_caps/domain/feature/feature.h>
#include <auco_caps/domain/tcp/tcp.h>
#include <auco_caps/domain/variable/feature_variable.h>
Include dependency graph for move_node.h:



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



Classes

- class [arcs::aubo_scope::MoveNode](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([MoveNode](#))

12.163 move__node.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_MOVE_NODE_H
00002 #define AUBO_SCOPE_MOVE_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005 #include <aubo_caps/domain/feature/feature.h>
00006 #include <aubo_caps/domain/tcp/tcp.h>
00007 #include <aubo_caps/domain/variable/feature_variable.h>
00008
00009 namespace arcs {
00010 namespace aubo_scope {
00011 ARCS_CLASS_FORWARD(MoveNode);
00012
00013 class ARCS_ABI_EXPORT MoveNode : public ProgramNode
00014 {
00015 public:
00016     enum ConfigType : int
00017     {
00018
00019         /**
00020          * <p>
00021          * Movement type is MoveJ.
00022          * </p>
00023          *
00024          * The config instance can be cast to {@link MoveJMoveNodeConfig}.
00025          */
00026         MOVE_J = 0,
00027
00028         /**
00029          * <p>
00030          * Movement type is MoveL.
00031          * </p>
00032          *
00033          * The config instance can be cast to {@link MoveLMoveNodeConfig}.
00034          */
00035         MOVE_L,
00036
00037         MOVE_S,
00038     };
00039
00040     enum TcpSelectionType : int
00041     {
00042
00043         /**
00044          * Motions are adjusted to the currently active TCP. The TCP is
00045          * determined during runtime of the program.
00046          */
00047         USE_ACTIVE_TCP = 0,
00048
00049         /**
00050          * No TCP is used and the motion under this Move command will be with
00051          * respect to the tool output flange.
00052          */
00053         IGNORE_ACTIVE_TCP,
00054
00055         /**
00056          * A specific TCP is selected. <br>
00057          * The TCPSelection instance can be cast to {@link MoveNodeTCP}.
00058          */
00059         MOVE_TCP
00060     };
00061
00062     enum FeatureSelectionType : int
00063     {
00064         /**
00065          * The selected feature is no longer available (e.g. because the user
00066          * has removed the feature in the Installation or the needed
00067          * Installation is not currently loaded)
00068          */
00069         UNRESOLVED_FEATURE,
00070
00071         /**
00072          * Feature is selected.
00073          * The Selection instance can be cast to {@link MoveNodeFeature}
00074          */
00075         FEATURE
00076     };
00077
00078     MoveNode(MoveNode &f);
00079     MoveNode(MoveNode &&f);
00080     virtual ~MoveNode();
00081
00082     /**

```

```

00083 * \chinese
00084 *         getter
00085 * @return
00086 * \endchinese
00087 * \english
00088 * This method returns the type of configuration. Cast this instance
00089 * appropriately to have access to specific getters.
00090 * @return the type of this config.
00091 * \endenglish
00092 */
00093 ConfigType getConfigType();
00094 void setConfigType(ConfigType type);
00095
00096 bool getUseJoints();
00097 void setUseJoints(bool use);
00098
00099 /**
00100 * \chinese
00101 * <p> MoveJ          </p>
00102 *         Move
00103 *
00104 * @param joint_speed
00105 * \endchinese
00106 * \english
00107 * <p>Set joint speed parameter for the MoveJ configuration.</p>
00108 * The parameter applies to the movement of the robot arm, from the previous
00109 * position through the waypoints under the Move node.
00110 *
00111 * @param joint_speed The shared joint speed to be achieved.
00112 * \endenglish
00113 */
00114 void setJointSpeed(double joint_speed);
00115 double getJointSpeed();
00116
00117 /**
00118 * \chinese
00119 * <p> MoveJ          </p>
00120 *         Move
00121 * @param joint_acceleration
00122 * \endchinese
00123 * \english
00124 * <p>Set joint acceleration parameter for the MoveJ configuration.</p>
00125 * The parameter applies to the movement of the robot arm, from the previous
00126 * position through the waypoints under the Move node.
00127 * @param joint_acceleration The shared joint acceleration to be used.
00128 * \endenglish
00129 */
00130 void setJointAcceleration(double joint_acceleration);
00131 double getJointAcceleration();
00132
00133 /**
00134 * \chinese
00135 * <p> MoveJ   TCP   </p>
00136 * <p>         TCP   TCP </p>
00137 * @param tcp_selection   TCP
00138 * \endchinese
00139 * \english
00140 * <p>Set the TCP selection for the MoveJ configuration.</p>
00141 * <p>The TCP will be set as the active TCP before the movement of the robot
00142 * arm is performed.</p>
00143 * @param tcp_selection The TCP setting to be applied.
00144 * \endenglish
00145 */
00146 void setTCPSelection(TCPPtr tcp_selection);
00147 TCPPtr getTCPSelection();
00148
00149 /**
00150 * \chinese
00151 * <p> MoveL          </p>
00152 *         Move
00153 * @param tool_speed
00154 * \endchinese
00155 * \english
00156 * <p>Set tool speed parameter for the MoveL configuration.</p>
00157 * The parameter applies to the movement of the robot arm, from the previous
00158 * position through the waypoints under the Move node.
00159 * @param tool_speed The shared tool speed to be achieved.
00160 * \endenglish
00161 */
00162 void setToolSpeed(double tool_speed);
00163 double getToolSpeed();
00164
00165 /**
00166 * \chinese
00167 * <p> MoveL          </p>
00168 *         Move
00169 * @param tool_acceleration

```

```

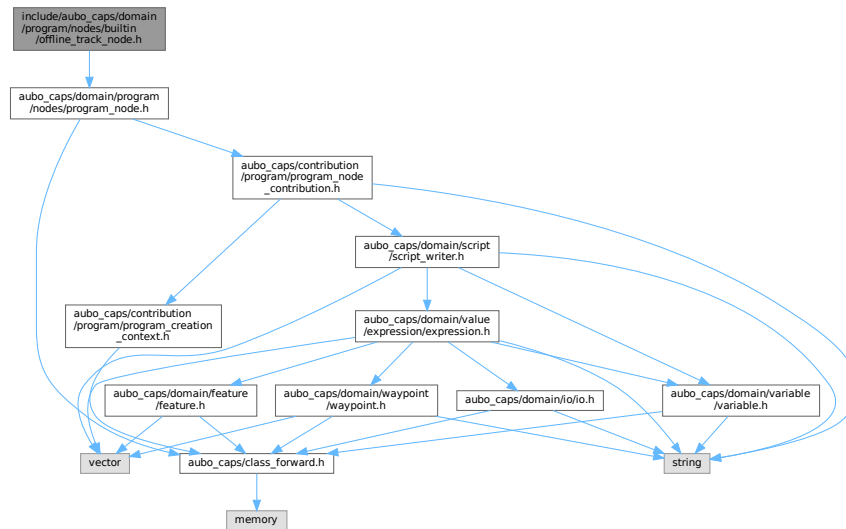
00170     * \endchinese
00171     * \english
00172     * <p>Set tool acceleration parameter for the MoveL configuration.</p>
00173     * The parameter applies to the movement of the robot arm, from the previous
00174     * position through the waypoints under the Move node.
00175     * @param tool_acceleration The shared tool acceleration to be used.
00176     * \endenglish
00177     */
00178     void setToolAcceleration(double tool_acceleration);
00179     double getToolAcceleration();
00180
00181     /**
00182     * \chinese
00183     * <p> MoveL      </p>
00184     *           Move
00185     * @param feature
00186     * \endchinese
00187     * \english
00188     * <p>Set feature for the MoveL configuration.</p>
00189     * The feature will be used for the movement of the robot arm, from the
00190     * previous position through the waypoints under the Move node.
00191     * @param feature The shared feature for this movement.
00192     * \endenglish
00193     */
00194     void setFeature(FeaturePtr feature);
00195     void setFeature(FeatureVariablePtr feature_variable);
00196     FeaturePtr getFeature();
00197
00198     /**
00199     * \chinese
00200     * <p> MoveP      </p>
00201     *           Move
00202     * @param blend
00203     * \endchinese
00204     * \english
00205     * <p>Set the blend parameter for the MoveP configuration.</p>
00206     * The blend parameter will be used between all waypoints in the movement of
00207     * the robot arm.
00208     * @param blend The shared blend used in the motion of this Move node.
00209     * \endenglish
00210     */
00211     void setBlend(double blend);
00212     double getBlend();
00213
00214     void setTcpSelectionType(TcpSelectionType type);
00215     TcpSelectionType getTcpSelectionType();
00216
00217 private:
00218     friend class DataSwitch;
00219     MoveNode();
00220     void *d_{ nullptr };
00221 };
00222
00223 } // namespace aubo_scope
00224 } // namespace arcs
00225 #endif // AUBO_SCOPE_MOVE_NODE_H

```

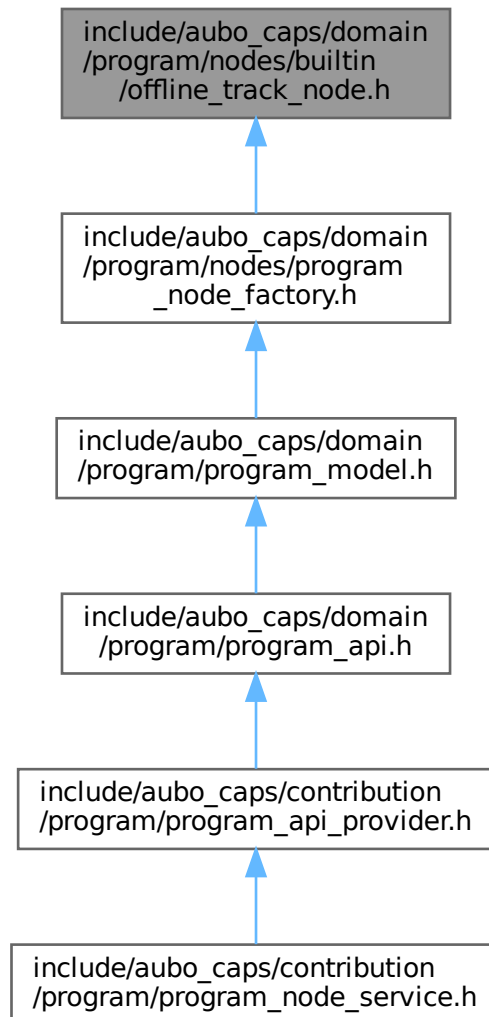
12.164 include/aubo_caps/domain/program/nodes/builtin/offline_track_node.h File Reference

#include <aubo_caps/domain/program/nodes/program_node.h>

Include dependency graph for offline_track_node.h:



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



Classes

- class [arcs::aubo_scope::OfflineTrackNode](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(OfflineTrackNode\)](#)

12.165 offline_track_node.h

[Go to the documentation of this file.](#)

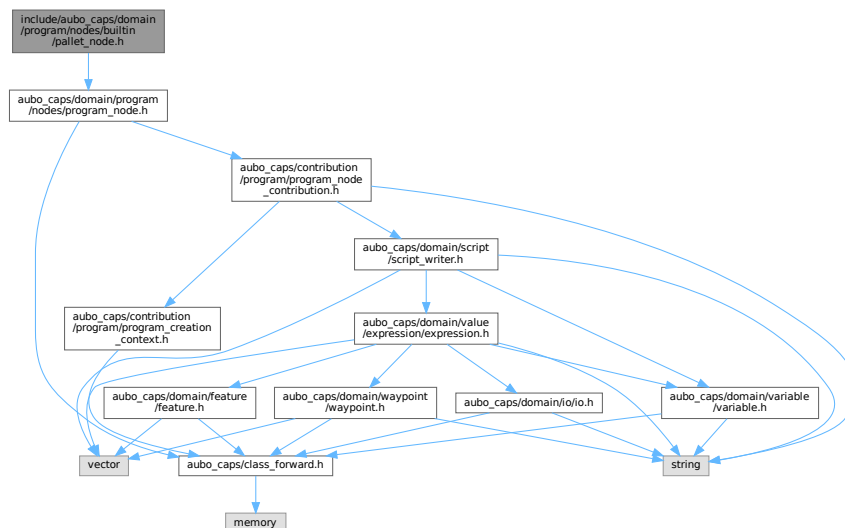
```

00001 #ifndef AUBO_SCOPE_OFFLINE_TRACK_NODE_H
00002 #define AUBO_SCOPE_OFFLINE_TRACK_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(OfflineTrackNode);
00009
00010 class ARCS_ABI_EXPORT OfflineTrackNode : public ProgramNode
00011 {
00012 public:
00013     OfflineTrackNode(OfflineTrackNode &f);
00014     OfflineTrackNode(OfflineTrackNode &&f);
00015     virtual ~OfflineTrackNode();
00016
00017 private:
00018     friend class DataSwitch;
00019     OfflineTrackNode();
00020     void *d_{ nullptr };
00021 };
00022
00023 } // namespace aubo_scope
00024 } // namespace arcs
00025 #endif // AUBO_SCOPE_OFFLINE_TRACK_NODE_H

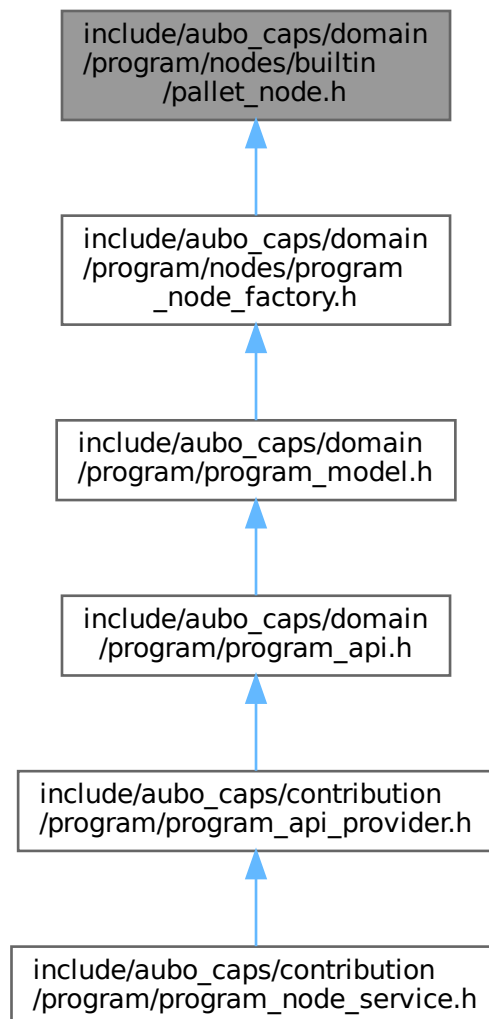
```

12.166 include/aubo_caps/domain/program/nodes/builtin/pallet_node.h File Reference

#include <aubo_caps/domain/program/nodes/program_node.h>
 Include dependency graph for pallet_node.h:



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



Classes

- class [arcs::aubo_scope::PalletNode](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(PalletNode\)](#)

12.167 pallet_node.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_PALLET_NODE_H
00002 #define AUBO_SCOPE_PALLET_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(PalletNode);
00009
00010 class ARCS_ABI_EXPORT PalletNode : public ProgramNode
00011 {
00012 public:
00013     PalletNode(PalletNode &f);
00014     PalletNode(PalletNode &&f);
00015     virtual ~PalletNode();
00016
00017 private:
00018     friend class DataSwitch;
00019     PalletNode();
00020     void *d_{ nullptr };
00021 };
00022
00023 } // namespace aubo_scope
00024 } // namespace arcs
00025 #endif // AUBO_SCOPE_PALLET_NODE_H

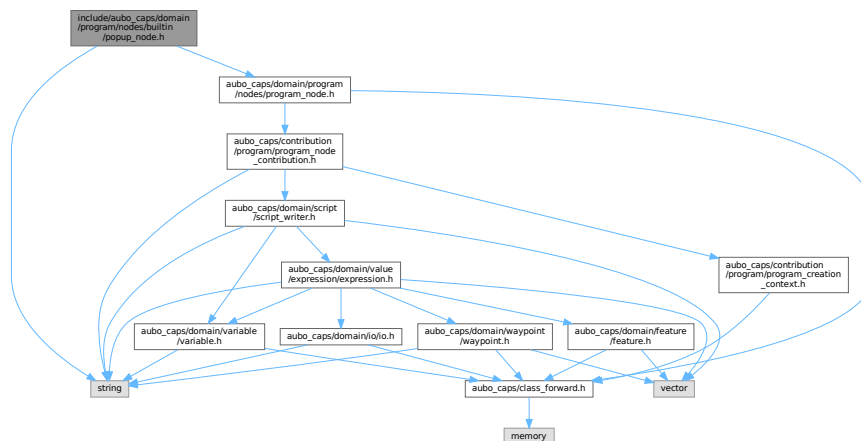
```

12.168 include/aubo_caps/domain/program/nodes/builtin/popup_node.h File Reference

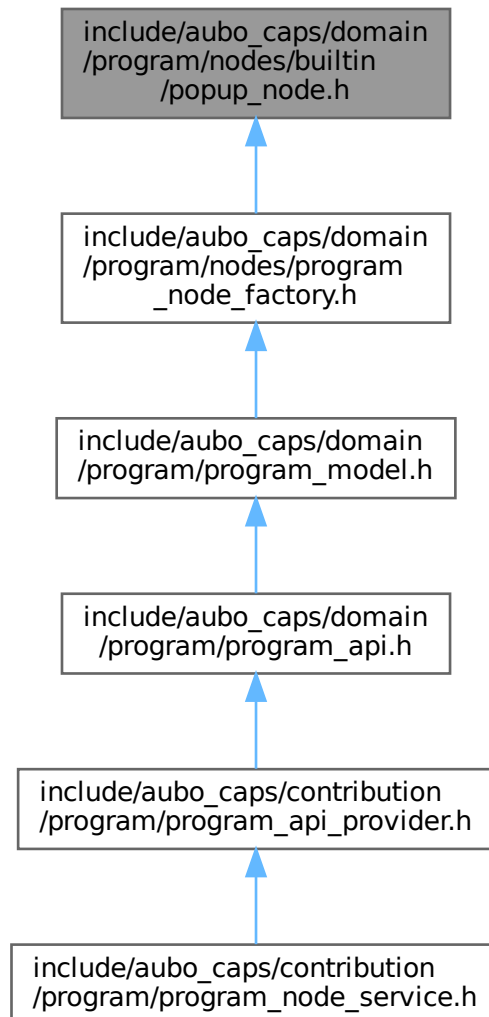
```
#include <string>
```

```
#include <aubo_caps/domain/program/nodes/program_node.h>
```

Include dependency graph for popup_node.h:



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



Classes

- class [arcs::aubo_scope::PopupNode](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(PopupNode\)](#)

12.169 popup_node.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_POPUP_NODE_H
00002 #define AUBO_SCOPE_POPUP_NODE_H
00003
00004 #include <string>
00005 #include <aubo_caps/domain/program/nodes/program_node.h>
00006
00007 namespace arcs {
00008 namespace aubo_scope {
00009 ARCS_CLASS_FORWARD(PopupNode);
00010
00011 class ARCS_ABI_EXPORT PopupNode : public ProgramNode
00012 {
00013 public:
00014     enum Type
00015     {
00016         MESSAGE,
00017         WARNING,
00018         ERROR
00019     };
00020
00021     PopupNode(PopupNode &f);
00022     PopupNode(PopupNode &&f);
00023     virtual ~PopupNode();
00024
00025     /**
00026     *
00027     * @param message the message in the pop up for the end user
00028     * @return this node
00029     * @throws IllegalStateException if called from a Qt-based AuboCap
00030     * outside of an {@link UndoableChanges} scope (see also {@link
00031     * UndoRedoManager})
00032     */
00033     void setMessage(std::string message);
00034     std::string getMessage();
00035     /**
00036     *
00037     * @param type the type of pop up
00038     * @return this node
00039     * @throws IllegalStateException if called from a Qt-based AuboCap
00040     * outside of an {@link UndoableChanges} scope (see also {@link
00041     * UndoRedoManager})
00042     */
00043     void setType(Type type);
00044     Type getType() const;
00045
00046 private:
00047     friend class DataSwitch;
00048     PopupNode();
00049     void *d_{ nullptr };
00050 };
00051
00052 } // namespace aubo_scope
00053 } // namespace arcs
00054 #endif // AUBO_SCOPE_POPUP_NODE_H

```

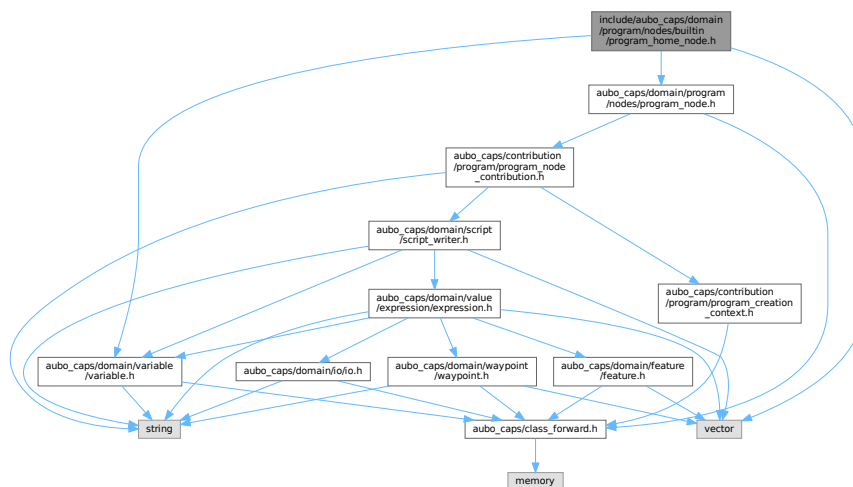
12.170 include/aubo_caps/domain/program/nodes/builtin/program_↵ home_node.h File Reference

```

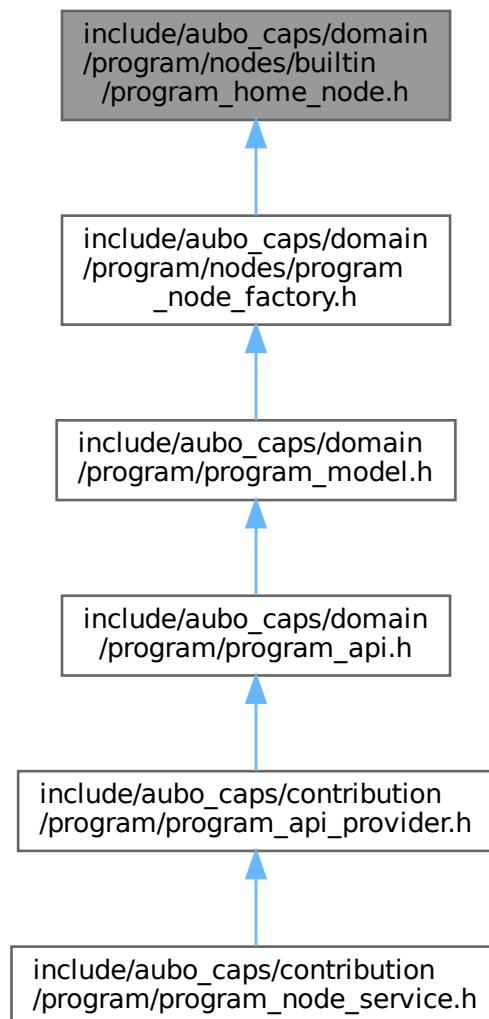
#include <vector>
#include <aubo_caps/domain/program/nodes/program_node.h>
#include <aubo_caps/domain/variable/variable.h>

```

Include dependency graph for `program_home_node.h`:



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



Classes

- class [arcs::aubo_scope::ProgramHomeNode](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([ProgramHomeNode](#))

12.171 program_home_node.h

[Go to the documentation of this file.](#)

```

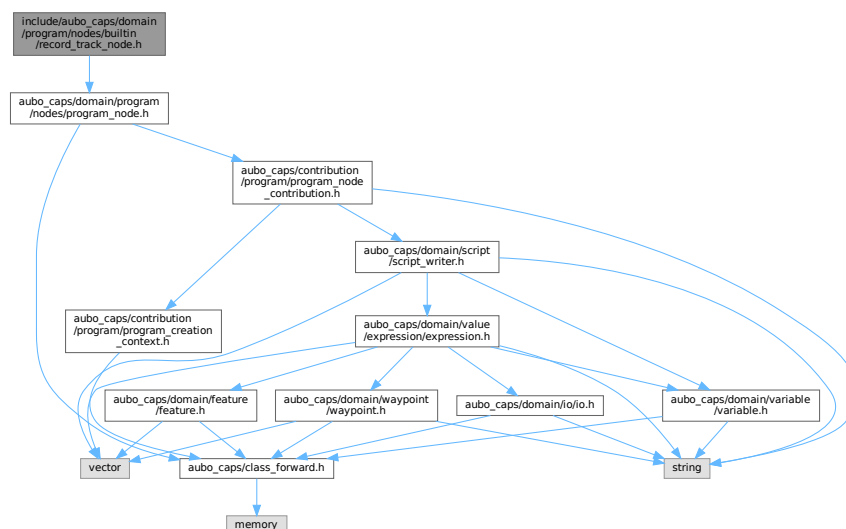
00001 #ifndef AUBO_SCOPE_PROGRAM_HOME_NODE_H
00002 #define AUBO_SCOPE_PROGRAM_HOME_NODE_H
00003
00004 #include <vector>
00005 #include <aubo_caps/domain/program/nodes/program_node.h>
00006 #include <aubo_caps/domain/variable/variable.h>
00007
00008 namespace arcs {
00009     namespace aubo_scope {
00010         ARCS_CLASS_FORWARD(ProgramHomeNode);
00011     }
00012     class ARCS_ABI_EXPORT ProgramHomeNode : public ProgramNode
00013     {
00014     public:
00015
00016         ProgramHomeNode(ProgramHomeNode &f);
00017         ProgramHomeNode(ProgramHomeNode &&f);
00018         virtual ~ProgramHomeNode();
00019
00020     private:
00021         friend class DataSwitch;
00022         ProgramHomeNode();
00023         void *d_{ nullptr };
00024     };
00025 };
00026
00027 } // namespace aubo_scope
00028 } // namespace arcs
00029 #endif // AUBO_SCOPE_PROGRAM_HOME_NODE_H

```

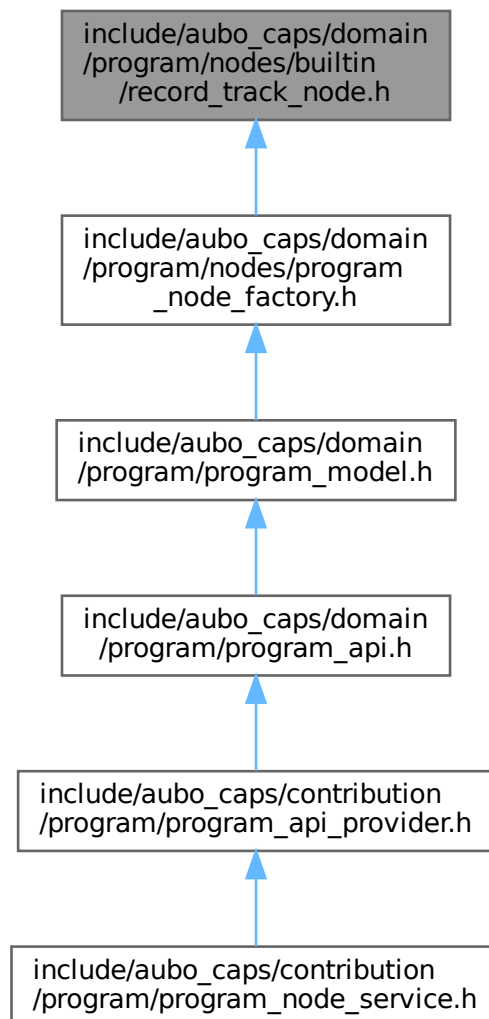
12.172 include/aubo_caps/domain/program/nodes/builtin/record_track_node.h File Reference

#include <aubo_caps/domain/program/nodes/program_node.h>

Include dependency graph for record_track_node.h:



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



Classes

- class [arcs::aubo_scope::RecordTrackNode](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([RecordTrackNode](#))

12.173 record_track_node.h

[Go to the documentation of this file.](#)

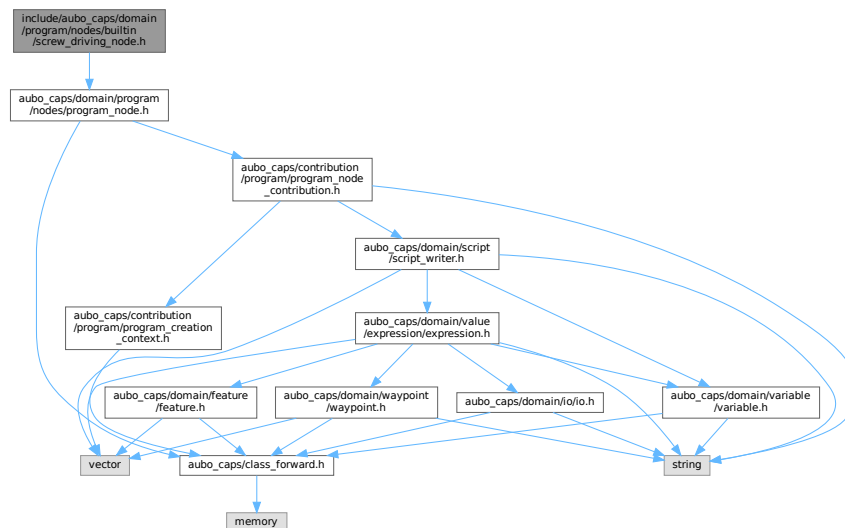
```

00001 #ifndef AUBO_SCOPE_RECORD_TRACK_NODE_H
00002 #define AUBO_SCOPE_RECORD_TRACK_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(RecordTrackNode);
00009
00010 class ARCS_ABI_EXPORT RecordTrackNode : public ProgramNode
00011 {
00012 public:
00013     RecordTrackNode(RecordTrackNode &f);
00014     RecordTrackNode(RecordTrackNode &&f);
00015     virtual ~RecordTrackNode();
00016
00017 private:
00018     friend class DataSwitch;
00019     RecordTrackNode();
00020     void *d_{ nullptr };
00021 };
00022
00023 } // namespace aubo_scope
00024 } // namespace arcs
00025 #endif // AUBO_SCOPE_RECORD_TRACK_NODE_H

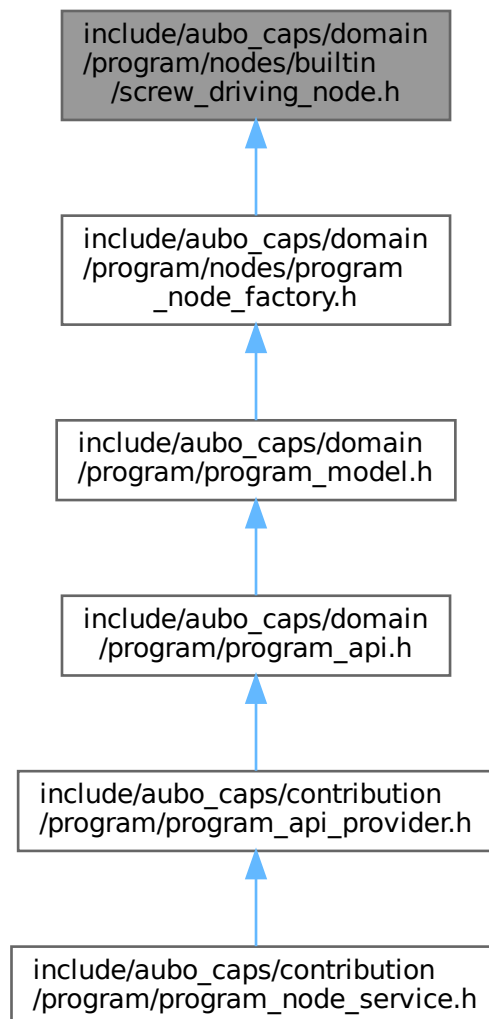
```

12.174 include/aubo_caps/domain/program/nodes/builtin/screw_driving_node.h File Reference

#include <aubo_caps/domain/program/nodes/program_node.h>
 Include dependency graph for screw_driving_node.h:



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



Classes

- class [arcs::aubo_scope::ScrewdrivingNode](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([ScrewdrivingNode](#))

12.175 screw_driving_node.h

[Go to the documentation of this file.](#)

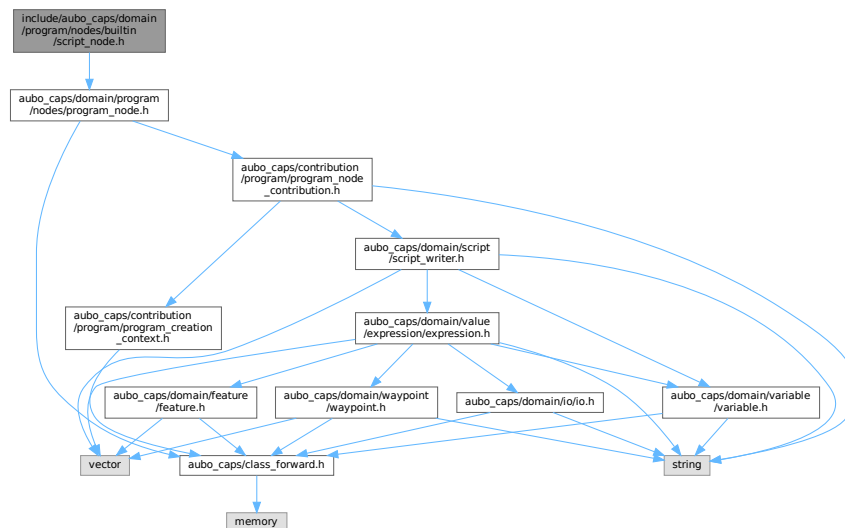
```

00001 #ifndef AUBO_SCOPE_SCREW_DRIVING_NODE_H
00002 #define AUBO_SCOPE_SCREW_DRIVING_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(ScrewdrivingNode);
00009
00010 class ARCS_ABI_EXPORT ScrewdrivingNode : public ProgramNode
00011 {
00012 public:
00013     ScrewdrivingNode(ScrewdrivingNode &f);
00014     ScrewdrivingNode(ScrewdrivingNode &&f);
00015     virtual ~ScrewdrivingNode();
00016
00017 private:
00018     friend class DataSwitch;
00019     ScrewdrivingNode();
00020     void *d_{ nullptr };
00021 };
00022
00023 } // namespace aubo_scope
00024 } // namespace arcs
00025 #endif // AUBO_SCOPE_SCREW_DRIVING_NODE_H

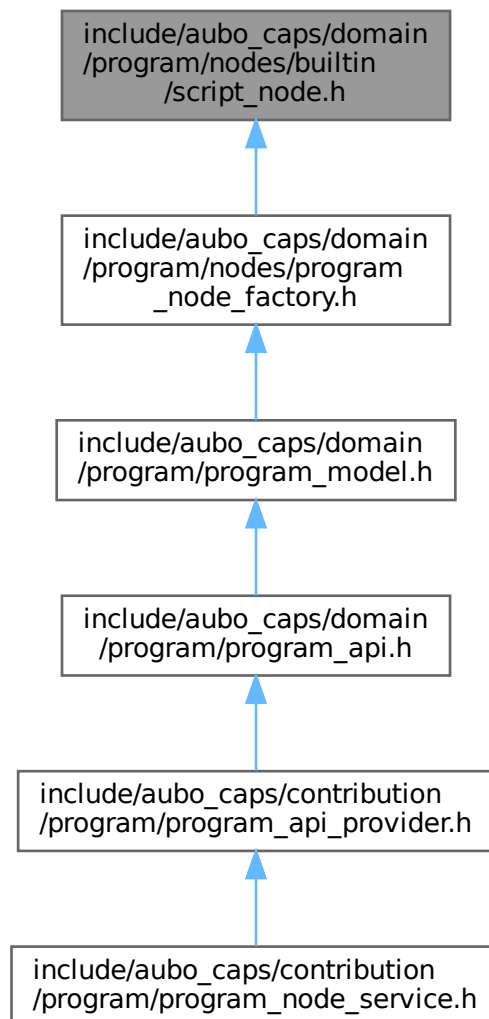
```

12.176 include/aubo_caps/domain/program/nodes/builtin/script_node.h File Reference

#include <aubo_caps/domain/program/nodes/program_node.h>
 Include dependency graph for script_node.h:



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



Classes

- class [arcs::aubo_scope::ScriptNode](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([ScriptNode](#))

12.177 script_node.h

[Go to the documentation of this file.](#)

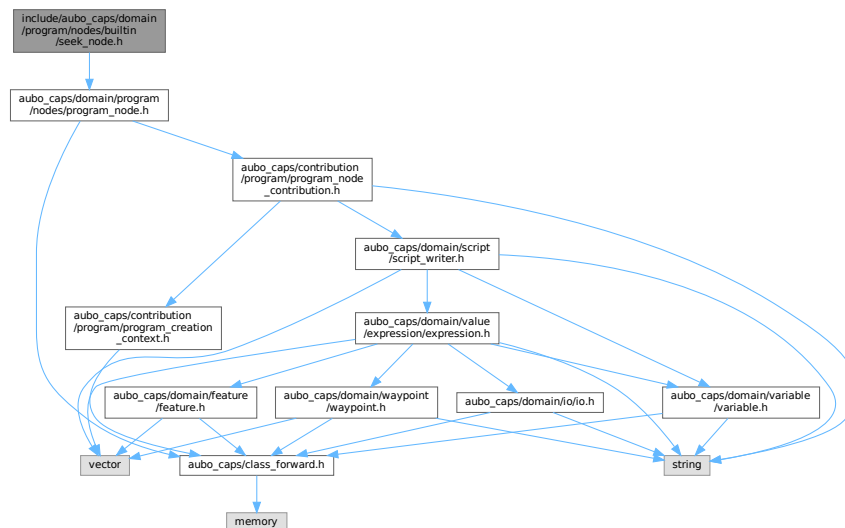
```

00001 #ifndef AUBO_SCOPE_SCRIPT_NODE_H
00002 #define AUBO_SCOPE_SCRIPT_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(ScriptNode);
00009
00010 class ARCS_ABI_EXPORT ScriptNode : public ProgramNode
00011 {
00012 public:
00013     ScriptNode(ScriptNode &f);
00014     ScriptNode(ScriptNode &&f);
00015     virtual ~ScriptNode();
00016
00017 private:
00018     friend class DataSwitch;
00019     ScriptNode();
00020     void *d_{ nullptr };
00021 };
00022
00023 } // namespace aubo_scope
00024 } // namespace arcs
00025 #endif // AUBO_SCOPE_SCRIPT_NODE_H

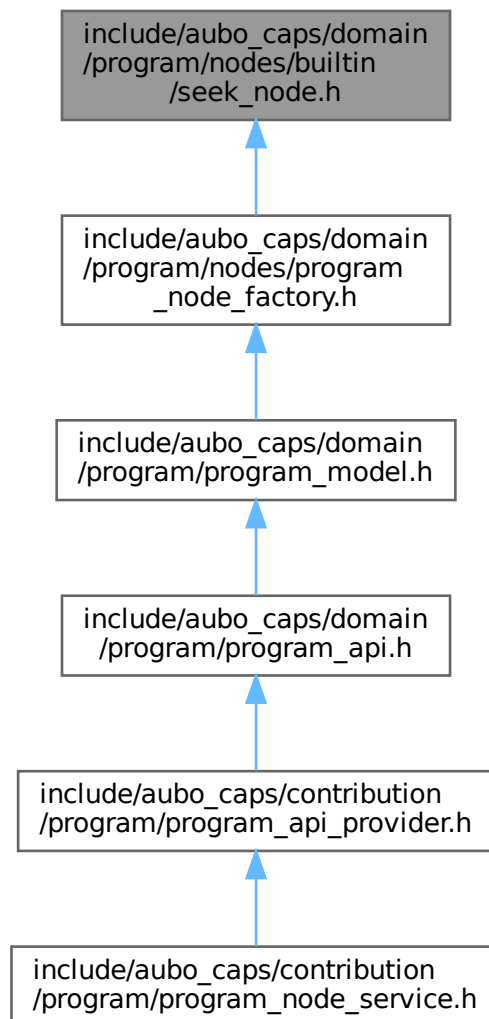
```

12.178 include/aubo_caps/domain/program/nodes/builtin/seek_node.h File Reference

#include <aubo_caps/domain/program/nodes/program_node.h>
 Include dependency graph for seek_node.h:



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



Classes

- class [arcs::aubo_scope::SeekNode](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([SeekNode](#))

12.179 seek_node.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_SEEK_NODE_H
00002 #define AUBO_SCOPE_SEEK_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(SeekNode);
00009
00010 class ARCS_ABI_EXPORT SeekNode : public ProgramNode
00011 {
00012 public:
00013     SeekNode(SeekNode &f);
00014     SeekNode(SeekNode &&f);
00015     virtual ~SeekNode();
00016
00017 private:
00018     friend class DataSwitch;
00019     SeekNode();
00020     void *d_{ nullptr };
00021 };
00022
00023 } // namespace aubo_scope
00024 } // namespace arcs
00025 #endif // AUBO_SCOPE_SEEK_NODE_H

```

12.180 include/aubo_caps/domain/program/nodes/builtin/set_node.h

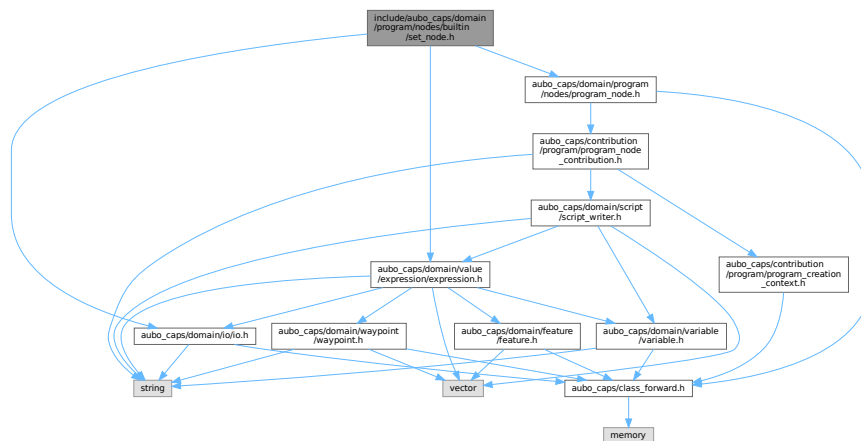
File Reference

```

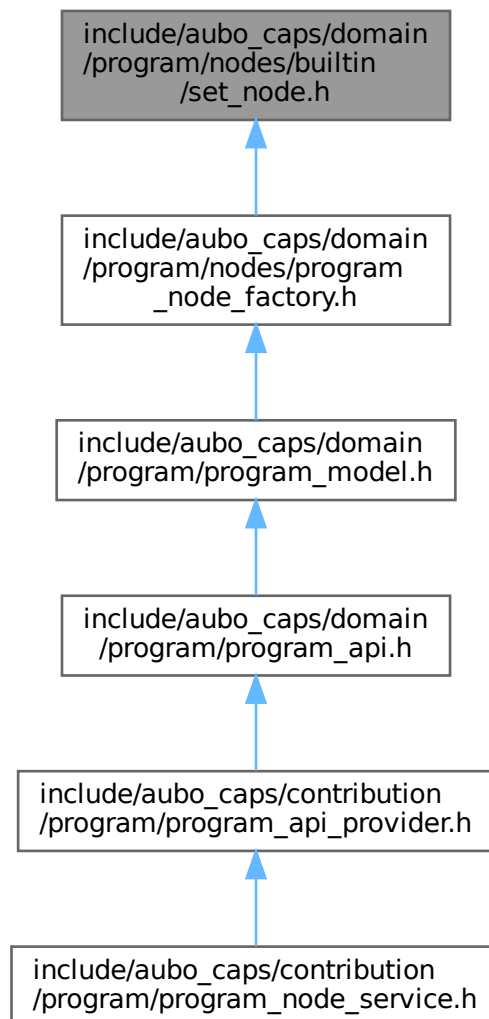
#include <aubo_caps/domain/program/nodes/program_node.h>
#include <aubo_caps/domain/value/expression/expression.h>
#include <aubo_caps/domain/io/io.h>

```

Include dependency graph for set_node.h:



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



Classes

- class [arcs::aubo_scope::SetNode](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([SetNode](#))

12.181 set_node.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_SET_NODE_H
00002 #define AUBO_SCOPE_SET_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005 #include <aubo_caps/domain/value/expression/expression.h>
00006 #include <aubo_caps/domain/io/io.h>
00007
00008 namespace arcs {
00009 namespace aubo_scope {
00010 ARCS_CLASS_FORWARD(SetNode);
00011
00012 class ARCS_ABI_EXPORT SetNode : public ProgramNode
00013 {
00014 public:
00015     /**
00016      * The configuration type used to determine which type of configuration this
00017      * instance is.
00018      */
00019     enum ConfigType
00020     {
00021         /**
00022          * No selection has been made. This type has no further information.
00023          */
00024         NO_TYPE,
00025
00026         /**
00027          * No action has been selected. This type has no further information.
00028          */
00029         NO_ACTION,
00030
00031         /**
00032          * <p>
00033          * Digital output has been selected.
00034          * </p>
00035          *
00036          * The config instance can be cast to {@link
00037          * DigitalOutputSetNodeConfig}.
00038          */
00039         DIGITAL_OUTPUT,
00040
00041         /**
00042          * <p>
00043          * Analog output configured for electric current or no output has been
00044          * selected.
00045          * </p>
00046          *
00047          * The config instance can be cast to {@link
00048          * AnalogOutputCurrentSetNodeConfig}.
00049          */
00050         ANALOG_OUTPUT_CURRENT,
00051
00052         /**
00053          * <p>
00054          * Analog output configured for voltage has been selected.
00055          * </p>
00056          *
00057          * The config instance can be cast to {@link
00058          * AnalogOutputVoltageSetNodeConfig}.
00059          */
00060         ANALOG_OUTPUT_VOLTAGE,
00061
00062         /**
00063          * <p>
00064          * Register using float type values has been selected.
00065          * </p>
00066          *
00067          * The config instance can be cast to {@link
00068          * FloatRegisterOutputSetNodeConfig}.
00069          */
00070         FLOAT_REGISTER_OUTPUT,
00071
00072         /**
00073          * <p>
00074          * Expression output has been selected.
00075          * </p>
00076          *
00077          * The config instance can be cast to {@link
00078          * ExpressionOutputSetNodeConfig}.
00079          */
00080         EXPRESSION_OUTPUT,
00081
00082         /**

```



```

00083     * <p>
00084     * Single digital pulse output has been selected.
00085     * </p>
00086     *
00087     * The config instance can be cast to {@link
00088     * SingleDigitalPulseSetNodeConfig}.
00089     */
00090     SINGLE_DIGITAL_PULSE,
00091
00092     /**
00093     * <p>
00094     * Increment variable has been selected. This type has no further
00095     * information.
00096     * </p>
00097     *
00098     * Setting this type of config will be ignored and will have no effect
00099     * on the node.
00100     */
00101     INCREMENT_VARIABLE,
00102 };
00103
00104 enum OutputSelectionType
00105 {
00106
00107     /**
00108     * No output is selected.
00109     */
00110     NO_OUTPUT,
00111
00112     /**
00113     * The selected output is no longer available (e.g. because the end user
00114     * has removed the output in the Installation or the needed Installation
00115     * is not currently loaded).
00116     */
00117     UNRESOLVED_OUTPUT,
00118
00119     /**
00120     * <p>
00121     * Output is selected.
00122     * </p>
00123     *
00124     * The Selection instance can be cast to {@link ExpressionOutput}.
00125     */
00126     OUTPUT
00127 };
00128 SetNode(SetNode &f);
00129 SetNode(SetNode &&f);
00130 virtual ~SetNode();
00131
00132 /**
00133  * \chinese
00134  *         getter
00135  * @return
00136  * \endchinese
00137  * \english
00138  * This method returns the type of configuration. Cast this instance
00139  * appropriately to have access to specific getters.
00140  * @return the type of this config.
00141  * \endenglish
00142  */
00143 ConfigType getConfigType();
00144 void setConfigType(ConfigType type);
00145
00146 /**
00147  * @return the time for the length of the pulse.
00148  */
00149 double getPulseTime();
00150 void setPulseTime(double time);
00151
00152 /**
00153  *
00154  * @return the float value to be applied to the float register output.
00155  */
00156 float getFloatValueToSet();
00157 void setFloatValueToSet(float value);
00158
00159 /**
00160  * @return the analog output value.
00161  */
00162 double getAnalogOutput();
00163 void setAnalogOutput(double value);
00164
00165 OutputSelectionType getOutputSelectionType();
00166
00167 /**
00168  *
00169  * @return the expression whose evaluation will be applied to the output.

```

```

00170  */
00171  ExpressionPtr getExpression();
00172  void setExpression(ExpressionPtr expression);
00173
00174  /**
00175   * The output selected can be an analog output, a digital output, a MODBUS
00176   * output or a register output.
00177   *
00178   * @return the output to which the result of the evaluation of the
00179   * expression will be applied.
00180   */
00181  IoPtr getOutput();
00182  void setOutput(IoPtr io);
00183
00184  /**
00185   * Depending on the type of output this is interpreted in different ways.
00186   * For digital output and MODBUS output,
00187   * <code>true</code> means the output will be set to HIGH,
00188   * <code>false</code> means the output will be set to LOW. For boolean
00189   * register the value is interpreted literally.
00190   *
00191   * @return the output value.
00192   */
00193  bool getBoolValueToSet();
00194  void setBoolValueToSet(bool value);
00195
00196  /**
00197   *
00198   * @return the voltage value/level to be applied to the output
00199   */
00200  double getVoltage();
00201  void setVoltage(double voltage);
00202
00203  /**
00204   *
00205   * @return the electric current value/level to be applied to the output.
00206   */
00207  double getCurrent();
00208  void setCurrent(double curr);
00209
00210 private:
00211  friend class DataSwitch;
00212  SetNode();
00213  void *d_{ nullptr };
00214 };
00215
00216 } // namespace aubo_scope
00217 } // namespace arcs
00218 #endif // AUBO_SCOPE_SET_NODE_H

```

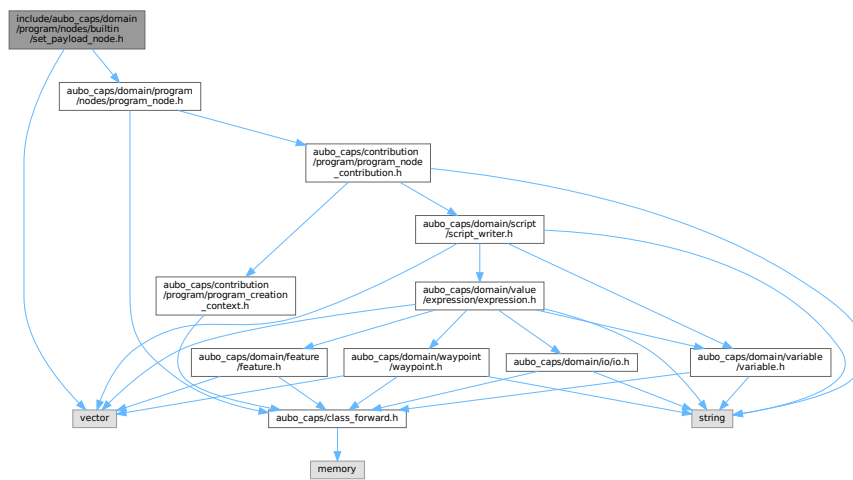
12.182 include/auco_caps/domain/program/nodes/builtin/set_ payload_node.h File Reference

```

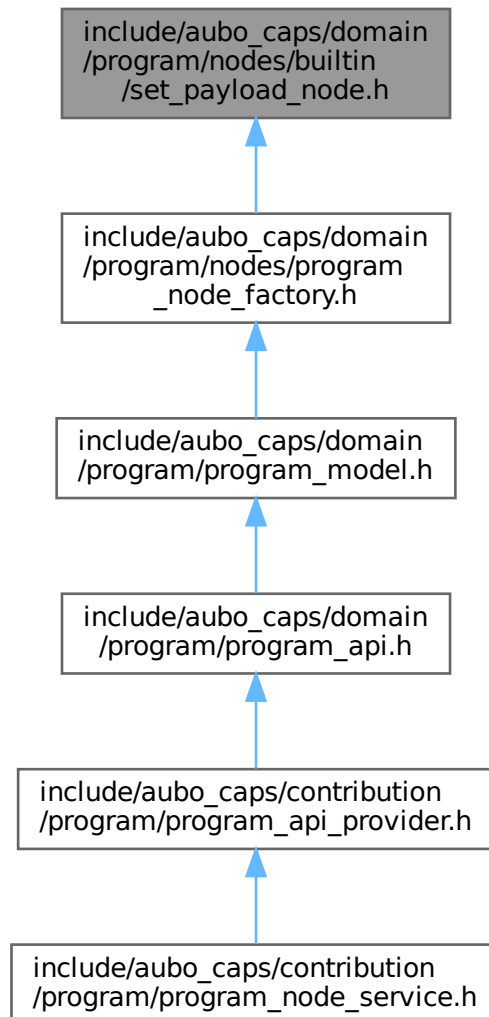
#include <vector>
#include <auco_caps/domain/program/nodes/program_node.h>

```

Include dependency graph for set_payload_node.h:



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



Classes

- class [arcs::aubo_scope::SetPayloadNode](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(SetPayloadNode\)](#)

12.183 set_payload_node.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_SET_PAYLOAD_NODE_H
00002 #define AUBO_SCOPE_SET_PAYLOAD_NODE_H
00003
00004 #include <vector>
00005 #include <aubo_caps/domain/program/nodes/program_node.h>
00006
00007 namespace arcs {
00008 namespace aubo_scope {
00009 ARCS_CLASS_FORWARD(SetPayloadNode);
00010
00011 class ARCS_ABI_EXPORT SetPayloadNode : public ProgramNode
00012 {
00013 public:
00014 /**
00015  * This base interface represents a configuration of the Set Payload program
00016  * node (i.e. a {@link SetPayloadNode}
00017  * instance).
00018  */
00019 enum ConfigType
00020 {
00021 /**
00022  * <p>
00023  * A payload (from the installation) is selected.
00024  * </p>
00025  *
00026  * The config instance can be cast to {@link
00027  * SelectionPayloadNodeConfig}.
00028  */
00029 PAYLOAD_SELECTION,
00030
00031 /**
00032  * <p>
00033  * Custom payload parameters are specified.
00034  * </p>
00035  *
00036  * The config instance can be cast to {@link
00037  * CustomParametersPayloadNodeConfig}.
00038  */
00039 CUSTOM_PARAMETERS
00040 };
00041
00042 SetPayloadNode(SetPayloadNode &f);
00043 SetPayloadNode(SetPayloadNode &&f);
00044 virtual ~SetPayloadNode();
00045
00046 /**
00047  * \chinese
00048  *      getter
00049  * @return
00050  * \endchinese
00051  * \english
00052  * This method returns the type of configuration. Cast this instance
00053  * appropriately to have access to specific getters.
00054  * @return The type of this config.
00055  * \endenglish
00056  */
00057 ConfigType getConfigType();
00058 void setConfigType(ConfigType type);
00059
00060 /**
00061  * \chinese
00062  * @return
00063  * \endchinese
00064  * \english
00065  * @return The total mass of the payload
00066  * \endenglish
00067  */
00068 double getMass();
00069 void setMass(double mass);
00070
00071 /**
00072  * \chinese
00073  *      CoG
00074  * @return
00075  * \endchinese
00076  * \english
00077  * Gets the payload's center of gravity (CoG), also referred to as center of
00078  * mass. It is defined as the offset between the center of the tool output
00079  * flange and the center of gravity of the attached payload.
00080  * @return The center of gravity (CoG) of the payload relative to the center
00081  * of the tool output flange
00082  * \endenglish

```

```

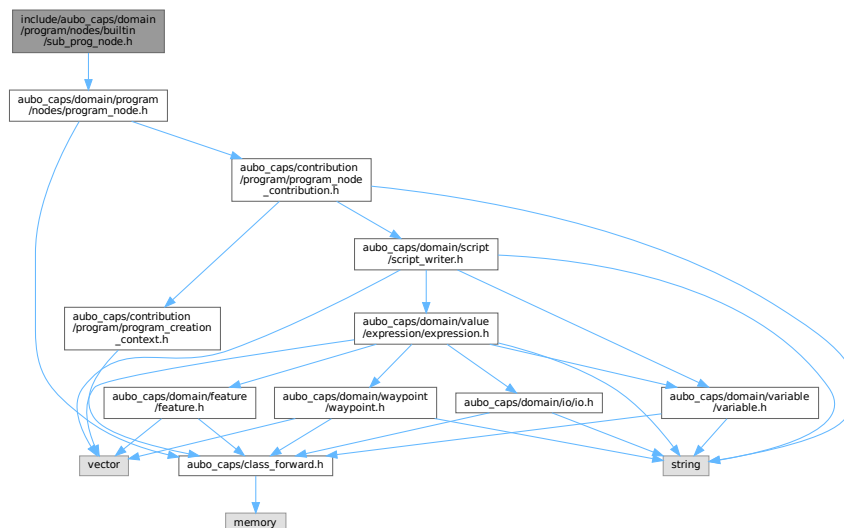
00083     */
00084     std::vector<double> getCenterOfGravity();
00085     void setCenterOfGravity(const std::vector<double> &cog);
00086
00087 private:
00088     friend class DataSwitch;
00089     SetPayloadNode();
00090     void *d_{ nullptr };
00091 };
00092
00093 } // namespace aubo_scope
00094 } // namespace arcs
00095 #endif // AUBO_SCOPE_SET_PAYLOAD_NODE_H

```

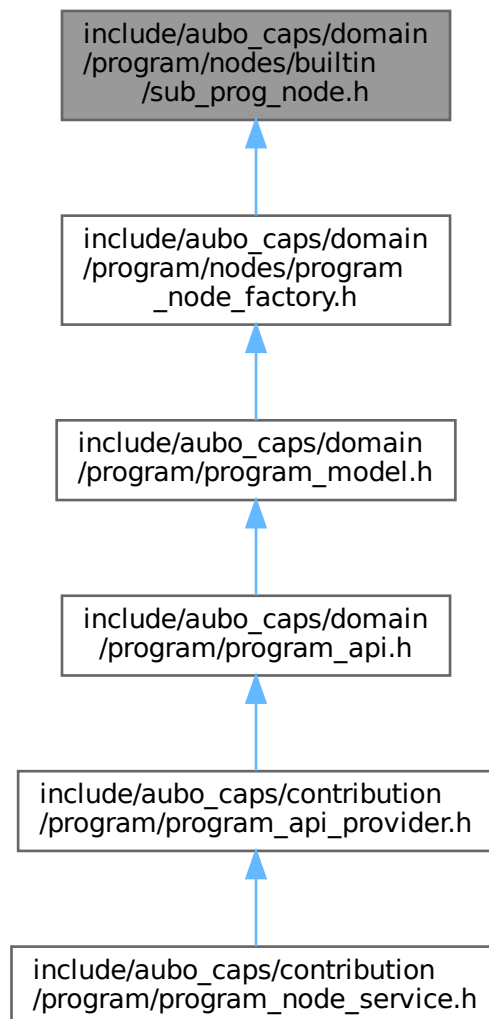
12.184 include/auco_caps/domain/program/nodes/builtin/sub_prog_node.h File Reference

#include <auco_caps/domain/program/nodes/program_node.h>

Include dependency graph for sub_prog_node.h:



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



Classes

- class [arcs::aubo_scope::SubProgNode](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) (SubProgNode)

12.185 sub_prog_node.h

[Go to the documentation of this file.](#)

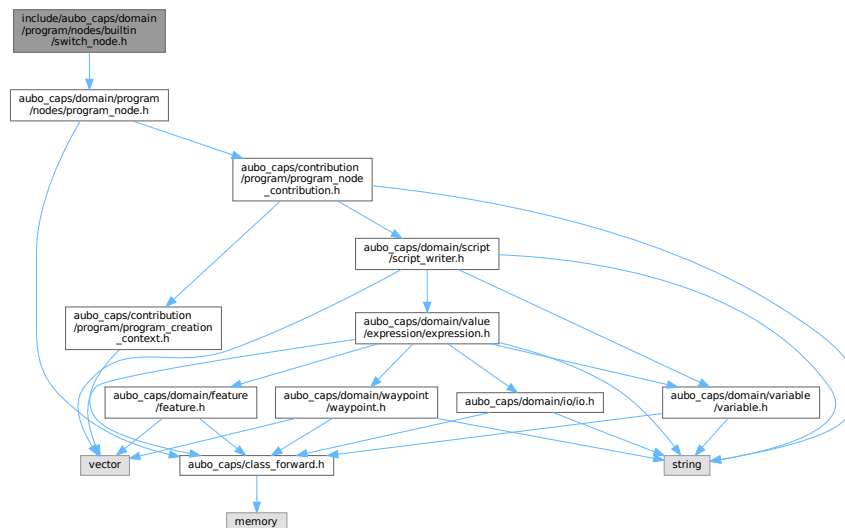
```

00001 #ifndef AUBO_SCOPE_SUB_PROG_NODE_H
00002 #define AUBO_SCOPE_SUB_PROG_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(SubProgNode);
00009
00010 class ARCS_ABI_EXPORT SubProgNode : public ProgramNode
00011 {
00012 public:
00013     SubProgNode(SubProgNode &f);
00014     SubProgNode(SubProgNode &&f);
00015     virtual ~SubProgNode();
00016
00017 private:
00018     friend class DataSwitch;
00019     SubProgNode();
00020     void *d_{ nullptr };
00021 };
00022
00023 } // namespace aubo_scope
00024 } // namespace arcs
00025 #endif // AUBO_SCOPE_SUB_PROG_NODE_H

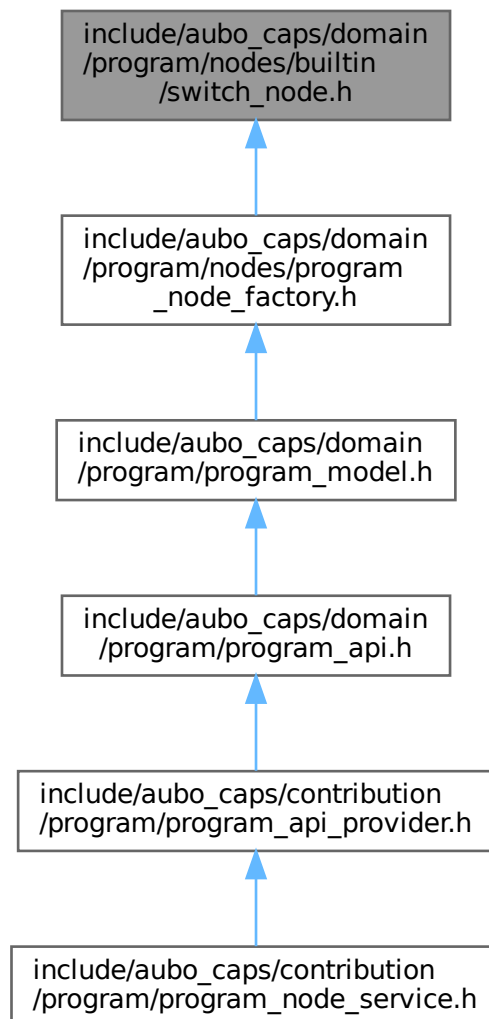
```

12.186 include/aubo_caps/domain/program/nodes/builtin/switch_node.h File Reference

#include <aubo_caps/domain/program/nodes/program_node.h>
 Include dependency graph for switch_node.h:



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



Classes

- class [arcs::aubo_scope::SwitchNode](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([SwitchNode](#))

12.187 switch_node.h

[Go to the documentation of this file.](#)

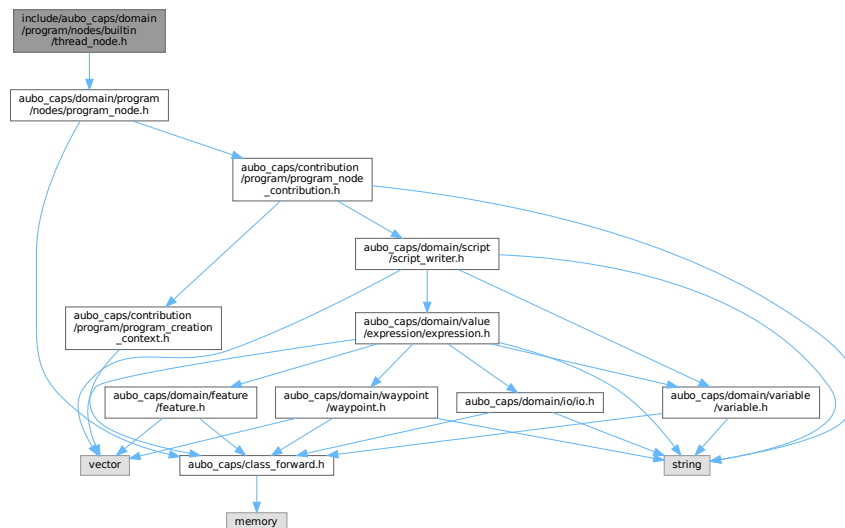
```

00001 #ifndef AUBO_SCOPE_SWITCH_NODE_H
00002 #define AUBO_SCOPE_SWITCH_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(SwitchNode);
00009
00010 class ARCS_ABI_EXPORT SwitchNode : public ProgramNode
00011 {
00012 public:
00013     SwitchNode(SwitchNode &f);
00014     SwitchNode(SwitchNode &&f);
00015     virtual ~SwitchNode();
00016
00017 private:
00018     friend class DataSwitch;
00019     SwitchNode();
00020     void *d_{ nullptr };
00021 };
00022
00023 } // namespace aubo_scope
00024 } // namespace arcs
00025 #endif // AUBO_SCOPE_SWITCH_NODE_H

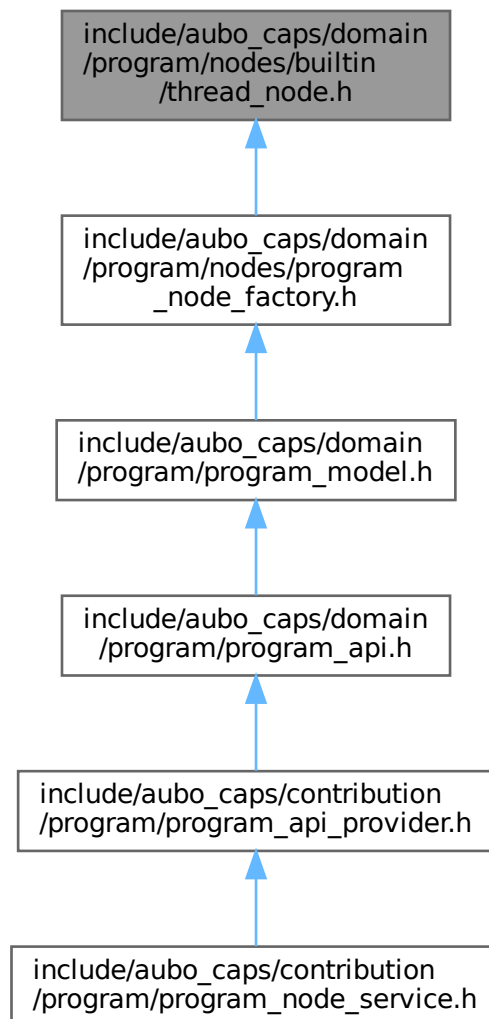
```

12.188 include/aubo_caps/domain/program/nodes/builtin/thread_node.h File Reference

#include <aubo_caps/domain/program/nodes/program_node.h>
 Include dependency graph for thread_node.h:



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



Classes

- class [arcs::aubo_scope::ThreadNode](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([ThreadNode](#))

12.189 thread_node.h

[Go to the documentation of this file.](#)

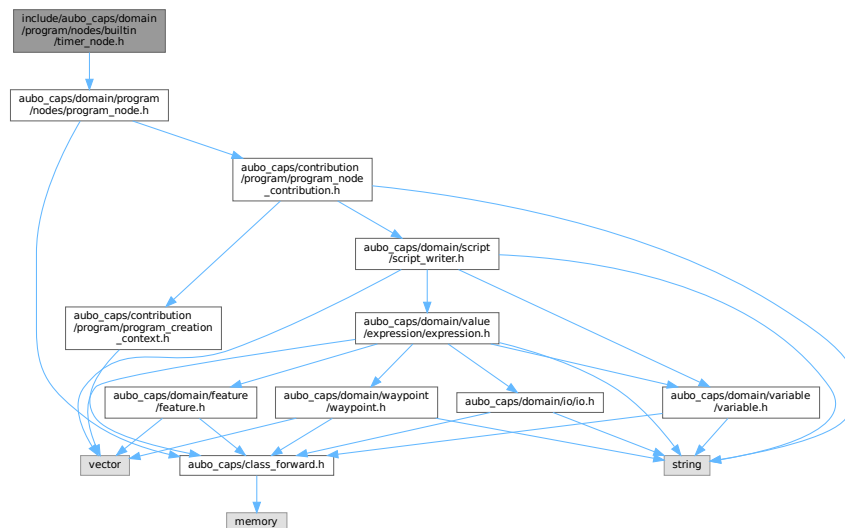
```

00001 #ifndef AUBO_SCOPE_THREAD_NODE_H
00002 #define AUBO_SCOPE_THREAD_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(ThreadNode);
00009
00010 class ARCS_ABI_EXPORT ThreadNode : public ProgramNode
00011 {
00012 public:
00013     ThreadNode(ThreadNode &f);
00014     ThreadNode(ThreadNode &&f);
00015     virtual ~ThreadNode();
00016
00017 private:
00018     friend class DataSwitch;
00019     ThreadNode();
00020     void *d_{ nullptr };
00021 };
00022
00023 } // namespace aubo_scope
00024 } // namespace arcs
00025 #endif // AUBO_SCOPE_THREAD_NODE_H

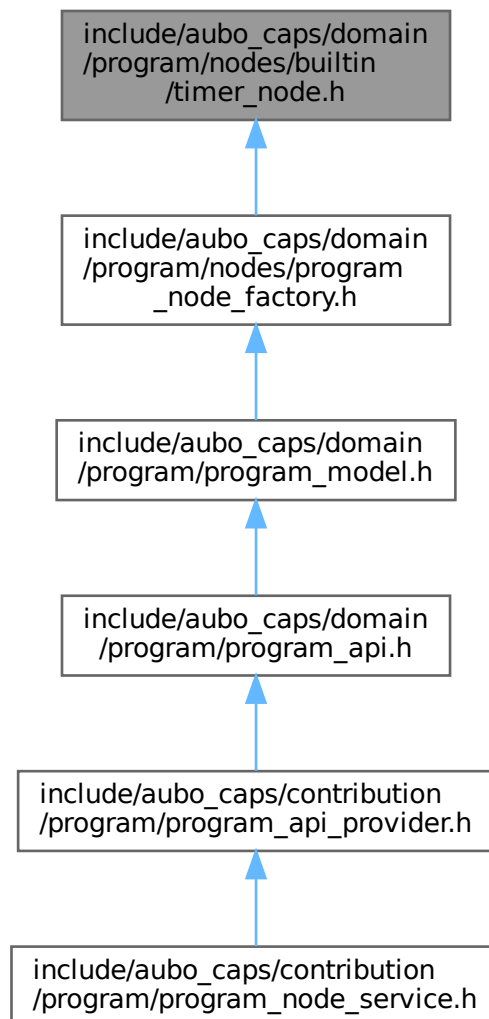
```

12.190 include/aubo_caps/domain/program/nodes/builtin/timer_node.h File Reference

#include <aubo_caps/domain/program/nodes/program_node.h>
 Include dependency graph for timer_node.h:



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



Classes

- class [arcs::aubo_scope::TimerNode](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([TimerNode](#))

12.191 timer_node.h

[Go to the documentation of this file.](#)

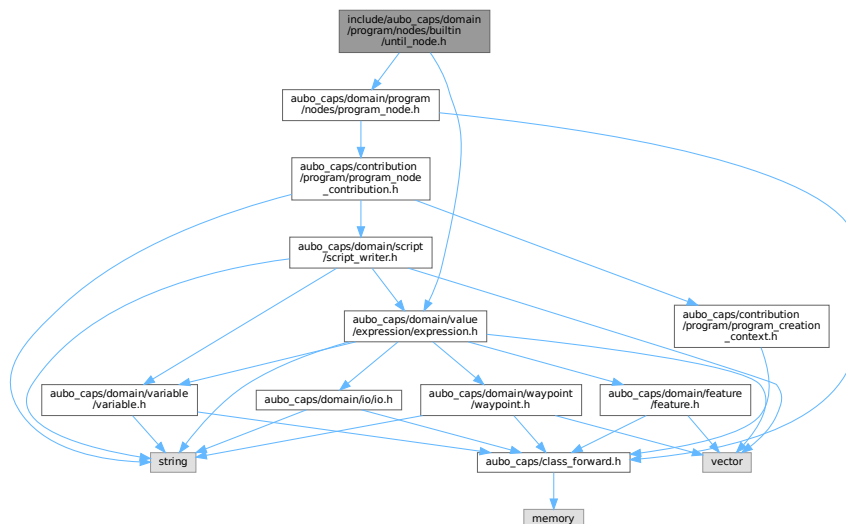
```

00001 #ifndef AUBO_SCOPE_TIMER_NODE_H
00002 #define AUBO_SCOPE_TIMER_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(TimerNode);
00009
00010 class ARCS_ABI_EXPORT TimerNode : public ProgramNode
00011 {
00012 public:
00013     TimerNode(TimerNode &f);
00014     TimerNode(TimerNode &&f);
00015     virtual ~TimerNode();
00016
00017 private:
00018     friend class DataSwitch;
00019     TimerNode();
00020     void *d_{ nullptr };
00021 };
00022
00023 } // namespace aubo_scope
00024 } // namespace arcs
00025 #endif // AUBO_SCOPE_TIMER_NODE_H

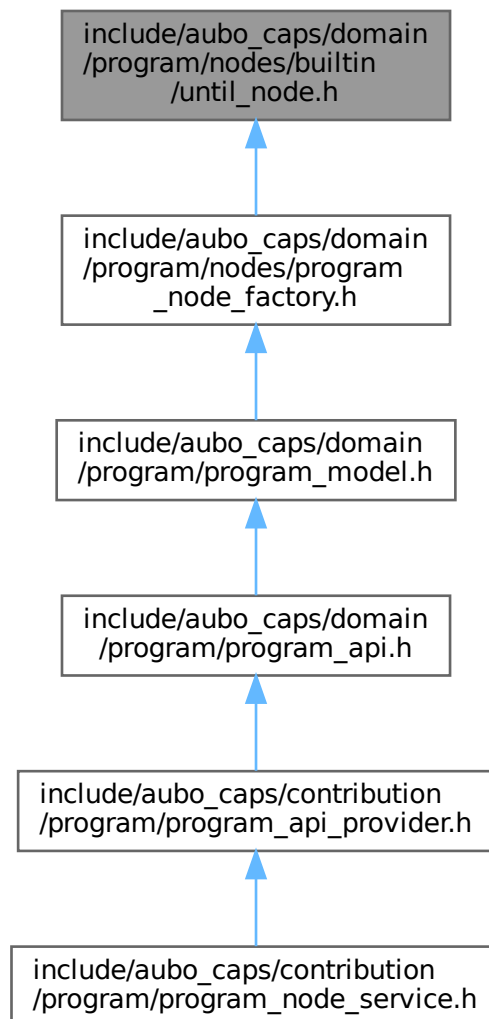
```

12.192 include/aubo_caps/domain/program/nodes/builtin/until_node.h File Reference

#include <aubo_caps/domain/program/nodes/program_node.h>
#include <aubo_caps/domain/value/expression/expression.h>
Include dependency graph for until_node.h:



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



Classes

- class [arcs::aubo_scope::UntilNode](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(UntilNode\)](#)

12.193 until_node.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_UNTIL_NODE_H
00002 #define AUBO_SCOPE_UNTIL_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005 #include <aubo_caps/domain/value/expression/expression.h>
00006
00007 namespace arcs {
00008 namespace aubo_scope {
00009 ARCS_CLASS_FORWARD(UntilNode);
00010
00011 /**
00012  * \chinese
00013  * <p>Until Waypoint Direction </p>
00014  * <p>TreeNode#addChild #removeChild Until
00015  * Until
00016  * </p>
00017  * \endchinese
00018  * \english
00019  * <p>The Until node is a type of program node that must be under a Waypoint
00020  * node or Direction node.</p>
00021  * <p>It has special behavior when calling {@link
00022  * TreeNode#addChild(ProgramNode)} and {@link
00023  * TreeNode#removeChild(TreeNode)}. Until nodes by default are leaf nodes,
00024  * however they can have actions added to them. When that happens they are
00025  * transformed into branch nodes that allow child nodes. When the last child
00026  * is removed they become leaf nodes again. This is all automatically
00027  * handled when {@link TreeNode#addChild(ProgramNode)} and {@link
00028  * TreeNode#removeChild(TreeNode)} are called.</p>
00029  * \endenglish
00030  */
00031 class ARCS_ABI_EXPORT UntilNode : public ProgramNode
00032 {
00033 public:
00034     /**
00035      * The configuration type used to determine which type of configuration this
00036      * instance is.
00037      */
00038     enum ConfigType : short
00039     {
00040
00041         /**
00042          * <p>
00043          * An Until node where the until condition is true once the
00044          * evaluation of an expression is true.
00045          * </p>
00046          *
00047          * The config instance can be cast to {@link
00048          * ExpressionUntilNodeConfig}.
00049          */
00050         EXPRESSION,
00051         /**
00052          * <p>
00053          * An Until node where the until condition is true once an impact on
00054          * the Tool is detected.
00055          * </p>
00056          *
00057          * This functionality is not supported by the API.
00058          */
00059         TOOL_CONTACT,
00060         /**
00061          * <p>
00062          * An Until node where the until condition is true once a given I/O
00063          * reaches a certain value.
00064          * </p>
00065          *
00066          * The config can be cast to {@link IOInputUntilNodeConfig}.
00067          */
00068         IO_INPUT,
00069         /**
00070          * <p>
00071          * An Until node where the until condition is true once a given
00072          * distance is reached.
00073          * </p>
00074          *
00075          * The config can be cast to {@link DistanceUntilNodeConfig}.
00076          */
00077         DISTANCE,
00078         /**
00079          * <p>
00080          * An Until node where the until condition is true once its parent
00081          * Waypoint is reached.
00082          */

```



```

00083     * </p>
00084     *
00085     * The config can be cast to {@link ReachedWaypointUntilNodeConfig}.
00086     */
00087     REACHED_WAYPOINT,
00088
00089     /**
00090     * An undefined Until node with no type set yet.
00091     */
00092     NONE_CONFIG_TYPE
00093 };
00094
00095 /**
00096 * Compare operator types LESS_THAN and GREATER_THAN used when waiting for
00097 * an analog input to go lower or higher, respectively, than a threshold
00098 */
00099 enum CompareOperator : int
00100 {
00101     LESS_THAN,
00102     GREATER_THAN
00103 };
00104
00105 /**
00106 * Blend Parameters can be of different types
00107 * The parameter type is used to determine the explicit blend parameters
00108 * (can be used in place of instanceof operations).
00109 */
00110 enum BlendParameterType
00111 {
00112
00113     /**
00114     * The robot arm will stop without performing a blend.
00115     */
00116     NO_BLEND,
00117
00118     /**
00119     * <p>
00120     * Custom blend has been selected.
00121     * </p>
00122     *
00123     * The blend instance can be cast to {@link CustomBlendParameters}.
00124     */
00125     CUSTOM_BLEND
00126 };
00127
00128 enum CustomDecelerationParameterType
00129 {
00130
00131     /**
00132     * This deceleration was either created specifying values for both
00133     * {@link AngularAcceleration} and {@link Acceleration} and/or is in use
00134     * in an Until node that is not yet in the program tree.
00135     */
00136     Unknown,
00137
00138     /**
00139     * This deceleration is currently used under a MoveJ node.
00140     */
00141     Angular,
00142
00143     /**
00144     * This deceleration is currently used under a MoveL or MoveP node.
00145     */
00146     Cartesian
00147 };
00148
00149 /**
00150 * The deceleration parameter type used to determine which type of
00151 * deceleration parameter this instance is.
00152 */
00153 enum DecelerationParameterType
00154 {
00155     /**
00156     * Shared deceleration has been selected. The acceleration values from
00157     * the parent Move node will be used as deceleration (absolute value).
00158     */
00159     SHARED_DECELERATION,
00160
00161     /**
00162     * <p>
00163     * Custom deceleration has been selected.
00164     * </p>
00165     *
00166     * This instance can be cast to {@link CustomDecelerationParameters}.
00167     */
00168     CUSTOM_DECELERATION
00169

```

```

00170 };
00171
00172 /**
00173  * The type of the input selection.
00174  */
00175 enum IOInputSelectionType
00176 {
00177     /**
00178      * The node is not fully configured.
00179      */
00180     NONE_INPUT_SELECT,
00181
00182     /**
00183      * <p>
00184      * Digital input is selected.
00185      * </p>
00186      *
00187      * The config instance can be cast to {@link
00188      * DigitalInputUntilNodeConfig}.
00189      */
00190     DIGITAL_INPUT,
00191
00192     /**
00193      * <p>
00194      * Analog input is selected.
00195      * </p>
00196      *
00197      * The config instance can be cast to {@link
00198      * AnalogInputUntilNodeConfig}.
00199      */
00200     ANALOG_INPUT,
00201
00202     /**
00203      * <p>
00204      * A float input register is selected.
00205      * </p>
00206      *
00207      * The config instance can be cast to {@link
00208      * FloatRegisterInputUntilNodeConfig}.
00209      */
00210     FLOAT_REGISTER_INPUT
00211 };
00212
00213 UntilNode(UntilNode &f);
00214 UntilNode(UntilNode &&f);
00215 virtual ~UntilNode();
00216
00217 /**
00218  * This method returns the type of configuration. Cast this instance
00219  * appropriately to have access to specific getters.
00220  *
00221  * @return the type of this config.
00222  */
00223 ConfigType getConfigType();
00224 void setConfigType(ConfigType type);
00225
00226 BlendParameterType getBlendParameterType();
00227 void setBlendParameterType(BlendParameterType type);
00228
00229 /**
00230  * Creates blend parameters for an Until node. Can be used in a Distance
00231  * configuration type.
00232  *
00233  * @param suggestedBlend This blend value may be (automatically) corrected
00234  * depending on the value of the distance parameter the Distance
00235  * configuration.
00236  *
00237  * @return the blend parameters for the Until node.
00238  */
00239 void setWaypointBlendParameter(double suggested_blend);
00240 double getWaypointBlendParameter();
00241
00242 DecelerationParameterType getDecelerationParameterType();
00243 void setDecelerationParameterType(DecelerationParameterType type);
00244
00245 /**
00246  * <p>
00247  * Creates deceleration parameter for an Until node. Can be used in an
00248  * Expression configuration type.
00249  * </p>
00250  *
00251  * An angular deceleration is used when the type of the parent Move node is
00252  * a MoveJ, whereas a Cartesian deceleration is used when the move type is a
00253  * MoveL or MoveP.
00254  *
00255  * @param deceleration the angular deceleration (the value for the Cartesian
00256  * deceleration will be the default value that AuboScope uses), not

```

```

00257 * <code>null</code>.
00258 * @param decelerationErrorHandler error handler for handling validation. If
00259 * using {@link ErrorHandler#AUTO_CORRECT} this will clamp the value to the
00260 * nearest valid angular deceleration value.
00261 * @return the deceleration parameter for the Until node.
00262 */
00263 void setDecelerationParameters(double deceleration);
00264 double getDecelerationParameters();
00265
00266 /**
00267 *
00268 * @return the deceleration type. This can change depending on the Move node
00269 * type above the Until node.
00270 */
00271 CustomDecelerationParameterType getCustomDecelerationParameterType();
00272
00273 void setCustomDecelerationParameterType(
00274     CustomDecelerationParameterType type);
00275
00276 /**
00277 * The rate at which to decelerate when the until condition is met.
00278 *
00279 * @return the specified angular deceleration of the until node (absolute
00280 * value).
00281 */
00282 double getAngularDeceleration();
00283 void setAngularDeceleration(double dec);
00284
00285 /**
00286 * The rate at which to decelerate when the until condition is met.
00287 *
00288 * @return the specified Cartesian deceleration of the until node (absolute
00289 * value).
00290 */
00291 double getCartesianDeceleration();
00292 void setCartesianDeceleration(double dec);
00293
00294 /**
00295 * <p>
00296 * Creates an Expression configuration for an Until node with an expression
00297 * and default value for the deceleration parameter. The until condition for
00298 * this configuration is true once the evaluation of the expression is true.
00299 * </p>
00300 *
00301 * Note that this default value may change between AuboScope versions.
00302 *
00303 * @param expression the expression whose evaluation will determine whether
00304 * the until condition is met.
00305 * @return the Until node configuration.
00306 */
00307 void setExpressionConfig(ExpressionPtr expression);
00308 ExpressionPtr getExpressionConfig();
00309
00310 /**
00311 * The distance the robot will travel.
00312 *
00313 * @return The distance.
00314 */
00315 double getDistance();
00316 void setDistance(double dis);
00317 IOInputSelectionType getIOInputSelectionType();
00318 void setIOInputSelectionType(IOInputSelectionType type);
00319
00320 /**
00321 * Creates an I/O Input configuration for an Until node. The until condition
00322 * for this configuration is true once the signal value of a given digital
00323 * input goes HIGH or LOW.
00324 *
00325 * @param input the digital input whose signal value will determine whether
00326 * the until condition is met, not <code>null</code>.
00327 * @param valueToProceedUntil <code>true</code>, if the until condition is
00328 * met when the input signal value goes HIGH, <code>false</code> otherwise
00329 * (condition met when signal value is LOW).
00330 * @return the Until node configuration.
00331 * @throws IllegalArgumentException if <code>input</code> is an output (does
00332 * not support reading of values). See {@link IO#isInput()}.
00333 */
00334 void setDigitalInputValueToProceed(bool value_to_proceed_until);
00335 bool getDigitalInputValueToProceed();
00336
00337 /**
00338 * Creates an I/O Input configuration for an Until node. The until condition
00339 * for this configuration is true once the signal value of a given
00340 * analog input goes past a specified electric Current threshold.
00341 *
00342 * @param input the analog input whose signal value will determine whether
00343 * the until condition is met, not <code>null</code>.

```

```

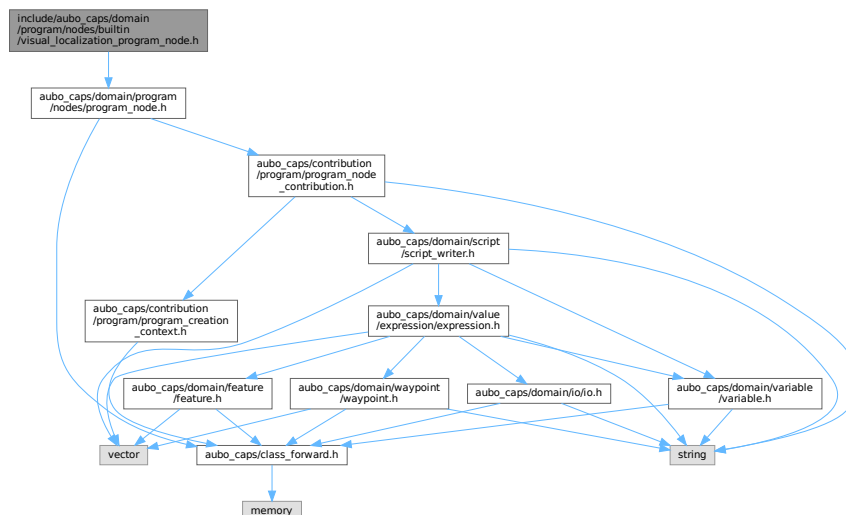
00344 * @param compareOperator the operator to use when comparing the signal
00345 * value of the selected analog input to the threshold value, not
00346 * <code>null</code>. Available options are "less than" ({@literal <}) or
00347 * "greater than"
00348 * ({@literal >}).
00349 * @param currentThreshold the electric Current threshold for the signal
00350 * value of the analog input. Execution will proceed until the signal value
00351 * of the selected analog input goes past this threshold.
00352 * @param errorHandler the error handler for validation errors. If using
00353 * {@link ErrorHandler#AUTO_CORRECT} this will clamp the value to the
00354 * nearest valid current value.
00355 * @return the Until node configuration.
00356 * @throws IllegalArgumentException if <code>input</code> is an output (does
00357 * not support reading of values). See {@link IO#isInput()}.
00358 */
00359 void setAnalogInputValueToProceed(double value);
00360 double getAnalogInputValueToProceed();
00361
00362 void setCompareOperator(CompareOperator compare_operator);
00363 CompareOperator getCompareOperator();
00364
00365 private:
00366 friend class DataSwitch;
00367 UntilNode();
00368 void *d_{ nullptr };
00369 };
00370
00371 // namespace aubo_scope
00372 // namespace arcs
00373 #endif // AUBO_SCOPE_UNTIL_NODE_H

```

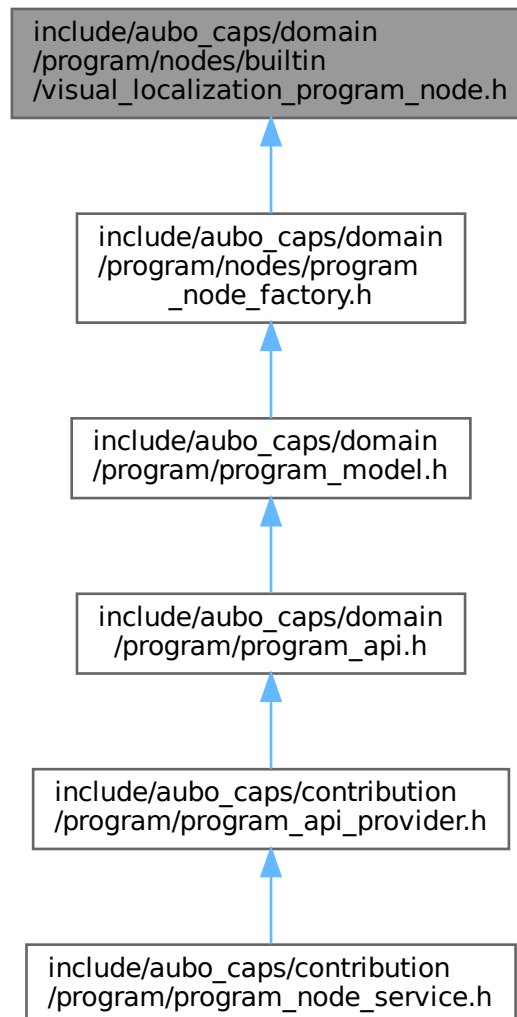
12.194 include/auco_caps/domain/program/nodes/builtin/visual_↵ localization_program_node.h File Reference

#include <auco_caps/domain/program/nodes/program_node.h>

Include dependency graph for visual_localization_program_node.h:



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



Classes

- class [arcs::aubo_scope::VisualLocalizationProgramNode](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([VisualLocalizationProgramNode](#))

12.195 visual_localization_program_node.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_VISUAL_LOCALIZATION_PROGRAM_NODE_H
00002 #define AUBO_SCOPE_VISUAL_LOCALIZATION_PROGRAM_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(VisualLocalizationProgramNode);
00009
00010 class ARCS_ABI_EXPORT VisualLocalizationProgramNode : public ProgramNode
00011 {
00012 public:
00013     VisualLocalizationProgramNode(VisualLocalizationProgramNode &f);
00014     VisualLocalizationProgramNode(VisualLocalizationProgramNode &&f);
00015     virtual ~VisualLocalizationProgramNode();
00016
00017 private:
00018     friend class DataSwitch;
00019     VisualLocalizationProgramNode();
00020     void *d_{ nullptr };
00021 };
00022
00023 } // namespace aubo_scope
00024 } // namespace arcs
00025
00026 #endif // AUBO_SCOPE_VISUAL_LOCALIZATION_PROGRAM_NODE_H

```

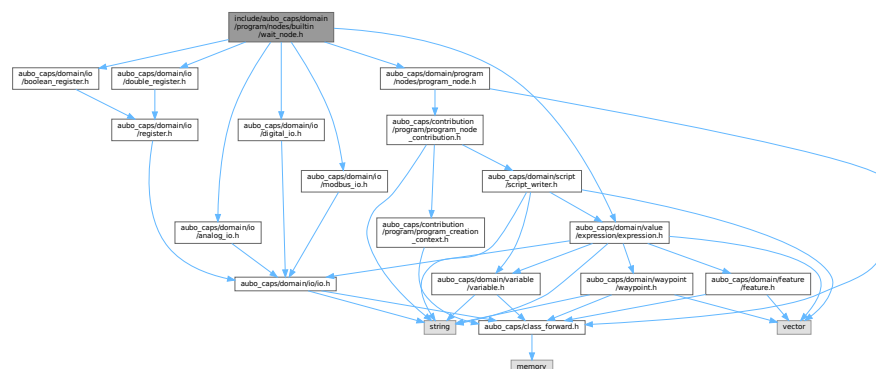
12.196 include/aubo_caps/domain/program/nodes/builtin/wait_node.h File Reference

```

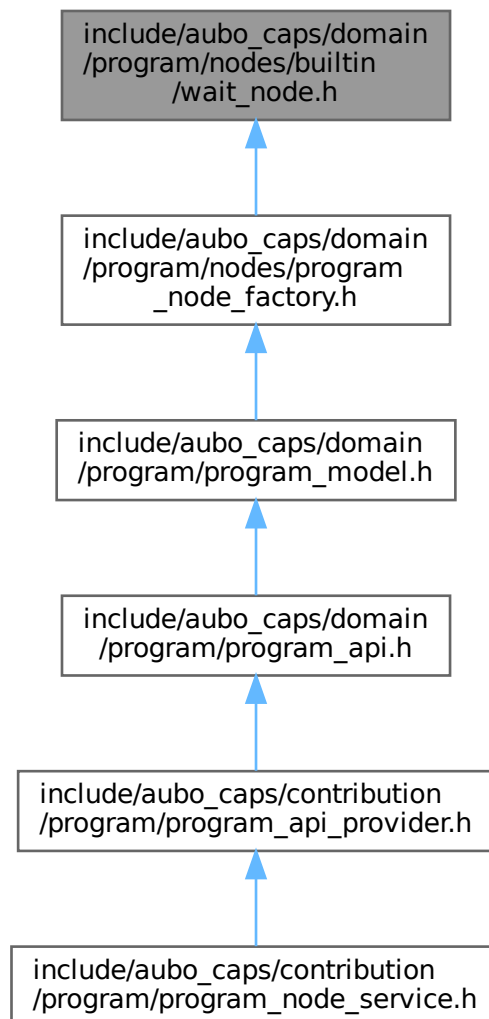
#include <aubo_caps/domain/program/nodes/program_node.h>
#include <aubo_caps/domain/io/analog_io.h>
#include <aubo_caps/domain/io/digital_io.h>
#include <aubo_caps/domain/io/modbus_io.h>
#include <aubo_caps/domain/io/boolean_register.h>
#include <aubo_caps/domain/io/double_register.h>
#include <aubo_caps/domain/value/expression/expression.h>

```

Include dependency graph for wait_node.h:



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



Classes

- class [arcs::aubo_scope::WaitNode](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(WaitNode\)](#)

12.197 wait_node.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_WAIT_NODE_H
00002 #define AUBO_SCOPE_WAIT_NODE_H
00003
00004 #include <aubo_caps/domain/program/nodes/program_node.h>
00005 #include <aubo_caps/domain/io/analog_io.h>
00006 #include <aubo_caps/domain/io/digital_io.h>
00007 #include <aubo_caps/domain/io/modbus_io.h>
00008 #include <aubo_caps/domain/io/boolean_register.h>
00009 #include <aubo_caps/domain/io/double_register.h>
00010 #include <aubo_caps/domain/value/expression/expression.h>
00011
00012 namespace arcs {
00013 namespace aubo_scope {
00014 ARCS_CLASS_FORWARD(WaitNode);
00015
00016 class ARCS_ABI_EXPORT WaitNode : public ProgramNode
00017 {
00018 public:
00019 /**
00020  * The configuration type used to determine which type of configuration this
00021  * instance is.
00022  */
00023 enum ConfigType : int
00024 {
00025     NO_TYPE,
00026
00027     /**
00028      * <p>
00029      * Nothing to wait.
00030      * </p>
00031      */
00032     NO_WAIT,
00033
00034     /**
00035      * <p>
00036      * Wait type is time.
00037      * </p>
00038      * The config instance can be cast to {@link TimeWaitNodeConfig}.
00039      */
00040     TIME,
00041
00042     /**
00043      * <p>
00044      * Wait type is digital_input.
00045      * </p>
00046      * The config instance can be cast to {@link
00047      * DigitalInputWaitNodeConfig}.
00048      */
00049     DIGITAL_INPUT,
00050
00051     /**
00052      * <p>
00053      * Wait type is analog_input_current.
00054      * </p>
00055      * The config instance can be cast to {@link
00056      * AnalogInputCurrentWaitNodeConfig}.
00057      */
00058     ANALOG_INPUT_CURRENT,
00059
00060     /**
00061      * <p>
00062      * Wait type is analog_input_voltage.
00063      * </p>
00064      * The config instance can be cast to {@link
00065      * AnalogInputVoltageWaitNodeConfig}.
00066      */
00067     ANALOG_INPUT_VOLTAGE,
00068
00069     /**
00070      * <p>
00071      * Wait type is float_register_input.
00072      * </p>
00073      * The config instance can be cast to {@link
00074      * FloatRegisterInputWaitNodeConfig}.
00075      */
00076
00077
00078
00079
00080
00081
00082

```



```

00083     */
00084     FLOAT_REGISTER_INPUT,
00085
00086     /**
00087     * <p>
00088     * Wait type is expression_input.
00089     * </p>
00090     *
00091     * The config instance can be cast to {@link
00092     * ExpressionInputWaitNodeConfig}.
00093     */
00094     EXPRESSION_INPUT
00095 };
00096
00097 /**
00098 * Compare operator types LESS_THAN and GREATER_THAN used when waiting for
00099 * an analog input to go lower or higher, respectively, than a threshold
00100 */
00101 enum CompareOperator : int
00102 {
00103     LESS_THAN,
00104     GREATER_THAN
00105 };
00106
00107 WaitNode(WaitNode &f);
00108 WaitNode(WaitNode &&f);
00109 virtual ~WaitNode();
00110
00111 /**
00112 * This method returns the type of configuration. Cast this instance
00113 * appropriately to have access to specific getters.
00114 *
00115 * @return the type of this config.
00116 */
00117 ConfigType getConfigType();
00118 void setConfigType(ConfigType type);
00119
00120 /**
00121 *
00122 * @return the time to wait for. second
00123 */
00124 double getWaitTime();
00125 void setWaitTime(double time);
00126
00127 /**
00128 * Depending on the type of input this is interpreted in different ways.
00129 * For digital inputs and MODBUS inputs, <code>true</code> means the input
00130 * to wait for must be HIGH, <code>false</code> means the input to wait for
00131 * must be LOW. For boolean registers the value is interpreted literally.
00132 *
00133 * @return the input value to wait for.
00134 */
00135 bool getBoolValueToWaitFor();
00136 void setBoolValueToWaitFor(bool value);
00137
00138 float getFloatValueToWaitFor();
00139 void setFloatValueToWaitFor(float value);
00140
00141 /**
00142 * Creates a configuration for waiting for a digital input to go high or
00143 * low.
00144 *
00145 * @param input the digital input on which to wait, not <code>null</code>.
00146 * @param value_to_wait_for <code>true</code>, if waiting for the input
00147 * signal value/level to go HIGH, <code>false</code> otherwise (waiting for
00148 * LOW signal value/level).
00149 * @return the configuration.
00150 * @throws IllegalArgumentException if <code>input</code> is an output (does
00151 * not support reading of values). See {@link IO#isInput()}.
00152 */
00153 void setDigitalInput(DigitalIoPtr input);
00154 DigitalIoPtr getDigitalInput();
00155
00156 /**
00157 * Creates a configuration for waiting for a digital MODBUS input to go high
00158 * or low.
00159 *
00160 * @param input the digital MODBUS input on which to wait, not
00161 * <code>null</code>.
00162 * @param value_to_wait_for <code>true</code>, if waiting for the input
00163 * signal value/level to go HIGH, <code>false</code> otherwise (waiting for
00164 * LOW signal value/level).
00165 * @return the configuration.
00166 * @throws IllegalArgumentException if <code>input</code> is an output (does
00167 * not support reading of values) or if <code>output</code> is not a digital
00168 * output. See {@link IO#isInput()} and {@link IO#getType()}.
00169 */

```

```

00170 void setModbusInput(ModbusIoPtr input);
00171 ModbusIoPtr getModbusInput();
00172
00173 /**
00174  * Creates a configuration for waiting for a boolean register to go
00175  * <code>true</code> or <code>>false</code>.
00176  *
00177  * @param input the boolean register input on which to wait, not
00178  * <code>null</code>.
00179  * @param value_to_wait_for <code>true</code>, if waiting for the register
00180  * to be <code>true</code>, <code>>false</code> otherwise.
00181  * @return the configuration.
00182  * @throws IllegalArgumentException if <code>input</code> is an output (does
00183  * not support reading of values). See {@link IO#isInput()}.
00184  */
00185 void setBoolRegister(BoolRegisterPtr input);
00186 BoolRegisterPtr getBoolRegister();
00187
00188 void setCompareOperator(CompareOperator compareOperator);
00189 CompareOperator getCompareOperator();
00190
00191 /**
00192  * Creates a configuration for waiting for an analog input to go past an
00193  * electric current threshold.
00194  *
00195  * @param input on which to wait, not <code>null</code>.
00196  * @param compareOperator the operator to use when comparing input to value,
00197  * not <code>null</code>. Available options are "less than" ({@literal <})
00198  * or "greater than" ({@literal >}).
00199  * @param current the electric current threshold to wait for, not
00200  * <code>null</code>.
00201  * @param errorHandler the error handler for validation errors. If using
00202  * {@link ErrorHandler#AUTO_CORRECT} this will clamp the value to the
00203  * nearest valid current value.
00204  * @return the configuration.
00205  * @throws IllegalArgumentException if <code>input</code> is an output (does
00206  * not support reading of values). See {@link IO#isInput()}.
00207  */
00208 void setAnalogInput(AnalogIoPtr input);
00209 AnalogIoPtr getAnalogInput();
00210
00211 /**
00212  * Creates a configuration for waiting for a register to go past a float
00213  * threshold.
00214  *
00215  * @param input the float register input on which to wait, not
00216  * <code>null</code>.
00217  * @param compareOperator the operator to use when comparing register to
00218  * value, not <code>null</code>. Available options are "less than"
00219  * ({@literal <}) or "greater than" ({@literal >}).
00220  * @param value_to_wait_for the float threshold to wait for.
00221  * @return the configuration.
00222  * @throws IllegalArgumentException if <code>input</code> is an output (does
00223  * not support reading of values). See {@link IO#isInput()}.
00224  */
00225 void setDoubleRegisterInput(DoubleRegisterPtr input);
00226 DoubleRegisterPtr getDoubleRegisterInput();
00227
00228 /**
00229  * Create a configuration for waiting for an expression to be evaluated to
00230  * true.
00231  *
00232  * @param expression the expression to evaluate, not <code>null</code>.
00233  * @return the configuration.
00234  */
00235 void setExpressionInputConfig(ExpressionPtr expression);
00236 ExpressionPtr getExpressionInputConfig();
00237
00238 private:
00239 friend class DataSwitch;
00240 WaitNode();
00241 void *d_{ nullptr };
00242 };
00243
00244 } // namespace aubo_scope
00245 } // namespace arcs
00246 #endif // AUBO_SCOPE_WAIT_NODE_H

```

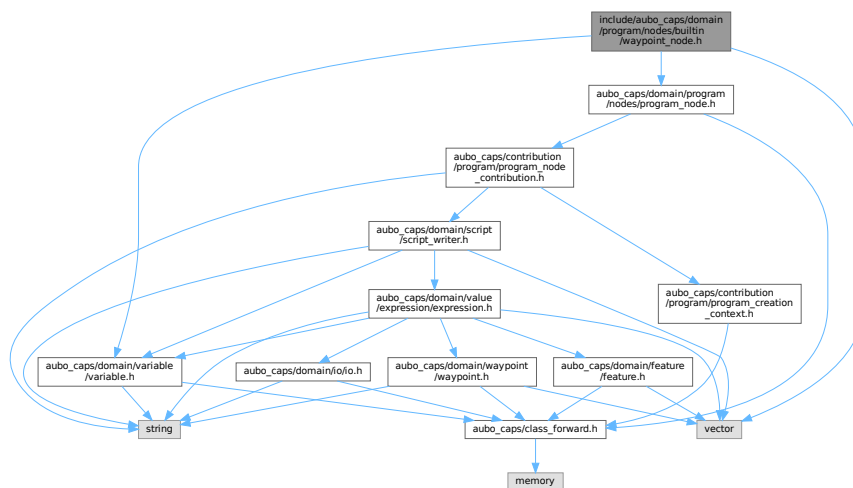
12.198 include/aubo_caps/domain/program/nodes/builtin/waypoint_node.h File Reference

```
#include <vector>
```

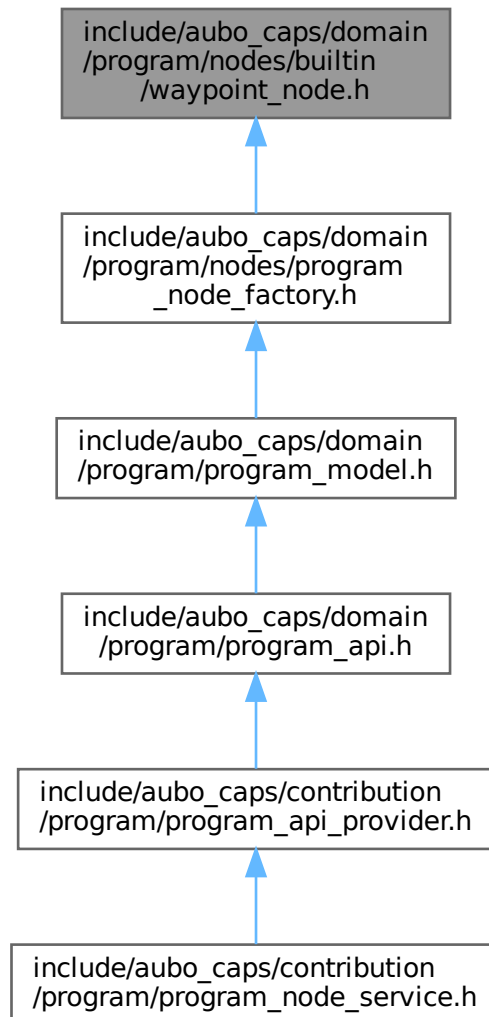
```
#include <aubo_caps/domain/program/nodes/program_node.h>
```

```
#include <aubo_caps/domain/variable/variable.h>
```

Include dependency graph for waypoint_node.h:



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



Classes

- class [arcs::aubo_scope::WaypointNode](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([WaypointNode](#))

12.199 waypoint_node.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_WAYPOINT_NODE_H
00002 #define AUBO_SCOPE_WAYPOINT_NODE_H
00003
00004 #include <vector>
00005 #include <aubo_caps/domain/program/nodes/program_node.h>
00006 #include <aubo_caps/domain/variable/variable.h>
00007
00008 namespace arcs {
00009 namespace aubo_scope {
00010
00011 ARCS_CLASS_FORWARD(WaypointNode);
00012
00013 class ARCS_ABI_EXPORT WaypointNode : public ProgramNode
00014 {
00015 public:
00016     /**
00017      * The configuration type used to determine which type of configuration this
00018      * instance is.
00019      */
00020     enum ConfigType : int
00021     {
00022         /**
00023          * <p>
00024          * Fixed position has been selected (fixed waypoint).
00025          * </p>
00026          *
00027          * The config instance can be cast to {@link
00028          * FixedPositionWaypointNodeConfig}.
00029          */
00030         FIXED_POSITION = 0,
00031
00032         /**
00033          * <p>
00034          * Variable position has been selected (variable waypoint).
00035          * </p>
00036          *
00037          * The config instance can be cast to {@link
00038          * VariablePositionWaypointNodeConfig}.
00039          */
00040         VARIABLE_POSITION,
00041
00042         /**
00043          * Relative position has been selected (relative waypoint).
00044          */
00045         RELATIVE_POSITION
00046     };
00047
00048     /**
00049      * The blend parameters type used to determine which type of blend
00050      * parameters this instance is.
00051      */
00052     enum BlendType : int
00053     {
00054         /**
00055          * No blend has been selected (the robot arm will stop at this point).
00056          */
00057         NO_BLEND = 0,
00058
00059         /**
00060          * Shared blend has been selected.
00061          */
00062         SHARED_BLEND,
00063
00064         /**
00065          * <p>
00066          * Waypoint blend has been selected.
00067          * </p>
00068          *
00069          * The blend instance can be cast to {@link WaypointBlendParameters}.
00070          */
00071         WAYPOINT_BLEND
00072     };
00073
00074     /**
00075      * The blend parameters type used to determine which type of blend
00076      * parameters this instance is.
00077      */
00078     enum MotionType : int
00079     {
00080         /**
00081          * <p>
00082          * Time motion parameters has been selected.

```

```

00083     * </p>
00084     *
00085     * This instance can be cast to {@link TimeMotionParameters}.
00086     */
00087     TIME_MOTION,
00088
00089     /**
00090     * <p>
00091     * Joint motion parameters has been selected.
00092     * </p>
00093     *
00094     * This instance can be cast to {@link JointMotionParameters}.
00095     */
00096     JOINT_MOTION,
00097
00098     /**
00099     * <p>
00100     * Tool motion parameters has been selected.
00101     * </p>
00102     *
00103     * This instance can be cast to {@link ToolMotionParameters}.
00104     */
00105     TOOL_MOTION,
00106
00107     /**
00108     * Shared motion parameters has been selected.
00109     */
00110     SHARED_MOTION
00111 };
00112
00113 enum VariableType : int
00114 {
00115     /**
00116     * This variable type pose parameters has been selected.
00117     */
00118     VARIABLE_POSE,
00119
00120     /**
00121     * This variable type joint parameters has been selected.
00122     */
00123     VARIABLE_JOINT
00124 };
00125
00126 enum RelativePositionType : int
00127 {
00128     /**
00129     * The teaching location has been selected.
00130     */
00131     TEACHING,
00132
00133     /**
00134     * The input offset has been selected.
00135     */
00136     INPUT
00137 };
00138
00139 enum RelativeOffsetType : int
00140 {
00141     /**
00142     * Custom offset has been selected.
00143     */
00144     CUSTOM,
00145
00146     /**
00147     * Variable type has been selected.
00148     */
00149     VARIABLE
00150 };
00151
00152 enum class PositionDefinition : int
00153 {
00154     /**
00155     * The position has not been defined.
00156     */
00157     UNDEFINED,
00158
00159     /**
00160     * <p>
00161     * The position of the waypoint has been defined.
00162     * </p>
00163     *
00164     * Cast this instance to {@link FixedPositionDefinedWaypointNodeConfig}.
00165     */
00166     DEFINED
00167 };
00168
00169 enum VariableDefinition : int

```

```

00170 {
00171     /**
00172      * A variable has not been selected. This instance has no further
00173      * information.
00174      */
00175     UNDEFINED,
00176
00177     /**
00178      * <p>
00179      * A variable has been selected making this config defined.
00180      * </p>
00181      *
00182      * Cast this instance to {@link
00183      * VariablePositionDefinedWaypointNodeConfig}.
00184      */
00185     DEFINED
00186 };
00187
00188 WaypointNode(WaypointNode &f);
00189 WaypointNode(WaypointNode &&f);
00190 virtual ~WaypointNode();
00191
00192 /**
00193  * This method returns the type of configuration. Cast this instance
00194  * appropriately to have access to specific getters.
00195  *
00196  * @return the type of this config.
00197  */
00198 ConfigType getConfigType();
00199
00200 /**
00201  * Get blend type of the waypoint
00202  * @return the blend type.
00203  */
00204 BlendType getBlendType();
00205
00206 MotionType getMotionType();
00207
00208 /**
00209  * Cast this instance appropriately to have access to specific getters.
00210  *
00211  * @return the position definition.
00212  */
00213 PositionDefinition getPositionDefinition();
00214
00215 /**
00216  * Cast this instance appropriately to have access to specific getters.
00217  *
00218  * @return the variable definition.
00219  */
00220 VariableDefinition getVariableDefinition();
00221
00222 /**
00223  * <p>
00224  * Get the blend to be used for the movement.
00225  * </p>
00226  *
00227  * Note that, if the configuration has not been applied to a waypoint node
00228  * in the program tree, the blend value returned is a "suggested" value,
00229  * since the blend may be (automatically) corrected depending on surrounding
00230  * waypoints in the program tree (when it is applied to a waypoint node).
00231  *
00232  * @return the blend for the waypoint.
00233  */
00234 double getBlend();
00235
00236 /**
00237  * @return the joint speed to be achieved.
00238  */
00239 double getJointSpeed();
00240
00241 /**
00242  * @return the joint acceleration to use for the movement.
00243  */
00244 double getJointAcceleration();
00245
00246 /**
00247  * @return the time for the waypoint movement.
00248  */
00249 double getTime();
00250
00251 /**
00252  * @return the tool speed to be achieved.
00253  */
00254 double getToolSpeed();
00255
00256 /**

```

```

00257     * @return the tool acceleration to use for the movement
00258     */
00259 double getToolAcceleration();
00260
00261 /**
00262  * @return the waypoint position and orientation.
00263  */
00264 std::vector<double> getPose();
00265
00266 /**
00267  * @return the joint positions corresponding to the pose of the robot (when
00268  * taking the TCP offset into account)
00269  */
00270 std::vector<double> getJointPositions();
00271
00272 /**
00273  * @return the pose for the TCP offset (used when defining the robot
00274  * position)
00275  */
00276 std::vector<double> getTcpOffset();
00277
00278 /**
00279  * @return the variable for the waypoint.
00280  */
00281 VariablePtr getVariable();
00282
00283 /**
00284  * <p>
00285  * Creates shared motion parameters for a waypoint.
00286  * </p>
00287  *
00288  * Can be used for waypoints under a MoveJ, MoveL or a MoveP.
00289  *
00290  * @return the motion parameters for the waypoint.
00291  */
00292 void setSharedMotionParameters();
00293
00294 /**
00295  * <p></p>
00296  * Creates time motion parameters for a waypoint.
00297  * </p>
00298  *
00299  * Can be used for waypoints under a MoveJ or a MoveL.
00300  *
00301  * @param duration the time the movement should take, not <code>null</code>.
00302  * @param errorHandler error handler for handling validation. If using
00303  * {&@link ErrorHandler#AUTO_CORRECT} this will clamp the value to the
00304  * nearest valid time value.
00305  * @return the time motion parameters for the waypoint.
00306  */
00307 void setTimeMotionParameters(unsigned int duration);
00308
00309 /**
00310  * <p>
00311  * Creates joint motion parameters for a waypoint.
00312  * </p>
00313  *
00314  * Can be used for waypoints under a MoveJ.
00315  *
00316  * @param jointSpeed the joint speed to be achieved, not <code>null</code>.
00317  * @param jointSpeedErrorHandler error handler for handling validation. If
00318  * using {&@link ErrorHandler#AUTO_CORRECT} this will clamp the value to the
00319  * nearest valid joint speed value.
00320  * @param joint_acceleration the joint acceleration to be used for the
00321  * movement, not <code>null</code>.
00322  * @param joint_accelerationErrorHandler error handler for handling
00323  * validation. If using {&@link ErrorHandler#AUTO_CORRECT} this will clamp
00324  * the value to the nearest valid joint acceleration value.
00325  * @return the joint motion parameters for the waypoint.
00326  */
00327 void setJointMotionParameters(double joint_speed,
00328                               double joint_acceleration);
00329
00330 /**
00331  * <p>
00332  * Creates tool motion parameters for a waypoint.
00333  * </p>
00334  *
00335  * Can be used for waypoints under a MoveL or a MoveP.
00336  *
00337  * @param tool_speed the tool speed to be achieved, not <code>null</code>.
00338  * @param tool_speedErrorHandler error handler for handling validation. If
00339  * using {&@link ErrorHandler#AUTO_CORRECT} this will clamp the value to the
00340  * nearest valid tool speed value.
00341  * @param tool_acceleration the tool acceleration to be used for the
00342  * movement, not <code>null</code>.
00343  * @param tool_accelerationErrorHandler error handler for handling

```



```

00344 * validation. If using {@link ErrorHandler#AUTO_CORRECT} this will clamp
00345 * the value to the nearest valid tool acceleration value.
00346 * @return the tool motion parameters for the waypoint.
00347 */
00348 void setToolMotionParameters(double tool_speed, double tool_acceleration);
00349
00350 /**
00351 * <p>
00352 * Creates a fixed position configuration with default values for a
00353 * waypoint.
00354 * </p>
00355 *
00356 * Note that these default values may change between AuboScope versions.
00357 *
00358 * @return the waypoint configuration.
00359 */
00360 void setFixedPositionConfig();
00361
00362 /**
00363 * <p>
00364 * Creates a fixed position configuration for a waypoint with a defined
00365 * pose.
00366 * </p>
00367 *
00368 * @param pose_including_tcp the position and orientation for the waypoint,
00369 * not <code>null</code>.
00370 * @param qNear the joint positions that will be used for the inverse
00371 * kinematics calculations as a robot configuration near the waypoint pose,
00372 * not <code>null</code>.
00373 * @param blendParameters the blend parameters for the waypoint, not
00374 * <code>null</code>.
00375 * @param waypoint_motion_parameters the motion parameters for the waypoint,
00376 * not <code>null</code>.
00377 * @return the waypoint configuration.
00378 */
00379 void setFixedPositionConfig(const std::vector<double> &pose_including_tcp,
00380                             const std::vector<double> &qNear);
00381
00382 /**
00383 * <p>
00384 * Creates a variable position configuration with default values for a
00385 * waypoint
00386 * </p>
00387 *
00388 * Note that these default values may change between AuboScope versions.
00389 *
00390 * @return the waypoint configuration.
00391 */
00392 void setVariablePositionConfig();
00393
00394 /**
00395 * Creates a variable position configuration for a waypoint with a defined
00396 * variable
00397 *
00398 * @param variable the variable position and orientation for the waypoint,
00399 * not <code>null</code>.
00400 * @param blendParameters the blend parameters for the waypoint, not
00401 * <code>null</code>.
00402 * @param waypoint_motion_parameters the motion parameters for the waypoint,
00403 * not <code>null</code>.
00404 * @return the waypoint configuration.
00405 */
00406 void setVariablePositionConfig(VariablePtr variable);
00407
00408 private:
00409 friend class DataSwitch;
00410 WaypointNode();
00411 void *d_{ nullptr };
00412 };
00413
00414 } // namespace aubo_scope
00415 } // namespace arcs
00416 #endif // AUBO_SCOPE_WAYPOINT_NODE_H

```

12.200 include/aubo_caps/domain/program/nodes/program_node.h File Reference

```

#include <aubo_caps/class_forward.h>
#include <aubo_caps/contribution/program/program_node_contribution.h>

```



```

00012 * \chinese
00013 *
00014 *   AuboScope
00015 *
00016 *   {@link ProgramNodeFactory}
00017 * \endchinese
00018 * \english
00019 * ProgramNode
00020 * This interface represents AuboScope program nodes.
00021 *
00022 * The program nodes can be created using {@link ProgramNodeFactory}.
00023 * \endenglish
00024 */
00025 class ARCS_ABI_EXPORT ProgramNode
00026 {
00027 public:
00028     ProgramNode(ProgramNode &f);
00029     ProgramNode(ProgramNode &&f);
00030     virtual ~ProgramNode();
00031
00032     /**
00033     * \chinese
00034     *   ProgramNodeContribution::generateScript
00035     *   ScriptWriter::writeChildren
00036     * @return
00037     * \endchinese
00038     * \english
00039     * Get the program node contribution. Not recommended to call
00040     * ProgramNodeContribution::generateScript directly, use
00041     * ScriptWriter::writeChildren instead.
00042     * @return the ProgramNodeContribution
00043     * \endenglish
00044     */
00045     [[deprecated]] ProgramNodeContributionPtr getProgramNodeContribution();
00046
00047 protected:
00048     ProgramNode();
00049
00050 private:
00051     friend class DataSwitch;
00052     void *d_{ nullptr };
00053 };
00054
00055 } // namespace aubo_scope
00056 } // namespace arcs
00057
00058 #endif // AUBO_SCOPE_PROGRAM_NODE_H

```

12.202 include/aubo_caps/domain/program/nodes/program_node_factory.h File Reference

```

#include <aubo_caps/domain/program/nodes/builtin/assignment_node.h>
#include <aubo_caps/domain/program/nodes/builtin/circle_move_node.h>
#include <aubo_caps/domain/program/nodes/builtin/comment_node.h>
#include <aubo_caps/domain/program/nodes/builtin/direction_node.h>
#include <aubo_caps/domain/program/nodes/builtin/folder_node.h>
#include <aubo_caps/domain/program/nodes/builtin/force_node.h>
#include <aubo_caps/domain/program/nodes/builtin/force_option_node.h>
#include <aubo_caps/domain/program/nodes/builtin/halt_node.h>
#include <aubo_caps/domain/program/nodes/builtin/program_home_node.h>
#include <aubo_caps/domain/program/nodes/builtin/if_node.h>
#include <aubo_caps/domain/program/nodes/builtin/else_if_node.h>
#include <aubo_caps/domain/program/nodes/builtin/else_node.h>
#include <aubo_caps/domain/program/nodes/builtin/loop_node.h>
#include <aubo_caps/domain/program/nodes/builtin/move_node.h>
#include <aubo_caps/domain/program/nodes/builtin/pallet_node.h>
#include <aubo_caps/domain/program/nodes/builtin/popup_node.h>
#include <aubo_caps/domain/program/nodes/builtin/timer_node.h>
#include <aubo_caps/domain/program/nodes/builtin/screw_driving_node.h>
#include <aubo_caps/domain/program/nodes/builtin/script_node.h>
#include <aubo_caps/domain/program/nodes/builtin/seek_node.h>

```

```

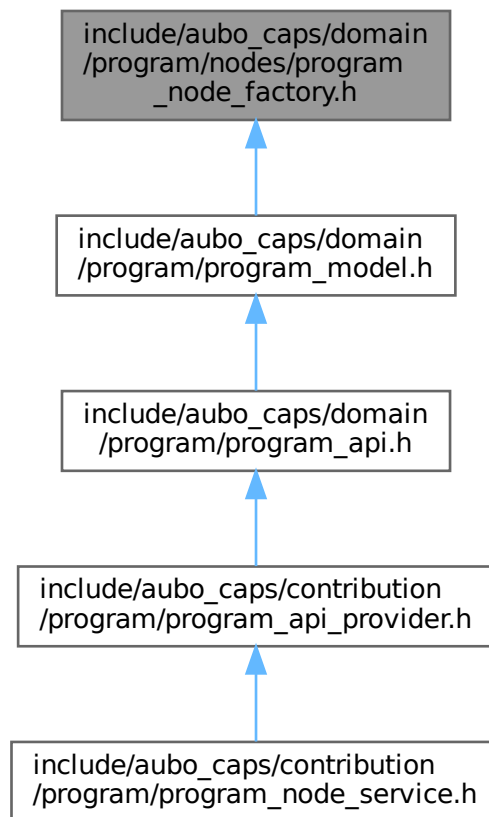
#include <aubo_caps/domain/program/nodes/builtin/set_node.h>
#include <aubo_caps/domain/program/nodes/builtin/set_payload_node.h>
#include <aubo_caps/domain/program/nodes/builtin/until_node.h>
#include <aubo_caps/domain/program/nodes/builtin/wait_node.h>
#include <aubo_caps/domain/program/nodes/builtin/waypoint_node.h>
#include <aubo_caps/domain/program/nodes/builtin/switch_node.h>
#include <aubo_caps/domain/program/nodes/builtin/case_node.h>
#include <aubo_caps/domain/program/nodes/builtin/default_case_node.h>
#include <aubo_caps/domain/program/nodes/builtin/break_node.h>
#include <aubo_caps/domain/program/nodes/builtin/sub_prog_node.h>
#include <aubo_caps/domain/program/nodes/builtin/call_node.h>
#include <aubo_caps/domain/program/nodes/builtin/thread_node.h>
#include <aubo_caps/domain/program/nodes/builtin/offline_track_node.h>
#include <aubo_caps/domain/program/nodes/builtin/record_track_node.h>
#include <aubo_caps/domain/program/nodes/builtin/gripper_node.h>
#include <aubo_caps/domain/program/nodes/builtin/assembly_node.h>
#include <aubo_caps/domain/program/nodes/builtin/visual_localization_program_node.h>
#include <aubo_caps/domain/program/nodes/builtin/conveyor_tracking_node.h>

```

Include dependency graph for program_node_factory.h:



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



Classes

- class [arcs::aubo_scope::ProgramNodeFactory](#)
[ProgramNodeFactory](#) This interfaces supplies methods to create different types of program nodes.

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(ProgramNodeFactory\)](#)
- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(ProgramNodeService\)](#)

12.203 program_node_factory.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_PROGRAM_NODE_FACTORY_H
00002 #define AUBO_SCOPE_PROGRAM_NODE_FACTORY_H
00003
00004 #include <aubo_caps/domain/program/nodes/builtin/assignment_node.h>
00005 #include <aubo_caps/domain/program/nodes/builtin/circle_move_node.h>
00006 #include <aubo_caps/domain/program/nodes/builtin/comment_node.h>
00007 #include <aubo_caps/domain/program/nodes/builtin/direction_node.h>
00008 #include <aubo_caps/domain/program/nodes/builtin/folder_node.h>
00009 #include <aubo_caps/domain/program/nodes/builtin/force_node.h>
00010 #include <aubo_caps/domain/program/nodes/builtin/force_option_node.h>
00011 #include <aubo_caps/domain/program/nodes/builtin/halt_node.h>
00012 #include <aubo_caps/domain/program/nodes/builtin/program_home_node.h>
00013 #include <aubo_caps/domain/program/nodes/builtin/if_node.h>
00014 #include <aubo_caps/domain/program/nodes/builtin/else_if_node.h>
00015 #include <aubo_caps/domain/program/nodes/builtin/else_node.h>
00016 #include <aubo_caps/domain/program/nodes/builtin/loop_node.h>
00017 #include <aubo_caps/domain/program/nodes/builtin/move_node.h>
00018 #include <aubo_caps/domain/program/nodes/builtin/pallet_node.h>
00019 #include <aubo_caps/domain/program/nodes/builtin/popup_node.h>
00020 #include <aubo_caps/domain/program/nodes/builtin/timer_node.h>
00021 #include <aubo_caps/domain/program/nodes/builtin/screw_driving_node.h>
00022 #include <aubo_caps/domain/program/nodes/builtin/script_node.h>
00023 #include <aubo_caps/domain/program/nodes/builtin/seek_node.h>
00024 #include <aubo_caps/domain/program/nodes/builtin/set_node.h>
00025 #include <aubo_caps/domain/program/nodes/builtin/set_payload_node.h>
00026 #include <aubo_caps/domain/program/nodes/builtin/until_node.h>
00027 #include <aubo_caps/domain/program/nodes/builtin/wait_node.h>
00028 #include <aubo_caps/domain/program/nodes/builtin/waypoint_node.h>
00029 #include <aubo_caps/domain/program/nodes/builtin/switch_node.h>
00030 #include <aubo_caps/domain/program/nodes/builtin/case_node.h>
00031 #include <aubo_caps/domain/program/nodes/builtin/default_case_node.h>
00032 #include <aubo_caps/domain/program/nodes/builtin/break_node.h>
00033 #include <aubo_caps/domain/program/nodes/builtin/sub_prog_node.h>
00034 #include <aubo_caps/domain/program/nodes/builtin/call_node.h>
00035 #include <aubo_caps/domain/program/nodes/builtin/thread_node.h>
00036 #include <aubo_caps/domain/program/nodes/builtin/offline_track_node.h>
00037 #include <aubo_caps/domain/program/nodes/builtin/record_track_node.h>
00038 #include <aubo_caps/domain/program/nodes/builtin/gripper_node.h>
00039 #include <aubo_caps/domain/program/nodes/builtin/assembly_node.h>
00040 #include <aubo_caps/domain/program/nodes/builtin/visual_localization_program_node.h>
00041 #include <aubo_caps/domain/program/nodes/builtin/conveyor_tracking_node.h>
00042 #include <aubo_caps/domain/program/nodes/builtin/gripper_node.h>
00043
00044 namespace arcs {
00045 namespace aubo_scope {
00046 ARCS_CLASS_FORWARD(ProgramNodeFactory);
00047 ARCS_CLASS_FORWARD(ProgramNodeService);
00048
00049 /**
00050  * \chinese
00051  *
00052  *
00053  *
00054  *      AUBO Robots      AuboCap
00055  *
00056  *
00057  *      AuboScope
00058  * \endchinese
00059  * \english
00060  * ProgramNodeFactory
00061  * This interfaces supplies methods to create different types of program nodes.
00062  *
00063  * Both built-in program nodes (provided by AUBO Robots) and AuboCap
00064  * program nodes can be created. Most methods will create a node with the same
00065  * default configuration as if the end user added the node manually.
00066  *
00067  * Note that this default configuration may change between AuboScope versions.
00068  * \endenglish
00069  */
00070 class ARCS_ABI_EXPORT ProgramNodeFactory
00071 {
00072 public:
00073     ProgramNodeFactory(ProgramNodeFactory &f);
00074     ProgramNodeFactory(ProgramNodeFactory &&f);
00075     virtual ~ProgramNodeFactory();
00076
00077     /**
00078      * \chinese
00079      *      AuboCap      ProgramNodeContribution
00080      *
00081      * @param typeid_service
00082      * @return ProgramNodeService      ProgramNodeContribution

```

```

00083 * ProgramNode   ProgramNodeService  AuboCap
00084 * <code>null</code>
00085 * @throws IllegalArgumentException   ProgramNodeService
00086 * <code>null</code>   ProgramNodeService
00087 * \endchinese
00088 * \english
00089 * This method creates a AuboCap program node which is an instance of a
00090 * {@link contribution/ProgramNodeContribution}.
00091 *
00092 * @param typeid_service is the service creating the program node
00093 * contribution
00094 * @return the {@link ProgramNode} which represents the {@link
00095 * contribution/ProgramNodeContribution} created by {@link
00096 * ProgramNodeService}. Returns <code>null</code> if AuboCap providing
00097 * ProgramNodeService is not installed.
00098 * @throws IllegalArgumentException if the argument ProgramNodeService
00099 * is <code>null</code> or ProgramNodeService does not implement {@link
00100 * ProgramNodeService}.
00101 * \endenglish
00102 */
00103 ProgramNodePtr createUserProgramNode(const char *typeid_service);
00104
00105 /**
00106 * \chinese
00107 *
00108 * @return
00109 * \endchinese
00110 * \english
00111 * Creates a default Move node with one default Waypoint node under it.
00112 * @return new Move Node.
00113 * \endenglish
00114 */
00115 MoveNodePtr createMoveNode();
00116
00117 /**
00118 * \chinese
00119 * <p>                </p>
00120 *
00121 * @return
00122 * \endchinese
00123 * \english
00124 * <p>Creates a default Move node without a Waypoint node under it (e.g. to be
00125 * used for inserting named waypoints).</p>
00126 * Note that at least one Waypoint node must be under it for a Move node to
00127 * be valid.
00128 * @return new Move Node.
00129 * \endenglish
00130 */
00131 MoveNodePtr createMoveNodeNoTemplate();
00132
00133 /**
00134 * \chinese
00135 *
00136 * @return
00137 * \endchinese
00138 * \english
00139 * Creates a Waypoint node with default waypoint name (corresponds to the
00140 * name generated if the end user added a new node manually).
00141 * @return new Waypoint Node.
00142 * \endenglish
00143 */
00144 WaypointNodePtr createWaypointNode();
00145
00146 /**
00147 * \chinese
00148 * <p>                </p>
00149 * <p>                </p>
00150 *
00151 *
00152 * @param suggested_name
00153 * [a-zA-Z][a-zA-Z0-9_]{0,14}
00154 * @return
00155 * @throws IllegalWaypointNameException
00156 * \endchinese
00157 * \english
00158 * <p>Creates a Waypoint node with a suggested name.</p>
00159 * <p>The final name can be different if the suggested name is already
00160 * registered. The waypoint name is registered when it is inserted into the
00161 * program tree.</p>
00162 * Variable waypoints will use the name of the variable instead of the
00163 * suggested name.
00164 *
00165 * @param suggested_name the suggested name of the waypoint. Valid names
00166 * must match regex [a-zA-Z][a-zA-Z0-9_]{0,14} for a total of 15 characters.
00167 * @return new Waypoint Node.
00168 * @throws IllegalWaypointNameException In case of an illegal name.
00169 * \endenglish

```

```

00170     */
00171     WaypointNodePtr createWaypointNode(const std::string &suggested_name);
00172
00173     /**
00174     * \chinese
00175     *     until    Until
00176     * @return    Until
00177     * \endchinese
00178     * \english
00179     * Creates an Until Node with no until type specified.
00180     * @return a new Until Node.
00181     * \endenglish
00182     */
00183     UntilNodePtr createUntilNode();
00184     DirectionNodePtr createDirectionNode();
00185     WaitNodePtr createWaitNode();
00186     SetNodePtr createSetNode();
00187     SetPayloadNodePtr createSetPayloadNode();
00188     PopupNodePtr createPopupNode();
00189     HaltNodePtr createHaltNode();
00190     CommentNodePtr createCommentNode();
00191     FolderNodePtr createFolderNode();
00192     LoopNodePtr createLoopNode();
00193     AssignmentNodePtr createAssignmentNode();
00194     IfNodePtr createIfNode();
00195     ElseIfNodePtr createElseIfNode();
00196     ElseNodePtr createElseNode();
00197     CircleMoveNodePtr createCircleMoveNode();
00198     PalletNodePtr createPalletNode();
00199     SeekNodePtr createSeekNode();
00200     ForceNodePtr createForceNode();
00201     ForceOptionNodePtr createForceOptionNode();
00202     AssemblyNodePtr createAssemblyNode();
00203     ScrewdrivingNodePtr createScrewdrivingNode();
00204     ScriptNodePtr createScriptNode();
00205     TimerNodePtr createTimerNode();
00206     SwitchNodePtr createSwitchNode();
00207     CaseNodePtr createCaseNode();
00208     DefaultCaseNodePtr createDefaultCaseNode();
00209     BreakNodePtr createBreakNode();
00210     SubProgNodePtr createSubProgNode();
00211     CallNodePtr createCallNode();
00212     OfflineTrackNodePtr createOfflineTrackNode();
00213     RecordTrackNodePtr createRecordTrackNode();
00214     ThreadNodePtr createThreadNode();
00215     ProgramHomeNodePtr createProgramHomeNode();
00216     VisualLocalizationProgramNodePtr createVisualLocalizationProgramNode();
00217     ConveyorTrackingNodePtr createConveyorTrackingNode();
00218
00219     /**
00220     * \chinese
00221     * <p>                </p>
00222     *     AuboScope
00223     *
00224     *
00225     * @param gripperDevice    <code>null</code>
00226     *
00227     * @return    TreeNode
00228     * @throws UnresolvedDeviceException    gripperDevice
00229     * \endchinese
00230     * \english
00231     * <p>
00232     * This method creates a new Gripper program node. This node type can be
00233     * used for programming grip and release actions with a selected gripper
00234     * device.
00235     * </p>
00236     * The program node will be created with the same default configuration as
00237     * if the end user added the node manually in AuboScope, i.e. an undefined
00238     * configuration where the gripper action is unselected.
00239     *
00240     * @param gripperDevice the gripper device to create a Gripper node for, not
00241     * <code>null</code>. The created Gripper node will program gripper actions
00242     * for this device.
00243     * @return a new Gripper program node (which can be inserted into the
00244     * program tree using the {@link TreeNode} interface).
00245     * @throws UnresolvedDeviceException when the specified
00246     * <code>gripperDevice</code> device is unresolved. See
00247     * {@link GripperDevice#isResolvable()}.
00248     * \endenglish
00249     */
00250     GripperNodePtr createGripperNode(GripperDevicePtr gripperDevice);
00251
00252 private:
00253     friend class DataSwitch;
00254     ProgramNodeFactory();
00255     void *d_{ nullptr };
00256 };

```



```

00257
00258 } // namespace aubo_scope
00259 } // namespace arcs
00260
00261 #endif

```

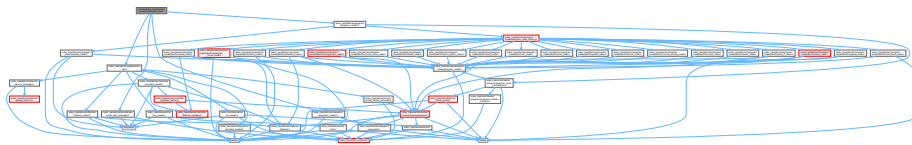
12.204 include/aubo_caps/domain/program/program_api.h File Reference

```

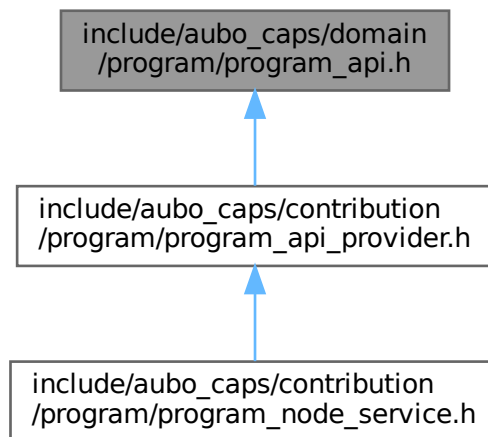
#include <aubo_caps/domain/application_api.h>
#include <aubo_caps/domain/program/program_model.h>
#include <aubo_caps/domain/undoredo/undo_redo_manager.h>
#include <aubo_caps/domain/waypoint/waypoint_model.h>

```

Include dependency graph for program_api.h:



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



Classes

- class [arcs::aubo_scope::ProgramApi](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([ProgramApi](#))

12.205 program_api.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_PROGRAM_API_H
00002 #define AUBO_SCOPE_PROGRAM_API_H
00003
00004 #include <aubo_caps/domain/application_api.h>
00005 #include <aubo_caps/domain/program/program_model.h>
00006 #include <aubo_caps/domain/undoredo/undo_redo_manager.h>
00007 #include <aubo_caps/domain/waypoint/waypoint_model.h>
00008
00009 namespace arcs {
00010 namespace aubo_scope {
00011 ARCS_CLASS_FORWARD(ProgramApi);
00012
00013 /**
00014  * \chinese
00015  * @defgroup ProgramApi ProgramApi( )
00016  * @ingroup ProgramApi
00017  *
00018  *         AuboScope
00019  * AuboScope
00020  * \endchinese
00021  * \english
00022  * @defgroup ProgramApi ProgramApi
00023  * @ingroup ProgramApi
00024  * ProgramApi
00025  * Provides access to functionality and services related to programs. This
00026  * includes what is available from within AuboScope, as well as creating
00027  * additional model elements to be used within AuboScope.
00028  * \endenglish
00029  */
00030 class ARCS_ABI_EXPORT ProgramApi : public ApplicationApi
00031 {
00032 public:
00033     ProgramApi(ProgramApi &f);
00034     ProgramApi(ProgramApi &&f);
00035     virtual ~ProgramApi();
00036
00037     /**
00038      * @ingroup ProgramApi
00039      * @ref ProgramModel
00040      * \chinese
00041      * @return
00042      * \endchinese
00043      * \english
00044      * @return An interface for working with the sub-trees of a program node
00045      * \endenglish
00046      */
00047     ProgramModelPtr getProgramModel();
00048
00049     /**
00050      * @ingroup ProgramApi
00051      * @ref WaypointModel
00052      * \chinese
00053      * @return
00054      * \endchinese
00055      * \english
00056      * @return An interface for working with waypoints
00057      * \endenglish
00058      */
00059     WaypointModelPtr getWaypointModel();
00060
00061     /**
00062      * @ingroup ProgramApi
00063      * @ref UndoRedoManager
00064      * \chinese
00065      * @return
00066      * \endchinese
00067      * \english
00068      * @return An interface for recording undoable changes for program node
00069      * contributions
00070      * \endenglish
00071      */
00072     UndoRedoManagerPtr getUndoRedoManager();
00073

```

```

00074 private:
00075     friend class DataSwitch;
00076     ProgramApi();
00077     void *d_{ nullptr };
00078 };
00079
00080 } // namespace aubo_scope
00081 } // namespace arcs
00082 #endif // AUBO_SCOPE_PROGRAM_API_H

```

12.206 include/aubo_caps/domain/program/program_model.h File Reference

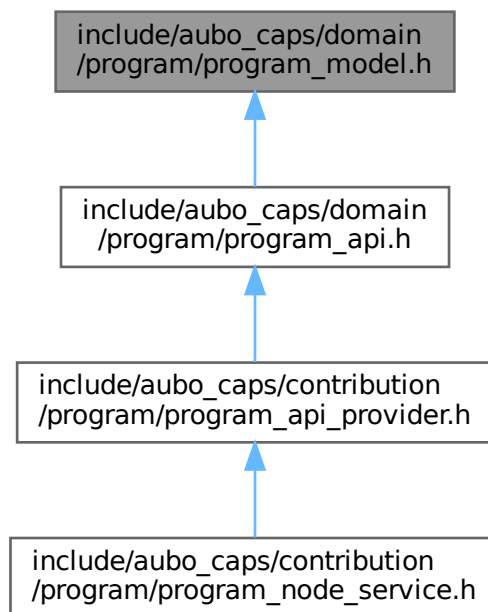
```

#include <aubo_caps/domain/program/structure/tree_node.h>
#include <aubo_caps/contribution/program/program_node_contribution.h>
#include <aubo_caps/domain/program/nodes/program_node_factory.h>
Include dependency graph for program_model.h:

```



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



Classes

- class `arcs::aubo_scope::ProgramModel`
`ProgramModel`

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([ProgramModel](#))

12.207 `program_model.h`

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_PROGRAM_MODEL_H
00002 #define AUBO_SCOPE_PROGRAM_MODEL_H
00003
00004 #include <aubo_caps/domain/program/structure/tree_node.h>
00005 #include <aubo_caps/contribution/program/program_node_contribution.h>
00006 #include <aubo_caps/domain/program/nodes/program_node_factory.h>
00007
00008 namespace arcs {
00009 namespace aubo_scope {
00010 ARCS_CLASS_FORWARD(ProgramModel);
00011
00012 /**
00013  * @ingroup ProgramApi
00014  * \chinese
00015  *
00016  * <h1> </h1>
00017  *   AuboStudio
00018  *   {@link TreeNode}      {@link ProgramNode}
00019  *
00020  * <h1> </h1>
00021  *   ProgramModel
00022  *   <ul>
00023  *     <li>getProgramNodeFactory() -          AuboScope
00024  *     AuboCap </li>
00025  *     <li>getRootTreeNode(ProgramNodeContribution root) -  AuboCap
00026  *     ProgramNodeContribution </li>
00027  *   </ul>
00028  *
00029  *   TreeNode
00030  *   lockChildSequence
00031  *
00032  * \endchinese
00033  * \english
00034  *   ProgramModel
00035  *   <h1>Summary</h1>
00036  *   This interface makes it possible to modify sub-trees of aubo_studio plugins.
00037  *   The structure of trees is separated from the contents of the nodes.
00038  *   The {@link TreeNode} interface is concerned with the structure of trees
00039  *   whereas the {@link ProgramNode} interface deals with the concrete content of
00040  *   the nodes of the tree.
00041  *
00042  * <h1>Getting started with program modification</h1>
00043  *   A ProgramModel contains a:
00044  *   <ul>
00045  *     <li>getProgramNodeFactory() - that makes it possible to create new
00046  *     program nodes (both built-in AuboScope nodes and AuboCap program nodes).
00047  *   </li> <li>getRootTreeNode(ProgramNodeContribution root)} - returns the
00048  *   sub-tree of a given AuboCap ProgramNodeContribution</li>
00049  *   </ul>
00050  *
00051  *   The TreeNode interface supports insertion and removal of nodes. With it you
00052  *   can retrieve the list of children of particular nodes. Calling
00053  *   lockChildSequence on a tree node, locks the immediate children under the
00054  *   node. I.e. children can not be rearranged, deleted or have other nodes
00055  *   inserted into the child sequence by the end user.
00056  * \endenglish
00057  */
00058 class ARCS_ABI_EXPORT ProgramModel
00059 {
00060 public:
00061     ProgramModel(ProgramModel &f);
00062     ProgramModel(ProgramModel &&f);
00063     virtual ~ProgramModel();
00064

```

```

00065  /**
00066  * \chinese
00067  *   ProgramNodeFactory
00068  *
00069  * @return
00070  * \endchinese
00071  * \english
00072  * This method returns a {@link ProgramNodeFactory} to create program nodes.
00073  *
00074  * @return the factory to create program nodes.
00075  * \endenglish
00076  */
00077  ProgramNodeFactoryPtr getProgramNodeFactory();
00078
00079  /**
00080  * \chinese
00081  *   ProgramNodeContribution   TreeNode
00082  *
00083  *
00084  * @param root   AuboCap
00085  * @return TreeNode
00086  * \endchinese
00087  * \english
00088  * Gets the {@link TreeNode} root from {@link ProgramNodeContribution}. From
00089  * here, children can be added to form a sub-tree.
00090  *
00091  * @param root The AuboCap program node where a sub-tree is to be rooted.
00092  * @return Returns a {@link TreeNode} root.
00093  * \endenglish
00094  */
00095  TreeNodePtr getTreeNode(ProgramNodeContribution *root);
00096
00097 private:
00098  friend class DataSwitch;
00099  ProgramModel();
00100  void *d_{ nullptr };
00101 };
00102
00103 } // namespace aubo_scope
00104 } // namespace arcs
00105
00106 #endif // AUBO_SCOPE_PROGRAM_MODEL_H

```

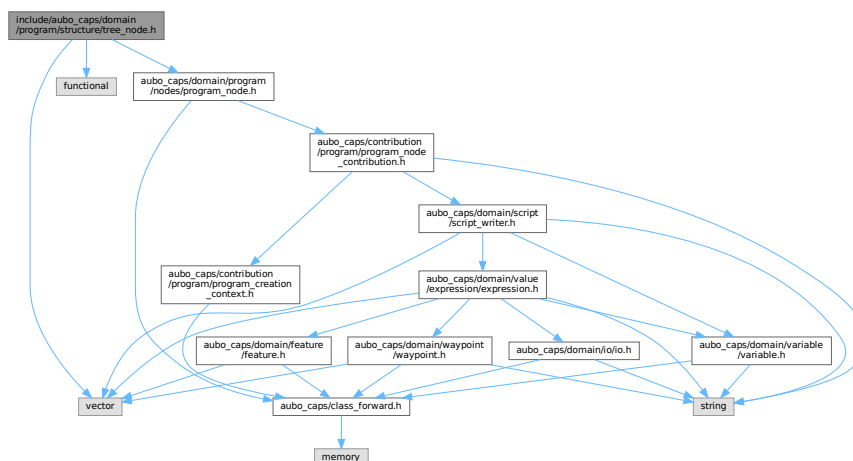
12.208 include/aubo_caps/domain/program/structure/tree_node.h File Reference

```

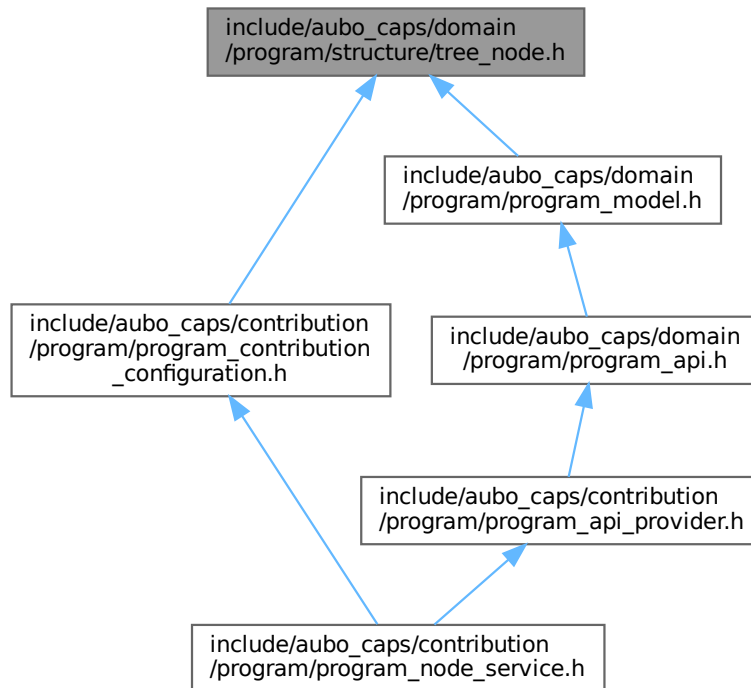
#include <vector>
#include <functional>
#include <aubo_caps/domain/program/nodes/program_node.h>

```

Include dependency graph for tree_node.h:



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



Classes

- class [arcs::aubo_scope::TreeNode](#)

[TreeNode](#) This interface represents a node in the program tree that can be used to construct a sub-tree rooted in a AuboCap program node.

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(TreeNode\)](#)

12.209 tree_node.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_TREE_NODE_H
00002 #define AUBO_SCOPE_TREE_NODE_H
00003
00004 #include <vector>
00005 #include <functional>

```

```

00006 #include <aubo_caps/domain/program/nodes/program_node.h>
00007
00008 namespace arcs {
00009 namespace aubo_scope {
00010 ARCS_CLASS_FORWARD(TreeNode);
00011
00012 /**
00013  * \chinese
00014  *
00015  *         AuboCap
00016  *
00017  *   {@link ProgramModel#getRootTreeNode(ProgramNodeContribution)}
00018  *   {@link TreeNode#addChild(ProgramNode)}
00019  *   {@link TreeNode}
00020  * \endchinese
00021  * \english
00022  * TreeNode
00023  * This interface represents a node in the program tree that can be used to
00024  * construct a sub-tree rooted in a AuboCap program node.
00025  *
00026  * Using the {@link ProgramModel#getRootTreeNode(ProgramNodeContribution)} to
00027  * obtain a root for the sub-tree, it is possible to add children. For each call
00028  * to {@link TreeNode#addChild(ProgramNode)}, a new {@link TreeNode} is
00029  * returned, that can, in turn, act as a root for yet another sub-tree.
00030  * \endenglish
00031  */
00032 class ARCS_ABI_EXPORT TreeNode
00033 {
00034 public:
00035     TreeNode(TreeNode &f);
00036     TreeNode(TreeNode &&f);
00037     virtual ~TreeNode();
00038
00039 /**
00040  * \chinese
00041  *
00042  *
00043  * @param program_node ProgramNodeFactory ProgramNode
00044  * @return TreeNode
00045  * @throws TreeStructureException ProgramNode
00046  * @throws IllegalStateException UndoableChanges Qt
00047  * AuboCap
00048  * \endchinese
00049  * \english
00050  * Add a child program node to the sub-tree.
00051  *
00052  * @param program_node the {@link ProgramNode} constructed using the {@link
00053  * ProgramNodeFactory}
00054  * @return Returns a TreeNode that can be used to add children to the newly
00055  * added child.
00056  * @throws TreeStructureException If it is not allowed to insert the {@link
00057  * ProgramNode} at this position a
00058  * {@link TreeStructureException} will be thrown.
00059  * @throws IllegalStateException if called from a Qt-based AuboCap
00060  * outside of an {@link UndoableChanges} scope (see also {@link
00061  * UndoRedoManager}).
00062  * \endenglish
00063  */
00064     TreeNodePtr addChild(ProgramNodePtr program_node);
00065
00066 /**
00067  * \chinese
00068  *
00069  *
00070  *
00071  * @param existingChildNode TreeNode TreeNode
00072  * @param program_node ProgramNodeFactory ProgramNode
00073  * @return TreeNode
00074  * @throws TreeStructureException
00075  * TreeNode
00076  * @throws IllegalStateException UndoableChanges
00077  * \endchinese
00078  * \english
00079  * Inserts a child program node in the sub-tree directly before the existing
00080  * selected child node. Shifts the selected child node and any subsequent
00081  * nodes to positions after the newly added child.
00082  *
00083  * @param existingChildNode existing TreeNode child of this TreeNode.
00084  * @param program_node the {@link ProgramNode} constructed using the {@link
00085  * ProgramNodeFactory}.
00086  * @return Returns a TreeNode that can be used to add children to the newly
00087  * added child.
00088  * @throws TreeStructureException If it is not allowed to insert the {@link
00089  * ProgramNode} at this position or if the selected child node is not a
00090  * child of this TreeNode a {@link TreeStructureException} will be thrown.
00091  * @throws IllegalStateException if called from a Qt-based AuboCap
00092  * outside of an {@link UndoableChanges} scope (see also {@link

```

```

00093     * UndoRedoManager}).
00094     * \endenglish
00095     */
00096     TreeNodePtr insertChildBefore(TreeNodePtr existingChildNode,
00097                                   ProgramNodePtr program_node);
00098
00099     /**
00100     * \chinese
00101     *
00102     *
00103     *
00104     * @param existingChildNode  TreeNode  TreeNode
00105     * @param program_node  ProgramNodeFactory  ProgramNode
00106     * @return  TreeNode
00107     * @throws TreeStructureException
00108     *         TreeNode
00109     * @throws IllegalStateException  UndoableChanges
00110     * \endchinese
00111     * \english
00112     * Inserts a child program node under in the sub-tree directly after the
00113     * existing selected child node. Shifts any subsequent nodes to positions
00114     * after the newly added child.
00115     *
00116     * @param existingChildNode existing  TreeNode child of this TreeNode.
00117     * @param program_node the {@link ProgramNode} constructed using the {@link
00118     * ProgramNodeFactory}.
00119     * @return Returns a TreeNode that can be used to add children to the newly
00120     * added child.
00121     * @throws TreeStructureException If it is not allowed to insert the {@link
00122     * ProgramNode} at this position or if the selected child node is not a
00123     * child of this TreeNode a {@link TreeStructureException} will be thrown.
00124     * @throws IllegalStateException if called from a Qt-based AudoCap
00125     * outside of an {@link UndoableChanges} scope (see also {@link
00126     * UndoRedoManager}).
00127     * \endenglish
00128     */
00129     TreeNodePtr insertChildAfter(TreeNodePtr existingChildNode,
00130                                  ProgramNodePtr program_node);
00131
00132     /**
00133     * \chinese
00134     *
00135     *
00136     * @param child  TreeNode  TreeNode
00137     * @return  TreeNode
00138     * @throws TreeStructureException  ProgramNode
00139     *         TreeNode
00140     * @throws IllegalStateException  UndoableChanges
00141     * \endchinese
00142     * \english
00143     * Cut a child program node under in the sub-tree directly after the
00144     * existing selected child node. Shifts any subsequent nodes to positions
00145     * after the newly added child.
00146     *
00147     * @param child existing  TreeNode child of this TreeNode.
00148     * @return Returns the TreeNode that has been cut off.
00149     * @throws TreeStructureException If it is not allowed to cut the {@link
00150     * ProgramNode} or if the selected child node is not a child of this
00151     * TreeNode a {@link TreeStructureException} will be thrown.
00152     * @throws IllegalStateException if called from a Qt-based AudoCap
00153     * outside of an {@link UndoableChanges} scope (see also {@link
00154     * UndoRedoManager}).
00155     * \endenglish
00156     */
00157     TreeNodePtr cutChildNode(TreeNodePtr child);
00158
00159     /**
00160     * \chinese
00161     *
00162     * {@literal <empty>}
00163     *
00164     * @param program_node  TreeNode
00165     * @return  <code>true</code>  <code>>false</code>
00166     * @throws TreeStructureException
00167     * @throws IllegalStateException  UndoableChanges
00168     * \endchinese
00169     * \english
00170     * Removes a child node from the sub-tree. Be aware that removing the last
00171     * child will trigger the insertion of an
00172     * {@literal <empty>} child node.
00173     *
00174     * @param program_node The TreeNode child to be removed.
00175     * @return Returns <code>true</code> if removed successfully.
00176     * <code>>false</code> otherwise.
00177     * @throws TreeStructureException If the removed child would leave the tree
00178     * in an illegal state a
00179     * {@link TreeStructureException} will be thrown.

```



```

00180 * @throws IllegalStateException if called from a Qt-based AuboCap
00181 * outside of an {@link UndoableChanges} scope (see also {@link
00182 * UndoRedoManager}).
00183 * \endenglish
00184 */
00185 bool removeChild(TreeNodePtr program_node);
00186
00187 /**
00188 * \chinese
00189 * @return TreeNode      TreeNode
00190 *
00191 * \endchinese
00192 * \english
00193 * @return a list of <code>TreeNode</code> objects that represents all the
00194 * children of this <code>TreeNode</code> (both programmatically added as
00195 * well as inserted by the end user).
00196 * \endenglish
00197 */
00198 std::vector<TreeNodePtr> getChildren();
00199
00200 std::vector<TreeNodePtr> getChildrenNotSuppressed();
00201
00202 /**
00203 * \chinese
00204 * @return      TreeNode
00205 * \endchinese
00206 * \english
00207 * @return Parent treeNode of the node
00208 * \endenglish
00209 */
00210 TreeNodePtr getParent();
00211
00212 /**
00213 * \chinese
00214 * @return      ProgramNode      AuboScope
00215 * \endchinese
00216 * \english
00217 * @return Returns the {@link ProgramNode} at this position in the sub-tree.
00218 * Can either be a built-in AuboScope program node (provided by AUBO
00219 * Robots) or a AuboCap program node.
00220 * \endenglish
00221 */
00222 ProgramNodePtr getProgramNode();
00223 bool isSuppressed();
00224
00225 /**
00226 * \chinese
00227 *
00228 *
00229 * @return <code>true</code>
00230 * <code>false</code>
00231 * \endchinese
00232 * \english
00233 * Whether the node is in the program tree; nodes not in the program tree
00234 * may have been cut or deleted.
00235 *
00236 * @return Returns <code>true</code> The node is in the program tree.
00237 * <code>false</code> The node is not in the program tree; such nodes may
00238 * have been cut or deleted.
00239 * \endenglish
00240 */
00241 bool isInProgramTree();
00242
00243 /**
00244 * \chinese
00245 * <p>      TreeNode      ProgramNode      TreeNode
00246 * </p>
00247 *
00248 *
00249 * @param program_node      TreeNode
00250 * <code>null</code>
00251 * @return      TreeNode
00252 * @throws ProgramNodeNotInSubTreeException
00253 * \endchinese
00254 * \english
00255 * <p>
00256 * Gets a corresponding {@link TreeNode} instance for a child program node
00257 * (a {@link ProgramNode} instance) in the sub-tree under this {@link
00258 * TreeNode}.
00259 * </p>
00260 * This method can for instance be used to gain access to the sub-tree under
00261 * a child program node encountered when iterating the sub-tree of the
00262 * parent node using the {@link ProgramNodeVisitor} (or any derived sub
00263 * class thereof).
00264 *
00265 * @param program_node program node to get a corresponding {@link TreeNode}

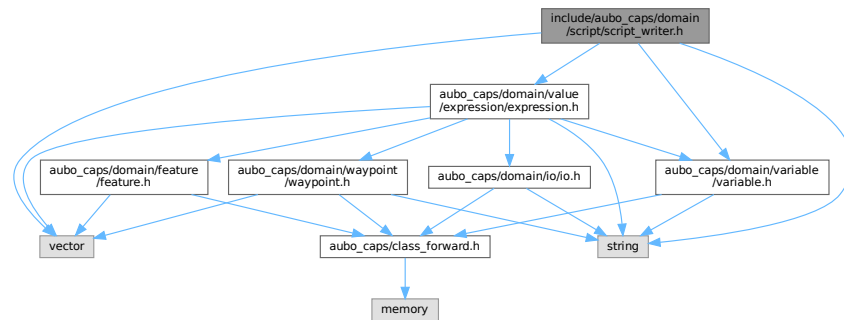
```

```

00267 * representation for, not null. Can either be a built-in
00268 * AuboScope program node (provided by AUBO Robots) or a AuboCap
00269 * program node.
00270 * @return the TreeNode instance for the specified program
00271 * node.
00272 * @throws ProgramNodeNotInSubTreeException when the program node cannot be
00273 * found because it is not in the sub-tree.
00274 * \endenglish
00275 */
00276 TreeNodePtr locateDescendantTreeNode(ProgramNodePtr program_node);
00277 /**
00278 * \chinese
00279 *
00280 *
00281 *
00282 * @param isChildSequenceLocked true
00283 * TreeNode
00284 * \endchinese
00285 * \english
00286 * Configures whether or not child nodes can be rearranged, deleted or have
00287 * other nodes inserted into the child sequence by the end user.
00288 *
00289 * @param isChildSequenceLocked If true then the immediate
00290 * children under this TreeNode will be locked.
00291 * \endenglish
00292 */
00293 void setChildSequenceLocked(bool isChildSequenceLocked);
00294 /**
00295 * \chinese
00296 *
00297 * <p>
00298 * <p>
00299 * <p> visit
00300 *
00301 * @param nodeVisitor
00302 * @return 0
00303 * \endchinese
00304 * \english
00305 * <p>
00306 * This method traverses the entire sub-tree under this tree node in a
00307 * depth-first fashion (this corresponds to a top-down approach in the
00308 * program tree).
00309 * </p>
00310 * <p>
00311 * A node visitor is used for callbacks to the visit-overloads you choose to
00312 * override. The overload called depends on the node type encountered.
00313 * Override the overloads for the node types you are concerned with. All
00314 * visit-methods have the program node, sibling index and depth as arguments
00315 * to help filter nodes if needed.
00316 * </p>
00317 * <p>
00318 * The node visitor can be either a {@link ProgramNodeVisitor}
00319 * implementation with optional overrides or a
00320 * {@link ProgramNodeInterfaceVisitor} implementation. In the latter
00321 * case, the
00322 * {@link ProgramNodeInterfaceVisitor#visitAs(Object, int, int)}
00323 * method must be implemented.
00324 * </p>
00325 * <p>
00326 * The {@link ProgramNodeInterfaceVisitor} can be used when targeting
00327 * AuboCap program nodes implementing the (generic) type parameter specified
00328 * in {@link ProgramNodeInterfaceVisitor} (see also
00329 * {@link ProgramNode#getAs(Class)}).
00330 * </p>
00331 * Note that this method is sometimes called accept() in the
00332 * Visitor software design pattern.
00333 *
00334 * @param nodeVisitor the instance callbacks are made to.
00335 * @return 0 for normal exit.
00336 * non-zero for exception or error causing premature exit.
00337 * \endenglish
00338 */
00339 int traverse(std::function<int(ProgramNodePtr, int, int)> nodeVisitor);
00340
00341 private:
00342 friend class DataSwitch;
00343 TreeNode();
00344 void *d_{ nullptr };
00345 };
00346
00347 } // namespace aubo_scope
00348 } // namespace arcs
00349 #endif // AUBO_SCOPE_TREE_NODE_H

```

```
#include <vector>
#include <string>
#include <aubo_caps/domain/variable/variable.h>
#include <aubo_caps/domain/value/expression/expression.h>
Include dependency graph for script_writer.h:
```



- `class arcs::aubo_scope::ScriptWriter`
`ScriptWriter` This interface provides support for generating Script code.

- namespace `arcs`
- namespace `arcs::aubo_scope`

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (ScriptWriter)`

12.211 script__writer.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_SCRIPT_WRITER_H
00002 #define AUBO_SCOPE_SCRIPT_WRITER_H
00003
00004 #include <vector>
00005 #include <string>
00006 #include <aubo_caps/domain/variable/variable.h>
00007 #include <aubo_caps/domain/value/expression/expression.h>
00008
00009 namespace arcs {
00010 namespace aubo_scope {
00011 ARCS_CLASS_FORWARD(ScriptWriter);
00012
00013 /**
00014  * \chinese
00015  *
00016  *
00017  * \endchinese
00018  * \english
00019  * ScriptWriter
00020  * This interface provides support for generating Script code.
00021  * \endenglish
00022  */
00023 class ARCS_ABI_EXPORT ScriptWriter
00024 {
00025 public:
00026     ScriptWriter(ScriptWriter &f);
00027     ScriptWriter(ScriptWriter &&f);
00028     virtual ~ScriptWriter();
00029
00030     void setLabel(int lineno, const std::string &comment);
00031
00032     /**
00033      * \chinese
00034      *
00035      * @param script_line
00036      * \endchinese
00037      * \english
00038      * Adds a single line of script code using auto-indentation.
00039      *
00040      * @param script_line single line of script code to append.
00041      * \endenglish
00042      */
00043     void appendLine(const std::string &script_line);
00044     void appendVectorDouble(const std::vector<double> &value);
00045
00046     /**
00047      * \chinese
00048      *
00049      * @param script
00050      * \endchinese
00051      * \english
00052      * Adds script code appending the script code as is without using
00053      * auto-indentation.
00054      *
00055      * @param script script code to append.
00056      * \endenglish
00057      */
00058     void appendRaw(const std::string &script);
00059
00060     /**
00061      * \chinese
00062      *
00063      * @return
00064      * \endchinese
00065      * \english
00066      * Generate a string with the full script code.
00067      *
00068      * @return the resulting script as a string.
00069      * \endenglish
00070      */
00071     std::string generateScript();
00072
00073     /**
00074      * \chinese
00075      *
00076      * \endchinese
00077      * \english
00078      * Use this method when generating script code for a node that has children
00079      * nodes. Let children nodes generate code.
00080      * \endenglish
00081      */
00082     void writeChildren();

```

```

00083
00084 void assign(VariablePtr variable, const std::string &expression,
00085             bool sync = false);
00086
00087 /**
00088  * \chinese
00089  * <p> local global </p>
00090  * @param variableName
00091  * @param expression
00092  * \endchinese
00093  * \english
00094  * <p>Variable assignment.
00095  * Assigns the variable without a local or global qualifier.</p>
00096  * @param variableName name of the variable, not null.
00097  * @param expression expression that is assigned to the variable, not null.
00098  * \endenglish
00099  */
00100 void assign(const std::string &variableName,
00101             const ExpressionPtr &expression, bool sync = false);
00102
00103 /**
00104  * \chinese
00105  * <p> </p>
00106  * @param variable
00107  * @param expression
00108  * \endchinese
00109  * \english
00110  * <p>Variable assignment.</p>
00111  * @param variable the variable to assign an expression to, not null.
00112  * @param expression expression that is assigned to the variable, not null.
00113  * \endenglish
00114  */
00115 void assign(VariablePtr variable, const ExpressionPtr &expression,
00116             bool sync = false);
00117
00118 /**
00119  * \chinese
00120  * 1
00121  * @param variable_name
00122  * \endchinese
00123  * \english
00124  * Add 1 to the variable value.
00125  * @param variable_name the variable to increment.
00126  * \endenglish
00127  */
00128 void incrementVariable(const std::string &variable_name);
00129
00130 /**
00131  * \chinese \endchinese
00132  * \english Add a note. @param expression the note expression. \endenglish
00133  */
00134 void note(const std::string &expression);
00135
00136 /**
00137  * \chinese \endchinese
00138  * \english Sleep for a number of seconds. @param seconds amount of time to sleep in seconds. \endenglish
00139  */
00140 void sleep(double seconds);
00141
00142 /** \chinese \endchinese \english Sync. \endenglish */
00143 void sync();
00144
00145 /** \chinese \endchinese \english Define a function. @param func_name function name. \endenglish */
00146 void defineFunction(const std::string &func_name);
00147 void anyonFunction(const std::string &func_name);
00148
00149 void setRobotIndex(int robot_index);
00150
00151 /** \chinese \endchinese \english Return from method. \endenglish */
00152 void returnMethod();
00153
00154 /** \chinese end \endchinese \english Insert an end. \endenglish */
00155 void end();
00156
00157 /** \chinese \endchinese \english Insert an empty line. \endenglish */
00158 void lineFeed();
00159
00160 /**
00161  * \chinese \endchinese
00162  * \english Sets the mass and Center of Gravity of the payload. \endenglish
00163  */
00164 void setPayload(double mass, double x, double y, double z);
00165
00166 /** \chinese TCP \endchinese \english Set the Tool Center Point. \endenglish */
00167 void setTcp(const std::vector<double> &pose);
00168
00169 /** \chinese if \endchinese \english Start an if-conditional. @param expression the expression. \endenglish */

```

```

00170 void ifCondition(const ExpressionPtr &expression);
00171
00172 /** \chinese if \endchinese \english Start a negated if-conditional. @param expression the expression. \endenglish */
00173 void ifNotCondition(const ExpressionPtr &expression);
00174
00175 /** \chinese else-if \endchinese \english Adds an else-if branch. @param expression the expression. \endenglish */
00176 void elifCondition(const ExpressionPtr &expression);
00177
00178 /** \chinese else \endchinese \english Adds an else branch. \endenglish */
00179 void elseCondition();
00180
00181 /** \chinese for \endchinese \english Starts a for-loop. @param count loop count, @param step loop step. \endenglish
*/
00182 void forCondition(int count, int step);
00183
00184 /** \chinese while true \endchinese \english Starts a while true loop. \endenglish */
00185 void whileTrue();
00186
00187 /** \chinese while \endchinese \english Starts a while-loop. @param expression the loop invariant. \endenglish */
00188 void whileCondition(const ExpressionPtr &expression);
00189
00190 /** \chinese while \endchinese \english Starts a negated while-loop. @param expression the loop invariant.
\endenglish */
00191 void whileNot(const ExpressionPtr &expression);
00192
00193 /**
00194  * \chinese \endchinese
00195  * \english Start a thread definition. @param thread_name the thread name, @param loop_or_not whether loops.
\endenglish
*/
00196
00197 void defineThread(const std::string &thread_name, bool loop_or_not);
00198
00199 /** \chinese \endchinese \english Start a thread. @param thread_name the thread name. \endenglish */
00200 void runThread(const std::string &thread_name);
00201
00202 /** \chinese \endchinese \english Kill a thread. @param thread_name the thread name. \endenglish */
00203 void killThread(const std::string &thread_name);
00204
00205 /**
00206  * \chinese
00207  *
00208  * @param variable_name
00209  * @return
00210  * \endchinese
00211  * \english
00212  * Returns a registered variable name that can be used in a script.
00213  * @param variable_name a variable name.
00214  * @return variable name that can be used in a script.
00215  * \endenglish
*/
00216
00217 std::string getResolvedVariable(const std::string &variable_name);
00218
00219 void increaseIndent();
00220 void decreaseIndent();
00221
00222 private:
00223 friend class DataSwitch;
00224 ScriptWriter();
00225 void *d_{ nullptr };
00226 };
00227
00228 } // namespace aubo_scope
00229 } // namespace arcs
00230
00231 #endif // AUBO_SCOPE_SCRIPT_WRITER_H

```

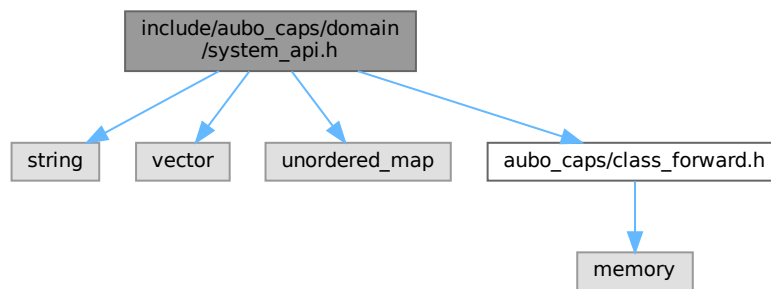
12.212 include/auco_caps/domain/system_api.h File Reference

```

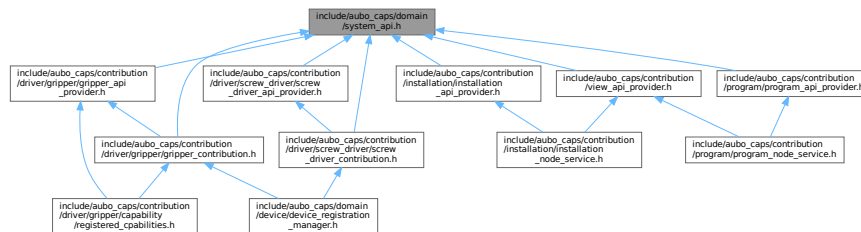
#include <string>
#include <vector>
#include <unordered_map>
#include <auco_caps/class_forward.h>

```

Include dependency graph for system_api.h:



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



Classes

- struct `arcs::aubo_scope::IoConfigure`
- class `arcs::aubo_scope::SystemApi`

Namespaces

- namespace `arcs`
- namespace `arcs::aubo_scope`

Enumerations

- enum class `arcs::aubo_scope::UnitType` { `arcs::aubo_scope::METRIC`, `arcs::aubo_scope::IMPERIAL` }

Functions

- `arcs::aubo_scope::ARCS_CLASS_FORWARD` (`SystemApi`)

12.213 system_api.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_SYSTEM_API_H
00002 #define AUBO_SCOPE_SYSTEM_API_H
00003
00004 #include <string>
00005 #include <vector>
00006 #include <unordered_map>
00007 #include <aubo_caps/class_forward.h>
00008
00009 namespace arcs {
00010 namespace aubo_scope {
00011
00012 enum class UnitType
00013 {
00014     METRIC, ///< mm Kg
00015     IMPERIAL ///<
00016 };
00017
00018 struct IoConfigure
00019 {
00020     std::string label;
00021     int num;
00022     std::vector<std::string> names;
00023     std::vector<int> sequences;
00024     std::vector<int> functions;
00025     std::vector<int> safety_functions;
00026     std::vector<int> safety_functions_fixed;
00027     bool function_lock{ false };
00028     std::vector<int> types;
00029     std::vector<double> mins;
00030     std::vector<double> maxs;
00031 };
00032
00033 ARCS_CLASS_FORWARD(SystemApi);
00034
00035 /**
00036  * \chinese
00037  * @defgroup SystemApi SystemApi( )
00038  * @ingroup SystemApi
00039  *
00040  *     AuboScope
00041  *
00042  * \endchinese
00043  * \english
00044  * @defgroup SystemApi SystemApi
00045  * @ingroup SystemApi
00046  * SystemApi
00047  * Provides access to system information of the AuboScope software, including
00048  * software version numbers, system locale, unit types, the robot serial number,
00049  * as well as robot joint position limits, velocity, acceleration, and payload
00050  * parameters.
00051  * \endenglish
00052  */
00053
00054 class ARCS_ABI_EXPORT SystemApi
00055 {
00056 public:
00057     SystemApi(SystemApi &f);
00058     SystemApi(SystemApi &&f);
00059     virtual ~SystemApi();
00060
00061     /**
00062      * @ingroup SystemApi
00063      * \chinese
00064      *
00065      * @return
00066      * \endchinese
00067      * \english
00068      * Get the version of Software.
00069      * @return the major version of Software.
00070      * \endenglish
00071      *
00072      */
00073     int getMajorVersion();
00074
00075     /**
00076      * @ingroup SystemApi
00077      * \chinese
00078      *
00079      * @return
00080      * \endchinese
00081      * \english
00082      * Get the minor version of Software.

```



```

00083     * @return the minor version of Software.
00084     * \endenglish
00085     */
00086 int getMinorVersion();
00087
00088 /**
00089  * @ingroup SystemApi
00090  * \chinese
00091  *     Bug
00092  * @return Bug
00093  * \endchinese
00094  * \english
00095  * Get the bug fix version of Software.
00096  * @return the bug fix version of Software.
00097  * \endenglish
00098  */
00099 int getBugfixVersion();
00100
00101 /**
00102  * @ingroup SystemApi
00103  * \chinese
00104  *
00105  * @return
00106  * \endchinese
00107  * \english
00108  * Get the build number of Software.
00109  * @return the build number of Software.
00110  * \endenglish
00111  */
00112 int getBuildNumber();
00113
00114 /**
00115  * @ingroup SystemApi
00116  * \chinese
00117  *     {@link Locale}
00118  *
00119  * @return
00120  * \endchinese
00121  * \english
00122  * Returns the {@link Locale} for the system.
00123  *
00124  * @return Locale.
00125  * \endenglish
00126  */
00127 std::string getLocale();
00128
00129 /**
00130  * @ingroup SystemApi
00131  * \chinese
00132  *     {@link Locale}
00133  *
00134  * @return
00135  * \endchinese
00136  * \english
00137  * Returns the {@link Locale} for the programming language used in the
00138  * system.
00139  *
00140  * @return Locale.
00141  * \endenglish
00142  */
00143 std::string getLocaleForProgrammingLanguage();
00144
00145 /**
00146  * @ingroup SystemApi
00147  * \chinese
00148  *
00149  *
00150  * @return
00151  * \endchinese
00152  * \english
00153  * Returns the UnitType of the system as an enum.
00154  *
00155  * @return UnitType.
00156  * \endenglish
00157  */
00158 UnitType getUnitType();
00159
00160 /**
00161  * @ingroup SystemApi
00162  * \chinese
00163  *
00164  * @return
00165  * \endchinese
00166  * \english
00167  * Returns the serial number of the robot.
00168  * @return the serial number of the robot.
00169  * \endenglish

```

```

00170     */
00171     std::string getSerialNumber();
00172
00173     /**
00174     * @ingroup SystemApi
00175     * \chinese
00176     *
00177     * @return <code>true</code>          <code>false</code>
00178     * \endchinese
00179     * \english
00180     * Returns <code>true</code> if controlling the real, physical robot and
00181     * <code>false</code> if in simulation mode.
00182     * @return <code>true</code> if controlling the real, physical robot and
00183     * <code>false</code> if in simulation mode.
00184     * \endenglish
00185     */
00186     bool isRealRobot();
00187
00188     /**
00189     * @ingroup SystemApi
00190     * \chinese
00191     *
00192     * @return
00193     * \endchinese
00194     * \english
00195     * Returns the max positions of robot joints.
00196     * @return Max Positions of robot joints in radian.
00197     * \endenglish
00198     */
00199     std::vector<double> getJointMaxPositions();
00200
00201     /**
00202     * @ingroup SystemApi
00203     * \chinese
00204     *
00205     * @return
00206     * \endchinese
00207     * \english
00208     * Returns the min positions of robot joints.
00209     * @return Min Positions of robot joints in radian.
00210     * \endenglish
00211     */
00212     std::vector<double> getJointMinPositions();
00213
00214     /**
00215     * @ingroup SystemApi
00216     * \chinese
00217     *
00218     * @return
00219     * \endchinese
00220     * \english
00221     * Returns the default velocity of robot joints.
00222     * @return the default velocity of robot joints.
00223     * \endenglish
00224     */
00225     double getJointDefaultVelocity();
00226
00227     /**
00228     * @ingroup SystemApi
00229     * \chinese
00230     *
00231     * @return
00232     * \endchinese
00233     * \english
00234     * Returns the default acceleration of robot joints.
00235     * @return the default acceleration of robot joints.
00236     * \endenglish
00237     */
00238     double getJointDefaultAcceleration();
00239
00240     /**
00241     * @ingroup SystemApi
00242     * \chinese
00243     *
00244     * @return
00245     * \endchinese
00246     * \english
00247     * Returns the max velocity of robot joints.
00248     * @return the max velocity of robot joints.
00249     * \endenglish
00250     */
00251     std::vector<double> getJointMaxVelocity();
00252
00253     /**
00254     * @ingroup SystemApi
00255     * \chinese
00256     *

```

```

00257     * @return
00258     * \endchinese
00259     * \english
00260     * Returns the max acceleration of robot joints.
00261     * @return the max acceleration of robot joints.
00262     * \endenglish
00263     */
00264     std::vector<double> getJointMaxAcceleration();
00265
00266     /**
00267     * @ingroup SystemApi
00268     * \chinese
00269     * TCP
00270     * @return TCP
00271     * \endchinese
00272     * \english
00273     * Returns the default velocity of TCP.
00274     * @return the default velocity of TCP.
00275     * \endenglish
00276     */
00277     double getTcpDefaultVelocity();
00278
00279     /**
00280     * @ingroup SystemApi
00281     * \chinese
00282     * TCP
00283     * @return TCP
00284     * \endchinese
00285     * \english
00286     * Returns the default acceleration of TCP.
00287     * @return the default acceleration of TCP.
00288     * \endenglish
00289     */
00290     double getTcpDefaultAcceleration();
00291
00292     /**
00293     * @ingroup SystemApi
00294     * \chinese
00295     * TCP
00296     * @return TCP
00297     * \endchinese
00298     * \english
00299     * Returns the max velocity of TCP.
00300     * @return the max velocity of TCP.
00301     * \endenglish
00302     */
00303     double getTcpMaxVelocity();
00304
00305     /**
00306     * @ingroup SystemApi
00307     * \chinese
00308     * TCP
00309     * @return TCP
00310     * \endchinese
00311     * \english
00312     * Returns the max acceleration of TCP.
00313     * @return the max acceleration of TCP.
00314     * \endenglish
00315     */
00316     double getTcpMaxAcceleration();
00317
00318     /**
00319     * @ingroup SystemApi
00320     * \chinese
00321     * TCP
00322     * @return TCP
00323     * \endchinese
00324     * \english
00325     * Returns the max distance of TCP.
00326     * @return the max distance of TCP.
00327     * \endenglish
00328     */
00329     double getTcpMaxDistance();
00330
00331     /**
00332     * @ingroup SystemApi
00333     * \chinese
00334     *
00335     * @return
00336     * \endchinese
00337     * \english
00338     * Returns the max payload mass.
00339     * @return the max payload mass.
00340     * \endenglish
00341     */
00342     double getMaxPayloadMass();
00343

```

```

00344  /**
00345  * @ingroup SystemApi
00346  * \chinese
00347  *
00348  * @return
00349  * \endchinese
00350  * \english
00351  * Returns the max payload distance.
00352  * @return the max payload distance.
00353  * \endenglish
00354  */
00355  double getMaxPayloadDistance();
00356
00357  /**
00358  * @ingroup SystemApi
00359  * \chinese
00360  *   Home
00361  * @return   Home
00362  * \endchinese
00363  * \english
00364  * Returns the default home position.
00365  * @return the default home position.
00366  * \endenglish
00367  */
00368  std::vector<double> getHomeDefaultPosition();
00369
00370  /**
00371  * @ingroup SystemApi
00372  * \chinese
00373  *   Home
00374  * @return   Home
00375  * \endchinese
00376  * \english
00377  * Returns the package home position.
00378  * @return the package home position.
00379  * \endenglish
00380  */
00381  std::vector<double> getHomePackagePosition();
00382
00383  /**
00384  * @ingroup SystemApi
00385  * \chinese
00386  *   DH
00387  * @param is_real
00388  * @return DH
00389  * \endchinese
00390  * \english
00391  * Returns the robot DH parameters.
00392  * @param is_real whether to get the real parameters.
00393  * @return the DH parameters.
00394  * \endenglish
00395  */
00396  std::unordered_map<std::string, std::vector<double>» getRobotDhParams(
00397      bool is_real);
00398
00399  /**
00400  * @ingroup SystemApi
00401  * \chinese
00402  *   IO
00403  * @return   IO
00404  * \endchinese
00405  * \english
00406  * Returns the tool digital I/O configuration.
00407  * @return the tool digital I/O configuration.
00408  * \endenglish
00409  */
00410  IoConfigure getToolDigitalIOConfigure();
00411
00412  /**
00413  * @ingroup SystemApi
00414  * \chinese
00415  *
00416  * @return
00417  * \endchinese
00418  * \english
00419  * Returns the standard digital input configuration.
00420  * @return the standard digital input configuration.
00421  * \endenglish
00422  */
00423  IoConfigure getStandardDigitalInputConfigure();
00424
00425  /**
00426  * @ingroup SystemApi
00427  * \chinese
00428  *
00429  * @return
00430  * \endchinese

```

```

00431     * \english
00432     * Returns the configurable digital input configuration.
00433     * @return the configurable digital input configuration.
00434     * \endenglish
00435     */
00436 IoConfigure getConfigurableDigitalInputConfigure();
00437
00438 /**
00439  * @ingroup SystemApi
00440  * \chinese
00441  *
00442  * @return
00443  * \endchinese
00444  * \english
00445  * Returns the standard digital output configuration.
00446  * @return the standard digital output configuration.
00447  * \endenglish
00448  */
00449 IoConfigure getStandardDigitalOutputConfigure();
00450
00451 /**
00452  * @ingroup SystemApi
00453  * \chinese
00454  *
00455  * @return
00456  * \endchinese
00457  * \english
00458  * Returns the configurable digital output configuration.
00459  * @return the configurable digital output configuration.
00460  * \endenglish
00461  */
00462 IoConfigure getConfigurableDigitalOutputConfigure();
00463
00464 /**
00465  * @ingroup SystemApi
00466  * \chinese
00467  *
00468  * @return
00469  * \endchinese
00470  * \english
00471  * Returns the standard analog input configuration.
00472  * @return the standard analog input configuration.
00473  * \endenglish
00474  */
00475 IoConfigure getStandardAnalogInputConfigure();
00476
00477 /**
00478  * @ingroup SystemApi
00479  * \chinese
00480  *
00481  * @return
00482  * \endchinese
00483  * \english
00484  * Returns the tool analog input configuration.
00485  * @return the tool analog input configuration.
00486  * \endenglish
00487  */
00488 IoConfigure getToolAnalogInputConfigure();
00489
00490 /**
00491  * @ingroup SystemApi
00492  * \chinese
00493  *
00494  * @return
00495  * \endchinese
00496  * \english
00497  * Returns the standard analog output configuration.
00498  * @return the standard analog output configuration.
00499  * \endenglish
00500  */
00501 IoConfigure getStandardAnalogOutputConfigure();
00502
00503 /**
00504  * @ingroup SystemApi
00505  * \chinese
00506  *
00507  * @return
00508  * \endchinese
00509  * \english
00510  * Returns the tool analog output configuration.
00511  * @return the tool analog output configuration.
00512  * \endenglish
00513  */
00514 IoConfigure getToolAnalogOutputConfigure();
00515
00516 /**
00517  * @ingroup SystemApi

```

```

00518     * \chinese
00519     *
00520     * @return
00521     * \endchinese
00522     * \english
00523     * Returns the static link input configuration.
00524     * @return the static link input configuration.
00525     * \endenglish
00526     */
00527 IoConfigure getStaticLinkInputConfigure();
00528
00529 /**
00530  * @ingroup SystemApi
00531  * \chinese
00532  *
00533  * @return
00534  * \endchinese
00535  * \english
00536  * Returns the static link output configuration.
00537  * @return the static link output configuration.
00538  * \endenglish
00539  */
00540 IoConfigure getStaticLinkOutputConfigure();
00541
00542 /**
00543  * @ingroup SystemApi
00544  * \chinese
00545  *
00546  * @return
00547  * \endchinese
00548  * \english
00549  * Returns the encoder configuration.
00550  * @return the encoder configuration.
00551  * \endenglish
00552  */
00553 IoConfigure getENCConfigure();
00554
00555 private:
00556     friend class DataSwitch;
00557     SystemApi();
00558     void *d_{ nullptr };
00559 };
00560
00561 } // namespace aubo_scope
00562 } // namespace arcs
00563
00564 #endif // SYSTEMAPI_H

```

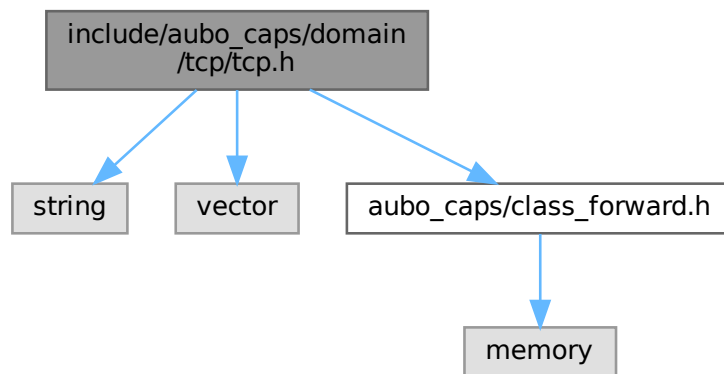
12.214 include/aubo_caps/domain/tcp/tcp.h File Reference

```

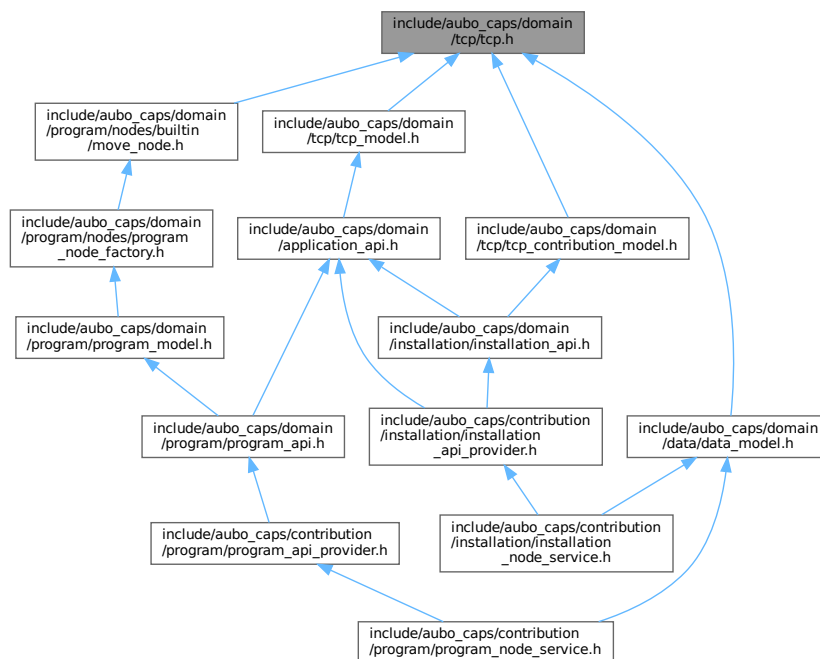
#include <string>
#include <vector>
#include <aubo_caps/class_forward.h>

```

Include dependency graph for tcp.h:



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



Classes

- class `arcs::aubo_scope::TCP`
[TCP](#) This interface represents a [TCP](#) in AuboScope.

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) (TCP)

12.215 tcp.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_TCP_H
00002 #define AUBO_SCOPE_TCP_H
00003
00004 #include <string>
00005 #include <vector>
00006 #include <aubo_caps/class_forward.h>
00007
00008 namespace arcs {
00009 namespace aubo_scope {
00010 ARCS_CLASS_FORWARD(TCP);
00011
00012 /**
00013  * @ingroup InstallationApi
00014  * \chinese
00015  * TCP
00016  *   AuboScope   TCP           TCP
00017  *   AuboCap     TCP
00018  * \endchinese
00019  * \english
00020  * TCP
00021  * This interface represents a TCP in AuboScope. This can either be a
00022  * user-defined TCP or a TCP added by a AuboCap.
00023  * \endenglish
00024  */
00025 class ARCS_ABI_EXPORT TCP
00026 {
00027 public:
00028     TCP(TCP &f);
00029     TCP(TCP &&f);
00030     ~TCP();
00031
00032     /**
00033      * @ingroup TCP
00034      * \chinese
00035      * TCP
00036      * @return TCP
00037      * \endchinese
00038      * \english
00039      * Returns the offset of the TCP with respect to the tool output flange.
00040      * @return The offset of the TCP.
00041      * \endenglish
00042      */
00043     std::vector<double> getOffset();
00044
00045     /**
00046      * @ingroup TCP
00047      * \chinese
00048      * TCP   TCP
00049      * @return TCP
00050      * \endchinese
00051      * \english
00052      * Note: The TCP can be renamed at any time.
00053      * @return The name of the TCP.
00054      * \endenglish
00055      */
00056     std::string getDisplayName();
00057
00058     /**
00059      * @ingroup TCP
00060      * \chinese
00061      * TCP           TCP   TCP
00062      *   TCP AuboCap   TCP
00063      * @return TCP  <code>true</code>  <code>false</code>
00064      * \endchinese

```



```

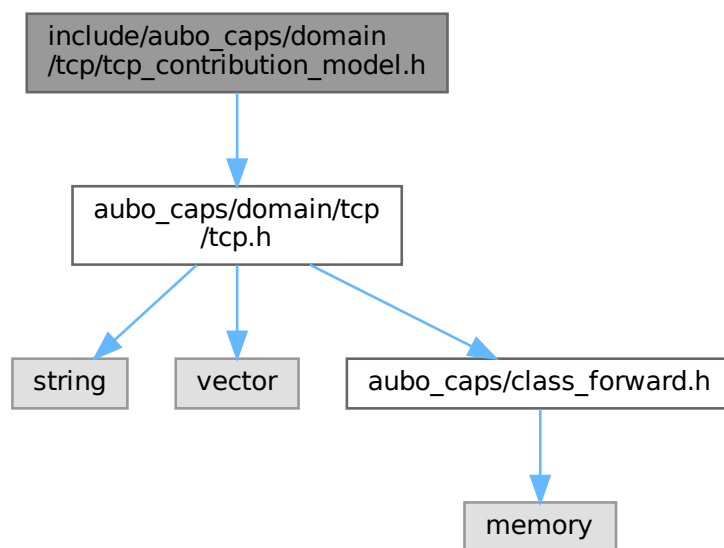
00065  * \english
00066  * A TCP cannot be guaranteed to be present in AboScope. The end user can
00067  * load a different installation (not containing the TCP) or the TCP was
00068  * removed by the end user or the AboCap that added the tcp. This method
00069  * can be used to determine, if the TCP is present.
00070  *
00071  * @return <code>true</code> if this TCP is present in AboScope, otherwise
00072  * <code>false</code>.
00073  * \endenglish
00074  */
00075  bool isResolvable();
00076
00077  /**
00078  * @ingroup TCP
00079  * \chinese
00080  *
00081  * @return
00082  * \endchinese
00083  * \english
00084  * Returns the hash code.
00085  * @return the hash code.
00086  * \endenglish
00087  */
00088  size_t hashCode();
00089
00090 private:
00091  friend class DataSwitch;
00092  TCP();
00093  void *d_{ nullptr };
00094 };
00095
00096 } // namespace aubo_scope
00097 } // namespace arcs
00098
00099 #endif // AUBO_SCOPE_TCP_H

```

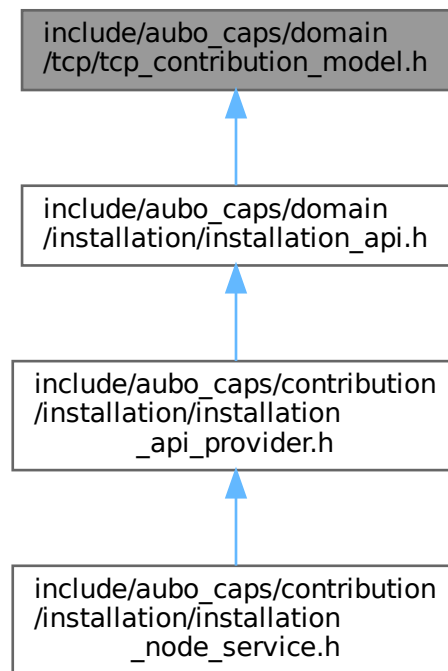
12.216 include/aubo_caps/domain/tcp/tcp_contribution_model.h File Reference

#include <aubo_caps/domain/tcp/tcp.h>

Include dependency graph for tcp_contribution_model.h:



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



Classes

- class [arcs::aubo_scope::TcpContributionModel](#)
[TcpContributionModel](#) This interface provides functionality for adding, updating and removing TCPs in AuboScope.

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([TcpContributionModel](#))

12.217 tcp_contribution_model.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_TCP_CONTRIBUTION_MODEL_H
00002 #define AUBO_SCOPE_TCP_CONTRIBUTION_MODEL_H
00003
00004 #include <aubo_caps/domain/tcp/tcp.h>

```

```

00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(TcpContributionModel);
00009
00010 /**
00011  * @ingroup InstallationApi
00012  * \chinese
00013  * TCP
00014  *     AuboScope      TCP
00015  *
00016  * <b> </b>
00017  * \endchinese
00018  * \english
00019  * TcpContributionModel
00020  * This interface provides functionality for adding, updating and removing TCPs
00021  * in AuboScope.
00022  *
00023  * <b>Note: </b> This interface must only be used in an installation
00024  * contribution.
00025  * \endenglish
00026  */
00027 class ARCS_ABI_EXPORT TcpContributionModel
00028 {
00029 public:
00030     TcpContributionModel(TcpContributionModel &f);
00031     TcpContributionModel(TcpContributionModel &&f);
00032     virtual ~TcpContributionModel();
00033
00034     /**
00035      * @ingroup TcpContributionModel
00036      * @ref TCP
00037      * \chinese
00038      *     AuboScope      TCP      TCP
00039      *
00040      * @param idKey      TCP      AuboCap
00041      * @param suggested_name  TCP
00042      * @param offset TCP
00043      * @return AuboScope      TCP
00044      * @throws TCPAlreadyAddedException      idKey TCP
00045      * @throws IllegalTCPNameException
00046      * @throws IllegalArgumentException
00047      * \endchinese
00048      * \english
00049      * Add a TCP to the current AuboScope installation. This makes it selectable
00050      * by the end user. The TCP is not modifiable by the end user.
00051      *
00052      * @param idKey The key to identify this TCP by.
00053      * @param suggested_name Suggested name for the TCP.
00054      * @param offset The offset of the TCP with respect to the tool output flange.
00055      * @return The TCP created and registered in AuboScope.
00056      * @throws TCPAlreadyAddedException If called twice using the same idKey.
00057      * @throws IllegalTCPNameException If the suggested name does not match regex.
00058      * @throws IllegalArgumentException If the offset is invalid.
00059      * \endenglish
00060      */
00061     TCPPtr addTcp(const std::string &idKey, const std::string &suggested_name,
00062                  const std::vector<double> &offset);
00063
00064     /**
00065      * @ingroup TcpContributionModel
00066      * @ref TCP
00067      * \chinese
00068      *     AuboCap idKey TCP
00069      * @param idKey TCP
00070      * @return TCP      <code>null</code>
00071      * \endchinese
00072      * \english
00073      * Returns the TCP previously added by this AuboCap using the same idKey.
00074      * @param idKey The key to identify this TCP by.
00075      * @return The TCP previously added by this AuboCap.
00076      * \endenglish
00077      */
00078     TCPPtr getTcp(const std::string &idKey);
00079
00080     /**
00081      * @ingroup TcpContributionModel
00082      * \chinese
00083      *     AuboCap TCP
00084      * @param idKey TCP
00085      * @param newOffset TCP
00086      * @throws TCPNotFoundException      idKey TCP
00087      * @throws IllegalArgumentException
00088      * \endchinese
00089      * \english
00090      * Update the offset of a TCP already added by this AuboCap.
00091      * @param idKey The key to identify the TCP.

```

```

00092  * @param newOffset The new offset to set for the TCP.
00093  * @throws TCPNotFoundException If no TCP exists with the provided idKey.
00094  * @throws IllegalArgumentException If the offset is invalid.
00095  * \endenglish
00096  */
00097  void updateTcp(const std::string &idKey,
00098               const std::vector<double> &newOffset);
00099
00100  /**
00101  * @ingroup TcpContributionModel
00102  * \chinese
00103  *   AuboCap    TCP    TCP
00104  * @param idKey  TCP
00105  * @throws TCPNotFoundException    idKey TCP
00106  * \endchinese
00107  * \english
00108  * Remove a TCP added by this AuboCap from AuboScope.
00109  * @param idKey The key used to add the TCP with.
00110  * @throws TCPNotFoundException If no TCP exists with the provided idKey.
00111  * \endenglish
00112  */
00113  void removeTcp(const std::string &idKey);
00114
00115 private:
00116  friend class DataSwitch;
00117  TcpContributionModel();
00118  void *d_{ nullptr };
00119 };
00120
00121 } // namespace aubo_scope
00122 } // namespace arcs
00123
00124 #endif // TCPCONTRIBUTIONMODEL_H

```

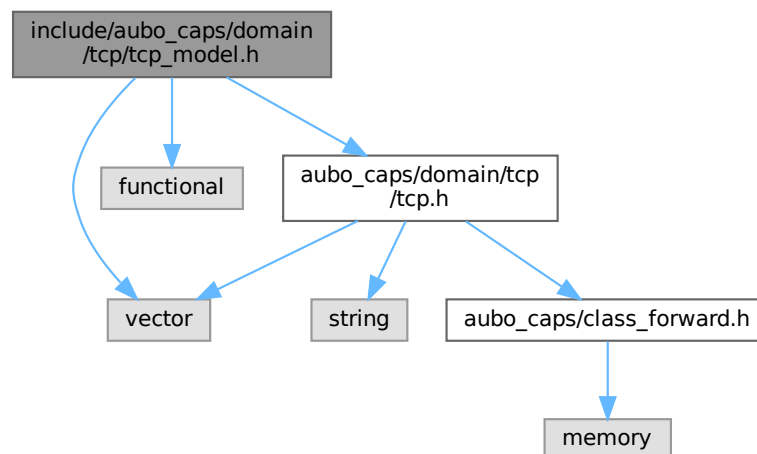
12.218 include/aubo_caps/domain/tcp/tcp_model.h File Reference

```

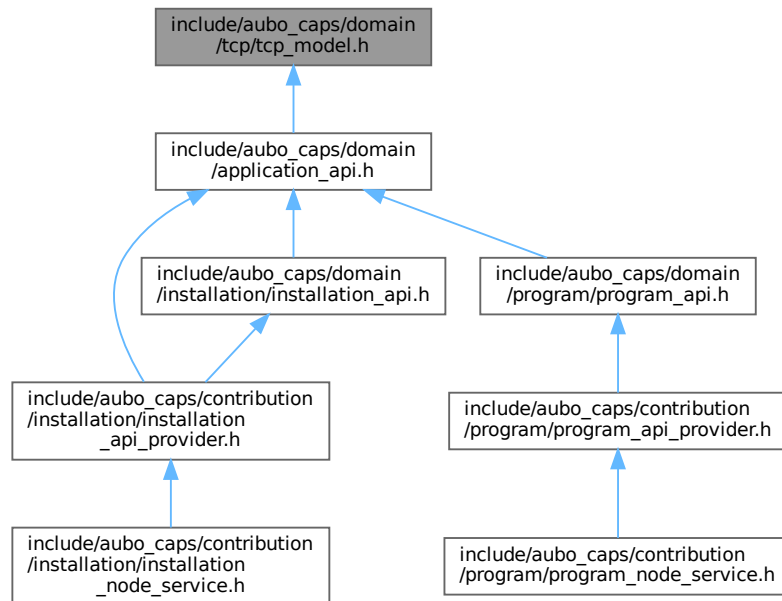
#include <vector>
#include <functional>
#include <aubo_caps/domain/tcp/tcp.h>

```

Include dependency graph for tcp_model.h:



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



Classes

- class [arcs::aubo_scope::TcpModel](#)
[TcpModel](#) This interface provides access to the TCPs that are currently available in AuboScope.

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(TcpModel\)](#)

12.219 tcp_model.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOP_TCP_MODEL_H
00002 #define AUBO_SCOP_TCP_MODEL_H
00003
00004 #include <vector>
00005 #include <functional>
00006 #include <aubo_caps/domain/tcp/tcp.h>
00007
00008 namespace arcs {
00009 namespace aubo_scope {
00010 ARCS_CLASS_FORWARD(TcpModel);
00011

```

```

00012 /**
00013  * @ingroup ApplicationApi
00014  * \chinese
00015  * TCP
00016  *   AuboScope   TCP
00017  * \endchinese
00018  * \english
00019  * TcpModel
00020  * This interface provides access to the TCPs that are currently available in
00021  * AuboScope.
00022  * \endenglish
00023  */
00024 class ARCS_ABI_EXPORT TcpModel
00025 {
00026 public:
00027     TcpModel(TcpModel &f);
00028     TcpModel(TcpModel &&f);
00029     virtual ~TcpModel();
00030
00031     /**
00032     * @ingroup TcpModel
00033     * \chinese
00034     *   AuboScope   TCP
00035     * @return   TCP
00036     * \endchinese
00037     * \english
00038     * Returns a collection of all the TCPs in AuboScope.
00039     * @return A collection of all the TCPs in AuboScope.
00040     * \endenglish
00041     */
00042     std::vector<TCPPtr> getAll();
00043
00044     /**
00045     * @ingroup TcpModel
00046     * @ref TCP
00047     * \chinese
00048     *   TCP
00049     * @param name TCP
00050     * @return   TCP
00051     * \endchinese
00052     * \english
00053     * Get a TCP by name.
00054     * @param name the name of the TCP.
00055     * @return the TCP with the given name.
00056     * \endenglish
00057     */
00058     TCPPtr getTcp(const std::string &name);
00059
00060     /**
00061     * @ingroup TcpModel
00062     * @ref TCP
00063     * \chinese
00064     *   TCP
00065     * @return   TCP
00066     * \endchinese
00067     * \english
00068     * Returns the default TCP.
00069     * @return the default TCP.
00070     * \endenglish
00071     */
00072     TCPPtr getDefaultTcp();
00073
00074     /**
00075     * @ingroup TcpModel
00076     * \chinese
00077     *
00078     * @param receiver
00079     * @param slot
00080     * @return
00081     * \endchinese
00082     * \english
00083     * Register callback for changed.
00084     * @param receiver the receiver identifier.
00085     * @param slot the callback function.
00086     * @return the connection identifier.
00087     * \endenglish
00088     */
00089     std::string connectedTo(const std::string &receiver,
00090                             const std::function<void(TcpModel *)> &slot);
00091
00092     /**
00093     * @ingroup TcpModel
00094     * \chinese
00095     *
00096     * @param receiver
00097     * \endchinese
00098     * \english
00099     * Disconnect a callback.

```

```

00099     * @param receiver the receiver identifier.
00100     * \endenglish
00101     */
00102     void disconnectFrom(const std::string &receiver);
00103
00104 private:
00105     friend class DataSwitch;
00106     TcpModel();
00107     void *d_{ nullptr };
00108 };
00109
00110 } // namespace aubo_scope
00111 } // namespace arcs
00112
00113 #endif // AUBO_SCOP_TCP_MODEL_H

```

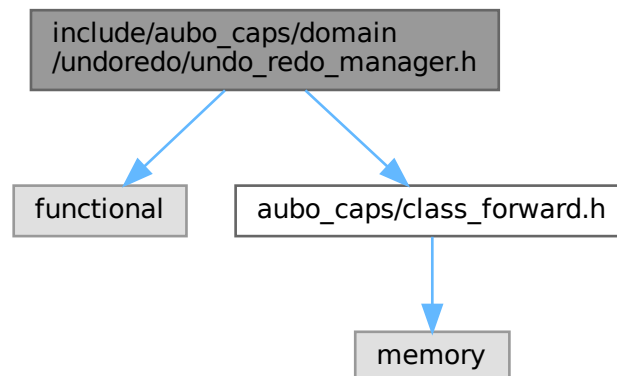
12.220 include/aubo_caps/domain/undoredo/undo_redo_manager.h File Reference

```

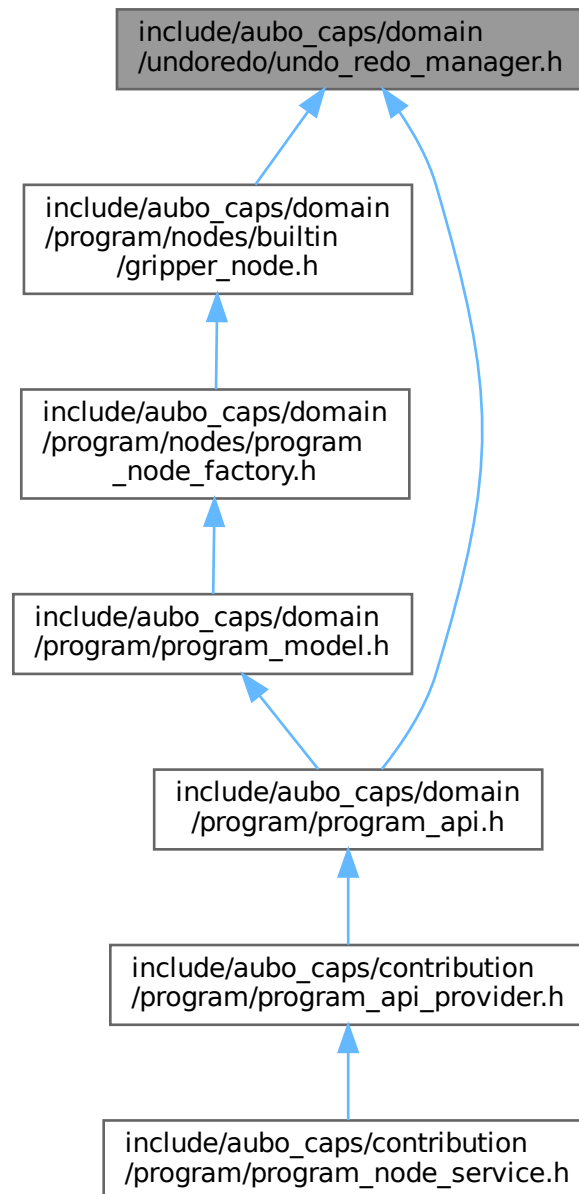
#include <functional>
#include <aubo_caps/class_forward.h>

```

Include dependency graph for undo_redo_manager.h:



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



Classes

- class [arcs::aubo_scope::UndoRedoManager](#)
[UndoRedoManager](#) This interface is used to record changes triggered by an end user in the screen of a AuboCap program node contribution (e.g.

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(UndoRedoManager\)](#)

12.221 undo_redo_manager.h

[Go to the documentation of this file.](#)

```

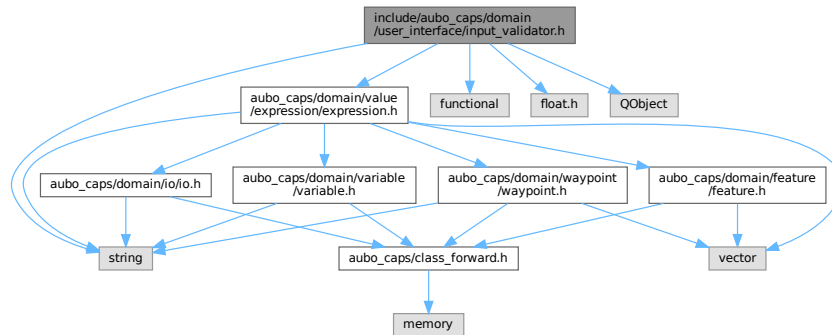
00001 #ifndef AUBO_SCOPE_UNDO_REDO_MANAGER_H
00002 #define AUBO_SCOPE_UNDO_REDO_MANAGER_H
00003
00004 #include <functional>
00005 #include <aubo_caps/class_forward.h>
00006
00007 namespace arcs {
00008 namespace aubo_scope {
00009 ARCS_CLASS_FORWARD(UndoRedoManager);
00010
00011 /**
00012  * @ingroup ProgramApi
00013  * \chinese
00014  * /
00015  *      AuboCap
00016  *      AuboCap      AuboScope  /
00017  *
00018  *      AuboCap      <br>
00019  *      HTML      AuboCap
00020  *
00021  * \endchinese
00022  * \english
00023  * UndoRedoManager
00024  * This interface is used to record changes triggered by an end user in the
00025  * screen of a AuboCap program node contribution (e.g. when a button is
00026  * clicked). It should be used in a AuboCap program node to support the
00027  * Undo/Redo functionality in AuboScope.
00028  *
00029  * NOTE: Do not use this functionality for AuboCap installation nodes, since it
00030  * is not supported (and has no effect). <br>
00031  * NOTE: This functionality is not relevant for AuboCap program nodes with a
00032  * HTML-based user interface, since automatic undo is supported for these.
00033  * \endenglish
00034  */
00035 class ARCS_ABI_EXPORT UndoRedoManager
00036 {
00037 public:
00038     UndoRedoManager(UndoRedoManager &f);
00039     UndoRedoManager(UndoRedoManager &&f);
00040     virtual ~UndoRedoManager();
00041
00042     /**
00043      * \chinese
00044      * <p>
00045      * /
00046      * </p>
00047      * /
00048      *
00049      * @param undoableChanges      /      AuboScope
00050      * " "
00051      * \endchinese
00052      * \english
00053      * <p>
00054      * Group changes into one Undo/Redo step.
00055      * </p>
00056      *
00057      * Currently Undo/Redo supports changes to the program tree and the data
00058      * model.
00059      *
00060      * @param undoableChanges Code to be recorded as one Undo/Redo step.
00061      * Pressing Undo in AuboScope will undo all changes
00062      * done to the program tree and the data model inside this block of code.
00063      * \endenglish
00064      */
00065     void recordChanges(const std::string &text,
00066                       std::function<void()> undoableChanges);
00067
00068 private:
00069     friend class DataSwitch;
00070     UndoRedoManager();
00071     void *d_{ nullptr };
00072 };
00073 } // namespace aubo_scope
00074 } // namespace arcs
00075
00076 #endif

```

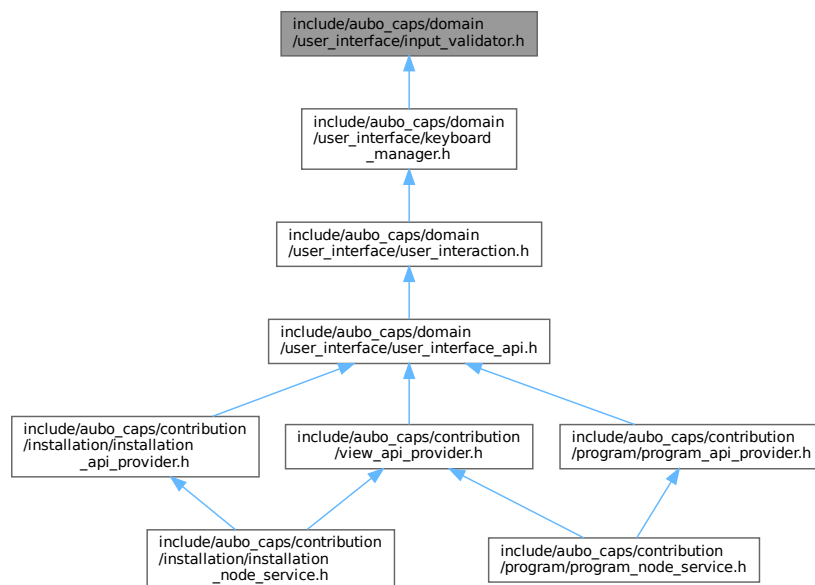
12.222 include/aubo_caps/domain/user_interface/input_validator.h

File Reference

```
#include <string>
#include <functional>
#include <float.h>
#include <QObject>
#include <aubo_caps/domain/value/expression/expression.h>
Include dependency graph for input_validator.h:
```



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



Classes

- struct `arcs::aubo_scope::InputValidator`

[InputValidator](#) Interface representing the input validators created by InputValidationFactory.

- struct [arcs::aubo_scope::IntegerRangeValidator](#)
- struct [arcs::aubo_scope::UIntegerRangeValidator](#)
- struct [arcs::aubo_scope::DoubleRangeValidator](#)
- struct [arcs::aubo_scope::StringLengthValidator](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

12.223 input_validator.h

[Go to the documentation of this file.](#)

```
00001 #ifndef AUBO_SCOPE_KEYBOARD_INPUT_VALIDATOR_H
00002 #define AUBO_SCOPE_KEYBOARD_INPUT_VALIDATOR_H
00003
00004 #include <string>
00005 #include <functional>
00006 #include <float.h>
00007
00008 #include <QObject>
00009
00010 #include <aubo_caps/domain/value/expression/expression.h>
00011
00012 namespace arcs {
00013 namespace aubo_scope {
00014
00015 /**
00016  * \chinese
00017  *
00018  * { @link InputValidationFactory }
00019  * /
00020  *
00021  * { @link InputValidationFactory }
00022  *
00023  *
00024  * @param <T> Integer Double
00025  * \endchinese
00026  * \english
00027  * InputValidator
00028  * Interface representing the input validators created by { @link
00029  * InputValidationFactory }. This factory provides a set of standard input
00030  * validators that can be used for validating the input entered by the user
00031  * using the virtual keyboard/keypad.
00032  *
00033  * If the standard validators available in the { @link InputValidationFactory }
00034  * does not satisfy your needs, you are free to implement your own custom
00035  * validator.
00036  *
00037  * @param <T> the (generic) type parameter for the interface representing the
00038  * type of input data being validated (e.g. Integer or Double).
00039  * \endenglish
00040  */
00041 struct InputValidator
00042 {
00043 /**
00044  * \chinese
00045  *
00046  *
00047  * @param value
00048  * @return <code>true</code>
00049  * \endchinese
00050  * \english
00051  * Validates whether the input value is valid.
00052  *
00053  * @param value to be validated.
00054  * @return <code>true</code> if value is valid.
00055  * \endenglish
00056  */
00057 virtual bool isValid(std::string value) = 0;
00058
00059 /**
00060  * \chinese
```

```

00061  *
00062  *
00063  * @param value
00064  * @return
00065  * \endchinese
00066  * \english
00067  * Returns a meaningful message in case the value is not valid.
00068  *
00069  * @param value the invalid value. Can be included in the message if it
00070  * makes sense.
00071  * @return message.
00072  * \endenglish
00073  */
00074 virtual std::string getMessage(std::string value) = 0;
00075 };
00076
00077 struct IntegerRangeValidator : InputValidator
00078 {
00079     IntegerRangeValidator(int min_value, int max_value);
00080
00081     IntegerRangeValidator(std::function<std::pair<int, int>(>> range_func);
00082
00083     bool isValid(std::string value) override;
00084
00085     std::string getMessage(std::string value) override;
00086
00087 private:
00088     void updateValue();
00089
00090 private:
00091     int min_value_{ INT_MIN };
00092     int max_value_{ INT_MAX };
00093
00094     std::function<std::pair<int, int>(>> range_func_{ nullptr };
00095 };
00096
00097 struct UIntegerRangeValidator : InputValidator
00098 {
00099     UIntegerRangeValidator(uint32_t min_value, uint32_t max_value);
00100
00101     UIntegerRangeValidator(
00102         std::function<std::pair<uint32_t, uint32_t>(>> range_func);
00103
00104     bool isValid(std::string value) override;
00105
00106     std::string getMessage(std::string value) override;
00107
00108 private:
00109     void updateValue();
00110
00111 private:
00112     uint32_t min_value_{ 0 };
00113     uint32_t max_value_{ UINT_MAX };
00114
00115     std::function<std::pair<uint32_t, uint32_t>(>> range_func_{ nullptr };
00116 };
00117
00118 struct DoubleRangeValidator : InputValidator
00119 {
00120     DoubleRangeValidator(double min_value, double max_value,
00121         bool is_min_inclusive = true,
00122         bool is_max_inclusive = true);
00123
00124     DoubleRangeValidator(std::function<std::pair<double, double>(>> range_func);
00125
00126     DoubleRangeValidator(
00127         std::function<std::tuple<double, double, bool, bool>(>> range_func);
00128
00129     bool isValid(std::string value) override;
00130
00131     std::string getMessage(std::string value) override;
00132
00133 private:
00134     void updateValue();
00135
00136 private:
00137     double min_value_{ DBL_MIN };
00138     double max_value_{ DBL_MAX };
00139     std::function<std::pair<double, double>(>> range_func_{ nullptr };
00140     std::function<std::tuple<double, double, bool, bool>(>>
00141         range_func_with_flags_{ nullptr };
00142     bool is_min_inclusive_{ true }; //
00143     bool is_max_inclusive_{ true }; //
00144 };
00145
00146 struct StringLengthValidator : InputValidator
00147 {

```

```

00148   StringLengthValidator(size_t min_length, size_t max_length);
00149
00150   StringLengthValidator(
00151       std::function<std::pair<size_t, size_t>()> range_func);
00152
00153   bool isValid(std::string value) override;
00154
00155   std::string getMessage(std::string value) override;
00156
00157 private:
00158   void updateValue();
00159
00160 private:
00161   size_t min_length_{ 0 };
00162   size_t max_length_{ ULONG_MAX };
00163   std::function<std::pair<size_t, size_t>()> range_func_{ nullptr };
00164 };
00165
00166 } // namespace aubo_scope
00167 } // namespace arcs
00168
00169 #endif

```

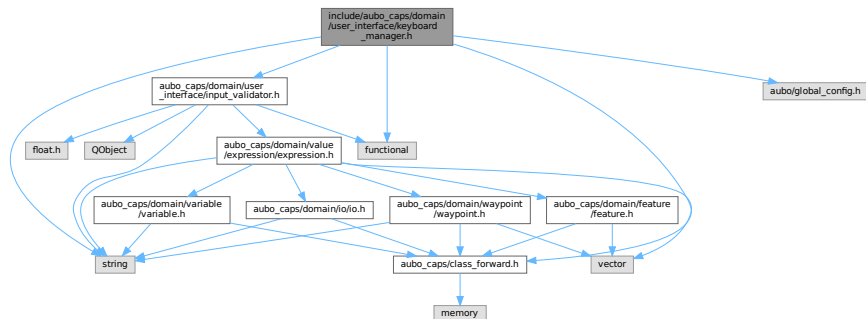
12.224 include/aubo_caps/domain/user_interface/keyboard_manager.h File Reference

```

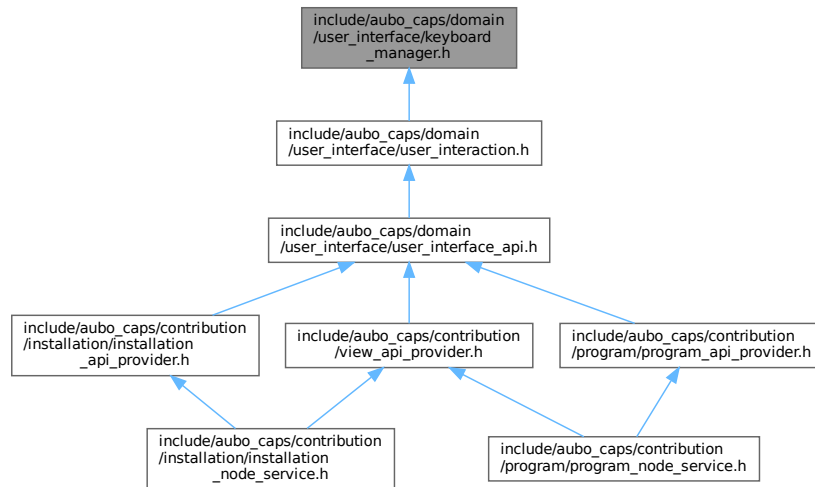
#include <string>
#include <functional>
#include <aubo_caps/class_forward.h>
#include <aubo_caps/domain/user_interface/input_validator.h>
#include <aubo/global_config.h>

```

Include dependency graph for keyboard_manager.h:



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



Classes

- class [arcs::aubo_scope::KeyboardManager](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([KeyboardManager](#))
- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([Expression](#))

12.225 keyboard_manager.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_CAPS_INTERFACE_KEYBOARD_MANAGER_H
00002 #define AUBO_CAPS_INTERFACE_KEYBOARD_MANAGER_H
00003
00004 #include <string>
00005 #include <functional>
00006
00007 #include <aubo_caps/class_forward.h>
00008 #include <aubo_caps/domain/user_interface/input_validator.h>
00009 #include <aubo/global_config.h>
00010 class QWidget;
00011
00012 namespace arcs {
00013 namespace aubo_scope {
00014
00015 ARCS_CLASS_FORWARD(KeyboardManager);
00016 ARCS_CLASS_FORWARD(Expression);
00017
00018 class ARCS_ABI_EXPORT KeyboardManager
00019 {

```

```

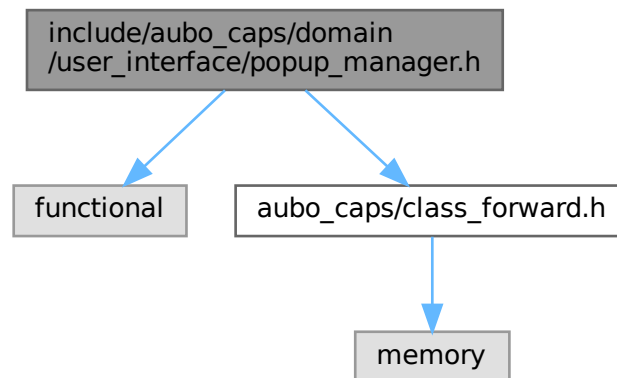
00020 public:
00021     KeyboardManager(KeyboardManager &f);
00022     KeyboardManager(KeyboardManager &&f);
00023     virtual ~KeyboardManager();
00024
00025     /**
00026      * @brief registerStandardKeyboard
00027      * @param wd:         ui->lineedit
00028      * @param init_cb:     edit
00029      * @param validator:
00030      * @param result_cb:   edit
00031      */
00032     ARCS_DEPRECATED void registerStandardKeyboard(
00033         QWidget *wd, std::function<std::string(void)> init_cb = nullptr,
00034         const std::shared_ptr<InputValidator> &validator = nullptr,
00035         std::function<void(std::string)> result_cb = nullptr);
00036     /**
00037      * @brief registerStandardKeyboard
00038      * @param wd:         ui->lineedit
00039      * @param init_cb:
00040      * @param result_cb    Expression    string
00041      * @param validator:
00042      */
00043     void registerStandardKeyboard(
00044         QWidget *wd, std::function<ExpressionPtr(void)> init_cb,
00045         std::function<void(ExpressionPtr)> result_cb,
00046         const std::shared_ptr<InputValidator> &validator = nullptr);
00047
00048     ///
00049     ARCS_DEPRECATED void registerStringKeyboard(
00050         QWidget *wd, std::function<std::string(void)> init_cb = nullptr,
00051         const std::shared_ptr<InputValidator> &validator = nullptr,
00052         std::function<void(std::string)> result_cb = nullptr);
00053     void registerStringKeyboard(
00054         QWidget *wd, std::function<ExpressionPtr(void)> init_cb,
00055         std::function<void(ExpressionPtr)> result_cb,
00056         const std::shared_ptr<InputValidator> &validator = nullptr);
00057
00058     ///
00059     ARCS_DEPRECATED void registerPasswordKeyboard(
00060         QWidget *wd, std::function<std::string(void)> init_cb = nullptr,
00061         const std::shared_ptr<InputValidator> &validator = nullptr,
00062         std::function<void(std::string)> result_cb = nullptr);
00063     void registerPasswordKeyboard(
00064         QWidget *wd, std::function<ExpressionPtr(void)> init_cb,
00065         std::function<void(ExpressionPtr)> result_cb,
00066         const std::shared_ptr<InputValidator> &validator = nullptr);
00067
00068     ///
00069     ARCS_DEPRECATED void registerCodeEditorKeyboard(
00070         QWidget *wd, std::function<std::string(void)> init_cb = nullptr,
00071         const std::shared_ptr<InputValidator> &validator = nullptr,
00072         std::function<void(std::string)> result_cb = nullptr);
00073     void registerCodeEditorKeyboard(
00074         QWidget *wd, std::function<ExpressionPtr(void)> init_cb,
00075         std::function<void(ExpressionPtr)> result_cb,
00076         const std::shared_ptr<InputValidator> &validator = nullptr);
00077
00078     void registerNumberKeyboard(
00079         QWidget *wd, std::function<std::string(void)> init_cb = nullptr,
00080         const std::shared_ptr<InputValidator> &validator = nullptr,
00081         std::function<void(std::string)> result_cb = nullptr);
00082     void registerDoubleKeyboard(
00083         QWidget *wd, std::function<std::string(void)> init_cb = nullptr,
00084         const std::shared_ptr<InputValidator> &validator = nullptr,
00085         std::function<void(std::string)> result_cb = nullptr);
00086     void registerPositiveDoubleKeyboard(
00087         QWidget *wd, std::function<std::string(void)> init_cb = nullptr,
00088         const std::shared_ptr<InputValidator> &validator = nullptr,
00089         std::function<void(std::string)> result_cb = nullptr);
00090     void registerIntegerKeyboard(
00091         QWidget *wd, std::function<std::string(void)> init_cb = nullptr,
00092         const std::shared_ptr<InputValidator> &validator = nullptr,
00093         std::function<void(std::string)> result_cb = nullptr);
00094     void registerPositiveIntegerKeyboard(
00095         QWidget *wd, std::function<std::string(void)> init_cb = nullptr,
00096         const std::shared_ptr<InputValidator> &validator = nullptr,
00097         std::function<void(std::string)> result_cb = nullptr);
00098     void registerIPAddressKeyboard(
00099         QWidget *wd, std::function<std::string(void)> init_cb = nullptr,
00100         std::function<void(std::string)> result_cb = nullptr);
00101     void registerExpressionKeyboard(
00102         QWidget *wd, std::function<ExpressionPtr(void)> init_cb = nullptr,
00103         std::function<void(ExpressionPtr result_value)> result_cb = nullptr,
00104         const std::shared_ptr<InputValidator> validator = nullptr);

```

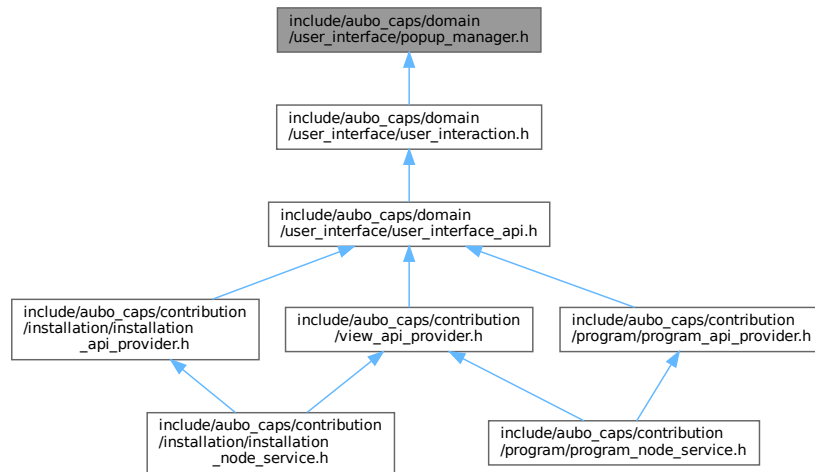
```
00107
00108 private:
00109     friend class DataSwitch;
00110     KeyboardManager();
00111     void *d_{ nullptr };
00112 };
00113
00114 } // namespace aubo_scope
00115 } // namespace arcs
00116
00117 #endif // AUBO_CAPS_INTERFACE_KEYBOARD_MANAGER_H
```

12.226 include/auco_caps/domain/user_interface/popup_manager.h File Reference

#include <functional>
#include <auco_caps/class_forward.h>
Include dependency graph for popup_manager.h:



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



Classes

- class `arcs::aubo_scope::PopupManager`

Namespaces

- namespace `arcs`
- namespace `arcs::aubo_scope`

Functions

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (PopupManager)`

12.227 popup_manager.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_POPUP_MANAGER_H
00002 #define AUBO_SCOPE_POPUP_MANAGER_H
00003
00004 #include <functional>
00005
00006 #include <aubo_caps/class_forward.h>
00007
00008 namespace arcs {
00009 namespace aubo_scope {
00010
00011 template <typename Type>
00012 struct DefaultSingletonTraits;
00013
00014 ARCS_CLASS_FORWARD(PopupManager);
00015
00016 class ARCS_ABI_EXPORT PopupManager
00017 {
00018 public:
00019     PopupManager(PopupManager &f);
00020     PopupManager(PopupManager &&f);
00021     virtual ~PopupManager();

```

```

00022
00023 //
00024 void notifyPopupShown();
00025
00026 //
00027 void notifyPopupHidden();
00028
00029 //
00030 int popupCount() const;
00031
00032 // 0
00033 bool isPopupCountZero() const;
00034
00035 // is_zero==true 0
00036 void setPopupCountChangedCallback(std::function<void(bool)> callback);
00037
00038 private:
00039 friend struct DefaultSingletonTraits<PopupManager>;
00040 PopupManager();
00041 void *d_{ nullptr };
00042 };
00043
00044 } // namespace aubo_scope
00045 } // namespace arcs
00046
00047 #endif // AUBO_SCOPE_POPUP_MANAGER_H

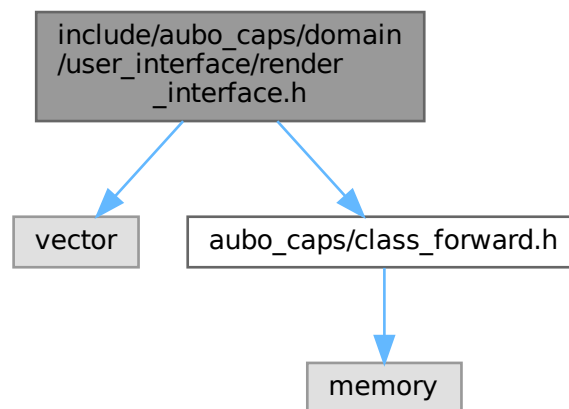
```

12.228 include/auco_caps/domain/user_interface/render_interface.h File Reference

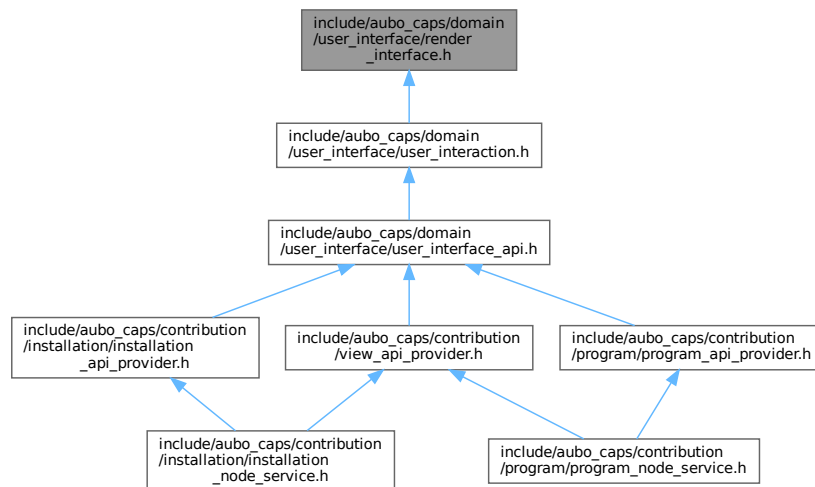
#include <vector>

#include <auco_caps/class_forward.h>

Include dependency graph for render_interface.h:



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



Classes

- class [arcs::aubo_scope::RenderInterface](#)
[RenderInterface](#) Provides display option controls related to 3D rendering, including show/hide of ground, robot models, coordinate systems, trajectories and points.

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(RenderInterface\)](#)

12.229 render_interface.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_CAPS_INTERFACE_RENDER_INTERFACE_H
00002 #define AUBO_CAPS_INTERFACE_RENDER_INTERFACE_H
00003
00004 #include <vector>
00005 #include <aubo_caps/class_forward.h>
00006
00007 class QGridLayout;
00008
00009 namespace arcs {
00010 namespace aubo_scope {
00011
00012 ARCS_CLASS_FORWARD(RenderInterface);
00013
00014 /**
00015  * @ingroup UserInterfaceApi
00016  * \chinese
00017  *

```

```

00018 * 3D
00019 * /
00020 * \endchinese
00021 * \english
00022 * RenderInterface
00023 * Provides display option controls related to 3D rendering, including show/hide
00024 * of ground, robot models, coordinate systems, trajectories and points.
00025 * \endenglish
00026 */
00027
00028 class ARCS_ABI_EXPORT RenderInterface
00029 {
00030 public:
00031     enum ShowOptions : uint16_t
00032     {
00033         None = 0,
00034         Ground = 0x01,
00035         TargetRobot = 0x02,
00036         RealRobot = 0x04,
00037         Coordinate = 0x08,
00038         Track = 0x10,
00039         Point = 0x20,
00040         View = 0x40,
00041         Obstacle = 0x80,
00042         SafetyPlane = 0x0100,
00043         ToolPosition = 0x0200,
00044         PointCloud = 0x0400,
00045         All = 0xFFFF,
00046     };
00047
00048 public:
00049     RenderInterface(RenderInterface &f);
00050     RenderInterface(RenderInterface &&f);
00051     virtual ~RenderInterface();
00052
00053     /**
00054     * @ingroup RenderInterface
00055     * \chinese
00056     * Render / (isEmbedded) gly
00057     *
00058     * @param is_embedded
00059     * @param gly
00060     * \endchinese
00061     * \english
00062     * Set the render window to be embedded/removed in the layout gly.
00063     *
00064     * @param is_embedded whether to embed
00065     * @param gly the layout
00066     * \endenglish
00067     */
00068     void setRenderEmbebed(bool is_embedded, QGridLayout *gly);
00069
00070     /**
00071     * @ingroup RenderInterface
00072     * \chinese
00073     *
00074     *
00075     * @param options ShowOptions
00076     * \endchinese
00077     * \english
00078     * Set the display mode to control the visibility of various elements.
00079     *
00080     * @param options option value, see ShowOptions enum for details
00081     * \endenglish
00082     */
00083     void setShowOptions(uint16_t options);
00084
00085     /**
00086     * @ingroup RenderInterface
00087     * \chinese
00088     * /
00089     *
00090     * @param name
00091     * @param points
00092     * @param color rgba
00093     * \endchinese
00094     * \english
00095     * Add/modify point cloud.
00096     *
00097     * @param name point cloud name
00098     * @param points point collection
00099     * @param color point cloud color rgba
00100     * \endenglish
00101     */
00102     void addPointCloud(const std::string &name,
00103                       const std::vector<std::vector<double> &points,
00104                       const std::vector<double> &color);

```

```

00105
00106 /**
00107  * @ingroup RenderInterface
00108  * \chinese
00109  *
00110  *
00111  * @param name
00112  * \endchinese
00113  * \english
00114  * Remove the point cloud with the specified name.
00115  *
00116  * @param name point cloud name
00117  * \endenglish
00118  */
00119 void removePointCloud(const std::string &name);
00120
00121 /**
00122  * @ingroup RenderInterface
00123  * \chinese
00124  *
00125  *
00126  * @param name
00127  * @retval true
00128  * @retval false
00129  * \endchinese
00130  * \english
00131  * Check whether the point cloud with the specified name exists.
00132  *
00133  * @param name point cloud name
00134  * @retval true exists
00135  * @retval false does not exist
00136  * \endenglish
00137  */
00138 bool hasPointCloud(const std::string &name);
00139
00140 /**
00141  * @ingroup RenderInterface
00142  * \chinese
00143  *
00144  * \endchinese
00145  * \english
00146  * Remove all point clouds.
00147  * \endenglish
00148  */
00149 void clearPointCloud();
00150
00151 /**
00152  * @ingroup RenderInterface
00153  * \chinese
00154  * /
00155  *
00156  * @param wp
00157  * @param pose x,y,z,rx,ry,rz
00158  * @param color {r, g, b, a}, 0.0~1.0
00159  * @param axis_visible
00160  * @retval true
00161  * @retval false
00162  * \endchinese
00163  * \english
00164  * Add/modify waypoint.
00165  *
00166  * @param wp the unique object representing the waypoint
00167  * @param pose waypoint pose x,y,z,rx,ry,rz
00168  * @param color waypoint color {r, g, b, a}, range 0.0~1.0
00169  * @param axis_visible whether to show pose direction arrows on the waypoint
00170  * @retval true success
00171  * @retval false failure
00172  * \endenglish
00173  */
00174 bool addWaypoint(void *wp, const std::vector<double> &pose,
00175                 const std::vector<double> &color = { 0.0, 1.0, 0.0, 1.0 },
00176                 bool axis_visible = false);
00177
00178 /**
00179  * @ingroup RenderInterface
00180  * \chinese
00181  *
00182  *
00183  * @param wp
00184  * \endchinese
00185  * \english
00186  * Remove the specified waypoint.
00187  *
00188  * @param wp the unique object representing the waypoint
00189  * \endenglish
00190  */
00191 void removeWaypoint(void *wp);

```

```

00192
00193 /**
00194  * @ingroup RenderInterface
00195  * \chinese
00196  *
00197  *
00198  * @param wp
00199  * @retval true
00200  * @retval false
00201  * \endchinese
00202  * \english
00203  * Check whether the specified waypoint exists.
00204  *
00205  * @param wp the unique object representing the waypoint
00206  * @retval true exists
00207  * @retval false does not exist
00208  * \endenglish
00209  */
00210 bool hasWaypoint(void *wp);
00211
00212 /**
00213  * @ingroup RenderInterface
00214  * \chinese
00215  *
00216  * \endchinese
00217  * \english
00218  * Remove all waypoints.
00219  * \endenglish
00220  */
00221 void clearWaypoints();
00222
00223 /**
00224  * @ingroup RenderInterface
00225  * \chinese
00226  * /
00227  *
00228  * @param wp
00229  * @param track x,y,z,rx,ry,rz
00230  * \endchinese
00231  * \english
00232  * Add/modify the move track for the specified waypoint.
00233  *
00234  * @param wp the unique object representing the waypoint
00235  * @param track waypoint pose collection x,y,z,rx,ry,rz
00236  * \endenglish
00237  */
00238 void setMoveTrack(void *wp, const std::vector<std::vector<double> &track);
00239
00240 /**
00241  * @ingroup RenderInterface
00242  * \chinese
00243  * @brief setBlendTrack
00244  * @param wp
00245  * @param track
00246  * \endchinese
00247  * \english
00248  * @brief setBlendTrack
00249  * @param wp
00250  * @param track
00251  * \endenglish
00252  */
00253 void setBlendTrack(void *wp, const std::vector<std::vector<double> &track);
00254
00255 /**
00256  * @ingroup RenderInterface
00257  * \chinese
00258  * @brief setMoveTrack
00259  * @param wp
00260  * @param track
00261  * @param color
00262  * \endchinese
00263  * \english
00264  * @brief setMoveTrack
00265  * @param wp
00266  * @param track
00267  * @param color
00268  * \endenglish
00269  */
00270 void setMoveTrack(void *wp, const std::vector<std::vector<double> &track,
00271                  const std::vector<double> &color);
00272
00273 /**
00274  * @ingroup RenderInterface
00275  * \chinese
00276  * @brief setBlendTrack
00277  * @param wp
00278  * @param track

```

```

00279     * @param color
00280     * \endchinese
00281     * \english
00282     * @brief setBlendTrack
00283     * @param wp
00284     * @param track
00285     * @param color
00286     * \endenglish
00287     */
00288 void setBlendTrack(void *wp, const std::vector<std::vector<double> &track,
00289                   const std::vector<double> &color);
00290
00291 /**
00292  * @ingroup RenderInterface
00293  * \chinese
00294  * /
00295  *
00296  * @param name
00297  * @param pose x,y,z,rx,ry,rz
00298  * \endchinese
00299  * \english
00300  * Add/modify coordinate axis.
00301  *
00302  * @param name coordinate name
00303  * @param pose pose x,y,z,rx,ry,rz
00304  * \endenglish
00305  */
00306 void addCoordinate(const std::string &name,
00307                  const std::vector<double> &pose);
00308
00309 /**
00310  * @ingroup RenderInterface
00311  * \chinese
00312  *
00313  *
00314  * @param name
00315  * \endchinese
00316  * \english
00317  * Remove the specified coordinate axis.
00318  *
00319  * @param name coordinate name
00320  * \endenglish
00321  */
00322 void removeCoordinate(const std::string &name);
00323
00324 /**
00325  * @ingroup RenderInterface
00326  * \chinese
00327  *
00328  *
00329  * @param name
00330  * @retval true
00331  * @retval false
00332  * \endchinese
00333  * \english
00334  * Check whether the specified coordinate axis exists.
00335  *
00336  * @param name coordinate name
00337  * @retval true exists
00338  * @retval false does not exist
00339  * \endenglish
00340  */
00341 bool hasCoordinate(const std::string &name);
00342
00343 /**
00344  * @ingroup RenderInterface
00345  * \chinese
00346  * /
00347  *
00348  * @param name
00349  * @param link_num: -1: TCP; 0: ; 1~6: ;
00350  * @param pose: , 6 0
00351  * @param points
00352  * @param color {r, g, b, a}, 0.0~1.0
00353  * \endchinese
00354  * \english
00355  * Add/modify line.
00356  *
00357  * @param name line name
00358  * @param link_num: -1: TCP; 0: base; 1~6: joints;
00359  * @param pose: reference coordinate system for points, default 6 zeros
00360  * @param points point collection
00361  * @param color line color {r, g, b, a}, range 0.0~1.0
00362  * \endenglish
00363  */
00364 void addLine(const std::string &name,
00365             const std::vector<std::vector<double> &points,

```

```

00366         const std::vector<double> &color);
00367
00368     /**
00369     * @ingroup RenderInterface
00370     * \chinese
00371     * @brief addLine
00372     * @param name
00373     * @param link_num
00374     * @param pose
00375     * @param points
00376     * @param color
00377     * \endchinese
00378     * \english
00379     * @brief addLine
00380     * @param name
00381     * @param link_num
00382     * @param pose
00383     * @param points
00384     * @param color
00385     * \endenglish
00386     */
00387 void addLine(const std::string &name, int link_num,
00388             const std::vector<double> &pose,
00389             const std::vector<std::vector<double>> &points,
00390             const std::vector<double> &color);
00391
00392     /**
00393     * @ingroup RenderInterface
00394     * \chinese
00395     *
00396     *
00397     * @param name
00398     * \endchinese
00399     * \english
00400     * Remove the specified line.
00401     *
00402     * @param name line name
00403     * \endenglish
00404     */
00405 void removeLine(const std::string &name);
00406
00407     /**
00408     * @ingroup RenderInterface
00409     * \chinese
00410     *
00411     * \endchinese
00412     * \english
00413     * Remove all lines.
00414     * \endenglish
00415     */
00416 void clearLines();
00417
00418     /**
00419     * @ingroup RenderInterface
00420     * \chinese
00421     * /
00422     *
00423     * @param name
00424     * @param link_num: -1: TCP; 0: ; 1~6: ;
00425     * @param pose
00426     * @param size
00427     * @param color {r, g, b, a}, 0.0~1.0
00428     * \endchinese
00429     * \english
00430     * Add/modify plane.
00431     *
00432     * @param name plane name
00433     * @param link_num: -1: TCP; 0: base; 1~6: joints;
00434     * @param pose position
00435     * @param size plane size (length, width)
00436     * @param color color {r, g, b, a}, range 0.0~1.0
00437     * \endenglish
00438     */
00439 void addPlane(const std::string &name, const std::vector<double> &pose,
00440             const std::vector<double> &size,
00441             const std::vector<double> &color);
00442
00443     /**
00444     * @ingroup RenderInterface
00445     * \chinese
00446     * @brief addPlane
00447     * @param name
00448     * @param link_num
00449     * @param pose
00450     * @param size
00451     * @param color
00452     * \endchinese

```



```

00453 * \english
00454 * @brief addPlane
00455 * @param name
00456 * @param link_num
00457 * @param pose
00458 * @param size
00459 * @param color
00460 * \endenglish
00461 */
00462 void addPlane(const std::string &name, int link_num,
00463              const std::vector<double> &pose,
00464              const std::vector<double> &size,
00465              const std::vector<double> &color);
00466
00467 /**
00468 * @ingroup RenderInterface
00469 * \chinese
00470 *
00471 *
00472 * @param name
00473 * \endchinese
00474 * \english
00475 * Remove the specified plane.
00476 *
00477 * @param name plane name
00478 * \endenglish
00479 */
00480 void removePlane(const std::string &name);
00481
00482 /**
00483 * @ingroup RenderInterface
00484 * \chinese
00485 *
00486 * \endchinese
00487 * \english
00488 * Remove all planes.
00489 * \endenglish
00490 */
00491 void clearPlanes();
00492
00493 /**
00494 * @ingroup RenderInterface
00495 * \chinese
00496 * /
00497 *
00498 * @param name
00499 * @param link_num: -1: TCP; 0: ; 1~6: ;
00500 * @param pose
00501 * @param size
00502 * @param color {r, g, b, a}, 0.0~1.0
00503 * @param type 0 6 1
00504 * \endchinese
00505 * \english
00506 * Add/modify cube.
00507 *
00508 * @param name cube name
00509 * @param link_num: -1: TCP; 0: base; 1~6: joints;
00510 * @param pose position
00511 * @param size cube size (length, width, height)
00512 * @param color cube color {r, g, b, a}, range 0.0~1.0
00513 * @param type cube rendering type: 0 for 6 planes, 1 for wireframe
00514 * \endenglish
00515 */
00516 void addCube(const std::string &name, const std::vector<double> &pose,
00517             const std::vector<double> &size,
00518             const std::vector<double> &color, int type = 0);
00519
00520 /**
00521 * @ingroup RenderInterface
00522 * \chinese
00523 * @brief addCube
00524 * @param name
00525 * @param link_num
00526 * @param pose
00527 * @param size
00528 * @param color
00529 * @param type
00530 * \endchinese
00531 * \english
00532 * @brief addCube
00533 * @param name
00534 * @param link_num
00535 * @param pose
00536 * @param size
00537 * @param color
00538 * @param type
00539 * \endenglish
00540 */

```

```

00540 void addCube(const std::string &name, int link_num,
00541             const std::vector<double> &pose,
00542             const std::vector<double> &size,
00543             const std::vector<double> &color, int type = 0);
00544
00545 /**
00546  * @ingroup RenderInterface
00547  * \chinese
00548  * /
00549  *
00550  * @param name
00551  * @param pose
00552  * @param length_min
00553  * @param length_max
00554  * @param width_min
00555  * @param width_max
00556  * @param height_min
00557  * @param height_max
00558  * @param color {r, g, b, a}, 0.0~1.0
00559  * @param type 0 6 1
00560  * \endchinese
00561  * \english
00562  * Add/modify cube with size ranges.
00563  *
00564  * @param name cube name
00565  * @param pose position
00566  * @param length_min
00567  * @param length_max
00568  * @param width_min
00569  * @param width_max
00570  * @param height_min
00571  * @param height_max
00572  * @param color cube color {r, g, b, a}, range 0.0~1.0
00573  * @param type cube rendering type: 0 for 6 planes, 1 for wireframe
00574  * \endenglish
00575  */
00576 void addCube(const std::string &name, const std::vector<double> &pose,
00577             float length_min, float length_max, float width_min,
00578             float width_max, float height_min, float height_max,
00579             const std::vector<double> &color, int type = 0);
00580
00581 /**
00582  * @ingroup RenderInterface
00583  * \chinese
00584  * @brief addCube
00585  * @param name
00586  * @param link_num
00587  * @param pose
00588  * @param length_min
00589  * @param length_max
00590  * @param width_min
00591  * @param width_max
00592  * @param height_min
00593  * @param height_max
00594  * @param color
00595  * @param type
00596  * \endchinese
00597  * \english
00598  * @brief addCube
00599  * @param name
00600  * @param link_num
00601  * @param pose
00602  * @param length_min
00603  * @param length_max
00604  * @param width_min
00605  * @param width_max
00606  * @param height_min
00607  * @param height_max
00608  * @param color
00609  * @param type
00610  * \endenglish
00611  */
00612 void addCube(const std::string &name, int link_num,
00613             const std::vector<double> &pose, float length_min,
00614             float length_max, float width_min, float width_max,
00615             float height_min, float height_max,
00616             const std::vector<double> &color, int type = 0);
00617
00618 /**
00619  * @ingroup RenderInterface
00620  * \chinese
00621  *
00622  *
00623  * @param name
00624  * \endchinese
00625  * \english
00626  * Remove the specified cube.

```

```

00627     *
00628     * @param name cube name
00629     * \endenglish
00630     */
00631 void removeCube(const std::string &name);
00632
00633 /**
00634  * @ingroup RenderInterface
00635  * \chinese
00636  *
00637  * \endchinese
00638  * \english
00639  * Remove all cubes.
00640  * \endenglish
00641  */
00642 void clearCubes();
00643
00644 //
00645 /**
00646  * @ingroup RenderInterface
00647  * @brief addMesh
00648  * @param name:
00649  * @param link_num: -1: TCP; 0:  ; 1~6:  ;
00650  * @param model_path: 3ds
00651  * @param pose:      {x, y, z, rx, ry, rz}
00652  * @return 0:  ;
00653  */
00654 int addMesh(const std::string &name, int link_num,
00655             const std::string &model_path, const std::vector<double> &pose);
00656
00657 /**
00658  * @ingroup RenderInterface
00659  * @brief getMeshes link_num
00660  * @param link_num: -1: TCP; 0:  ; 1~6:  ;
00661  * @return
00662  */
00663 std::vector<std::string> getMeshes(int link_num);
00664
00665 /**
00666  * @ingroup RenderInterface
00667  * @brief removeMesh
00668  * @param name
00669  * @return
00670  */
00671 int removeMesh(const std::string &name);
00672
00673 /**
00674  * @ingroup RenderInterface
00675  * @brief clearMeshes
00676  * @param link_num
00677  * @return
00678  */
00679 int clearMeshes(int link_num);
00680
00681 private:
00682 friend class DataSwitch;
00683 RenderInterface();
00684 void *d_{ nullptr };
00685 };
00686
00687 } // namespace aubo_scope
00688 } // namespace arcs
00689
00690 #endif // AUBO_CAPS_INTERFACE_RENDER_INTERFACE_H

```

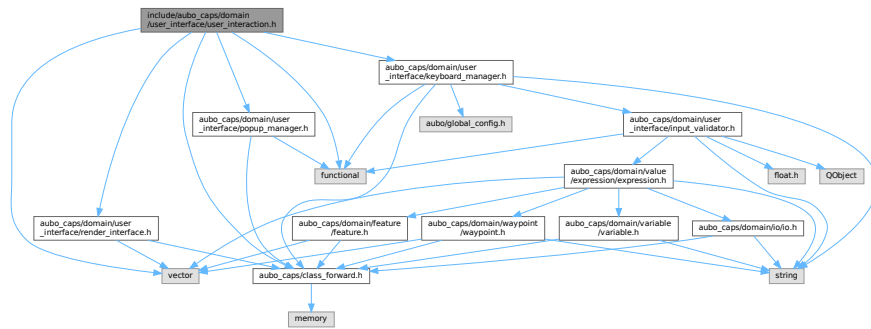
12.230 include/aubo_caps/domain/user_interface/user_interaction.h File Reference

```

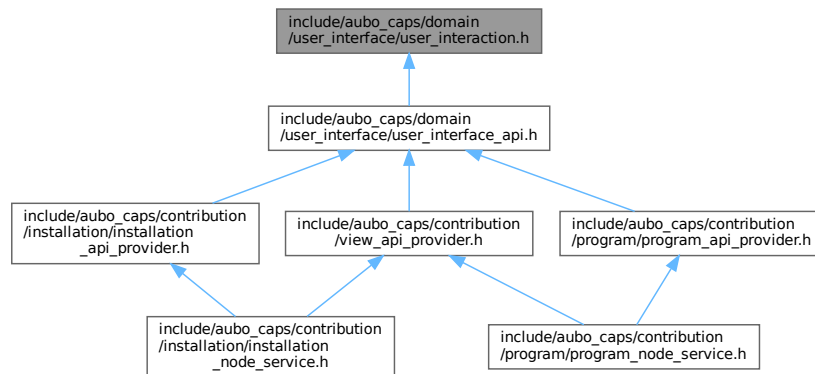
#include <vector>
#include <functional>
#include <aubo_caps/class_forward.h>
#include <aubo_caps/domain/user_interface/keyboard_manager.h>
#include <aubo_caps/domain/user_interface/render_interface.h>
#include <aubo_caps/domain/user_interface/popup_manager.h>

```

Include dependency graph for `user_interaction.h`:



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



Classes

- class [arcs::aubo_scope::UserInteraction](#)
[UserInteraction](#) This interface provides functionality for requesting input or actions from end users.

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Typedefs

- using [arcs::aubo_scope::RobotPositionCallback](#)
- using [arcs::aubo_scope::RobotMovementCallback](#) = `std::function<void(bool accepted, int error)>`

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(UserInteraction\)](#)
- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(TCP\)](#)

12.231 user_interaction.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_USER_INTERACTION_H
00002 #define AUBO_SCOPE_USER_INTERACTION_H
00003
00004 #include <vector>
00005 #include <functional>
00006
00007 #include <aubo_caps/class_forward.h>
00008 #include <aubo_caps/domain/user_interface/keyboard_manager.h>
00009 #include <aubo_caps/domain/user_interface/render_interface.h>
00010 #include <aubo_caps/domain/user_interface/popup_manager.h>
00011
00012 namespace arcs {
00013 namespace aubo_scope {
00014
00015 ARCS_CLASS_FORWARD(UserInteraction);
00016 ARCS_CLASS_FORWARD(TCP);
00017
00018 using RobotPositionCallback = std::function<void(
00019     bool accepted, const std::vector<double> &pose,
00020     const std::vector<double> &q, const std::vector<double> &offset)>;
00021 using RobotMovementCallback = std::function<void(bool accepted, int error)>;
00022
00023 /**
00024  * @ingroup UserInterfaceApi
00025  * \chinese
00026  *
00027  *
00028  * \endchinese
00029  * \english
00030  * UserInteraction
00031  * This interface provides functionality for requesting input or actions from
00032  * end users.
00033  * \endenglish
00034  */
00035 class ARCS_ABI_EXPORT UserInteraction
00036 {
00037 public:
00038     UserInteraction(UserInteraction &f);
00039     UserInteraction(UserInteraction &&f);
00040     virtual ~UserInteraction();
00041
00042     /**
00043      * @ingroup UserInteraction
00044      * <p>
00045      * Request the end user to use the robot to define a robot position.
00046      * Override the
00047      * {@link RobotPositionCallback2#onOk(PositionParameters)} method to execute
00048      * code once the end user is done. Optionally override the {@link
00049      * RobotPositionCallback2#onCancel()} method to execute code if the end user
00050      * cancels the operation.
00051      * </p>
00052      *
00053      * <p>
00054      * This method is asynchronous, i.e. the method will return immediately.
00055      * Only when the end user is done, either
00056      * {@link RobotPositionCallback2#onOk(PositionParameters)} or {@link
00057      * RobotPositionCallback2#onCancel()} will be called.
00058      * </p>
00059      *
00060      * NOTE: This functionality should not be used in a toolbar contribution
00061      * (see {@link ToolbarService}).
00062      *
00063      * @param callback the instance callbacks will be made to.
00064      */
00065     void getUserDefinedRobotPosition(RobotPositionCallback callback);
00066
00067     /**
00068      * @ingroup UserInteraction
00069      * \chinese
00070      * tcpOffset
00071      *
00072      * @param tcpOffset TCP
00073      * @param callback
00074      * \endchinese
00075      * \english
00076      * Request the end user to define a robot position with specified tcpOffset.
00077      *
00078      * @param tcpOffset
00079      * @param callback
00080      * \endenglish
00081      */
00082     void getUserDefinedRobotPosition(const std::vector<double> &tcpOffset,

```

```

00083         RobotPositionCallback callback);
00084
00085     /**
00086     * @ingroup UserInteraction
00087     * \chinese
00088     * tcpOffset
00089     *
00090     * @param tcpOffset TCP
00091     * @param callback
00092     * \endchinese
00093     * \english
00094     * Request the end user to define a robot position with specified tcpOffset string.
00095     *
00096     * @param tcpOffset
00097     * @param callback
00098     * \endenglish
00099     */
00100 void getUserDefinedRobotPosition(const std::string &tcpOffset,
00101                                 RobotPositionCallback callback);
00102
00103     /**
00104     * @ingroup UserInteraction
00105     * \chinese
00106     * TCP
00107     *
00108     * @param tcpOffset TCP
00109     * @param callback
00110     * \endchinese
00111     * \english
00112     * Request the end user to define a robot position with specified TCP object.
00113     *
00114     * @param tcpOffset
00115     * @param callback
00116     * \endenglish
00117     */
00118 void getUserDefinedRobotPosition(const TCPPtr tcpOffset,
00119                                 RobotPositionCallback callback);
00120
00121     /**
00122     * @ingroup UserInteraction
00123     * <p>
00124     * This method provides a factory for creating keyboard inputs which are
00125     * used to configure a virtual keyboard/keypad and to request it to be
00126     * displayed for a Qt GUI component.
00127     * </p>
00128     *
00129     * NOTE: This functionality is only relevant for AuboCap nodes with a
00130     * Qt-based user interface (see {@link ProgramNodeService},
00131     * {@link InstallationNodeService} and {@link ToolbarService}).
00132     *
00133     * @return factory providing access to keyboard inputs.
00134     */
00135 KeyboardManagerPtr getKeyboardManager();
00136
00137     /**
00138     * @ingroup UserInteraction
00139     * <p>
00140     * Transition to the move tab widget to allow/request the end user to move
00141     * the robot (either automatically or manually) to a desired target position
00142     * specified by joint positions.
00143     * </p>
00144     *
00145     * <p>
00146     * The Automove function will move the robot to the target joint positions
00147     * linearly in joint space.
00148     * </p>
00149     *
00150     * <b>NOTE:</b> This method does not take the currently active TCP offset
00151     * into account, i.e. the Automove function will have the specified joint
00152     * positions as target.
00153     *
00154     * @param q The target joint positions to move to.
00155     * @param callback The instance which callbacks will be made to.
00156     * @throws IllegalStateException If the plugin screen is not active
00157     * (visible).
00158     */
00159 int requestUserToMoveJoint(const std::vector<double> &q,
00160                           RobotMovementCallback callback);
00161
00162     /**
00163     * @ingroup UserInteraction
00164     * @brief requestUserToMoveJoint
00165     * @param q
00166     * @param callback
00167     * @return
00168     */
00169 int requestUserToMoveJoint(const std::string &q,
00170                           RobotMovementCallback callback);

```

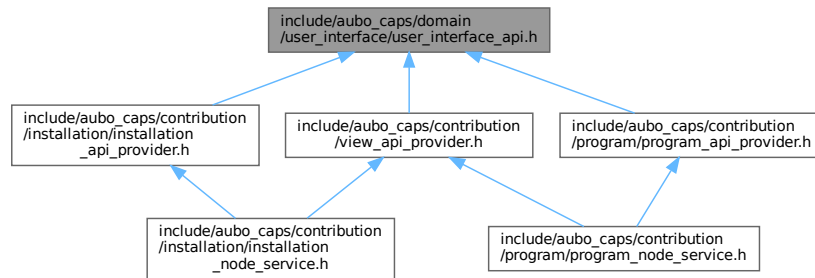
```

00170  /**
00171  * @ingroup UserInteraction
00172  * @param pose The target pose the robot TCP will move to.
00173  * @param q The reference joint Angle.
00174  * @param callback The instance which callbacks will be made to.
00175  * @throws IllegalStateException If the plugin screen is not active
00176  * (visible).
00177  */
00178  int requestUserToMoveJoint(const std::vector<double> &pose,
00179                          const std::vector<double> &q,
00180                          RobotMovementCallback callback);
00181
00182  /**
00183  * @ingroup UserInteraction
00184  * <p>
00185  * Transition to the move tab widget to allow/request the end user to move
00186  * the robot (either automatically or manually) to a desired target position
00187  * specified by a pose.
00188  * </p>
00189  *
00190  * <p>
00191  * The Automove function will move the robot TCP to the target position
00192  * linearly in Cartesian space. If this it not possible, the robot will move
00193  * to the target linearly in joint space.
00194  * </p>
00195  *
00196  * The current joint positions of the robot will be used as starting point
00197  * for inverse kinematics to calculate a target joint vector corresponding
00198  * to desired pose, taking the currently active TCP offset into account.
00199  *
00200  * @param pose The target pose the robot TCP will move to.
00201  * @param callback The instance which callbacks will be made to.
00202  * @throws IllegalStateException If the plugin screen is not active
00203  * (visible).
00204  */
00205  QT_DEPRECATED int requestUserToMoveLine(const std::vector<double> &pose,
00206                                         RobotMovementCallback callback);
00207
00208  /**
00209  * @ingroup UserInteraction
00210  * <p>
00211  * Transition to the move tab widget to allow/request the end user to move
00212  * the robot (either automatically or manually) to a desired target position
00213  * specified by a pose.
00214  * </p>
00215  *
00216  * <p>
00217  * The Automove function will move the robot TCP to the target position
00218  * linearly in Cartesian space. If this it not possible, the robot will move
00219  * to the target linearly in joint space.
00220  * </p>
00221  *
00222  * The current joint positions of the robot will be used as starting point
00223  * for inverse kinematics to calculate a target joint vector corresponding
00224  * to desired pose, taking the currently active TCP offset into account.
00225  *
00226  * @param pose The target pose the robot TCP will move to.
00227  * @param tcpOffset
00228  * @param callback The instance which callbacks will be made to.
00229  * @throws IllegalStateException If the plugin screen is not active
00230  * (visible).
00231  */
00232  int requestUserToMoveLine(const std::vector<double> &pose,
00233                          const std::vector<double> &tcpOffset,
00234                          RobotMovementCallback callback);
00235
00236  /**
00237  * @ingroup UserInteraction
00238  * \chinese
00239  *
00240  *
00241  * @param q
00242  * @param pose
00243  * @return
00244  * \endchinese
00245  * \english
00246  * Get robot thumbnail.
00247  *
00248  * @param q joint angles
00249  * @param pose pose
00250  * @return robot thumbnail
00251  * \endenglish
00252  */
00253  QPixmap *getRobotThumb(const std::vector<double> &q,
00254                       const std::vector<double> &pose);
00255
00256  /**

```

12.232 include/aubo_caps/domain/user_interface/user_interface_↵
api.h File Reference

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



Classes

- class [arcs::aubo_scope::UserInterfaceApi](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(UserInterfaceApi\)](#)

12.233 user_interface_api.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_USER_INTERFACE_API_H
00002 #define AUBO_SCOPE_USER_INTERFACE_API_H
00003
00004 #include <aubo_caps/domain/user_interface/user_interaction.h>
00005
00006 namespace arcs {
00007     namespace aubo_scope {
00008         ARCS_CLASS_FORWARD(UserInterfaceApi);
00009
00010         /**
00011          * \chinese
00012          * @defgroup UserInterfaceApi UserInterfaceApi( )
00013          * @ingroup UserInterfaceApi
00014          *
00015          *   AuboScope
00016          * \endchinese
00017          * \english
00018          * @defgroup UserInterfaceApi UserInterfaceApi
00019          * @ingroup UserInterfaceApi
00020          *   UserInterfaceApi
00021          * Provides access to functionality and services available from within AuboScope
00022          * related to user interface and end user interaction.
00023          * \endenglish
00024          */
00025         class ARCS_ABI_EXPORT UserInterfaceApi
00026         {
00027         public:
00028             UserInterfaceApi(UserInterfaceApi &f);
00029             UserInterfaceApi(UserInterfaceApi &&f);
00030             virtual ~UserInterfaceApi();

```


Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([Expression](#))

12.235 expression.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_EXPRESSION_H
00002 #define AUBO_SCOPE_EXPRESSION_H
00003
00004 #include <string>
00005 #include <vector>
00006
00007 #include <aubo_caps/domain/variable/variable.h>
00008 #include <aubo_caps/domain/feature/feature.h>
00009 #include <aubo_caps/domain/waypoint/waypoint.h>
00010 #include <aubo_caps/domain/io/io.h>
00011
00012 namespace arcs {
00013     namespace aubo_scope {
00014         ARCS_CLASS_FORWARD(Expression);
00015
00016         /**
00017          * \chinese
00018          *
00019          *         If Wait
00020          *     true false
00021          *
00022          *     {@link ExpressionBuilder}
00023          * \endchinese
00024          * \english
00025          * Expression
00026          * An expression can be used for, e.g. configuration of various built-in program
00027          * nodes, such as If and Wait nodes. Expressions are typically conditional
00028          * expressions (which evaluates to true or false).
00029          *
00030          * An Expression is built using the {@link ExpressionBuilder} interface.
00031          * \endenglish
00032          */
00033         class ARCS_ABI_EXPORT Expression
00034         {
00035         public:
00036             Expression(Expression &f);
00037             Expression(Expression &&f);
00038             virtual ~Expression();
00039
00040             std::string toString();
00041
00042             /**
00043              * \chinese
00044              *
00045              * @param expressionPart
00046              * \endchinese
00047              * \english
00048              * Append a string part of the expression.
00049              * @param expressionPart the expression part to be appended.
00050              * \endenglish
00051              */
00052             void append(const std::string &expressionPart);
00053
00054             /**
00055              * \chinese          \endchinese
00056              * \english Append a variable object to the expression. @param variable the variable. \endenglish
00057              */
00058             void appendVariable(VariablePtr variable);
00059
00060             /**
00061              * \chinese          \endchinese
00062              * \english Append a Feature object to the expression. @param feature the Feature. \endenglish
00063              */
00064             void appendFeature(FeaturePtr feature);

```

```

00065
00066 /**
00067  * \chinese      \endchinese
00068  * \english Append a waypoint to the expression. @param waypoint the waypoint. \endenglish
00069  */
00070 void appendWaypoint(WaypointPtr waypoint);
00071
00072 /**
00073  * \chinese      I/O \endchinese
00074  * \english Append an I/O object to the expression. @param io the I/O. \endenglish
00075  */
00076 void appendIo(IoPtr io);
00077
00078 private:
00079 friend class DataSwitch;
00080 Expression();
00081 void *d_{ nullptr };
00082 };
00083
00084 } // namespace aubo_scope
00085 } // namespace arcs
00086 #endif // AUBO_SCOPE_EXPRESSION_H

```

12.236 include/auco_caps/domain/value/value_factory_provider.h

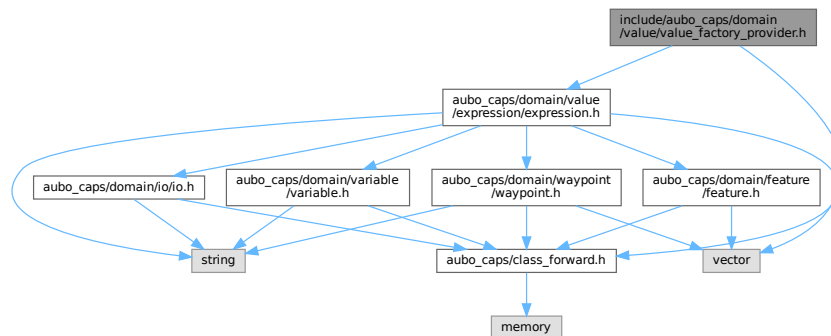
File Reference

```

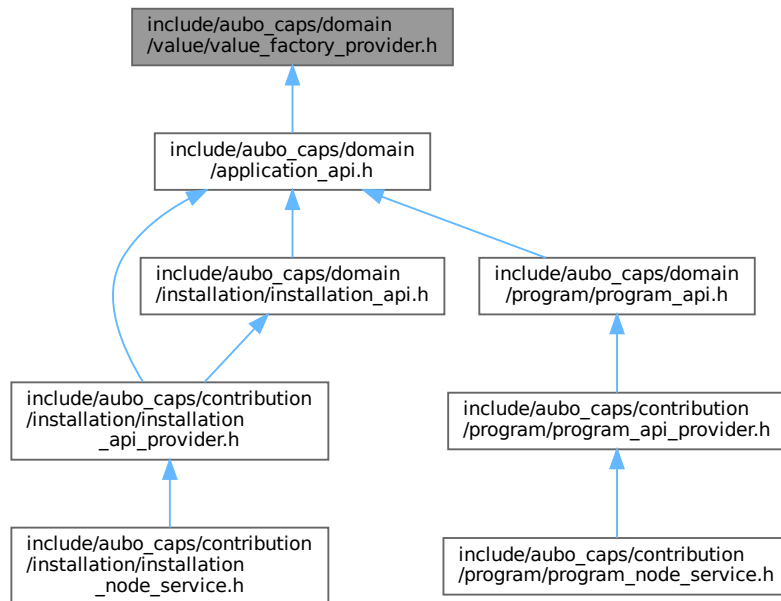
#include <auco_caps/class_forward.h>
#include <auco_caps/domain/value/expression/expression.h>

```

Include dependency graph for value_factory_provider.h:



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



Classes

- class `arcs::aubo_scope::ValueFactory`
ValueFactory

Namespaces

- namespace `arcs`
- namespace `arcs::aubo_scope`

Functions

- `arcs::aubo_scope::ARCS_CLASS_FORWARD (ValueFactory)`

12.237 value_factory_provider.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_VALUE_FACTORY_H
00002 #define AUBO_SCOPE_VALUE_FACTORY_H
00003
00004 #include <aubo_caps/class_forward.h>
00005 #include <aubo_caps/domain/value/expression/expression.h>
00006
00007 namespace arcs {
00008 namespace aubo_scope {
00009 ARCS_CLASS_FORWARD(ValueFactory);
00010
00011 /**

```

```

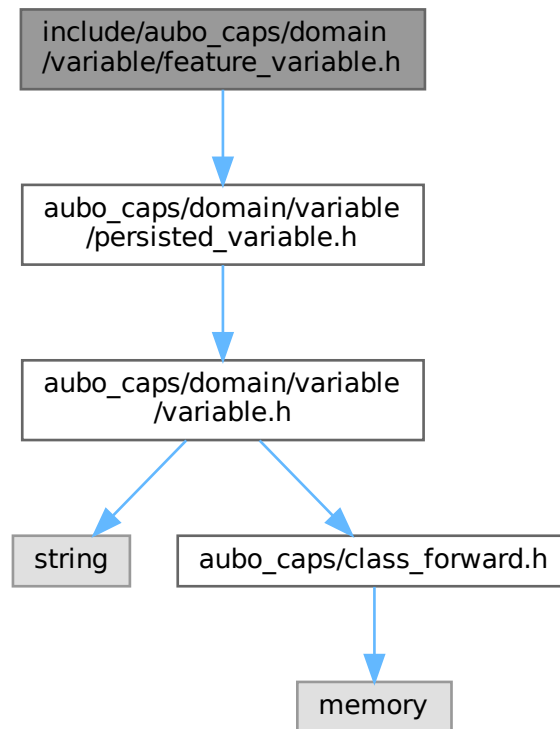
00012 * @ingroup ApplicationApi
00013 * \chinese
00014 *
00015 * <h1> </h1>
00016 *      {@link SimpleValueFactory}
00017 *      {@link ExpressionBuilder}      {@link BlendFactory}
00018 *      {@link JointPositionFactory}
00019 *      {@link PoseFactory}
00020 * \endchinese
00021 * \english
00022 * ValueFactory
00023 * <h1>Summary</h1>
00024 * This interface provides a mean to create immutable value objects using
00025 * factories. The {@link SimpleValueFactory} helps creating simple value
00026 * objects. The {@link ExpressionBuilder} helps building valid expressions. The
00027 * {@link BlendFactory} helps creating blend objects. The {@link
00028 * JointPositionFactory} helps creating joint positions objects. The {@link
00029 * PoseFactory} helps creating Pose objects.
00030 * \endenglish
00031 */
00032 class ARCS_ABI_EXPORT ValueFactory
00033 {
00034 public:
00035     ValueFactory(ValueFactory &f);
00036     ValueFactory(ValueFactory &&f);
00037     virtual ~ValueFactory();
00038
00039     /**
00040     * \chinese
00041     *      {@link ExpressionBuilder}
00042     * @return
00043     * \endchinese
00044     * \english
00045     * This method instantiates a new {@link ExpressionBuilder} to create a
00046     * single valid expression.
00047     *
00048     * @return the builder to build a single expression.
00049     * \endenglish
00050     */
00051     ExpressionPtr createExpression();
00052     ExpressionPtr createExpression(const std::string &str);
00053
00054 private:
00055     friend class DataSwitch;
00056     ValueFactory();
00057     void *d_{ nullptr };
00058 };
00059 } // namespace aubo_scope
00060 } // namespace arcs
00061
00062 #endif

```

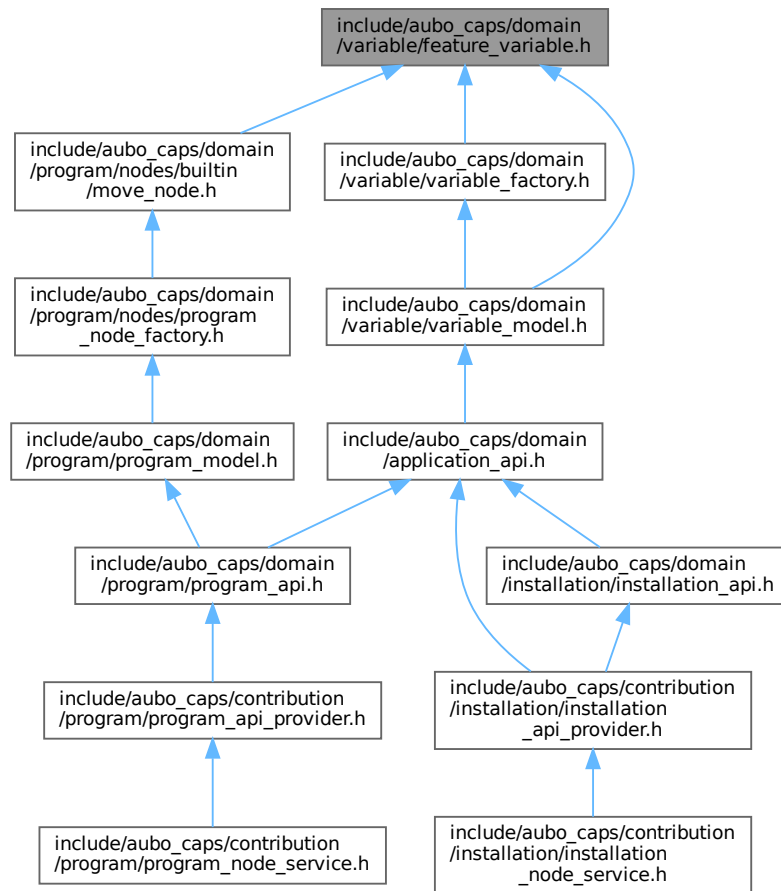
12.238 include/aubo_caps/domain/variable/feature_variable.h File Reference

#include <aubo_caps/domain/variable/persisted_variable.h>

Include dependency graph for feature_variable.h:



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



Classes

- class `arcs::aubo_scope::FeatureVariable`
`FeatureVariable` A `Feature` marked as a `Variable` in the installation will exist even when a robot is turned off and on.
- struct `arcs::aubo_scope::FeatureVariableFilter`

Namespaces

- namespace `arcs`
- namespace `arcs::aubo_scope`

Functions

- `arcs::aubo_scope::ARCS_CLASS_FORWARD` (`FeatureVariable`)

12.239 feature__variable.h

[Go to the documentation of this file.](#)

```

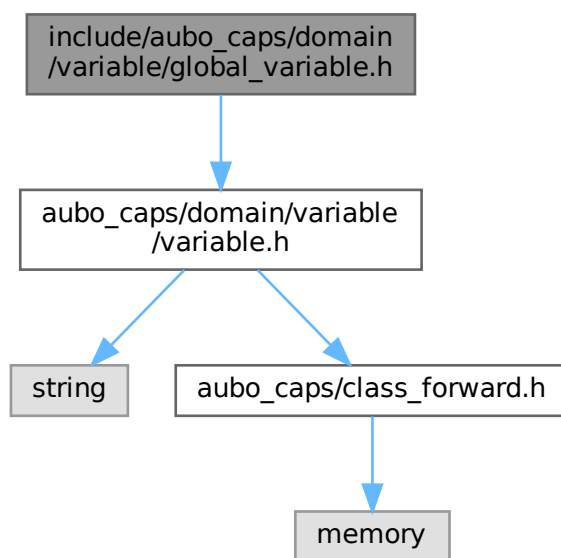
00001 #ifndef AUBO_SCOPE_FEATURE_VARIABLE_H
00002 #define AUBO_SCOPE_FEATURE_VARIABLE_H
00003
00004 #include <aubo_caps/domain/variable/persisted__variable.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(FeatureVariable);
00009
00010 /**
00011  * \chinese
00012  *
00013  *
00014  * \endchinese
00015  * \english
00016  * FeatureVariable
00017  * A Feature marked as a Variable in the installation will exist even when a
00018  * robot is turned off and on.
00019  * \endenglish
00020  */
00021 class ARCS_ABI_EXPORT FeatureVariable : public PersistedVariable
00022 {
00023 public:
00024     FeatureVariable(FeatureVariable &f);
00025     FeatureVariable(FeatureVariable &&f);
00026     virtual ~FeatureVariable();
00027
00028 private:
00029     friend class DataSwitch;
00030     FeatureVariable();
00031     void *d_{ nullptr };
00032 };
00033
00034 struct FeatureVariableFilter
00035 {
00036     bool operator()(VariablePtr var)
00037     {
00038         return std::dynamic_pointer_cast<FeatureVariable>(var) != nullptr &&
00039             var->getType() == Variable::FEATURE;
00040     }
00041 };
00042
00043 } // namespace aubo_scope
00044 } // namespace arcs
00045 #endif // AUBO_SCOPE_FEATURE_VARIABLE_H

```

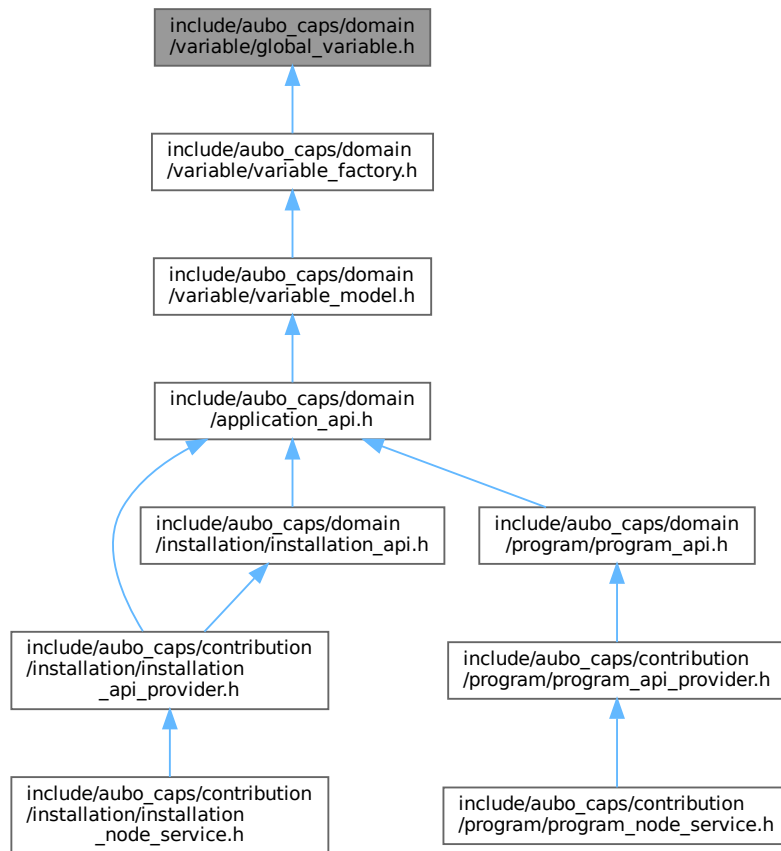
12.240 include/aubo_caps/domain/variable/global_variable.h File Reference

`#include <aubo_caps/domain/variable/variable.h>`

Include dependency graph for `global_variable.h`:



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



Classes

- class [arcs::aubo_scope::GlobalVariable](#)
Global variables are available everywhere in a program.
- struct [arcs::aubo_scope::GlobalVariableFilter](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([GlobalVariable](#))

12.241 global_variable.h

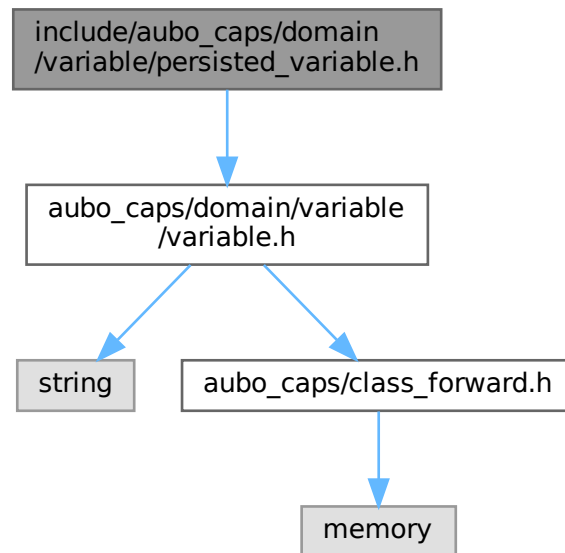
[Go to the documentation of this file.](#)

```
00001 #ifndef AUBO_SCOPE_GLOBAL_VARIABLE_H
00002 #define AUBO_SCOPE_GLOBAL_VARIABLE_H
00003
00004 #include <aubo_caps/domain/variable/variable.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(GlobalVariable);
00009
00010 /**
00011  * Global variables are available everywhere in a program.
00012  */
00013 class ARCS_ABI_EXPORT GlobalVariable : public Variable
00014 {
00015 public:
00016     GlobalVariable(GlobalVariable &f);
00017     GlobalVariable(GlobalVariable &&f);
00018     virtual ~GlobalVariable();
00019
00020 protected:
00021     GlobalVariable();
00022
00023 private:
00024     friend class DataSwitch;
00025     void *d_{ nullptr };
00026 };
00027
00028 struct GlobalVariableFilter
00029 {
00030     bool operator()(VariablePtr var)
00031     {
00032         return std::dynamic_pointer_cast<GlobalVariable>(var) != nullptr &&
00033             var->getType() == Variable::GLOBAL;
00034     }
00035 };
00036
00037 } // namespace aubo_scope
00038 } // namespace arcs
00039
00040 #endif // AUBO_SCOPE_GLOBAL_VARIABLE_H
```

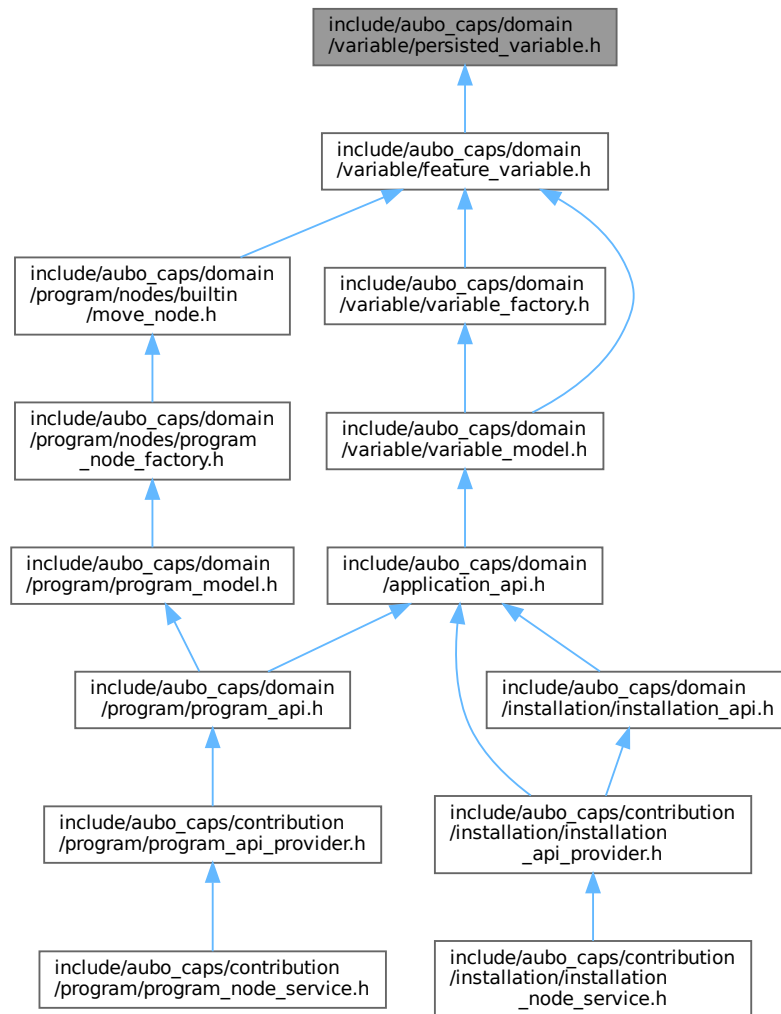
12.242 include/aubo_caps/domain/variable/persisted_variable.h File Reference

#include <aubo_caps/domain/variable/variable.h>

Include dependency graph for persisted_variable.h:



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



Classes

- class [arcs::aubo_scope::PersistedVariable](#)
ingroup [ApplicationApi](#) This interface represents variables stored with the installation.
- struct [arcs::aubo_scope::PersistedVariableFilter](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([PersistedVariable](#))

12.243 persisted_variable.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_PERSISTED_VARIABLE_H
00002 #define AUBO_SCOPE_PERSISTED_VARIABLE_H
00003
00004 #include <aubo_caps/domain/variable/variable.h>
00005
00006 namespace arcs {
00007 namespace aubo_scope {
00008 ARCS_CLASS_FORWARD(PersistedVariable);
00009
00010 /**
00011  * ingroup ApplicationApi
00012  * This interface represents variables stored with the installation. These
00013  * variables are defined in the installation and will exist even when a robot is
00014  * turned off and on.
00015  */
00016 class ARCS_ABI_EXPORT PersistedVariable : public Variable
00017 {
00018 public:
00019     PersistedVariable(PersistedVariable &f);
00020     PersistedVariable(PersistedVariable &&f);
00021     virtual ~PersistedVariable();
00022
00023     /**
00024      * @ingroup PersistedVariable
00025      * \chinese
00026      *
00027      *         PersistedVariable
00028      *
00029      * @return          <code>true</code>
00030      * \endchinese
00031      * \english
00032      * A persisted variable can not be guaranteed to be present. This method can
00033      * be used to determine, if the variable is present.
00034      *
00035      * The persisted variable will not be present, if the end user loads a
00036      * different installation which does not contain the variable, or if the
00037      * variable is deleted by the end user.
00038      *
00039      * A program node storing a <code>PersistedVariable</code> in its Data
00040      * Model will be undefined if the variable cannot be resolved (i.e. the
00041      * variable is not present).
00042      *
00043      * @return <code>true</code>, if this persisted variable is resolvable (i.e.
00044      * present in the current installation).
00045      * \endenglish
00046      */
00047     bool isResolvable();
00048
00049 protected:
00050     PersistedVariable();
00051
00052 private:
00053     friend class DataSwitch;
00054     void *d_{ nullptr };
00055 };
00056
00057 struct PersistedVariableFilter
00058 {
00059     bool operator()(VariablePtr var)
00060     {
00061         return std::dynamic_pointer_cast<PersistedVariable>(var) != nullptr &&
00062             var->getType() == Variable::VALUE_PERSISTED;
00063     }
00064 };
00065
00066 } // namespace aubo_scope
00067 } // namespace arcs
00068 #endif // AUBO_SCOPE_PERSISTED_VARIABLE_H

```

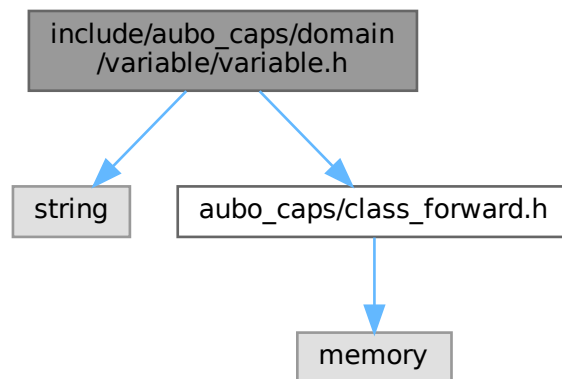
12.244 include/aubo_caps/domain/variable/variable.h File Reference

```

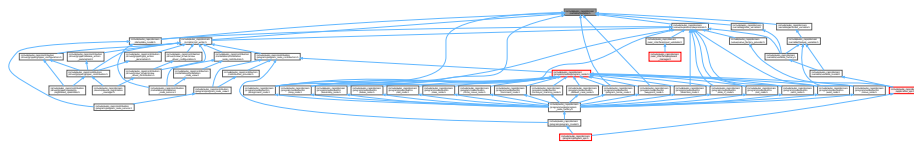
#include <string>
#include <aubo_caps/class_forward.h>

```

Include dependency graph for variable.h:



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



Classes

- class [arcs::aubo_scope::Variable](#)
[Variable](#) Represents a variable in AuboScope.

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([Variable](#))

12.245 variable.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_VARIABLE_H
00002 #define AUBO_SCOPE_VARIABLE_H
00003
00004 #include <string>
00005 #include <aubo_caps/class_forward.h>
00006
00007 namespace arcs {
00008 namespace aubo_scope {
00009 ARCS_CLASS_FORWARD(Variable);
00010
00011 /**
00012  * @ingroup ApplicationApi
00013  * \chinese
00014  *
00015  *   AuboScope
00016  * \endchinese
00017  * \english
00018  * Variable
00019  * Represents a variable in AuboScope. Variables can be used to store and pass
00020  * data in programs.
00021  * \endenglish
00022  */
00023
00024 class ARCS_ABI_EXPORT Variable
00025 {
00026 public:
00027     Variable(Variable &f);
00028     Variable(Variable &&f);
00029     virtual ~Variable();
00030
00031     /**
00032      * The variable type used to determine which type of variable this instance
00033      * is.
00034      */
00035     enum Type : int
00036     {
00037
00038         /**
00039          * <p>
00040          * GLOBAL means that the variable is available everywhere in a program.
00041          * </p>
00042          *
00043          * This variable instance can be cast to {@link GlobalVariable}.
00044          */
00045         GLOBAL,
00046
00047         /**
00048          * <p>
00049          * VALUE_PERSISTED means that the variable is stored with the
00050          * installation. These variables are defined in the installation and
00051          * will retain their values even when a robot is turned off and on.
00052          * </p>
00053          *
00054          * This variable instance can be cast to {@link ValuePersistedVariable}.
00055          */
00056         VALUE_PERSISTED,
00057
00058         /**
00059          * <p>
00060          * FEATURE means that a Feature is marked as a variable. The variable is
00061          * defined in the installation and will exist even when a robot is
00062          * turned off and on.
00063          * </p>
00064          *
00065          * This variable instance can be cast to {@link FeatureVariable}.
00066          */
00067         FEATURE
00068     };
00069
00070     /**
00071      * @ingroup Variable
00072      * \chinese
00073      *
00074      * @return
00075      * \endchinese
00076      * \english
00077      * Returns the type of variable.
00078      * @return the type of variable.
00079      * \endenglish
00080      */
00081     Type getType();
00082

```

```

00083  /**
00084  * @ingroup Variable
00085  * \chinese
00086  *
00087  * @return
00088  * \endchinese
00089  * \english
00090  * Note: The variable can be renamed at any time.
00091  * @return the display name of the variable.
00092  * \endenglish
00093  */
00094  std::string getDisplayName();
00095
00096  /**
00097  * @ingroup Variable
00098  * \chinese
00099  *
00100  * @return          true
00101  * \endchinese
00102  * \english
00103  * Checks if the object references the same internal Variable.
00104  * @return true if object references the same internal Variable.
00105  * \endenglish
00106  */
00107  bool equals(VariablePtr object);
00108
00109  /**
00110  * @ingroup Variable
00111  * \chinese
00112  *
00113  * @return
00114  * \endchinese
00115  * \english
00116  * Returns the hash code for this object.
00117  * @return hashCode for this object.
00118  * \endenglish
00119  */
00120  size_t hashCode();
00121
00122  /**
00123  * @ingroup Variable
00124  * \chinese
00125  *
00126  * @return
00127  * \endchinese
00128  * \english
00129  * Returns a string representation of the variable.
00130  * @return a string representation of the variable.
00131  * \endenglish
00132  */
00133  std::string toString();
00134
00135 protected:
00136     Variable();
00137
00138 private:
00139     friend class DataSwitch;
00140     void *d_{ nullptr };
00141 };
00142
00143 } // namespace aubo_scope
00144 } // namespace arcs
00145 #endif // AUBO_SCOPE_VARIABLE_H

```

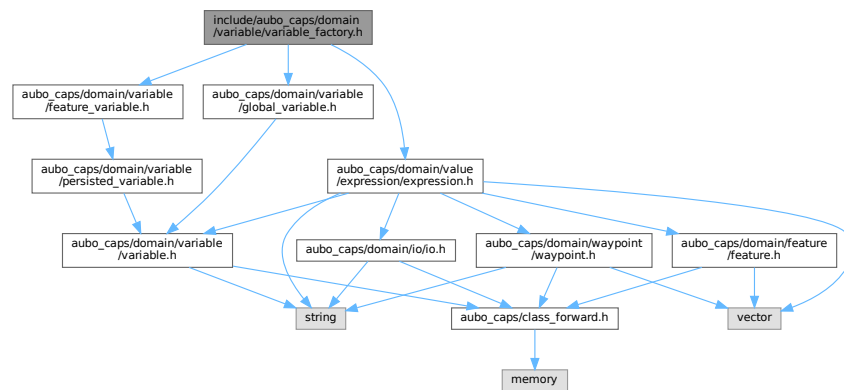
12.246 include/aubo_caps/domain/variable/variable_factory.h File Reference

```

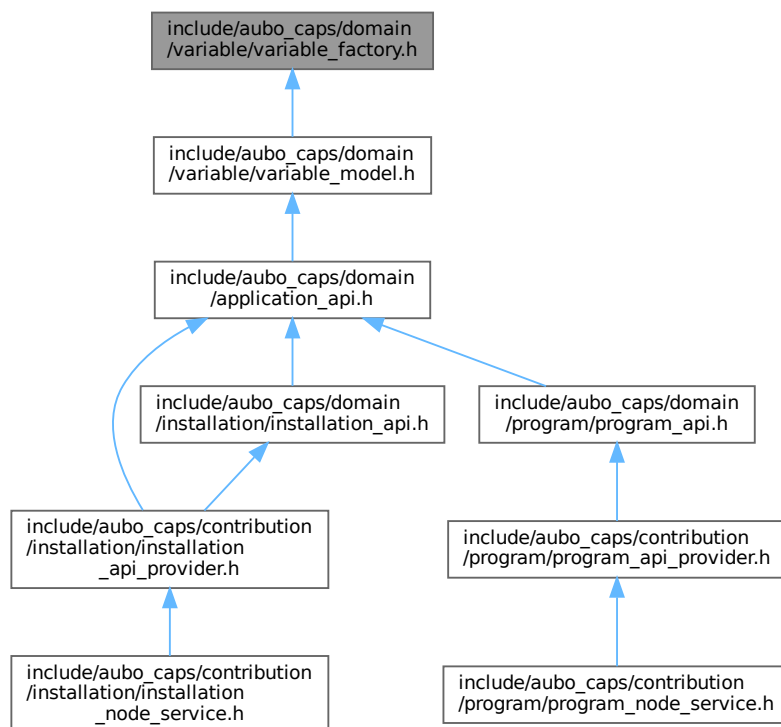
#include <aubo_caps/domain/value/expression/expression.h>
#include <aubo_caps/domain/variable/global_variable.h>
#include <aubo_caps/domain/variable/feature_variable.h>

```

Include dependency graph for variable_factory.h:



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



Classes

- class [arcs::aubo_scope::VariableFactory](#)

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD \(VariableFactory\)](#)

12.247 variable_factory.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_VARIABLE_FACTORY_H
00002 #define AUBO_SCOPE_VARIABLE_FACTORY_H
00003
00004 #include <aubo_caps/domain/value/expression/expression.h>
00005 #include <aubo_caps/domain/variable/global_variable.h>
00006 #include <aubo_caps/domain/variable/feature_variable.h>
00007
00008 namespace arcs {
00009 namespace aubo_scope {
00010 ARCS_CLASS_FORWARD(VariableFactory);
00011
00012 /**
00013  * @ingroup ApplicationApi
00014  * <p>
00015  * This interface supplies methods to create variable objects.
00016  * </p>
00017  *
00018  * <b>NOTE:</b> Only use this functionality in a AuboCap program node (not
00019  * relevant for an installation node).
00020  */
00021 class ARCS_ABI_EXPORT VariableFactory
00022 {
00023 public:
00024     VariableFactory(VariableFactory &f);
00025     VariableFactory(VariableFactory &&f);
00026     ~VariableFactory();
00027
00028     /**
00029      * @ingroup VariableFactory
00030      * \chinese
00031      * <p>
00032      * AuboScope
00033      * </p>
00034      * @param suggested_name
00035      * @return GlobalVariable
00036      * @throws VariableException
00037      * \endchinese
00038      * \english
00039      * <p>
00040      * Creates a global variable. Registering the variable in AuboScope (by
00041      * storing it in a {@link DataModel} instance or using it for the
00042      * configuration of a built-in AuboScope program node) will make it
00043      * accessible to all program nodes.
00044      * </p>
00045      *
00046      * @param suggested_name Suggested name of the variable.
00047      * @return GlobalVariable.
00048      * @throws VariableException In case of an illegal name.
00049      * \endenglish
00050      */
00051     GlobalVariablePtr createGlobalVariable(const std::string &suggested_name);
00052
00053     /**
00054      * @ingroup VariableFactory
00055      * \chinese
00056      * <p>
00057      * @param suggested_name
00058      * @param initialValue
00059      * @return GlobalVariable
00060      * \endchinese
00061      * \english
00062      * <p>
00063      * Creates a global variable with an initial value.
00064      * </p>
00065      *
00066      * @param suggested_name Suggested name of the variable.
00067      * @param initialValue The expression that will be evaluated in order to
00068      * initialize the value of the variable.
00069      * @return GlobalVariable.
00070      * @throws VariableException In case of an illegal name.
00071      * \endenglish
00072      */
00073     GlobalVariablePtr createGlobalVariable(const std::string &suggested_name,
                                         ExpressionPtr initialValue);

```

```

00074
00075 /**
00076  * @ingroup VariableFactory
00077  * \chinese
00078  * <p>    </p>
00079  * @param suggested_name
00080  * @param pose
00081  * @return FeatureVariable
00082  * @throws VariableException
00083  * \endchinese
00084  * \english
00085  * <p>
00086  * Creates a feature variable.
00087  * </p>
00088  * @param suggested_name Suggested name of the variable.
00089  * @param pose the pose of the feature.
00090  * @return FeatureVariable.
00091  * @throws VariableException In case of an illegal name.
00092  * \endenglish
00093  */
00094 FeatureVariablePtr createFeatureVariable(const std::string &suggested_name,
00095                                         const std::vector<double> &pose);
00096
00097 private:
00098 friend class DataSwitch;
00099 VariableFactory();
00100 void *d_{ nullptr };
00101 };
00102
00103 } // namespace aubo_scope
00104 } // namespace arcs
00105 #endif // AUBO_SCOPE_VARIABLE_FACTORY_H

```

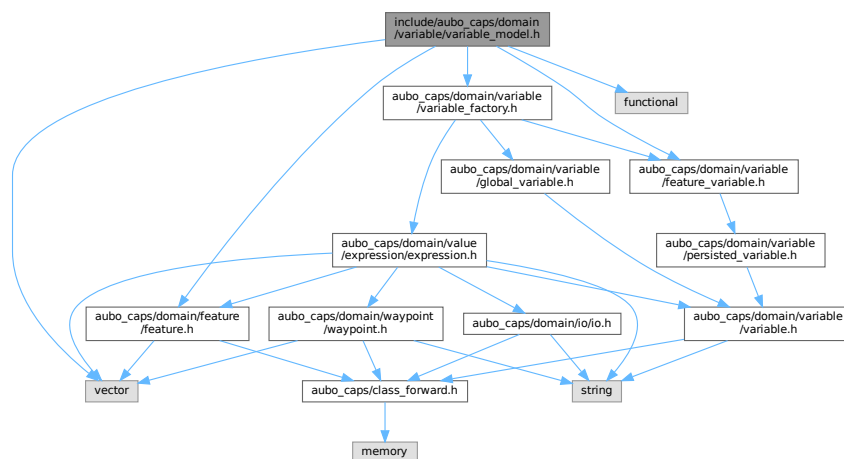
12.248 include/aubo_caps/domain/variable/variable_model.h File Reference

```

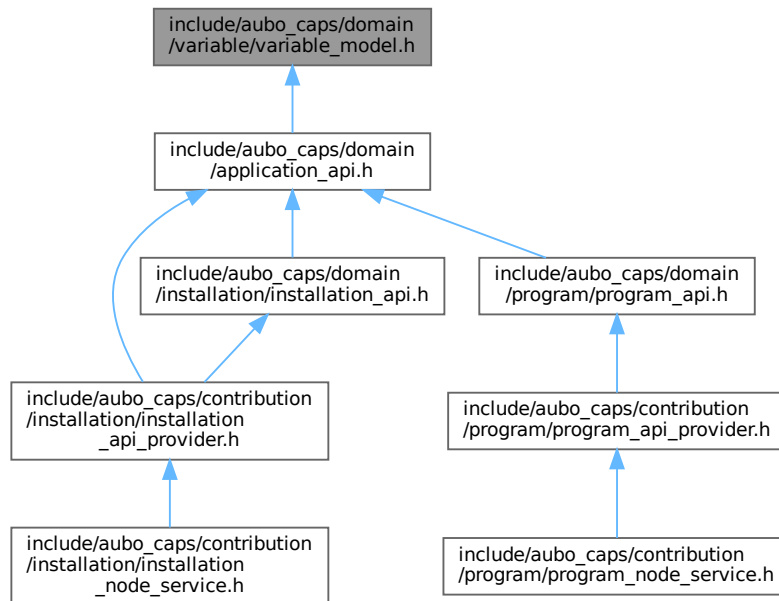
#include <vector>
#include <functional>
#include <aubo_caps/domain/feature/feature.h>
#include <aubo_caps/domain/variable/variable_factory.h>
#include <aubo_caps/domain/variable/feature_variable.h>

```

Include dependency graph for variable_model.h:



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



Classes

- class [arcs::aubo_scope::VariableModel](#)
[VariableModel](#) Provides methods that returns the variables that currently are registered in the system.

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([VariableModel](#))

12.249 variable_model.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_VARIABLE_MODEL_H
00002 #define AUBO_SCOPE_VARIABLE_MODEL_H
00003
00004 #include <vector>
00005 #include <functional>
00006 #include <aubo_caps/domain/feature/feature.h>
00007 #include <aubo_caps/domain/variable/variable_factory.h>
00008 #include <aubo_caps/domain/variable/feature_variable.h>
00009
00010 namespace arcs {
00011 namespace aubo_scope {

```

```

00012 ARCS_CLASS_FORWARD(VariableModel);
00013
00014 /**
00015  * @ingroup ApplicationApi
00016  * \chinese
00017  *
00018  *      {@link DataModel}
00019  *      AuboScope
00020  * \endchinese
00021  * \english
00022  * VariableModel
00023  * Provides methods that returns the variables that currently are registered in
00024  * the system. A variable is registered if it has been stored in a {@link
00025  * DataModel} instance or used for the configuration of a built-in AuboScope
00026  * program node.
00027  * \endenglish
00028  */
00029 class ARCS_ABI_EXPORT VariableModel
00030 {
00031 public:
00032     VariableModel(VariableModel &f);
00033     VariableModel(VariableModel &&f);
00034     ~VariableModel();
00035
00036     /**
00037      * @ingroup VariableModel
00038      * \chinese
00039      *
00040      * @return
00041      * \endchinese
00042      * \english
00043      * Returns the collection of all variables that currently are registered in
00044      * the system.
00045      * @return the collection of all registered variables.
00046      * \endenglish
00047      */
00048     std::vector<VariablePtr> getAll();
00049
00050     VariablePtr get(const std::string &name);
00051
00052     /**
00053      * @ingroup VariableModel
00054      * \chinese
00055      *
00056      * @param filter
00057      * @return
00058      * \endchinese
00059      * \english
00060      * Get a subset of all the variables registered in the system using a filter.
00061      *
00062      * @param filter see {@link VariableFilterFactory} for examples.
00063      * @return the collection of Variables that are accepted by the filter.
00064      * \endenglish
00065      */
00066     std::vector<VariablePtr> get(std::function<bool(VariablePtr)> filter =
00067         [](VariablePtr) { return true; });
00068
00069     /**
00070      * @ingroup VariableModel
00071      * \chinese
00072      *
00073      * @return
00074      * \endchinese
00075      * \english
00076      * Returns a {@link VariableFactory} to create variable objects.
00077      * @return a {@link VariableFactory} to create variable objects.
00078      * \endenglish
00079      */
00080     VariableFactoryPtr getVariableFactory();
00081
00082     /**
00083      * @ingroup VariableModel
00084      * \chinese
00085      *
00086      * @param feature
00087      * @return
00088      * @throws IllegalArgumentException
00089      * \endchinese
00090      * \english
00091      * Get the feature variable for a {@link Feature}.
00092      *
00093      * @param feature the feature with a variable.
00094      * @return the variable for the feature.
00095      * @throws IllegalArgumentException If feature not is present in the
00096      * installation or feature has no variable, see {@link Feature#isVariable()}.
00097      * \endenglish
00098      */

```

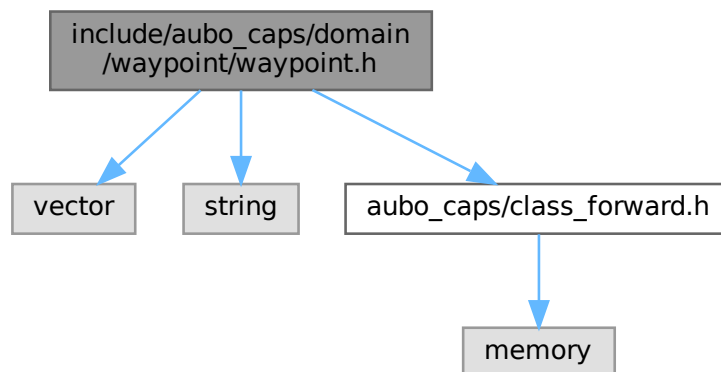
```

00099 FeatureVariablePtr getFeatureVariable(FeaturePtr feature);
00100
00101 /**
00102  * @ingroup VariableModel
00103  * \chinese
00104  *
00105  * @param receiver
00106  * @param slot
00107  * @return
00108  * \endchinese
00109  * \english
00110  * Register callback for changed.
00111  *
00112  * @param receiver the receiver identifier.
00113  * @param slot the callback function.
00114  * @return the connection identifier.
00115  * \endenglish
00116  */
00117 std::string connectedTo(const std::string &receiver,
00118                        const std::function<void(VariableModel *)> &slot);
00119 /**
00120  * @ingroup VariableModel
00121  * \chinese
00122  *
00123  * @param receiver
00124  * \endchinese
00125  * \english
00126  * Disconnect a callback.
00127  * @param receiver the receiver identifier.
00128  * \endenglish
00129  */
00130 void disconnectFrom(const std::string &receiver);
00131
00132 private:
00133     friend class DataSwitch;
00134     VariableModel();
00135     void *d_{ nullptr };
00136 };
00137
00138 } // namespace aubo_scope
00139 } // namespace arcs
00140 #endif // AUBO_SCOPE_VARIABLE_MODEL_H

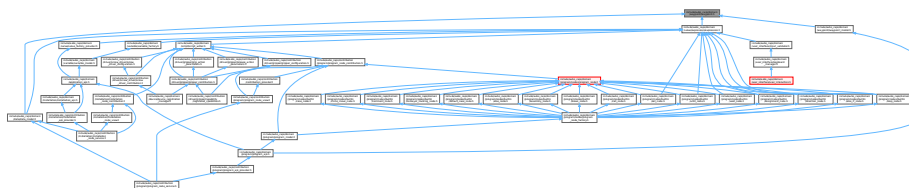
```

12.250 include/aubo_caps/domain/waypoint/waypoint.h File Reference

#include <vector>
#include <string>
#include <aubo_caps/class_forward.h>
Include dependency graph for waypoint.h:



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



Classes

- class [arcs::aubo_scope::Waypoint](#)
[Waypoint](#) This interface composes parameters that define a position of the robot.

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([Waypoint](#))

12.251 waypoint.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_WAYPOINT_H
00002 #define AUBO_SCOPE_WAYPOINT_H
00003
00004 #include <vector>
00005 #include <string>
00006 #include <aubo_caps/class_forward.h>
00007
00008 namespace arcs {
00009 namespace aubo_scope {
00010 ARCS_CLASS_FORWARD(Waypoint);
00011
00012 /**
00013  * @ingroup ProgramApi
00014  * \chinese
00015  *
00016  *
00017  * \endchinese
00018  * \english
00019  * Waypoint
00020  * This interface composes parameters that define a position of the robot.
00021  * \endenglish
00022  */
00023 class ARCS_ABI_EXPORT Waypoint
00024 {
00025 public:
00026     Waypoint(Waypoint &f);
00027     Waypoint(Waypoint &&f);
00028     ~Waypoint();
00029
00030     /**
00031      * @ingroup Waypoint
00032      * \chinese
00033      *
00034      * @return
00035      * \endchinese
00036      * \english
00037      * Returns the display name of the Waypoint.

```

```

00038     * @return Display name.
00039     * \endenglish
00040     */
00041     std::string getDisplayName();
00042
00043     /**
00044     * @ingroup Waypoint
00045     * \chinese
00046     *     TCP
00047     * @return
00048     * \endchinese
00049     * \english
00050     * Returns the pose for the position of the robot (including the TCP offset).
00051     * @return the pose for the position of the robot.
00052     * \endenglish
00053     */
00054     std::vector<double> getPose();
00055
00056     /**
00057     * @ingroup Waypoint
00058     * \chinese
00059     *     TCP
00060     * @return
00061     * \endchinese
00062     * \english
00063     * Returns the joint positions corresponding to the pose of the robot (when
00064     * taking the TCP offset into account).
00065     * @return the joint positions.
00066     * \endenglish
00067     */
00068     std::vector<double> getJointPositions();
00069
00070     /**
00071     * @ingroup Waypoint
00072     * \chinese
00073     *     TCP
00074     * @return TCP
00075     * \endchinese
00076     * \english
00077     * Returns the pose for the TCP offset (used when defining the robot
00078     * position).
00079     * @return the TCP offset.
00080     * \endenglish
00081     */
00082     std::vector<double> getTcpOffset();
00083
00084     /**
00085     * @ingroup Waypoint
00086     * \chinese
00087     *
00088     * @return
00089     * \endchinese
00090     * \english
00091     * Returns whether the Waypoint is defined.
00092     * @return <code>true</code> if defined.
00093     * \endenglish
00094     */
00095     bool isDefined();
00096
00097     /**
00098     * @ingroup Waypoint
00099     * \chinese
00100     *
00101     * @return
00102     * \endchinese
00103     * \english
00104     * Returns the hash code of the Waypoint.
00105     * @return the hash code.
00106     * \endenglish
00107     */
00108     size_t hashCode();
00109
00110 private:
00111     friend class DataSwitch;
00112     Waypoint();
00113     void *d_{ nullptr };
00114 };
00115
00116 } // namespace aubo_scope
00117 } // namespace arcs
00118
00119 #endif // AUBO_SCOPE_WAYPOINT_H

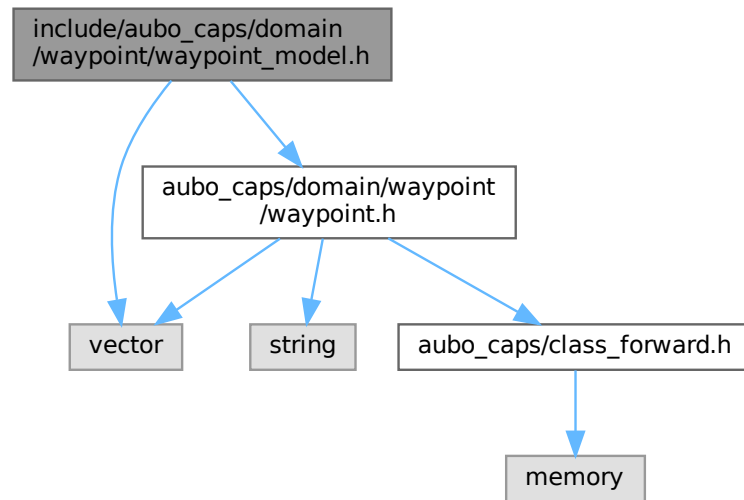
```

12.252 include/aubo_caps/domain/waypoint/waypoint_model.h File Reference

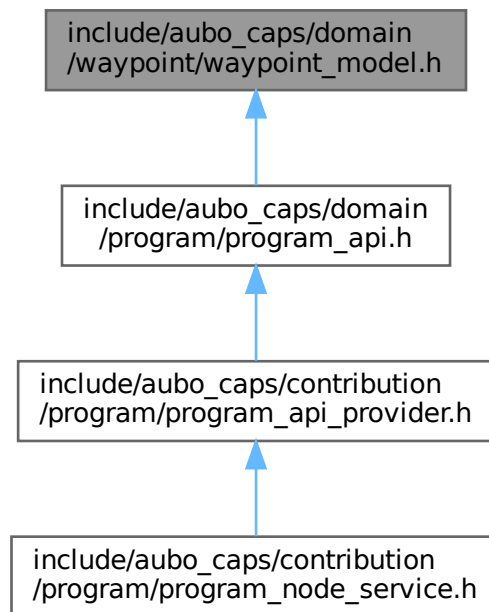
```
#include <vector>
```

```
#include <aubo_caps/domain/waypoint/waypoint.h>
```

Include dependency graph for waypoint_model.h:



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



Classes

- class [arcs::aubo_scope::WaypointModel](#)
[WaypointModel](#) Provides methods that returns Waypoints from the current robot installation.

Namespaces

- namespace [arcs](#)
- namespace [arcs::aubo_scope](#)

Functions

- [arcs::aubo_scope::ARCS_CLASS_FORWARD](#) ([WaypointModel](#))

12.253 waypoint_model.h

[Go to the documentation of this file.](#)

```

00001 #ifndef AUBO_SCOPE_WAYPOINT_MODEL_H
00002 #define AUBO_SCOPE_WAYPOINT_MODEL_H
00003
00004 #include <vector>
00005 #include <aubo_caps/domain/waypoint/waypoint.h>
00006
00007 namespace arcs {
00008 namespace aubo_scope {

```

```

00009 ARCS_CLASS_FORWARD(WaypointModel);
00010
00011 /**
00012  * @ingroup ProgramApi
00013  * \chinese
00014  *
00015  *
00016  * \endchinese
00017  * \english
00018  * WaypointModel
00019  * Provides methods that returns Waypoints from the current robot installation.
00020  * \endenglish
00021  */
00022 class ARCS_ABI_EXPORT WaypointModel
00023 {
00024 public:
00025     WaypointModel(WaypointModel &f);
00026     WaypointModel(WaypointModel &&f);
00027     ~WaypointModel();
00028
00029     /**
00030      * @ingroup WaypointModel
00031      * \chinese
00032      *
00033      * @return
00034      * \endchinese
00035      * \english
00036      * Returns the geometric Waypoints of the current installation.
00037      * @return the geometric Waypoints of the current installation.
00038      * \endenglish
00039      */
00040     std::vector<WaypointPtr> getWaypoints();
00041
00042     /**
00043      * @ingroup WaypointModel
00044      * \chinese
00045      *
00046      * @param name
00047      * @return
00048      * \endchinese
00049      * \english
00050      * Returns the Waypoint by name.
00051      * @param name the name of the Waypoint.
00052      * @return the Waypoint with the given name.
00053      * \endenglish
00054      */
00055     WaypointPtr getByName(const std::string &name);
00056
00057     /**
00058      * @ingroup WaypointModel
00059      * \chinese
00060      *     AuboScope
00061      *
00062      * @param suggestedName
00063      * [a-zA-Z][a-zA-Z0-9_]{0,14} 15 Waypoint
00064      * @param pose
00065      * @return AuboScope
00066      * @throws WaypointAlreadyAddedException <code>idKey</code>
00067      * { @link #getWaypoint(String) }
00068      * { @link #updateWaypoint(String, Pose) }
00069      * @throws IllegalWaypointNameException
00070      * \endchinese
00071      * \english
00072      * Add a Waypoint to the current AuboScope installation. This makes it
00073      * selectable by the end user. The Waypoint is not modifiable by the end
00074      * user.
00075      *
00076      * @param suggestedName Suggested name for the Waypoint. Valid names must
00077      * match regex [a-zA-Z][a-zA-Z0-9_]{0,14} for a total of 15 characters. The
00078      * final name can be retrieved from the returned Waypoint instance.
00079      * @param pose The pose of the Waypoint with respect to the robot base.
00080      * @return The Waypoint created and registered in AuboScope.
00081      * @throws WaypointAlreadyAddedException If a Waypoint has previously been
00082      * added the same <code>idKey</code> identifier. Use { @link
00083      * #getWaypoint(String) } to check if the Waypoint has already been added to
00084      * the current installation. Use { @link #updateWaypoint(String, Pose) } to
00085      * update the Waypoint.
00086      * @throws IllegalWaypointNameException If the suggested name does not match
00087      * required regex.
00088      * \endenglish
00089      */
00090     WaypointPtr addWaypoint(const std::string &suggestedName,
00091                             const std::vector<double> &pose);
00092
00093     /**
00094      * @ingroup WaypointModel
00095      * \chinese
00096      *     AuboScope

```

```

00096  *
00097  * @param suggestedName
00098  * [a-zA-Z][a-zA-Z0-9_]{0,14} 15 Waypoint
00099  * @param pose
00100  * @param joint
00101  * @param offset TCP
00102  * @return AuboScope
00103  * @throws WaypointAlreadyAddedException <code>idKey</code>
00104  *
00105  * @throws IllegalWaypointNameException
00106  * \endchinese
00107  * \english
00108  * Add a Waypoint to the current AuboScope installation with joint angles
00109  * and offset.
00110  *
00111  * @param suggestedName Suggested name for the Waypoint. Valid names must
00112  * match regex [a-zA-Z][a-zA-Z0-9_]{0,14} for a total of 15 characters. The
00113  * final name can be retrieved from the returned Waypoint instance.
00114  * @param pose The pose of the Waypoint with respect to the robot base.
00115  * @param joint The joint of the Waypoint with respect to the robot base.
00116  * @param offset The tcp of the Waypoint with respect to the robot base.
00117  * @return The Waypoint created and registered in AuboScope.
00118  * @throws WaypointAlreadyAddedException If a Waypoint has previously been
00119  * added the same <code>idKey</code> identifier. Use {@link
00120  * #getWaypoint(String)} to check if the Waypoint has already been added to
00121  * the current installation. Use {@link #updateWaypoint(String, Pose)} to
00122  * update the Waypoint.
00123  * @throws IllegalWaypointNameException If the suggested name does not match
00124  * required regex.
00125  * \endenglish
00126  */
00127  WaypointPtr addWaypoint(const std::string &suggestedName,
00128                          const std::vector<double> &pose,
00129                          const std::vector<double> &joint,
00130                          const std::vector<double> &offset);
00131  /**
00132  * @ingroup WaypointModel
00133  * \chinese
00134  * AuboCap
00135  *
00136  *
00137  * @param name
00138  * @throws WaypointNotFoundException <code>idKey</code>
00139  *
00140  * \endchinese
00141  * \english
00142  * Remove a Waypoint added by this AuboCap from AuboScope. Program nodes
00143  * using the Waypoint will be become undefined because the Waypoint is no
00144  * longer resolvable.
00145  *
00146  * @param name The key used to add the Waypoint with.
00147  * @throws WaypointNotFoundException If no Waypoint exists with the provided
00148  * <code>idKey</code>.
00149  * \endenglish
00150  */
00151  void removeWaypoint(const std::string &name);
00152
00153  /**
00154  * @ingroup WaypointModel
00155  * \chinese
00156  *
00157  * @param name
00158  * @param newName
00159  * \endchinese
00160  * \english
00161  * Rename the waypoint in the model by the name.
00162  * @param name the original name.
00163  * @param newName the new name.
00164  * \endenglish
00165  */
00166  void renameWaypoint(const std::string &name, const std::string &newName);
00167
00168 private:
00169  friend class DataSwitch;
00170  WaypointModel();
00171  void *d_{ nullptr };
00172 };
00173
00174 } // namespace aubo_scope
00175 } // namespace arcs
00176
00177 #endif // AUBO_SCOPE_WAYPOINT_MODEL_H

```

Index

- ~AnalogIo
 - arcs::aubo_scope::AnalogIo, [102](#)
- ~ApplicationApi
 - arcs::aubo_scope::ApplicationApi, [107](#)
- ~AssemblyNode
 - arcs::aubo_scope::AssemblyNode, [112](#)
- ~AssignmentNode
 - arcs::aubo_scope::AssignmentNode, [116](#)
- ~AuboCapInfo
 - arcs::aubo_scope::AuboCapInfo, [118](#)
- ~BaseFeature
 - arcs::aubo_scope::BaseFeature, [122](#)
- ~BoolRegister
 - arcs::aubo_scope::BoolRegister, [125](#)
- ~BreakNode
 - arcs::aubo_scope::BreakNode, [127](#)
- ~CallNode
 - arcs::aubo_scope::CallNode, [131](#)
- ~CaseNode
 - arcs::aubo_scope::CaseNode, [134](#)
- ~CircleMoveNode
 - arcs::aubo_scope::CircleMoveNode, [137](#)
- ~CommentNode
 - arcs::aubo_scope::CommentNode, [140](#)
- ~ContributionProvider
 - arcs::aubo_scope::ContributionProvider< T>, [142](#)
- ~ConveyorTrackingNode
 - arcs::aubo_scope::ConveyorTrackingNode, [146](#)
- ~DataModel
 - arcs::aubo_scope::DataModel, [150](#)
- ~DefaultCaseNode
 - arcs::aubo_scope::DefaultCaseNode, [169](#)
- ~Device
 - arcs::aubo_scope::Device, [171](#)
- ~DeviceManager
 - arcs::aubo_scope::DeviceManager, [173](#)
- ~DeviceRegistrationManager
 - arcs::aubo_scope::DeviceRegistrationManager, [174](#)
- ~DigitalIo
 - arcs::aubo_scope::DigitalIo, [178](#)
- ~DirectionNode
 - arcs::aubo_scope::DirectionNode, [185](#)
- ~DoubleRegister
 - arcs::aubo_scope::DoubleRegister, [194](#)
- ~DriveScrewNotOKParameters
 - arcs::aubo_scope::DriveScrewNotOKParameters, [195](#)
- ~DriveScrewOKParameters
 - arcs::aubo_scope::DriveScrewOKParameters, [195](#)
- ~ElseIfNode
 - arcs::aubo_scope::ElseIfNode, [198](#)
- ~ElseNode
 - arcs::aubo_scope::ElseNode, [202](#)
- ~EtherNetIP
 - arcs::aubo_scope::EtherNetIP, [206](#)
- ~Expression
 - arcs::aubo_scope::Expression, [211](#)
- ~Feature
 - arcs::aubo_scope::Feature, [215](#)
- ~FeatureContributionModel
 - arcs::aubo_scope::FeatureContributionModel, [219](#)
- ~FeatureLine
 - arcs::aubo_scope::FeatureLine, [226](#)
- ~FeatureModel
 - arcs::aubo_scope::FeatureModel, [229](#)
- ~FeaturePlane
 - arcs::aubo_scope::FeaturePlane, [234](#)
- ~FeaturePoint
 - arcs::aubo_scope::FeaturePoint, [237](#)
- ~FeatureTool
 - arcs::aubo_scope::FeatureTool, [241](#)
- ~FeatureVariable
 - arcs::aubo_scope::FeatureVariable, [245](#)
- ~FeedScrewParameters
 - arcs::aubo_scope::FeedScrewParameters, [247](#)
- ~FolderNode
 - arcs::aubo_scope::FolderNode, [249](#)
- ~ForceNode
 - arcs::aubo_scope::ForceNode, [253](#)
- ~ForceOptionNode
 - arcs::aubo_scope::ForceOptionNode, [257](#)
- ~Function
 - arcs::aubo_scope::Function, [259](#)
- ~FunctionModel
 - arcs::aubo_scope::FunctionModel, [263](#)
- ~GlobalVariable
 - arcs::aubo_scope::GlobalVariable, [268](#)
- ~GripActionParameters
 - arcs::aubo_scope::GripActionParameters, [270](#)
- ~GripForceCapability
 - arcs::aubo_scope::GripForceCapability, [273](#)
- ~GripVacuumCapability
 - arcs::aubo_scope::GripVacuumCapability, [293](#)

- ~GripperApiProvider
 - arcs::aubo_scope::GripperApiProvider, [274](#)
- ~GripperConfiguration
 - arcs::aubo_scope::GripperConfiguration, [276](#)
- ~GripperContribution
 - arcs::aubo_scope::GripperContribution, [283](#)
- ~GripperDevice
 - arcs::aubo_scope::GripperDevice, [287](#)
- ~GripperNode
 - arcs::aubo_scope::GripperNode, [291](#)
- ~HaltNode
 - arcs::aubo_scope::HaltNode, [296](#)
- ~IfNode
 - arcs::aubo_scope::IfNode, [300](#)
- ~InstallationApi
 - arcs::aubo_scope::InstallationApi, [306](#)
- ~InstallationApiProvider
 - arcs::aubo_scope::InstallationApiProvider, [309](#)
- ~InstallationContributionConfiguration
 - arcs::aubo_scope::InstallationContributionConfiguration, [311](#)
- ~InstallationCreationContext
 - arcs::aubo_scope::InstallationCreationContext, [313](#)
- ~InstallationNodeContribution
 - arcs::aubo_scope::InstallationNodeContribution, [314](#)
- ~InstallationNodeService
 - arcs::aubo_scope::InstallationNodeService, [315](#)
- ~InstallationNodeView
 - arcs::aubo_scope::InstallationNodeView, [318](#)
- ~IntegerRegister
 - arcs::aubo_scope::IntegerRegister, [324](#)
- ~Io
 - arcs::aubo_scope::Io, [327](#)
- ~IoModel
 - arcs::aubo_scope::IoModel, [334](#)
- ~KeyboardManager
 - arcs::aubo_scope::KeyboardManager, [339](#)
- ~LoopNode
 - arcs::aubo_scope::LoopNode, [346](#)
- ~ModbusIo
 - arcs::aubo_scope::ModbusIo, [352](#)
- ~MoveNode
 - arcs::aubo_scope::MoveNode, [359](#)
- ~OfflineTrackNode
 - arcs::aubo_scope::OfflineTrackNode, [366](#)
- ~PalletNode
 - arcs::aubo_scope::PalletNode, [370](#)
- ~Payload
 - arcs::aubo_scope::Payload, [373](#)
- ~PayloadContributionModel
 - arcs::aubo_scope::PayloadContributionModel, [378](#)
- ~PayloadModel
 - arcs::aubo_scope::PayloadModel, [385](#)
- ~PersistedVariable
 - arcs::aubo_scope::PersistedVariable, [390](#)
- ~PopupManager
 - arcs::aubo_scope::PopupManager, [394](#)
- ~PopupNode
 - arcs::aubo_scope::PopupNode, [398](#)
- ~PrepareToStartScrewdriverParameters
 - arcs::aubo_scope::PrepareToStartScrewdriverParameters, [401](#)
- ~Profinet
 - arcs::aubo_scope::Profinet, [404](#)
- ~ProgramApi
 - arcs::aubo_scope::ProgramApi, [410](#)
- ~ProgramApiProvider
 - arcs::aubo_scope::ProgramApiProvider, [413](#)
- ~ProgramContributionConfiguration
 - arcs::aubo_scope::ProgramContributionConfiguration, [416](#)
- ~ProgramCreationContext
 - arcs::aubo_scope::ProgramCreationContext, [423](#)
- ~ProgramHomeNode
 - arcs::aubo_scope::ProgramHomeNode, [427](#)
- ~ProgramModel
 - arcs::aubo_scope::ProgramModel, [429](#)
- ~ProgramNode
 - arcs::aubo_scope::ProgramNode, [434](#)
- ~ProgramNodeContribution
 - arcs::aubo_scope::ProgramNodeContribution, [435](#)
- ~ProgramNodeFactory
 - arcs::aubo_scope::ProgramNodeFactory, [440](#)
- ~ProgramNodeService
 - arcs::aubo_scope::ProgramNodeService, [448](#)
- ~ProgramNodeView
 - arcs::aubo_scope::ProgramNodeView, [451](#)
- ~ProgramSelectionParameters
 - arcs::aubo_scope::ProgramSelectionParameters, [452](#)
- ~RecordTrackNode
 - arcs::aubo_scope::RecordTrackNode, [455](#)
- ~Register
 - arcs::aubo_scope::Register, [457](#)
- ~RegisteredCapabilities
 - arcs::aubo_scope::RegisteredCapabilities, [458](#)
- ~ReleaseActionParameters
 - arcs::aubo_scope::ReleaseActionParameters, [460](#)
- ~RenderInterface
 - arcs::aubo_scope::RenderInterface, [466](#)
- ~ScrewdriverApiProvider
 - arcs::aubo_scope::ScrewdriverApiProvider, [480](#)
- ~ScrewdriverConfiguration
 - arcs::aubo_scope::ScrewdriverConfiguration, [481](#)
- ~ScrewdriverContribution

- arcs::aubo_scope::ScrewdriverContribution, 489
- ~ScrewdriverParameters
 - arcs::aubo_scope::ScrewdriverParameters, 492
- ~ScrewdriverProgram
 - arcs::aubo_scope::ScrewdriverProgram, 493
- ~ScrewdriverReadyParameters
 - arcs::aubo_scope::ScrewdriverReadyParameters, 494
- ~ScrewdrivingNode
 - arcs::aubo_scope::ScrewdrivingNode, 497
- ~ScriptNode
 - arcs::aubo_scope::ScriptNode, 500
- ~ScriptWriter
 - arcs::aubo_scope::ScriptWriter, 504
- ~SeekNode
 - arcs::aubo_scope::SeekNode, 515
- ~SetNode
 - arcs::aubo_scope::SetNode, 520
- ~SetPayloadNode
 - arcs::aubo_scope::SetPayloadNode, 528
- ~SpeedCapability
 - arcs::aubo_scope::SpeedCapability, 530
- ~SubProgNode
 - arcs::aubo_scope::SubProgNode, 537
- ~SwitchNode
 - arcs::aubo_scope::SwitchNode, 540
- ~SystemApi
 - arcs::aubo_scope::SystemApi, 544
- ~TCP
 - arcs::aubo_scope::TCP, 546
- ~TcpContributionModel
 - arcs::aubo_scope::TcpContributionModel, 550
- ~TcpModel
 - arcs::aubo_scope::TcpModel, 555
- ~ThreadNode
 - arcs::aubo_scope::ThreadNode, 560
- ~TimerNode
 - arcs::aubo_scope::TimerNode, 563
- ~TreeNode
 - arcs::aubo_scope::TreeNode, 566
- ~UndoRedoManager
 - arcs::aubo_scope::UndoRedoManager, 577
- ~UntilNode
 - arcs::aubo_scope::UntilNode, 584
- ~UserInteraction
 - arcs::aubo_scope::UserInteraction, 593
- ~UserInterfaceApi
 - arcs::aubo_scope::UserInterfaceApi, 601
- ~ValueFactory
 - arcs::aubo_scope::ValueFactory, 603
- ~Variable
 - arcs::aubo_scope::Variable, 607
- ~VariableFactory
 - arcs::aubo_scope::VariableFactory, 611
- ~VariableModel
 - arcs::aubo_scope::VariableModel, 615
- ~ViewApiProvider
 - arcs::aubo_scope::ViewApiProvider, 620
- ~VisualLocalizationProgramNode
 - arcs::aubo_scope::VisualLocalizationProgramNode, 624
- ~WaitNode
 - arcs::aubo_scope::WaitNode, 629
- ~Waypoint
 - arcs::aubo_scope::Waypoint, 637
- ~WaypointModel
 - arcs::aubo_scope::WaypointModel, 641
- ~WaypointNode
 - arcs::aubo_scope::WaypointNode, 650
- ~WidthCapability
 - arcs::aubo_scope::WidthCapability, 657
- ABANDON_TRACKING
 - arcs::aubo_scope::ConveyorTrackingNode, 145
- addAssemblyUntilNode
 - arcs::aubo_scope::AssemblyNode, 112
- addChild
 - arcs::aubo_scope::TreeNode, 567
- addCoordinate
 - arcs::aubo_scope::RenderInterface, 466
- addCube
 - arcs::aubo_scope::RenderInterface, 466–468
- addFeature
 - arcs::aubo_scope::FeatureContributionModel, 220
- addFunction
 - arcs::aubo_scope::FunctionModel, 263
- addLine
 - arcs::aubo_scope::RenderInterface, 469
- addMesh
 - arcs::aubo_scope::RenderInterface, 470
- addPayload
 - arcs::aubo_scope::PayloadContributionModel, 378, 379
- addPlane
 - arcs::aubo_scope::RenderInterface, 470, 471
- addPointCloud
 - arcs::aubo_scope::RenderInterface, 471
- addScrewdriverProgram
 - arcs::aubo_scope::ScrewdriverConfiguration, 481
- addScrewdriverPrograms
 - arcs::aubo_scope::ScrewdriverConfiguration, 482
- addTcp
 - arcs::aubo_scope::TcpContributionModel, 551
- addWaypoint
 - arcs::aubo_scope::RenderInterface, 471
 - arcs::aubo_scope::WaypointModel, 641, 642
- All
 - arcs::aubo_scope::RenderInterface, 465
- ALWAYS
 - arcs::aubo_scope::LoopNode, 345
- ANALOG
 - arcs::aubo_scope::Io, 326

- ANALOG_INPUT
 - arcs::aubo_scope::UntilNode, 583
- ANALOG_INPUT_CURRENT
 - arcs::aubo_scope::WaitNode, 628
- ANALOG_INPUT_VOLTAGE
 - arcs::aubo_scope::WaitNode, 628
- ANALOG_OUTPUT_CURRENT
 - arcs::aubo_scope::SetNode, 519
- ANALOG_OUTPUT_VOLTAGE
 - arcs::aubo_scope::SetNode, 519
- AnalogIo
 - arcs::aubo_scope::AnalogIo, 101, 102
- AnalogIoFilter
 - arcs::aubo_scope::AnalogIoFilter, 104
- Angular
 - arcs::aubo_scope::UntilNode, 583
- anonyFunction
 - arcs::aubo_scope::ScriptWriter, 504
- append
 - arcs::aubo_scope::Expression, 211
- appendFeature
 - arcs::aubo_scope::Expression, 211
- appendIo
 - arcs::aubo_scope::Expression, 212
- appendLine
 - arcs::aubo_scope::ScriptWriter, 504
- appendRaw
 - arcs::aubo_scope::ScriptWriter, 504
- appendVariable
 - arcs::aubo_scope::Expression, 212
- appendVectorDouble
 - arcs::aubo_scope::ScriptWriter, 505
- appendWaypoint
 - arcs::aubo_scope::Expression, 212
- ApplicationApi, 37
 - arcs::aubo_scope::ApplicationApi, 106, 107
 - getDeviceManager, 38
 - getFeatureModel, 38
 - getIoModel, 39
 - getPayloadModel, 39
 - getTcpModel, 39
 - getValueFactory, 39
 - getVariableModel, 39
- arcs, 73
- arcs::aubo_scope, 73
 - ARCS_CLASS_FORWARD, 82–96
 - ContributionProviderPtr, 81
 - IMPERIAL, 82
 - LOOSEN, 82
 - METRIC, 82
 - RobotMovementCallback, 81
 - RobotPositionCallback, 81
 - ScrewdriverOperationType, 81
 - TIGHTEN, 82
 - UnitType, 82
 - WidgetHandle, 81
- arcs::aubo_scope::AnalogIo, 99
 - ~AnalogIo, 102
- AnalogIo, 101, 102
 - d_, 103
 - DataSwitch, 103
 - getMaxRangeValue, 102
 - getMinRangeValue, 102
 - getValue, 102
 - isCurrent, 102
 - isVoltage, 103
 - setValue, 103
- arcs::aubo_scope::AnalogIoFilter, 104
 - AnalogIoFilter, 104
 - interface_, 104
 - is_input_, 104
 - operator(), 104
- arcs::aubo_scope::ApplicationApi, 105
 - ~ApplicationApi, 107
 - ApplicationApi, 106, 107
 - d_, 109
 - DataSwitch, 109
 - getInstallationNode, 107, 108
 - getRobotProxy, 108
- arcs::aubo_scope::AssemblyNode, 109
 - ~AssemblyNode, 112
 - addAssemblyUntilNode, 112
 - AssemblyNode, 111, 112
 - COMPLIANT_INSERTION, 111
 - ConfigType, 110
 - d_, 113
 - DataSwitch, 112
 - HOLE_SEARCH, 111
 - NONE, 111
- arcs::aubo_scope::AssignmentNode, 113
 - ~AssignmentNode, 116
 - AssignmentNode, 115, 116
 - ConfigType, 114
 - d_, 118
 - DataSwitch, 117
 - EXPRESSION, 115
 - getConfigType, 116
 - getExpression, 116
 - getVariable, 117
 - INPUT_BOOL, 115
 - INPUT_FLOAT, 115
 - INPUT_INTEGER, 115
 - INPUT_STRING, 115
 - InputMode, 115
 - OPERATOR, 115
 - setConfigType, 117
 - setExpression, 117
 - setVariable, 117
- arcs::aubo_scope::AuboCapInfo, 118
 - ~AuboCapInfo, 118
 - getName, 119
 - getSymbolicName, 119
 - getVendor, 119
- arcs::aubo_scope::BaseFeature, 119
 - ~BaseFeature, 122
 - BaseFeature, 121, 122

- d_, 122
- DataSwitch, 122
- arcs::aubo_scope::BoolRegister, 123
 - ~BoolRegister, 125
- arcs::aubo_scope::BreakNode, 125
 - ~BreakNode, 127
 - BreakNode, 126, 127
 - d_, 128
 - DataSwitch, 128
- arcs::aubo_scope::CallNode, 128
 - ~CallNode, 131
 - CallNode, 130, 131
 - d_, 131
 - DataSwitch, 131
- arcs::aubo_scope::CaseNode, 132
 - ~CaseNode, 134
 - CaseNode, 133, 134
 - d_, 134
 - DataSwitch, 134
- arcs::aubo_scope::CircleMoveNode, 135
 - ~CircleMoveNode, 137
 - CircleMoveNode, 136, 137
 - d_, 137
 - DataSwitch, 137
- arcs::aubo_scope::CommentNode, 138
 - ~CommentNode, 140
 - CommentNode, 139, 140
 - d_, 141
 - DataSwitch, 141
 - getComment, 140
 - setComment, 140
- arcs::aubo_scope::ContributionProvider< T >, 141
 - ~ContributionProvider, 142
 - get, 142
- arcs::aubo_scope::ConveyorTrackingNode, 143
 - ~ConveyorTrackingNode, 146
 - ABANDON_TRACKING, 145
 - ASK, 144
 - CLEAR_QUEUE, 144
 - CONTINUE_TRACKING, 145
 - ConveyorStartActionQueueType, 144
 - ConveyorStartActionType, 144
 - ConveyorTrackingNode, 145, 146
 - ConveyorTrackingTargetType, 145
 - d_, 146
 - DataSwitch, 146
 - NEW_WORKPIECE, 145
 - PREVIOUS_WORKPIECE, 145
 - RESERVE_QUEUE, 144
- arcs::aubo_scope::DataModel, 147
 - ~DataModel, 150
 - d_, 167
 - DataModel, 149, 150
 - DataSwitch, 166
 - get, 150–157
 - getKeys, 157
 - isSet, 157
 - remove, 157
 - set, 158–166
- arcs::aubo_scope::DefaultCaseNode, 167
 - ~DefaultCaseNode, 169
 - d_, 170
 - DataSwitch, 170
 - DefaultCaseNode, 169, 170
- arcs::aubo_scope::DefaultSingletonTraits< Type >, 170
- arcs::aubo_scope::Device, 171
 - ~Device, 171
 - getContributorInfo, 172
 - getDisplayName, 172
 - getId, 172
 - isResolvable, 172
- arcs::aubo_scope::DeviceManager, 173
 - ~DeviceManager, 173
 - getGrippers, 173
 - getScrewDrivers, 173
- arcs::aubo_scope::DeviceRegistrationManager, 174
 - ~DeviceRegistrationManager, 174
 - registerGripperContribution, 174
 - registerScrewDriverContribution, 175
- arcs::aubo_scope::DigitalIo, 175
 - ~DigitalIo, 178
 - d_, 180
 - DataSwitch, 179
 - DigitalIo, 177, 178
 - getValue, 179
 - setValue, 179
- arcs::aubo_scope::DigitalIoFilter, 180
 - DigitalIoFilter, 180
 - interface_, 181
 - is_input_, 181
 - operator(), 181
- arcs::aubo_scope::DirectionNode, 181
 - ~DirectionNode, 185
 - AXIS, 184
 - Axis, 183
 - d_, 188
 - DataSwitch, 188
 - DirectionNode, 185, 186
 - DirectionSelectionType, 183
 - EXPRESSION, 184
 - FEATURE, 184
 - FeatureSelectionType, 184
 - getAxis, 186
 - getDirectionSelectionType, 186
 - getExpression, 186
 - getFeature, 186
 - getFeatureSelectionType, 186
 - getToolAcceleration, 186
 - getToolSpeed, 187
 - getType, 187
 - ParameterType, 184
 - setAxis, 187
 - setDirectionSelectionType, 187
 - setExpression, 187
 - setFeature, 187

- setFeatureSelectionType, 187
- setToolAcceleration, 187
- setToolSpeed, 188
- setType, 188
- SHARED, 184
- SHARED_MOTION, 184
- TOOL_MOTION, 184
- UNRESOLVED_FEATURE, 184
- X_MINUS, 183
- X_PLUS, 183
- Y_MINUS, 183
- Y_PLUS, 183
- Z_MINUS, 183
- Z_PLUS, 183
- arcs::aubo_scope::DoubleRangeValidator, 189
 - DoubleRangeValidator, 190
 - getMessage, 190
 - is_max_inclusive_, 191
 - is_min_inclusive_, 191
 - isValid, 191
 - max_value_, 191
 - min_value_, 192
 - range_func_, 192
 - range_func_with_flags_, 192
 - updateValue, 191
- arcs::aubo_scope::DoubleRegister, 192
 - ~DoubleRegister, 194
- arcs::aubo_scope::DriveScrewNotOKParameters, 194
 - ~DriveScrewNotOKParameters, 195
- arcs::aubo_scope::DriveScrewOKParameters, 195
 - ~DriveScrewOKParameters, 195
- arcs::aubo_scope::ElseIfNode, 196
 - ~ElseIfNode, 198
 - d_, 199
 - DataSwitch, 199
 - ElseIfNode, 197, 198
 - getExpression, 198
 - setExpression, 198
- arcs::aubo_scope::ElseNode, 200
 - ~ElseNode, 202
 - d_, 202
 - DataSwitch, 202
 - ElseNode, 201, 202
- arcs::aubo_scope::EtherNetIP, 203
 - ~EtherNetIP, 206
 - d_, 208
 - DataSwitch, 207
 - EtherNetIP, 205, 206
 - getBitOffset, 206
 - getByteOffset, 206
 - getValue, 206
 - isSlave, 207
 - setValue, 207
- arcs::aubo_scope::EtherNetIPFilter, 208
 - EtherNetIPFilter, 208
 - is_input_, 209
 - operator(), 209
 - type_, 209
- arcs::aubo_scope::Expression, 209
 - ~Expression, 211
 - append, 211
 - appendFeature, 211
 - appendIo, 212
 - appendVariable, 212
 - appendWaypoint, 212
 - d_, 213
 - DataSwitch, 213
 - Expression, 210, 211
 - toString, 212
- arcs::aubo_scope::Feature, 213
 - ~Feature, 215
 - d_, 217
 - DataSwitch, 217
 - Feature, 215, 216
 - getDisplayName, 216
 - getPose, 216
 - hashCode, 216
 - isDefined, 216
 - isJoggable, 216
 - isResolvable, 217
 - isVariable, 217
- arcs::aubo_scope::FeatureContributionModel, 218
 - ~FeatureContributionModel, 219
 - addFeature, 220
 - d_, 222
 - DataSwitch, 222
 - FeatureContributionModel, 219, 220
 - getFeature, 220
 - removeFeature, 221
 - renameFeature, 221
 - updateFeature, 221
- arcs::aubo_scope::FeatureDefinedFilter, 222
 - operator(), 223
- arcs::aubo_scope::FeatureJoggableFilter, 223
 - operator(), 223
- arcs::aubo_scope::FeatureLine, 224
 - ~FeatureLine, 226
 - d_, 227
 - DataSwitch, 227
 - FeatureLine, 225, 226
- arcs::aubo_scope::FeatureModel, 227
 - ~FeatureModel, 229
 - d_, 231
 - DataSwitch, 231
 - FeatureModel, 228, 229
 - getAll, 229
 - getBaseFeature, 229
 - getByName, 229
 - getFeatures, 230
 - getGeomFeture, 230
 - getToolFeature, 230
- arcs::aubo_scope::FeaturePlane, 231
 - ~FeaturePlane, 234
 - d_, 234
 - DataSwitch, 234

- FeaturePlane, 233, 234
- arcs::aubo_scope::FeaturePoint, 235
 - ~FeaturePoint, 237
 - d_, 238
 - DataSwitch, 238
 - FeaturePoint, 237, 238
- arcs::aubo_scope::FeatureTool, 238
 - ~FeatureTool, 241
 - d_, 241
 - DataSwitch, 241
 - FeatureTool, 240, 241
- arcs::aubo_scope::FeatureVariable, 242
 - ~FeatureVariable, 245
 - d_, 245
 - DataSwitch, 245
 - FeatureVariable, 244, 245
- arcs::aubo_scope::FeatureVariableFilter, 246
 - operator(), 246
- arcs::aubo_scope::FeedScrewParameters, 246
 - ~FeedScrewParameters, 247
- arcs::aubo_scope::FolderNode, 247
 - ~FolderNode, 249
 - d_, 250
 - DataSwitch, 250
 - FolderNode, 248, 249
 - setFolderName, 250
- arcs::aubo_scope::ForceNode, 251
 - ~ForceNode, 253
 - ConfigType, 252
 - d_, 254
 - DataSwitch, 254
 - ForceNode, 253, 254
 - FRAME, 252
 - MOTION, 252
 - POINT, 252
 - SIMPLE, 252
- arcs::aubo_scope::ForceOptionNode, 254
 - ~ForceOptionNode, 257
 - d_, 257
 - DataSwitch, 257
 - ForceOptionNode, 256, 257
- arcs::aubo_scope::Function, 257
 - ~Function, 259
 - d_, 261
 - DataSwitch, 260
 - Function, 258, 259
 - getArguments, 260
 - getContributorInfo, 260
 - getName, 260
- arcs::aubo_scope::FunctionModel, 261
 - ~FunctionModel, 263
 - addFunction, 263
 - d_, 265
 - DataSwitch, 265
 - FunctionModel, 262, 263
 - getFunction, 264
 - getFunctions, 264
 - removeFunction, 264
- arcs::aubo_scope::GlobalVariable, 266
 - ~GlobalVariable, 268
 - d_, 269
 - DataSwitch, 269
 - GlobalVariable, 268, 269
- arcs::aubo_scope::GlobalVariableFilter, 269
 - operator(), 270
- arcs::aubo_scope::GripActionParameters, 270
 - ~GripActionParameters, 270
 - getForce, 270
 - getSpeed, 271
 - getVacuum, 271
 - getWidth, 271
 - isGripDetectionEnabled, 272
- arcs::aubo_scope::GripForceCapability, 273
 - ~GripForceCapability, 273
 - updateCapability, 273
- arcs::aubo_scope::GripperApiProvider, 274
 - ~GripperApiProvider, 274
 - getIoModel, 275
 - getSystemApi, 275
- arcs::aubo_scope::GripperConfiguration, 275
 - ~GripperConfiguration, 276
 - registerGripDetectedCapability, 276
 - registerGrippingForceCapability, 277
 - registerGrippingVacuumCapability, 278
 - registerReleaseDetectedCapability, 279
 - registerSpeedCapability, 280
 - registerWidthCapability, 281
- arcs::aubo_scope::GripperContribution, 282
 - ~GripperContribution, 283
 - configureGripper, 283
 - generateGripActionScript, 284
 - generateReleaseActionScript, 284
 - getTitle, 284
- arcs::aubo_scope::GripperDevice, 286
 - ~GripperDevice, 287
- arcs::aubo_scope::GripperNode, 287
 - ~GripperNode, 291
 - ConfigType, 289
 - CUSTOM_PARAMETERS, 290
 - d_, 292
 - DataSwitch, 292
 - getConfigType, 291
 - getGripperDevice, 291
 - getSettingType, 291
 - GRIP_ACTION, 289
 - GripperNode, 290, 291
 - PAYLOAD_SELECTION, 289
 - RELEASE_ACTION, 289
 - setConfigType, 292
 - SettingType, 289
 - UNDEFINED, 289
 - UNDEFINED0, 289
- arcs::aubo_scope::GripVacuumCapability, 292
 - ~GripVacuumCapability, 293
 - updateCapability, 293
- arcs::aubo_scope::HaltNode, 294

- ~HaltNode, 296
- d_, 297
- DataSwitch, 297
- HaltNode, 296, 297
- arcs::aubo_scope::IfNode, 297
 - ~IfNode, 300
 - d_, 301
 - DataSwitch, 301
 - getExpression, 300
 - IfNode, 299, 300
 - setExpression, 300
- arcs::aubo_scope::InputIoFilter, 301
 - operator(), 302
- arcs::aubo_scope::InputValidator, 302
 - getMessage, 303
 - isValid, 303
- arcs::aubo_scope::InstallationApi, 304
 - ~InstallationApi, 306
 - d_, 307
 - DataSwitch, 307
 - InstallationApi, 306, 307
- arcs::aubo_scope::InstallationApiProvider, 307
 - ~InstallationApiProvider, 309
 - d_, 310
 - DataSwitch, 310
 - getInstallationApi, 309
 - getSystemApi, 309
 - getUserInterfaceApi, 309
 - InstallationApiProvider, 308, 309
- arcs::aubo_scope::InstallationContributionConfiguration, 310
 - ~InstallationContributionConfiguration, 311
- arcs::aubo_scope::InstallationCreationContext, 311
 - ~InstallationCreationContext, 313
 - d_, 314
 - DataSwitch, 313
 - getCreatedVersion, 313
 - getLastSavedVersion, 313
 - getNodeCreationType, 313
 - InstallationCreationContext, 312, 313
 - LOAD, 312
 - NEW, 312
 - NodeCreationType, 311
- arcs::aubo_scope::InstallationNodeContribution, 314
 - ~InstallationNodeContribution, 314
 - closeView, 314
 - generateScript, 314
 - openView, 315
- arcs::aubo_scope::InstallationNodeService, 315
 - ~InstallationNodeService, 315
 - configureContribution, 316
 - createInstallationNode, 316
 - createView, 316
 - getIcon, 317
 - getTitle, 317
- arcs::aubo_scope::InstallationNodeView, 318
 - ~InstallationNodeView, 318
 - buildUi, 318
- arcs::aubo_scope::IntegerRangeValidator, 319
 - getMessage, 320
 - IntegerRangeValidator, 320
 - isValid, 321
 - max_value_, 321
 - min_value_, 321
 - range_func_, 321
 - updateValue, 321
- arcs::aubo_scope::IntegerRegister, 322
 - ~IntegerRegister, 324
- arcs::aubo_scope::Io, 324
 - ~Io, 327
 - ANALOG, 326
 - BOOLEAN, 326
 - CONFIGURABLE, 326
 - d_, 330
 - DataSwitch, 330
 - DIGITAL, 326
 - DOUBLE, 326
 - ETHERNET_IP, 326
 - GENERAL_PURPOSE, 326
 - getAction, 328
 - getDefaultName, 328
 - getDisplayName, 328
 - getInterfaceType, 328
 - getType, 328
 - getValueStr, 329
 - hashCode, 329
 - INTEGER, 326
 - InterfaceType, 325
 - Io, 326, 327
 - IoType, 326
 - isInput, 329
 - isResolvable, 329
 - LINKIO, 326
 - MODBUS, 326
 - PROFINET, 326
 - STANDARD, 326
 - TOOL, 326
- arcs::aubo_scope::IoConfigure, 330
 - function_lock, 331
 - functions, 331
 - label, 331
 - maxs, 331
 - mins, 331
 - names, 331
 - num, 332
 - safety_functions, 332
 - safety_functions_fixed, 332
 - sequences, 332
 - types, 332
- arcs::aubo_scope::IoModel, 332
 - ~IoModel, 334
 - d_, 336
 - DataSwitch, 336
 - getAll, 335

- getIo, 335
- getIos, 335
- IoModel, 333, 334
- arcs::aubo_scope::KeyboardManager, 336
 - ~KeyboardManager, 339
 - d_, 342
 - DataSwitch, 342
 - KeyboardManager, 338, 339
 - registerCodeEditorKeyboard, 339
 - registerDoubleKeyboard, 339
 - registerExpressionKeyboard, 340
 - registerIntegerKeyboard, 340
 - registerIPAddressKeyboard, 340
 - registerNumberKeyboard, 340
 - registerPasswordKeyboard, 340
 - registerPositiveDoubleKeyboard, 341
 - registerPositiveIntegerKeyboard, 341
 - registerStandardKeyboard, 341
 - registerStringKeyboard, 342
- arcs::aubo_scope::LoopNode, 343
 - ~LoopNode, 346
 - ALWAYS, 345
 - ConfigType, 344
 - COUNTER, 345
 - d_, 348
 - DataSwitch, 348
 - EXPRESSION, 345
 - getConfigType, 346
 - getExpression, 346
 - getLoopCount, 346
 - getLoopCountVariable, 346
 - isEvaluateContinously, 347
 - LoopNode, 345, 346
 - setConfigType, 347
 - setEvaluateContinously, 347
 - setExpression, 347
 - setLoopCount, 347
 - setLoopCountVariable, 348
- arcs::aubo_scope::ModbusIo, 349
 - ~ModbusIo, 352
 - d_, 354
 - DataSwitch, 353
 - getIpAddress, 352
 - getSignalAddress, 352
 - getValue, 352
 - isSlave, 353
 - ModbusIo, 351, 352
 - setValue, 353
- arcs::aubo_scope::ModbusIoFilter, 354
 - is_input_, 355
 - ModbusIoFilter, 354
 - operator(), 355
 - type_, 355
- arcs::aubo_scope::MoveNode, 355
 - ~MoveNode, 359
 - ConfigType, 357
 - d_, 363
 - DataSwitch, 363
 - FEATURE, 358
 - FeatureSelectionType, 357
 - getBlend, 359
 - getConfigType, 359
 - getFeature, 360
 - getJointAcceleration, 360
 - getJointSpeed, 360
 - getTCPSelection, 360
 - getTcpSelectionType, 360
 - getToolAcceleration, 360
 - getToolSpeed, 360
 - getUseJoints, 360
 - IGNORE_ACTIVE_TCP, 358
 - MOVE_J, 357
 - MOVE_L, 357
 - MOVE_S, 357
 - MOVE_TCP, 358
 - MoveNode, 358, 359
 - setBlend, 360
 - setConfigType, 361
 - setFeature, 361
 - setJointAcceleration, 361
 - setJointSpeed, 362
 - setTCPSelection, 362
 - setTcpSelectionType, 362
 - setToolAcceleration, 362
 - setToolSpeed, 363
 - setUseJoints, 363
 - TcpSelectionType, 358
 - UNRESOLVED_FEATURE, 358
 - USE_ACTIVE_TCP, 358
- arcs::aubo_scope::OfflineTrackNode, 364
 - ~OfflineTrackNode, 366
 - d_, 366
 - DataSwitch, 366
 - OfflineTrackNode, 365, 366
- arcs::aubo_scope::OutputIoFilter, 367
 - operator(), 367
- arcs::aubo_scope::PalletNode, 367
 - ~PalletNode, 370
 - d_, 370
 - DataSwitch, 370
 - PalletNode, 369, 370
- arcs::aubo_scope::Payload, 371
 - ~Payload, 373
 - d_, 376
 - DataSwitch, 375
 - getCenterOfGravity, 373
 - getDisplayName, 373
 - getInertiaMatrix, 374
 - getMass, 374
 - getParametersWarning, 374
 - hashCode, 374
 - isResolvable, 375
 - isUsable, 375
 - MASS_0KG, 372
 - MASS_AND_CENTER_OF_GRAVITY_EXCEEDED, 372

- NO_WARNING, 372
- ParametersWarning, 371
- Payload, 372, 373
- RATED_MASS_EXCEEDED, 372
- arcs::aubo_scope::PayloadContributionModel, 376
 - ~PayloadContributionModel, 378
 - addPayload, 378, 379
 - d_, 383
 - DataSwitch, 383
 - getPayload, 381
 - PayloadContributionModel, 377, 378
 - removePayload, 381
 - updatePayload, 382
- arcs::aubo_scope::PayloadModel, 384
 - ~PayloadModel, 385
 - d_, 387
 - DataSwitch, 387
 - getCurrentPayload, 386
 - getDefaultPayload, 386
 - getPayload, 386
 - getPayloads, 387
 - PayloadModel, 385, 386
- arcs::aubo_scope::PersistedVariable, 388
 - ~PersistedVariable, 390
 - d_, 391
 - DataSwitch, 391
 - isResolvable, 391
 - PersistedVariable, 390, 391
- arcs::aubo_scope::PersistedVariableFilter, 392
 - operator(), 392
- arcs::aubo_scope::PopupManager, 392
 - ~PopupManager, 394
 - d_, 395
 - DefaultSingletonTraits< PopupManager >, 395
 - isPopupCountZero, 394
 - notifyPopupHidden, 394
 - notifyPopupShown, 394
 - popupCount, 394
 - PopupManager, 393, 394
 - setPopupCountChangedCallback, 394
- arcs::aubo_scope::PopupNode, 396
 - ~PopupNode, 398
 - d_, 400
 - DataSwitch, 400
 - ERROR, 397
 - getMessage, 399
 - getType, 399
 - MESSAGE, 397
 - PopupNode, 398, 399
 - setMessage, 399
 - setType, 399
 - Type, 397
 - WARNING, 397
- arcs::aubo_scope::PrepareToStartScrewdriverParameters, ProgramCreationContext, 422, 423
 - 400
 - ~PrepareToStartScrewdriverParameters, 401
- arcs::aubo_scope::Profinet, 401
 - ~Profinet, 404
 - d_, 406
 - DataSwitch, 406
 - getBitOffset, 405
 - getByteOffset, 405
 - getValue, 405
 - isSlave, 405
 - Profinet, 403, 404
 - setValue, 405
- arcs::aubo_scope::ProfinetFilter, 406
 - is_input_, 407
 - operator(), 407
 - ProfinetFilter, 407
 - type_, 407
- arcs::aubo_scope::ProgramApi, 408
 - ~ProgramApi, 410
 - d_, 411
 - DataSwitch, 411
 - ProgramApi, 410, 411
- arcs::aubo_scope::ProgramApiProvider, 411
 - ~ProgramApiProvider, 413
 - d_, 414
 - DataSwitch, 414
 - getProgramApi, 413
 - getSystemApi, 413
 - getUiThreadApi, 413
 - ProgramApiProvider, 412, 413
- arcs::aubo_scope::ProgramContributionConfiguration, 414
 - ~ProgramContributionConfiguration, 416
 - d_, 420
 - DataSwitch, 420
 - ProgramContributionConfiguration, 416, 417
 - setAllowBreakpointOnChildNodesInSubtree, 417
 - setAllowBreakpointOnNode, 417
 - setAllowInsertOnlyIntoNodes, 417
 - setAllowStartFromChildNodesInSubtree, 418
 - setAllowStartFromNode, 418
 - setChildrenAllowed, 418
 - setChildrenAllowedOnly, 419
 - setDeprecated, 419
 - setDisplayOrderId, 419
 - setUserInsertable, 420
- arcs::aubo_scope::ProgramCreationContext, 421
 - ~ProgramCreationContext, 423
 - d_, 424
 - DataSwitch, 424
 - getCreatedVersion, 423
 - getLastSavedVersion, 423
 - getNodeCreationType, 423
 - LOAD, 422
 - NEW, 422
 - NodeCreationType, 421
- arcs::aubo_scope::ProgramHomeNode, 424
 - ~ProgramHomeNode, 427
 - d_, 427

- DataSwitch, [427](#)
- ProgramHomeNode, [426](#), [427](#)
- arcs::aubo_scope::ProgramModel, [427](#)
 - ~ProgramModel, [429](#)
 - d_, [431](#)
 - DataSwitch, [430](#)
 - getProgramNodeFactory, [430](#)
 - getTreeNode, [430](#)
 - ProgramModel, [429](#), [430](#)
- arcs::aubo_scope::ProgramNode, [431](#)
 - ~ProgramNode, [434](#)
 - d_, [435](#)
 - DataSwitch, [434](#)
 - getProgramNodeContribution, [434](#)
 - ProgramNode, [433](#), [434](#)
- arcs::aubo_scope::ProgramNodeContribution, [435](#)
 - ~ProgramNodeContribution, [435](#)
 - closeView, [436](#)
 - generateScript, [436](#)
 - getAction, [436](#)
 - getTitle, [436](#)
 - isDefined, [437](#)
 - openView, [437](#)
 - postAction, [437](#)
- arcs::aubo_scope::ProgramNodeFactory, [438](#)
 - ~ProgramNodeFactory, [440](#)
 - createAssemblyNode, [441](#)
 - createAssignmentNode, [441](#)
 - createBreakNode, [441](#)
 - createCallNode, [441](#)
 - createCaseNode, [441](#)
 - createCircleMoveNode, [441](#)
 - createCommentNode, [441](#)
 - createConveyorTrackingNode, [441](#)
 - createDefaultCaseNode, [441](#)
 - createDirectionNode, [442](#)
 - createElseIfNode, [442](#)
 - createElseNode, [442](#)
 - createFolderNode, [442](#)
 - createForceNode, [442](#)
 - createForceOptionNode, [442](#)
 - createGripperNode, [442](#)
 - createHaltNode, [443](#)
 - createIfNode, [443](#)
 - createLoopNode, [443](#)
 - createMoveNode, [443](#)
 - createMoveNodeNoTemplate, [443](#)
 - createOfflineTrackNode, [444](#)
 - createPalletNode, [444](#)
 - createPopupNode, [444](#)
 - createProgramHomeNode, [444](#)
 - createRecordTrackNode, [444](#)
 - createScrewdrivingNode, [444](#)
 - createScriptNode, [444](#)
 - createSeekNode, [444](#)
 - createSetNode, [444](#)
 - createSetPayloadNode, [445](#)
 - createSubProgNode, [445](#)
 - createSwitchNode, [445](#)
 - createThreadNode, [445](#)
 - createTimerNode, [445](#)
 - createUntilNode, [445](#)
 - createUserProgramNode, [445](#)
 - createVisualLocalizationProgramNode, [446](#)
 - createWaitNode, [446](#)
 - createWaypointNode, [446](#)
 - d_, [447](#)
 - DataSwitch, [447](#)
 - ProgramNodeFactory, [440](#), [441](#)
- arcs::aubo_scope::ProgramNodeService, [447](#)
 - ~ProgramNodeService, [448](#)
 - configureContribution, [448](#)
 - createNode, [448](#)
 - createView, [449](#)
 - getIcon, [449](#)
 - getId, [449](#)
 - getTitle, [450](#)
- arcs::aubo_scope::ProgramNodeView, [450](#)
 - ~ProgramNodeView, [451](#)
 - buildUi, [451](#)
- arcs::aubo_scope::ProgramSelectionParameters, [451](#)
 - ~ProgramSelectionParameters, [452](#)
 - getScrewdriverOperationType, [452](#)
 - getScrewdriverProgram, [452](#)
- arcs::aubo_scope::RecordTrackNode, [453](#)
 - ~RecordTrackNode, [455](#)
 - d_, [455](#)
 - DataSwitch, [455](#)
 - RecordTrackNode, [454](#), [455](#)
- arcs::aubo_scope::Register, [456](#)
 - ~Register, [457](#)
- arcs::aubo_scope::RegisteredCapabilities, [457](#)
 - ~RegisteredCapabilities, [458](#)
 - getGripForceCapability, [458](#)
 - getGripVacuumCapability, [459](#)
 - getSpeedCapability, [459](#)
 - getWidthCapability, [459](#)
- arcs::aubo_scope::ReleaseActionParameters, [460](#)
 - ~ReleaseActionParameters, [460](#)
 - getSpeed, [460](#)
 - getWidth, [461](#)
 - isReleaseDetectionEnabled, [461](#)
- arcs::aubo_scope::RenderInterface, [462](#)
 - ~RenderInterface, [466](#)
 - addCoordinate, [466](#)
 - addCube, [466–468](#)
 - addLine, [469](#)
 - addMesh, [470](#)
 - addPlane, [470](#), [471](#)
 - addPointCloud, [471](#)
 - addWaypoint, [471](#)
 - All, [465](#)
 - clearCubes, [472](#)
 - clearLines, [472](#)
 - clearMeshes, [472](#)

- clearPlanes, 472
- clearPointCloud, 473
- clearWaypoints, 473
- Coordinate, 464
- d_, 479
- DataSwitch, 479
- getMeshes, 473
- Ground, 464
- hasCoordinate, 473
- hasPointCloud, 474
- hasWaypoint, 474
- None, 464
- Obstacle, 465
- Point, 464
- PointCloud, 465
- RealRobot, 464
- removeCoordinate, 475
- removeCube, 475
- removeLine, 475
- removeMesh, 476
- removePlane, 476
- removePointCloud, 476
- removeWaypoint, 477
- RenderInterface, 465, 466
- SafetyPlane, 465
- setBlendTrack, 477
- setMoveTrack, 478
- setRenderEmbedded, 478
- setShowOptions, 479
- ShowOptions, 464
- TargetRobot, 464
- ToolPosition, 465
- Track, 464
- View, 464
- arcs::aubo_scope::ScrewdriverApiProvider, 480
 - ~ScrewdriverApiProvider, 480
 - getIOModel, 480
 - getSystemAPI, 480
- arcs::aubo_scope::ScrewdriverConfiguration, 481
 - ~ScrewdriverConfiguration, 481
 - addScrewdriverProgram, 481
 - addScrewdriverPrograms, 482
 - registerDriveScrewNotOKCapability, 482
 - registerDriveScrewOKCapability, 484
 - registerFeedScrewCapability, 485
 - registerOperationTypeCapability, 485
 - registerPrepareToStartScrewdriverCapability, 486
 - registerProgramSelectionCapability, 486
 - registerScrewdriverReadyCapability, 487
- arcs::aubo_scope::ScrewdriverContribution, 488
 - ~ScrewdriverContribution, 489
 - configureScrewdriver, 489
 - generateStartScrewdriverScript, 490
 - generateStopScrewdriverScript, 490
 - getTitle, 490
- arcs::aubo_scope::ScrewdriverParameters, 491
 - ~ScrewdriverParameters, 492
 - getScrewdriverOperationType, 492
 - getScrewdriverProgram, 492
- arcs::aubo_scope::ScrewdriverProgram, 493
 - ~ScrewdriverProgram, 493
 - getDisplayName, 493
 - getId, 493
- arcs::aubo_scope::ScrewdriverReadyParameters, 494
 - ~ScrewdriverReadyParameters, 494
- arcs::aubo_scope::ScrewdrivingNode, 495
 - ~ScrewdrivingNode, 497
 - d_, 497
 - DataSwitch, 497
 - ScrewdrivingNode, 496, 497
- arcs::aubo_scope::ScriptNode, 498
 - ~ScriptNode, 500
 - d_, 501
 - DataSwitch, 500
 - ScriptNode, 499, 500
- arcs::aubo_scope::ScriptWriter, 501
 - ~ScriptWriter, 504
 - anonyFunction, 504
 - appendLine, 504
 - appendRaw, 504
 - appendVectorDouble, 505
 - assign, 505, 506
 - d_, 513
 - DataSwitch, 512
 - decreaseIndent, 506
 - defineFunction, 506
 - defineThread, 507
 - elseCondition, 507
 - elseifCondition, 507
 - end, 507
 - forCondition, 507
 - generateScript, 508
 - getResolvedVariable, 508
 - ifCondition, 508
 - ifNotCondition, 508
 - increaseIndent, 509
 - incrementVariable, 509
 - killThread, 509
 - lineFeed, 509
 - note, 509
 - returnMethod, 510
 - runThread, 510
 - ScriptWriter, 503, 504
 - setLabel, 510
 - setPayload, 510
 - setRobotIndex, 510
 - setTcp, 510
 - sleep, 511
 - sync, 511
 - whileCondition, 511
 - whileNot, 512
 - whileTrue, 512
 - writeChildren, 512
- arcs::aubo_scope::SeekNode, 513

- ~SeekNode, 515
- d_, 516
- DataSwitch, 516
- SeekNode, 514, 515
- arcs::aubo_scope::SetNode, 516
 - ~SetNode, 520
 - ANALOG_OUTPUT_CURRENT, 519
 - ANALOG_OUTPUT_VOLTAGE, 519
 - ConfigType, 518
 - d_, 524
 - DataSwitch, 524
 - DIGITAL_OUTPUT, 519
 - EXPRESSION_OUTPUT, 519
 - FLOAT_REGISTER_OUTPUT, 519
 - getAnalogOutput, 521
 - getBoolValueToSet, 521
 - getConfigType, 521
 - getCurrent, 521
 - getExpression, 522
 - getFloatValueToSet, 522
 - getOutput, 522
 - getOutputSelectionType, 522
 - getPulseTime, 522
 - getVoltage, 522
 - INCREMENT_VARIABLE, 519
 - NO_ACTION, 518
 - NO_OUTPUT, 519
 - NO_TYPE, 518
 - OUTPUT, 519
 - OutputSelectionType, 519
 - setAnalogOutput, 523
 - setBoolValueToSet, 523
 - setConfigType, 523
 - setCurrent, 523
 - setExpression, 523
 - setFloatValueToSet, 523
 - SetNode, 520, 521
 - setOutput, 523
 - setPulseTime, 523
 - setVoltage, 524
 - SINGLE_DIGITAL_PULSE, 519
 - UNRESOLVED_OUTPUT, 519
- arcs::aubo_scope::SetPayloadNode, 525
 - ~SetPayloadNode, 528
 - ConfigType, 526
 - CUSTOM_PARAMETERS, 527
 - d_, 529
 - DataSwitch, 529
 - getCenterOfGravity, 528
 - getConfigType, 528
 - getMass, 528
 - PAYLOAD_SELECTION, 526
 - setCenterOfGravity, 529
 - setConfigType, 529
 - setMass, 529
 - SetPayloadNode, 527, 528
- arcs::aubo_scope::SpeedCapability, 530
 - ~SpeedCapability, 530
 - updateCapability, 530
- arcs::aubo_scope::StringLengthValidator, 531
 - getMessage, 533
 - isValid, 533
 - max_length_, 534
 - min_length_, 534
 - range_func_, 534
 - StringLengthValidator, 532
 - updateValue, 533
- arcs::aubo_scope::SubProgNode, 534
 - ~SubProgNode, 537
 - d_, 537
 - DataSwitch, 537
 - SubProgNode, 536, 537
- arcs::aubo_scope::SwitchNode, 538
 - ~SwitchNode, 540
 - d_, 541
 - DataSwitch, 540
 - SwitchNode, 539, 540
- arcs::aubo_scope::SystemApi, 541
 - ~SystemApi, 544
 - d_, 544
 - DataSwitch, 544
 - SystemApi, 543, 544
- arcs::aubo_scope::TCP, 545
 - ~TCP, 546
 - d_, 548
 - DataSwitch, 548
 - getDisplayName, 547
 - getOffset, 547
 - hashCode, 547
 - isResolvable, 547
 - TCP, 546, 547
- arcs::aubo_scope::TcpContributionModel, 548
 - ~TcpContributionModel, 550
 - addTcp, 551
 - d_, 553
 - DataSwitch, 552
 - getTcp, 551
 - removeTcp, 551
 - TcpContributionModel, 549, 550
 - updateTcp, 552
- arcs::aubo_scope::TcpModel, 553
 - ~TcpModel, 555
 - connectedTo, 555
 - d_, 557
 - DataSwitch, 557
 - disconnectFrom, 556
 - getAll, 556
 - getDefaultTcp, 556
 - getTcp, 556
 - TcpModel, 554, 555
- arcs::aubo_scope::ThreadNode, 558
 - ~ThreadNode, 560
 - d_, 561
 - DataSwitch, 560
 - ThreadNode, 559, 560
- arcs::aubo_scope::TimerNode, 561

- ~TimerNode, 563
- d_, 564
- DataSwitch, 564
- TimerNode, 562, 563
- arcs::aubo_scope::TreeNode, 564
 - ~TreeNode, 566
 - addChild, 567
 - cutChildNode, 567
 - d_, 573
 - DataSwitch, 572
 - getChildren, 568
 - getChildrenNotSuppressed, 568
 - getParent, 568
 - getProgramNode, 568
 - insertChildAfter, 568
 - insertChildBefore, 569
 - isInProgramTree, 570
 - isSuppressed, 570
 - locateDescendantTreeNode, 570
 - removeChild, 571
 - setChildSequenceLocked, 571
 - traverse, 572
 - TreeNode, 566, 567
- arcs::aubo_scope::UIntegerRangeValidator, 573
 - getMessage, 574
 - isValid, 575
 - max_value_, 575
 - min_value_, 575
 - range_func_, 575
 - UIntegerRangeValidator, 574
 - updateValue, 575
- arcs::aubo_scope::UndoRedoManager, 576
 - ~UndoRedoManager, 577
 - d_, 578
 - DataSwitch, 578
 - recordChanges, 578
 - UndoRedoManager, 577, 578
- arcs::aubo_scope::UntilNode, 579
 - ~UntilNode, 584
 - ANALOG_INPUT, 583
 - Angular, 583
 - BlendParameterType, 581
 - Cartesian, 583
 - CompareOperator, 581
 - ConfigType, 582
 - CUSTOM_BLEND, 581
 - CUSTOM_DECELERATION, 583
 - CustomDecelerationParameterType, 582
 - d_, 591
 - DataSwitch, 591
 - DecelerationParameterType, 583
 - DIGITAL_INPUT, 583
 - DISTANCE, 582
 - EXPRESSION, 582
 - FLOAT_REGISTER_INPUT, 583
 - getAnalogInputValueToProceed, 585
 - getAngularDeceleration, 585
 - getBlendParameterType, 585
 - getCartesianDeceleration, 585
 - getCompareOperator, 585
 - getConfigType, 585
 - getCustomDecelerationParameterType, 586
 - getDecelerationParameters, 586
 - getDecelerationParameterType, 586
 - getDigitalInputValueToProceed, 586
 - getDistance, 586
 - getExpressionConfig, 586
 - getIOInputSelectionType, 586
 - getWaypointBlendParameter, 587
 - GREATER_THAN, 582
 - IO_INPUT, 582
 - IOInputSelectionType, 583
 - LESS_THAN, 582
 - NO_BLEND, 581
 - NONE_CONFIG_TYPE, 582
 - NONE_INPUT_SELECT, 583
 - REACHED_WAYPOINT, 582
 - setAnalogInputValueToProceed, 587
 - setAngularDeceleration, 587
 - setBlendParameterType, 588
 - setCartesianDeceleration, 588
 - setCompareOperator, 588
 - setConfigType, 588
 - setCustomDecelerationParameterType, 588
 - setDecelerationParameters, 588
 - setDecelerationParameterType, 589
 - setDigitalInputValueToProceed, 589
 - setDistance, 589
 - setExpressionConfig, 590
 - setIOInputSelectionType, 590
 - setWaypointBlendParameter, 590
 - SHARED_DECELERATION, 583
 - TOOL_CONTACT, 582
 - Unknown, 582
 - UntilNode, 584, 585
- arcs::aubo_scope::UserInteraction, 591
 - ~UserInteraction, 593
 - d_, 599
 - DataSwitch, 599
 - getKeyboardManager, 593
 - getPopupManager, 593
 - getRenderInterface, 593
 - getRobotThumb, 594
 - getUserDefinedRobotPosition, 594, 595
 - requestUserToMoveJoint, 595, 596
 - requestUserToMoveLine, 597
 - setOperationLogDisplayName, 598
 - UserInteraction, 592, 593
- arcs::aubo_scope::UserInterfaceApi, 599
 - ~UserInterfaceApi, 601
 - d_, 601
 - DataSwitch, 601
 - UserInterfaceApi, 600, 601
- arcs::aubo_scope::ValueFactory, 601
 - ~ValueFactory, 603
 - createExpression, 604

- d_, 604
- DataSwitch, 604
- ValueFactory, 602, 603
- arcs::aubo_scope::Variable, 605
 - ~Variable, 607
- d_, 609
- DataSwitch, 609
- equals, 608
- FEATURE, 606
- getDisplayName, 608
- getType, 608
- GLOBAL, 606
- hashCode, 608
- toString, 608
- Type, 606
- VALUE_PERSISTED, 606
- Variable, 607, 608
- arcs::aubo_scope::VariableFactory, 609
 - ~VariableFactory, 611
 - createFeatureVariable, 611
 - createGlobalVariable, 612
- d_, 613
- DataSwitch, 613
- VariableFactory, 610, 611
- arcs::aubo_scope::VariableModel, 613
 - ~VariableModel, 615
- connectedTo, 615
- d_, 618
- DataSwitch, 618
- disconnectFrom, 616
- get, 616
- getAll, 617
- getFeatureVariable, 617
- getVariableFactory, 617
- VariableModel, 614, 615
- arcs::aubo_scope::ViewApiProvider, 618
 - ~ViewApiProvider, 620
- d_, 621
- DataSwitch, 621
- getSystemApi, 620
- getUserInterfaceApi, 620
- ViewApiProvider, 619
- arcs::aubo_scope::VisualLocalizationProgramNode, 621
 - ~VisualLocalizationProgramNode, 624
- d_, 624
- DataSwitch, 624
- VisualLocalizationProgramNode, 623, 624
- arcs::aubo_scope::WaitNode, 625
 - ~WaitNode, 629
- ANALOG_INPUT_CURRENT, 628
- ANALOG_INPUT_VOLTAGE, 628
- CompareOperator, 627
- ConfigType, 627
- d_, 635
- DataSwitch, 635
- DIGITAL_INPUT, 627
- EXPRESSION_INPUT, 628
- FLOAT_REGISTER_INPUT, 628
- getAnalogInput, 629
- getBoolRegister, 629
- getBoolValueToWaitFor, 629
- getCompareOperator, 629
- getConfigType, 630
- getDigitalInput, 630
- getDoubleRegisterInput, 630
- getExpressionInputConfig, 630
- getFloatValueToWaitFor, 630
- getModbusInput, 630
- getWaitTime, 630
- GREATER_THAN, 627
- LESS_THAN, 627
- NO_TYPE, 627
- NO_WAIT, 627
- setAnalogInput, 630
- setBoolRegister, 631
- setBoolValueToWaitFor, 632
- setCompareOperator, 632
- setConfigType, 632
- setDigitalInput, 632
- setDoubleRegisterInput, 633
- setExpressionInputConfig, 633
- setFloatValueToWaitFor, 634
- setModbusInput, 634
- setWaitTime, 634
- TIME, 627
- WaitNode, 628, 629
- arcs::aubo_scope::Waypoint, 635
 - ~Waypoint, 637
- d_, 639
- DataSwitch, 638
- getDisplayName, 637
- getJointPositions, 637
- getPose, 637
- getTcpOffset, 638
- hashCode, 638
- isDefined, 638
- Waypoint, 636, 637
- arcs::aubo_scope::WaypointModel, 639
 - ~WaypointModel, 641
- addWaypoint, 641, 642
- d_, 644
- DataSwitch, 644
- getByName, 642
- getWaypoints, 643
- removeWaypoint, 643
- renameWaypoint, 643
- WaypointModel, 640, 641
- arcs::aubo_scope::WaypointNode, 644
 - ~WaypointNode, 650
- BlendType, 646
- ConfigType, 647
- CUSTOM, 648
- d_, 657
- DataSwitch, 656
- DEFINED, 648, 649

- FIXED_POSITION, 647
- getBlend, 650
- getBlendType, 650
- getConfigType, 651
- getJointAcceleration, 651
- getJointPositions, 651
- getJointSpeed, 651
- getMotionType, 651
- getPose, 652
- getPositionDefinition, 652
- getTcpOffset, 652
- getTime, 652
- getToolAcceleration, 652
- getToolSpeed, 652
- getVariable, 653
- getVariableDefinition, 653
- INPUT, 648
- JOINT_MOTION, 647
- MotionType, 647
- NO_BLEND, 647
- PositionDefinition, 647
- RELATIVE_POSITION, 647
- RelativeOffsetType, 648
- RelativePositionType, 648
- setFixedPositionConfig, 653
- setJointMotionParameters, 654
- setSharedMotionParameters, 654
- setTimeMotionParameters, 654
- setToolMotionParameters, 655
- setVariablePositionConfig, 655, 656
- SHARED_BLEND, 647
- SHARED_MOTION, 647
- TEACHING, 648
- TIME_MOTION, 647
- TOOL_MOTION, 647
- UNDEFINED, 648, 649
- VARIABLE, 648
- VARIABLE_JOINT, 649
- VARIABLE_POSE, 649
- VARIABLE_POSITION, 647
- VariableDefinition, 648
- VariableType, 649
- WAYPOINT_BLEND, 647
- WaypointNode, 649, 650
- arcs::aubo_scope::WidthCapability, 657
 - ~WidthCapability, 657
 - updateCapability, 657
- arcs::aubo_sdk, 97
 - ARCS_CLASS_FORWARD, 97
- ARCS_ABI_EXPORT
 - class_forward.h, 660
- ARCS_ABI_IMPORT
 - class_forward.h, 660
- ARCS_ABI_LOCAL
 - class_forward.h, 660
- ARCS_CLASS_FORWARD
 - arcs::aubo_scope, 82–96
 - arcs::aubo_sdk, 97
 - class_forward.h, 660
- ARCS_DECLARE_PTR
 - class_forward.h, 660
- ARCS_DECLARE_PTR_MEMBER
 - class_forward.h, 660
- ARCS_STRUCT_FORWARD
 - class_forward.h, 661
- ASK
 - arcs::aubo_scope::ConveyorTrackingNode, 144
- AssemblyNode
 - arcs::aubo_scope::AssemblyNode, 111, 112
- assign
 - arcs::aubo_scope::ScriptWriter, 505, 506
- AssignmentNode
 - arcs::aubo_scope::AssignmentNode, 115, 116
- AuboStudio SDK, 1
- AXIS
 - arcs::aubo_scope::DirectionNode, 184
- Axis
 - arcs::aubo_scope::DirectionNode, 183
- BaseFeature
 - arcs::aubo_scope::BaseFeature, 121, 122
- BlendParameterType
 - arcs::aubo_scope::UntilNode, 581
- BlendType
 - arcs::aubo_scope::WaypointNode, 646
- BOOLEAN
 - arcs::aubo_scope::Io, 326
- BreakNode
 - arcs::aubo_scope::BreakNode, 126, 127
- buildUi
 - arcs::aubo_scope::InstallationNodeView, 318
 - arcs::aubo_scope::ProgramNodeView, 451
- CallNode
 - arcs::aubo_scope::CallNode, 130, 131
- Cartesian
 - arcs::aubo_scope::UntilNode, 583
- CaseNode
 - arcs::aubo_scope::CaseNode, 133, 134
- CircleMoveNode
 - arcs::aubo_scope::CircleMoveNode, 136, 137
- class_forward.h
 - ARCS_ABI_EXPORT, 660
 - ARCS_ABI_IMPORT, 660
 - ARCS_ABI_LOCAL, 660
 - ARCS_CLASS_FORWARD, 660
 - ARCS_DECLARE_PTR, 660
 - ARCS_DECLARE_PTR_MEMBER, 660
 - ARCS_STRUCT_FORWARD, 661
- CLEAR_QUEUE
 - arcs::aubo_scope::ConveyorTrackingNode, 144
- clearCubes
 - arcs::aubo_scope::RenderInterface, 472
- clearLines
 - arcs::aubo_scope::RenderInterface, 472

- clearMeshes
 - arcs::aubo_scope::RenderInterface, [472](#)
- clearPlanes
 - arcs::aubo_scope::RenderInterface, [472](#)
- clearPointCloud
 - arcs::aubo_scope::RenderInterface, [473](#)
- clearWaypoints
 - arcs::aubo_scope::RenderInterface, [473](#)
- closeView
 - arcs::aubo_scope::InstallationNodeContribution, [createAssemblyNode](#)
[314](#)
 - arcs::aubo_scope::ProgramNodeContribution, [436](#)
- CommentNode
 - arcs::aubo_scope::CommentNode, [139](#), [140](#)
- CompareOperator
 - arcs::aubo_scope::UntilNode, [581](#)
 - arcs::aubo_scope::WaitNode, [627](#)
- COMPLIANT_INSERTION
 - arcs::aubo_scope::AssemblyNode, [111](#)
- ConfigType
 - arcs::aubo_scope::AssemblyNode, [110](#)
 - arcs::aubo_scope::AssignmentNode, [114](#)
 - arcs::aubo_scope::ForceNode, [252](#)
 - arcs::aubo_scope::GripperNode, [289](#)
 - arcs::aubo_scope::LoopNode, [344](#)
 - arcs::aubo_scope::MoveNode, [357](#)
 - arcs::aubo_scope::SetNode, [518](#)
 - arcs::aubo_scope::SetPayloadNode, [526](#)
 - arcs::aubo_scope::UntilNode, [582](#)
 - arcs::aubo_scope::WaitNode, [627](#)
 - arcs::aubo_scope::WaypointNode, [647](#)
- CONFIGURABLE
 - arcs::aubo_scope::Io, [326](#)
- configureContribution
 - arcs::aubo_scope::InstallationNodeService, [316](#)
 - arcs::aubo_scope::ProgramNodeService, [448](#)
- configureGripper
 - arcs::aubo_scope::GripperContribution, [283](#)
- configureScrewdriver
 - arcs::aubo_scope::ScrewdriverContribution, [489](#)
- connectedTo
 - arcs::aubo_scope::TcpModel, [555](#)
 - arcs::aubo_scope::VariableModel, [615](#)
- CONTINUE_TRACKING
 - arcs::aubo_scope::ConveyorTrackingNode, [145](#)
- ContributionProviderPtr
 - arcs::aubo_scope, [81](#)
- ConveyorStartActionQueueType
 - arcs::aubo_scope::ConveyorTrackingNode, [144](#)
- ConveyorStartActionType
 - arcs::aubo_scope::ConveyorTrackingNode, [144](#)
- ConveyorTrackingNode
 - arcs::aubo_scope::ConveyorTrackingNode, [145](#), [146](#)
- ConveyorTrackingTargetType
 - arcs::aubo_scope::ConveyorTrackingNode, [145](#)
- Coordinate
 - arcs::aubo_scope::RenderInterface, [464](#)
- COUNTER
 - arcs::aubo_scope::LoopNode, [345](#)
- createAssemblyNode
 - arcs::aubo_scope::ProgramNodeFactory, [441](#)
- createAssignmentNode
 - arcs::aubo_scope::ProgramNodeFactory, [441](#)
- createBreakNode
 - arcs::aubo_scope::ProgramNodeFactory, [441](#)
- createCallNode
 - arcs::aubo_scope::ProgramNodeFactory, [441](#)
- createCaseNode
 - arcs::aubo_scope::ProgramNodeFactory, [441](#)
- createCircleMoveNode
 - arcs::aubo_scope::ProgramNodeFactory, [441](#)
- createCommentNode
 - arcs::aubo_scope::ProgramNodeFactory, [441](#)
- createConveyorTrackingNode
 - arcs::aubo_scope::ProgramNodeFactory, [441](#)
- createDefaultCaseNode
 - arcs::aubo_scope::ProgramNodeFactory, [441](#)
- createDirectionNode
 - arcs::aubo_scope::ProgramNodeFactory, [442](#)
- createElseIfNode
 - arcs::aubo_scope::ProgramNodeFactory, [442](#)
- createElseNode
 - arcs::aubo_scope::ProgramNodeFactory, [442](#)
- createExpression
 - arcs::aubo_scope::ValueFactory, [604](#)
- createFeatureVariable
 - arcs::aubo_scope::VariableFactory, [611](#)
- createFolderNode
 - arcs::aubo_scope::ProgramNodeFactory, [442](#)
- createForceNode
 - arcs::aubo_scope::ProgramNodeFactory, [442](#)
- createForceOptionNode
 - arcs::aubo_scope::ProgramNodeFactory, [442](#)
- createGlobalVariable
 - arcs::aubo_scope::VariableFactory, [612](#)
- createGripperNode
 - arcs::aubo_scope::ProgramNodeFactory, [442](#)
- createHaltNode
 - arcs::aubo_scope::ProgramNodeFactory, [443](#)
- createIfNode
 - arcs::aubo_scope::ProgramNodeFactory, [443](#)
- createInstallationNode
 - arcs::aubo_scope::InstallationNodeService, [316](#)
- createLoopNode
 - arcs::aubo_scope::ProgramNodeFactory, [443](#)
- createMoveNode
 - arcs::aubo_scope::ProgramNodeFactory, [443](#)

- createMoveNodeNoTemplate
 - arcs::aubo_scope::ProgramNodeFactory, [443](#)
- createNode
 - arcs::aubo_scope::ProgramNodeService, [448](#)
- createOfflineTrackNode
 - arcs::aubo_scope::ProgramNodeFactory, [444](#)
- createPalletNode
 - arcs::aubo_scope::ProgramNodeFactory, [444](#)
- createPopupNode
 - arcs::aubo_scope::ProgramNodeFactory, [444](#)
- createProgramHomeNode
 - arcs::aubo_scope::ProgramNodeFactory, [444](#)
- createRecordTrackNode
 - arcs::aubo_scope::ProgramNodeFactory, [444](#)
- createScrewdrivingNode
 - arcs::aubo_scope::ProgramNodeFactory, [444](#)
- createScriptNode
 - arcs::aubo_scope::ProgramNodeFactory, [444](#)
- createSeekNode
 - arcs::aubo_scope::ProgramNodeFactory, [444](#)
- createSetNode
 - arcs::aubo_scope::ProgramNodeFactory, [444](#)
- createSetPayloadNode
 - arcs::aubo_scope::ProgramNodeFactory, [445](#)
- createSubProgNode
 - arcs::aubo_scope::ProgramNodeFactory, [445](#)
- createSwitchNode
 - arcs::aubo_scope::ProgramNodeFactory, [445](#)
- createThreadNode
 - arcs::aubo_scope::ProgramNodeFactory, [445](#)
- createTimerNode
 - arcs::aubo_scope::ProgramNodeFactory, [445](#)
- createUntilNode
 - arcs::aubo_scope::ProgramNodeFactory, [445](#)
- createUserProgramNode
 - arcs::aubo_scope::ProgramNodeFactory, [445](#)
- createView
 - arcs::aubo_scope::InstallationNodeService, [316](#)
 - arcs::aubo_scope::ProgramNodeService, [449](#)
- createVisualLocalizationProgramNode
 - arcs::aubo_scope::ProgramNodeFactory, [446](#)
- createWaitNode
 - arcs::aubo_scope::ProgramNodeFactory, [446](#)
- createWaypointNode
 - arcs::aubo_scope::ProgramNodeFactory, [446](#)
- CUSTOM
 - arcs::aubo_scope::WaypointNode, [648](#)
- CUSTOM_BLEND
 - arcs::aubo_scope::UntilNode, [581](#)
- CUSTOM_DECELERATION
 - arcs::aubo_scope::UntilNode, [583](#)
- CUSTOM_PARAMETERS
 - arcs::aubo_scope::GripperNode, [290](#)
 - arcs::aubo_scope::SetPayloadNode, [527](#)
- CustomDecelerationParameterType
 - arcs::aubo_scope::UntilNode, [582](#)
- cutChildNode
 - arcs::aubo_scope::TreeNode, [567](#)
- d_
 - arcs::aubo_scope::AnalogIo, [103](#)
 - arcs::aubo_scope::ApplicationApi, [109](#)
 - arcs::aubo_scope::AssemblyNode, [113](#)
 - arcs::aubo_scope::AssignmentNode, [118](#)
 - arcs::aubo_scope::BaseFeature, [122](#)
 - arcs::aubo_scope::BreakNode, [128](#)
 - arcs::aubo_scope::CallNode, [131](#)
 - arcs::aubo_scope::CaseNode, [134](#)
 - arcs::aubo_scope::CircleMoveNode, [137](#)
 - arcs::aubo_scope::CommentNode, [141](#)
 - arcs::aubo_scope::ConveyorTrackingNode, [146](#)
 - arcs::aubo_scope::DataModel, [167](#)
 - arcs::aubo_scope::DefaultCaseNode, [170](#)
 - arcs::aubo_scope::DigitalIo, [180](#)
 - arcs::aubo_scope::DirectionNode, [188](#)
 - arcs::aubo_scope::ElseIfNode, [199](#)
 - arcs::aubo_scope::ElseNode, [202](#)
 - arcs::aubo_scope::EtherNetIP, [208](#)
 - arcs::aubo_scope::Expression, [213](#)
 - arcs::aubo_scope::Feature, [217](#)
 - arcs::aubo_scope::FeatureContributionModel, [222](#)
 - arcs::aubo_scope::FeatureLine, [227](#)
 - arcs::aubo_scope::FeatureModel, [231](#)
 - arcs::aubo_scope::FeaturePlane, [234](#)
 - arcs::aubo_scope::FeaturePoint, [238](#)
 - arcs::aubo_scope::FeatureTool, [241](#)
 - arcs::aubo_scope::FeatureVariable, [245](#)
 - arcs::aubo_scope::FolderNode, [250](#)
 - arcs::aubo_scope::ForceNode, [254](#)
 - arcs::aubo_scope::ForceOptionNode, [257](#)
 - arcs::aubo_scope::Function, [261](#)
 - arcs::aubo_scope::FunctionModel, [265](#)
 - arcs::aubo_scope::GlobalVariable, [269](#)
 - arcs::aubo_scope::GripperNode, [292](#)
 - arcs::aubo_scope::HaltNode, [297](#)
 - arcs::aubo_scope::IfNode, [301](#)
 - arcs::aubo_scope::InstallationApi, [307](#)
 - arcs::aubo_scope::InstallationApiProvider, [310](#)
 - arcs::aubo_scope::InstallationCreationContext, [314](#)
 - arcs::aubo_scope::Io, [330](#)
 - arcs::aubo_scope::IoModel, [336](#)
 - arcs::aubo_scope::KeyboardManager, [342](#)
 - arcs::aubo_scope::LoopNode, [348](#)
 - arcs::aubo_scope::ModbusIo, [354](#)
 - arcs::aubo_scope::MoveNode, [363](#)
 - arcs::aubo_scope::OfflineTrackNode, [366](#)
 - arcs::aubo_scope::PalletNode, [370](#)
 - arcs::aubo_scope::Payload, [376](#)
 - arcs::aubo_scope::PayloadContributionModel, [383](#)
 - arcs::aubo_scope::PayloadModel, [387](#)
 - arcs::aubo_scope::PersistedVariable, [391](#)

- arcs::aubo_scope::PopupManager, 395
- arcs::aubo_scope::PopupNode, 400
- arcs::aubo_scope::Profinet, 406
- arcs::aubo_scope::ProgramApi, 411
- arcs::aubo_scope::ProgramApiProvider, 414
- arcs::aubo_scope::ProgramContributionConfiguration, 420
- arcs::aubo_scope::ProgramCreationContext, 424
- arcs::aubo_scope::ProgramHomeNode, 427
- arcs::aubo_scope::ProgramModel, 431
- arcs::aubo_scope::ProgramNode, 435
- arcs::aubo_scope::ProgramNodeFactory, 447
- arcs::aubo_scope::RecordTrackNode, 455
- arcs::aubo_scope::RenderInterface, 479
- arcs::aubo_scope::ScrewdrivingNode, 497
- arcs::aubo_scope::ScriptNode, 501
- arcs::aubo_scope::ScriptWriter, 513
- arcs::aubo_scope::SeekNode, 516
- arcs::aubo_scope::SetNode, 524
- arcs::aubo_scope::SetPayloadNode, 529
- arcs::aubo_scope::SubProgNode, 537
- arcs::aubo_scope::SwitchNode, 541
- arcs::aubo_scope::SystemApi, 544
- arcs::aubo_scope::TCP, 548
- arcs::aubo_scope::TcpContributionModel, 553
- arcs::aubo_scope::TcpModel, 557
- arcs::aubo_scope::ThreadNode, 561
- arcs::aubo_scope::TimerNode, 564
- arcs::aubo_scope::TreeNode, 573
- arcs::aubo_scope::UndoRedoManager, 578
- arcs::aubo_scope::UntilNode, 591
- arcs::aubo_scope::UserInteraction, 599
- arcs::aubo_scope::UserInterfaceApi, 601
- arcs::aubo_scope::ValueFactory, 604
- arcs::aubo_scope::Variable, 609
- arcs::aubo_scope::VariableFactory, 613
- arcs::aubo_scope::VariableModel, 618
- arcs::aubo_scope::ViewApiProvider, 621
- arcs::aubo_scope::VisualLocalizationProgramNode, 624
- arcs::aubo_scope::WaitNode, 635
- arcs::aubo_scope::Waypoint, 639
- arcs::aubo_scope::WaypointModel, 644
- arcs::aubo_scope::WaypointNode, 657
- DataModel
 - arcs::aubo_scope::DataModel, 149, 150
- DataSwitch
 - arcs::aubo_scope::AnalogIo, 103
 - arcs::aubo_scope::ApplicationApi, 109
 - arcs::aubo_scope::AssemblyNode, 112
 - arcs::aubo_scope::AssignmentNode, 117
 - arcs::aubo_scope::BaseFeature, 122
 - arcs::aubo_scope::BreakNode, 128
 - arcs::aubo_scope::CallNode, 131
 - arcs::aubo_scope::CaseNode, 134
 - arcs::aubo_scope::CircleMoveNode, 137
 - arcs::aubo_scope::CommentNode, 141
 - arcs::aubo_scope::ConveyorTrackingNode, 146
 - arcs::aubo_scope::DataModel, 166
 - arcs::aubo_scope::DefaultCaseNode, 170
 - arcs::aubo_scope::DigitalIo, 179
 - arcs::aubo_scope::DirectionNode, 188
 - arcs::aubo_scope::ElseIfNode, 199
 - arcs::aubo_scope::ElseNode, 202
 - arcs::aubo_scope::EtherNetIP, 207
 - arcs::aubo_scope::Expression, 213
 - arcs::aubo_scope::Feature, 217
 - arcs::aubo_scope::FeatureContributionModel, 222
 - arcs::aubo_scope::FeatureLine, 227
 - arcs::aubo_scope::FeatureModel, 231
 - arcs::aubo_scope::FeaturePlane, 234
 - arcs::aubo_scope::FeaturePoint, 238
 - arcs::aubo_scope::FeatureTool, 241
 - arcs::aubo_scope::FeatureVariable, 245
 - arcs::aubo_scope::FolderNode, 250
 - arcs::aubo_scope::ForceNode, 254
 - arcs::aubo_scope::ForceOptionNode, 257
 - arcs::aubo_scope::Function, 260
 - arcs::aubo_scope::FunctionModel, 265
 - arcs::aubo_scope::GlobalVariable, 269
 - arcs::aubo_scope::GripperNode, 292
 - arcs::aubo_scope::HaltNode, 297
 - arcs::aubo_scope::IfNode, 301
 - arcs::aubo_scope::InstallationApi, 307
 - arcs::aubo_scope::InstallationApiProvider, 310
 - arcs::aubo_scope::InstallationCreationContext, 313
 - arcs::aubo_scope::Io, 330
 - arcs::aubo_scope::IoModel, 336
 - arcs::aubo_scope::KeyboardManager, 342
 - arcs::aubo_scope::LoopNode, 348
 - arcs::aubo_scope::ModbusIo, 353
 - arcs::aubo_scope::MoveNode, 363
 - arcs::aubo_scope::OfflineTrackNode, 366
 - arcs::aubo_scope::PalletNode, 370
 - arcs::aubo_scope::Payload, 375
 - arcs::aubo_scope::PayloadContributionModel, 383
 - arcs::aubo_scope::PayloadModel, 387
 - arcs::aubo_scope::PersistedVariable, 391
 - arcs::aubo_scope::PopupNode, 400
 - arcs::aubo_scope::Profinet, 406
 - arcs::aubo_scope::ProgramApi, 411
 - arcs::aubo_scope::ProgramApiProvider, 414
 - arcs::aubo_scope::ProgramContributionConfiguration, 420
 - arcs::aubo_scope::ProgramCreationContext, 424
 - arcs::aubo_scope::ProgramHomeNode, 427
 - arcs::aubo_scope::ProgramModel, 430
 - arcs::aubo_scope::ProgramNode, 434
 - arcs::aubo_scope::ProgramNodeFactory, 447

- arcs::aubo_scope::RecordTrackNode, [455](#)
- arcs::aubo_scope::RenderInterface, [479](#)
- arcs::aubo_scope::ScrewdrivingNode, [497](#)
- arcs::aubo_scope::ScriptNode, [500](#)
- arcs::aubo_scope::ScriptWriter, [512](#)
- arcs::aubo_scope::SeekNode, [516](#)
- arcs::aubo_scope::SetNode, [524](#)
- arcs::aubo_scope::SetPayloadNode, [529](#)
- arcs::aubo_scope::SubProgNode, [537](#)
- arcs::aubo_scope::SwitchNode, [540](#)
- arcs::aubo_scope::SystemApi, [544](#)
- arcs::aubo_scope::TCP, [548](#)
- arcs::aubo_scope::TcpContributionModel, [552](#)
- arcs::aubo_scope::TcpModel, [557](#)
- arcs::aubo_scope::ThreadNode, [560](#)
- arcs::aubo_scope::TimerNode, [564](#)
- arcs::aubo_scope::TreeNode, [572](#)
- arcs::aubo_scope::UndoRedoManager, [578](#)
- arcs::aubo_scope::UntilNode, [591](#)
- arcs::aubo_scope::UserInteraction, [599](#)
- arcs::aubo_scope::UserInterfaceApi, [601](#)
- arcs::aubo_scope::ValueFactory, [604](#)
- arcs::aubo_scope::Variable, [609](#)
- arcs::aubo_scope::VariableFactory, [613](#)
- arcs::aubo_scope::VariableModel, [618](#)
- arcs::aubo_scope::ViewApiProvider, [621](#)
- arcs::aubo_scope::VisualLocalizationProgramNode, [624](#)
- arcs::aubo_scope::WaitNode, [635](#)
- arcs::aubo_scope::Waypoint, [638](#)
- arcs::aubo_scope::WaypointModel, [644](#)
- arcs::aubo_scope::WaypointNode, [656](#)
- DecelerationParameterType
 - arcs::aubo_scope::UntilNode, [583](#)
- decreaseIndent
 - arcs::aubo_scope::ScriptWriter, [506](#)
- DefaultCaseNode
 - arcs::aubo_scope::DefaultCaseNode, [169](#), [170](#)
- DefaultSingletonTraits< PopupManager >
 - arcs::aubo_scope::PopupManager, [395](#)
- DEFINED
 - arcs::aubo_scope::WaypointNode, [648](#), [649](#)
- defineFunction
 - arcs::aubo_scope::ScriptWriter, [506](#)
- defineThread
 - arcs::aubo_scope::ScriptWriter, [507](#)
- DIGITAL
 - arcs::aubo_scope::Io, [326](#)
- DIGITAL_INPUT
 - arcs::aubo_scope::UntilNode, [583](#)
 - arcs::aubo_scope::WaitNode, [627](#)
- DIGITAL_OUTPUT
 - arcs::aubo_scope::SetNode, [519](#)
- DigitalIo
 - arcs::aubo_scope::DigitalIo, [177](#), [178](#)
- DigitalIoFilter
 - arcs::aubo_scope::DigitalIoFilter, [180](#)
- DirectionNode
 - arcs::aubo_scope::DirectionNode, [185](#), [186](#)
- DirectionSelectionType
 - arcs::aubo_scope::DirectionNode, [183](#)
- disconnectFrom
 - arcs::aubo_scope::TcpModel, [556](#)
 - arcs::aubo_scope::VariableModel, [616](#)
- DISTANCE
 - arcs::aubo_scope::UntilNode, [582](#)
- doc Directory Reference, [59](#)
- doc/SDK_overview_en.md, [659](#)
- DOUBLE
 - arcs::aubo_scope::Io, [326](#)
- DoubleRangeValidator
 - arcs::aubo_scope::DoubleRangeValidator, [190](#)
- elseCondition
 - arcs::aubo_scope::ScriptWriter, [507](#)
- elseifCondition
 - arcs::aubo_scope::ScriptWriter, [507](#)
- ElseIfNode
 - arcs::aubo_scope::ElseIfNode, [197](#), [198](#)
- ElseNode
 - arcs::aubo_scope::ElseNode, [201](#), [202](#)
- end
 - arcs::aubo_scope::ScriptWriter, [507](#)
- equals
 - arcs::aubo_scope::Variable, [608](#)
- ERROR
 - arcs::aubo_scope::PopupNode, [397](#)
- ETHERNET_IP
 - arcs::aubo_scope::Io, [326](#)
- EtherNetIP
 - arcs::aubo_scope::EtherNetIP, [205](#), [206](#)
- EtherNetIPFilter
 - arcs::aubo_scope::EtherNetIPFilter, [208](#)
- EXPRESSION
 - arcs::aubo_scope::AssignmentNode, [115](#)
 - arcs::aubo_scope::DirectionNode, [184](#)
 - arcs::aubo_scope::LoopNode, [345](#)
 - arcs::aubo_scope::UntilNode, [582](#)
- Expression
 - arcs::aubo_scope::Expression, [210](#), [211](#)
- EXPRESSION_INPUT
 - arcs::aubo_scope::WaitNode, [628](#)
- EXPRESSION_OUTPUT
 - arcs::aubo_scope::SetNode, [519](#)
- FEATURE
 - arcs::aubo_scope::DirectionNode, [184](#)
 - arcs::aubo_scope::MoveNode, [358](#)
 - arcs::aubo_scope::Variable, [606](#)
- Feature
 - arcs::aubo_scope::Feature, [215](#), [216](#)
- FeatureContributionModel
 - arcs::aubo_scope::FeatureContributionModel, [219](#), [220](#)
- FeatureLine
 - arcs::aubo_scope::FeatureLine, [225](#), [226](#)
- FeatureModel

- arcs::aubo_scope::FeatureModel, 228, 229
- FeaturePlane
 - arcs::aubo_scope::FeaturePlane, 233, 234
- FeaturePoint
 - arcs::aubo_scope::FeaturePoint, 237, 238
- FeatureSelectionType
 - arcs::aubo_scope::DirectionNode, 184
 - arcs::aubo_scope::MoveNode, 357
- FeatureTool
 - arcs::aubo_scope::FeatureTool, 240, 241
- FeatureVariable
 - arcs::aubo_scope::FeatureVariable, 244, 245
- FIXED_POSITION
 - arcs::aubo_scope::WaypointNode, 647
- FLOAT_REGISTER_INPUT
 - arcs::aubo_scope::UntilNode, 583
 - arcs::aubo_scope::WaitNode, 628
- FLOAT_REGISTER_OUTPUT
 - arcs::aubo_scope::SetNode, 519
- FolderNode
 - arcs::aubo_scope::FolderNode, 248, 249
- ForceNode
 - arcs::aubo_scope::ForceNode, 253, 254
- ForceOptionNode
 - arcs::aubo_scope::ForceOptionNode, 256, 257
- forCondition
 - arcs::aubo_scope::ScriptWriter, 507
- FRAME
 - arcs::aubo_scope::ForceNode, 252
- Function
 - arcs::aubo_scope::Function, 258, 259
- function_lock
 - arcs::aubo_scope::IoConfigure, 331
- FunctionModel
 - arcs::aubo_scope::FunctionModel, 262, 263
- functions
 - arcs::aubo_scope::IoConfigure, 331
- GENERAL_PURPOSE
 - arcs::aubo_scope::Io, 326
- generateGripActionScript
 - arcs::aubo_scope::GripperContribution, 284
- generateReleaseActionScript
 - arcs::aubo_scope::GripperContribution, 284
- generateScript
 - arcs::aubo_scope::InstallationNodeContribution, 314
 - arcs::aubo_scope::ProgramNodeContribution, 436
 - arcs::aubo_scope::ScriptWriter, 508
- generateStartScrewdriverScript
 - arcs::aubo_scope::ScrewdriverContribution, 490
- generateStopScrewdriverScript
 - arcs::aubo_scope::ScrewdriverContribution, 490
- get
 - arcs::aubo_scope::ContributionProvider< T >, 142
- arcs::aubo_scope::DataModel, 150–157
- arcs::aubo_scope::VariableModel, 616
- getAction
 - arcs::aubo_scope::Io, 328
 - arcs::aubo_scope::ProgramNodeContribution, 436
- getAll
 - arcs::aubo_scope::FeatureModel, 229
 - arcs::aubo_scope::IoModel, 335
 - arcs::aubo_scope::TcpModel, 556
 - arcs::aubo_scope::VariableModel, 617
- getAnalogInput
 - arcs::aubo_scope::WaitNode, 629
- getAnalogInputValueToProceed
 - arcs::aubo_scope::UntilNode, 585
- getAnalogOutput
 - arcs::aubo_scope::SetNode, 521
- getAngularDeceleration
 - arcs::aubo_scope::UntilNode, 585
- getArguments
 - arcs::aubo_scope::Function, 260
- getAxis
 - arcs::aubo_scope::DirectionNode, 186
- getBaseFeature
 - arcs::aubo_scope::FeatureModel, 229
- getBitOffset
 - arcs::aubo_scope::EtherNetIP, 206
 - arcs::aubo_scope::Profinet, 405
- getBlend
 - arcs::aubo_scope::MoveNode, 359
 - arcs::aubo_scope::WaypointNode, 650
- getBlendParameterType
 - arcs::aubo_scope::UntilNode, 585
- getBlendType
 - arcs::aubo_scope::WaypointNode, 650
- getBoolRegister
 - arcs::aubo_scope::WaitNode, 629
- getBoolValueToSet
 - arcs::aubo_scope::SetNode, 521
- getBoolValueToWaitFor
 - arcs::aubo_scope::WaitNode, 629
- getBugfixVersion
 - SystemApi, 45
- getBuildNumber
 - SystemApi, 45
- getByName
 - arcs::aubo_scope::FeatureModel, 229
 - arcs::aubo_scope::WaypointModel, 642
- getByteOffset
 - arcs::aubo_scope::EtherNetIP, 206
 - arcs::aubo_scope::Profinet, 405
- getCartesianDeceleration
 - arcs::aubo_scope::UntilNode, 585
- getCenterOfGravity
 - arcs::aubo_scope::Payload, 373
 - arcs::aubo_scope::SetPayloadNode, 528
- getChildren
 - arcs::aubo_scope::TreeNode, 568

- getChildrenNotSuppressed
 - arcs::aubo_scope::TreeNode, 568
- getComment
 - arcs::aubo_scope::CommentNode, 140
- getCompareOperator
 - arcs::aubo_scope::UntilNode, 585
 - arcs::aubo_scope::WaitNode, 629
- getConfigType
 - arcs::aubo_scope::AssignmentNode, 116
 - arcs::aubo_scope::GripperNode, 291
 - arcs::aubo_scope::LoopNode, 346
 - arcs::aubo_scope::MoveNode, 359
 - arcs::aubo_scope::SetNode, 521
 - arcs::aubo_scope::SetPayloadNode, 528
 - arcs::aubo_scope::UntilNode, 585
 - arcs::aubo_scope::WaitNode, 630
 - arcs::aubo_scope::WaypointNode, 651
- getConfigurableDigitalInputConfigure
 - SystemApi, 45
- getConfigurableDigitalOutputConfigure
 - SystemApi, 45
- getContributorInfo
 - arcs::aubo_scope::Device, 172
 - arcs::aubo_scope::Function, 260
- getCreatedVersion
 - arcs::aubo_scope::InstallationCreationContext, 313
 - arcs::aubo_scope::ProgramCreationContext, 423
- getCurrent
 - arcs::aubo_scope::SetNode, 521
- getCurrentPayload
 - arcs::aubo_scope::PayloadModel, 386
- getCustomDecelerationParameterType
 - arcs::aubo_scope::UntilNode, 586
- getDecelerationParameters
 - arcs::aubo_scope::UntilNode, 586
- getDecelerationParameterType
 - arcs::aubo_scope::UntilNode, 586
- getDefaultName
 - arcs::aubo_scope::Io, 328
- getDefaultPayload
 - arcs::aubo_scope::PayloadModel, 386
- getDefaultTcp
 - arcs::aubo_scope::TcpModel, 556
- getDeviceManager
 - ApplicationApi, 38
- getDigitalInput
 - arcs::aubo_scope::WaitNode, 630
- getDigitalInputValueToProceed
 - arcs::aubo_scope::UntilNode, 586
- getDirectionSelectionType
 - arcs::aubo_scope::DirectionNode, 186
- getDisplayName
 - arcs::aubo_scope::Device, 172
 - arcs::aubo_scope::Feature, 216
 - arcs::aubo_scope::Io, 328
 - arcs::aubo_scope::Payload, 373
- arcs::aubo_scope::ScrewdriverProgram, 493
- arcs::aubo_scope::TCP, 547
- arcs::aubo_scope::Variable, 608
- arcs::aubo_scope::Waypoint, 637
- getDistance
 - arcs::aubo_scope::UntilNode, 586
- getDoubleRegisterInput
 - arcs::aubo_scope::WaitNode, 630
- getENCConfigure
 - SystemApi, 45
- getExpression
 - arcs::aubo_scope::AssignmentNode, 116
 - arcs::aubo_scope::DirectionNode, 186
 - arcs::aubo_scope::ElseIfNode, 198
 - arcs::aubo_scope::IfNode, 300
 - arcs::aubo_scope::LoopNode, 346
 - arcs::aubo_scope::SetNode, 522
- getExpressionConfig
 - arcs::aubo_scope::UntilNode, 586
- getExpressionInputConfig
 - arcs::aubo_scope::WaitNode, 630
- getFeature
 - arcs::aubo_scope::DirectionNode, 186
 - arcs::aubo_scope::FeatureContributionModel, 220
 - arcs::aubo_scope::MoveNode, 360
- getFeatureContributionModel
 - InstallationApi, 41
- getFeatureModel
 - ApplicationApi, 38
- getFeatures
 - arcs::aubo_scope::FeatureModel, 230
- getFeatureSelectionType
 - arcs::aubo_scope::DirectionNode, 186
- getFeatureVariable
 - arcs::aubo_scope::VariableModel, 617
- getFloatValueToSet
 - arcs::aubo_scope::SetNode, 522
- getFloatValueToWaitFor
 - arcs::aubo_scope::WaitNode, 630
- getForce
 - arcs::aubo_scope::GripActionParameters, 270
- getFunction
 - arcs::aubo_scope::FunctionModel, 264
- getFunctionModel
 - InstallationApi, 41
- getFunctions
 - arcs::aubo_scope::FunctionModel, 264
- getGeomFeture
 - arcs::aubo_scope::FeatureModel, 230
- getGripForceCapability
 - arcs::aubo_scope::RegisteredCapabilities, 458
- getGripperDevice
 - arcs::aubo_scope::GripperNode, 291
- getGrippers
 - arcs::aubo_scope::DeviceManager, 173
- getGripVacuumCapability
 - arcs::aubo_scope::RegisteredCapabilities, 459

- getHomeDefaultPosition
 - SystemApi, [45](#)
- getHomePackagePosition
 - SystemApi, [46](#)
- getIcon
 - arcs::aubo_scope::InstallationNodeService, [317](#)
 - arcs::aubo_scope::ProgramNodeService, [449](#)
- getId
 - arcs::aubo_scope::Device, [172](#)
 - arcs::aubo_scope::ProgramNodeService, [449](#)
 - arcs::aubo_scope::ScrewdriverProgram, [493](#)
- getInertiaMatrix
 - arcs::aubo_scope::Payload, [374](#)
- getInstallationApi
 - arcs::aubo_scope::InstallationApiProvider, [309](#)
- getInstallationNode
 - arcs::aubo_scope::ApplicationApi, [107](#), [108](#)
- getInterfaceType
 - arcs::aubo_scope::Io, [328](#)
- getIo
 - arcs::aubo_scope::IoModel, [335](#)
- getIOInputSelectionType
 - arcs::aubo_scope::UntilNode, [586](#)
- getIOModel
 - arcs::aubo_scope::ScrewdriverApiProvider, [480](#)
- getIoModel
 - ApplicationApi, [39](#)
 - arcs::aubo_scope::GripperApiProvider, [275](#)
- getIos
 - arcs::aubo_scope::IoModel, [335](#)
- getIpAddress
 - arcs::aubo_scope::ModbusIo, [352](#)
- getJointAcceleration
 - arcs::aubo_scope::MoveNode, [360](#)
 - arcs::aubo_scope::WaypointNode, [651](#)
- getJointDefaultAcceleration
 - SystemApi, [46](#)
- getJointDefaultVelocity
 - SystemApi, [46](#)
- getJointMaxAcceleration
 - SystemApi, [46](#)
- getJointMaxPositions
 - SystemApi, [46](#)
- getJointMaxVelocity
 - SystemApi, [47](#)
- getJointMinPositions
 - SystemApi, [47](#)
- getJointPositions
 - arcs::aubo_scope::Waypoint, [637](#)
 - arcs::aubo_scope::WaypointNode, [651](#)
- getJointSpeed
 - arcs::aubo_scope::MoveNode, [360](#)
 - arcs::aubo_scope::WaypointNode, [651](#)
- getKeyboardManager
 - arcs::aubo_scope::UserInteraction, [593](#)
- getKeys
 - arcs::aubo_scope::DataModel, [157](#)
- getLastSavedVersion
 - arcs::aubo_scope::InstallationCreationContext, [313](#)
 - arcs::aubo_scope::ProgramCreationContext, [423](#)
- getLocale
 - SystemApi, [47](#)
- getLocaleForProgrammingLanguage
 - SystemApi, [47](#)
- getLoopCount
 - arcs::aubo_scope::LoopNode, [346](#)
- getLoopCountVariable
 - arcs::aubo_scope::LoopNode, [346](#)
- getMajorVersion
 - SystemApi, [47](#)
- getMass
 - arcs::aubo_scope::Payload, [374](#)
 - arcs::aubo_scope::SetPayloadNode, [528](#)
- getMaxPayloadDistance
 - SystemApi, [48](#)
- getMaxPayloadMass
 - SystemApi, [48](#)
- getMaxRangeValue
 - arcs::aubo_scope::AnalogIo, [102](#)
- getMeshes
 - arcs::aubo_scope::RenderInterface, [473](#)
- getMessage
 - arcs::aubo_scope::DoubleRangeValidator, [190](#)
 - arcs::aubo_scope::InputValidator, [303](#)
 - arcs::aubo_scope::IntegerRangeValidator, [320](#)
 - arcs::aubo_scope::PopupNode, [399](#)
 - arcs::aubo_scope::StringLengthValidator, [533](#)
 - arcs::aubo_scope::UIntegerRangeValidator, [574](#)
- getMinorVersion
 - SystemApi, [48](#)
- getMinRangeValue
 - arcs::aubo_scope::AnalogIo, [102](#)
- getModbusInput
 - arcs::aubo_scope::WaitNode, [630](#)
- getMotionType
 - arcs::aubo_scope::WaypointNode, [651](#)
- getName
 - arcs::aubo_scope::AuboCapInfo, [119](#)
 - arcs::aubo_scope::Function, [260](#)
- getNodeCreationType
 - arcs::aubo_scope::InstallationCreationContext, [313](#)
 - arcs::aubo_scope::ProgramCreationContext, [423](#)
- getOffset
 - arcs::aubo_scope::TCP, [547](#)
- getOutput
 - arcs::aubo_scope::SetNode, [522](#)
- getOutputSelectionType
 - arcs::aubo_scope::SetNode, [522](#)

- getParametersWarning
 - arcs::aubo_scope::Payload, [374](#)
- getParent
 - arcs::aubo_scope::TreeNode, [568](#)
- getPayload
 - arcs::aubo_scope::PayloadContributionModel, [381](#)
 - arcs::aubo_scope::PayloadModel, [386](#)
- getPayloadContributionModel
 - InstallationApi, [41](#)
- getPayloadModel
 - ApplicationApi, [39](#)
- getPayloads
 - arcs::aubo_scope::PayloadModel, [387](#)
- getPopupManager
 - arcs::aubo_scope::UserInteraction, [593](#)
- getPose
 - arcs::aubo_scope::Feature, [216](#)
 - arcs::aubo_scope::Waypoint, [637](#)
 - arcs::aubo_scope::WaypointNode, [652](#)
- getPositionDefinition
 - arcs::aubo_scope::WaypointNode, [652](#)
- getProgramApi
 - arcs::aubo_scope::ProgramApiProvider, [413](#)
- getProgramModel
 - ProgramApi, [42](#)
- getProgramNode
 - arcs::aubo_scope::TreeNode, [568](#)
- getProgramNodeContribution
 - arcs::aubo_scope::ProgramNode, [434](#)
- getProgramNodeFactory
 - arcs::aubo_scope::ProgramModel, [430](#)
- getPulseTime
 - arcs::aubo_scope::SetNode, [522](#)
- getRenderInterface
 - arcs::aubo_scope::UserInteraction, [593](#)
- getResolvedVariable
 - arcs::aubo_scope::ScriptWriter, [508](#)
- getRobotDhParams
 - SystemApi, [48](#)
- getRobotProxy
 - arcs::aubo_scope::ApplicationApi, [108](#)
- getRobotThumb
 - arcs::aubo_scope::UserInteraction, [594](#)
- getScrewdriverOperationType
 - arcs::aubo_scope::ProgramSelectionParameters, [452](#)
 - arcs::aubo_scope::ScrewdriverParameters, [492](#)
- getScrewdriverProgram
 - arcs::aubo_scope::ProgramSelectionParameters, [452](#)
 - arcs::aubo_scope::ScrewdriverParameters, [492](#)
- getScrewDrivers
 - arcs::aubo_scope::DeviceManager, [173](#)
- getSerialNumber
 - SystemApi, [49](#)
- getSettingType
 - arcs::aubo_scope::GripperNode, [291](#)
- getSignalAddress
 - arcs::aubo_scope::ModbusIo, [352](#)
- getSpeed
 - arcs::aubo_scope::GripActionParameters, [271](#)
 - arcs::aubo_scope::ReleaseActionParameters, [460](#)
- getSpeedCapability
 - arcs::aubo_scope::RegisteredCapabilities, [459](#)
- getStandardAnalogInputConfigure
 - SystemApi, [49](#)
- getStandardAnalogOutputConfigure
 - SystemApi, [49](#)
- getStandardDigitalInputConfigure
 - SystemApi, [49](#)
- getStandardDigitalOutputConfigure
 - SystemApi, [49](#)
- getStaticLinkInputConfigure
 - SystemApi, [50](#)
- getStaticLinkOutputConfigure
 - SystemApi, [50](#)
- getSymbolicName
 - arcs::aubo_scope::AuboCapInfo, [119](#)
- getSystemAPI
 - arcs::aubo_scope::ScrewdriverApiProvider, [480](#)
- getSystemApi
 - arcs::aubo_scope::GripperApiProvider, [275](#)
 - arcs::aubo_scope::InstallationApiProvider, [309](#)
 - arcs::aubo_scope::ProgramApiProvider, [413](#)
 - arcs::aubo_scope::ViewApiProvider, [620](#)
- getTcp
 - arcs::aubo_scope::TcpContributionModel, [551](#)
 - arcs::aubo_scope::TcpModel, [556](#)
- getTcpContributionModel
 - InstallationApi, [41](#)
- getTcpDefaultAcceleration
 - SystemApi, [50](#)
- getTcpDefaultVelocity
 - SystemApi, [50](#)
- getTcpMaxAcceleration
 - SystemApi, [50](#)
- getTcpMaxDistance
 - SystemApi, [51](#)
- getTcpMaxVelocity
 - SystemApi, [51](#)
- getTcpModel
 - ApplicationApi, [39](#)
- getTcpOffset
 - arcs::aubo_scope::Waypoint, [638](#)
 - arcs::aubo_scope::WaypointNode, [652](#)
- getTCPSelection
 - arcs::aubo_scope::MoveNode, [360](#)
- getTcpSelectionType
 - arcs::aubo_scope::MoveNode, [360](#)
- getTime
 - arcs::aubo_scope::WaypointNode, [652](#)
- getTitle

- arcs::aubo_scope::GripperContribution, 284
- arcs::aubo_scope::InstallationNodeService, 317
- arcs::aubo_scope::ProgramNodeContribution, 436
- arcs::aubo_scope::ProgramNodeService, 450
- arcs::aubo_scope::ScrewdriverContribution, 490
- getToolAcceleration
 - arcs::aubo_scope::DirectionNode, 186
 - arcs::aubo_scope::MoveNode, 360
 - arcs::aubo_scope::WaypointNode, 652
- getToolAnalogInputConfigure
 - SystemApi, 51
- getToolAnalogOutputConfigure
 - SystemApi, 51
- getToolDigitalIOConfigure
 - SystemApi, 51
- getToolFeature
 - arcs::aubo_scope::FeatureModel, 230
- getToolSpeed
 - arcs::aubo_scope::DirectionNode, 187
 - arcs::aubo_scope::MoveNode, 360
 - arcs::aubo_scope::WaypointNode, 652
- getTreeNode
 - arcs::aubo_scope::ProgramModel, 430
- getType
 - arcs::aubo_scope::DirectionNode, 187
 - arcs::aubo_scope::Io, 328
 - arcs::aubo_scope::PopupNode, 399
 - arcs::aubo_scope::Variable, 608
- getUndoRedoManager
 - ProgramApi, 42
- getUnitType
 - SystemApi, 52
- getUseJoints
 - arcs::aubo_scope::MoveNode, 360
- getUserDefinedRobotPosition
 - arcs::aubo_scope::UserInteraction, 594, 595
- getUserInteraction
 - UserInterfaceApi, 53
- getUserInterfaceApi
 - arcs::aubo_scope::InstallationApiProvider, 309
 - arcs::aubo_scope::ProgramApiProvider, 413
 - arcs::aubo_scope::ViewApiProvider, 620
- getVacuum
 - arcs::aubo_scope::GripActionParameters, 271
- getValue
 - arcs::aubo_scope::AnalogIo, 102
 - arcs::aubo_scope::DigitalIo, 179
 - arcs::aubo_scope::EtherNetIP, 206
 - arcs::aubo_scope::ModbusIo, 352
 - arcs::aubo_scope::Profinet, 405
- getValueFactory
 - ApplicationApi, 39
- getValueStr
 - arcs::aubo_scope::Io, 329
- getVariable
 - arcs::aubo_scope::AssignmentNode, 117
 - arcs::aubo_scope::WaypointNode, 653
- getVariableDefinition
 - arcs::aubo_scope::WaypointNode, 653
- getVariableFactory
 - arcs::aubo_scope::VariableModel, 617
- getVariableModel
 - ApplicationApi, 39
- getVendor
 - arcs::aubo_scope::AuboCapInfo, 119
- getVoltage
 - arcs::aubo_scope::SetNode, 522
- getWaitTime
 - arcs::aubo_scope::WaitNode, 630
- getWaypointBlendParameter
 - arcs::aubo_scope::UntilNode, 587
- getWaypointModel
 - ProgramApi, 42
- getWaypoints
 - arcs::aubo_scope::WaypointModel, 643
- getWidth
 - arcs::aubo_scope::GripActionParameters, 271
 - arcs::aubo_scope::ReleaseActionParameters, 461
- getWidthCapability
 - arcs::aubo_scope::RegisteredCapabilities, 459
- GLOBAL
 - arcs::aubo_scope::Variable, 606
- GlobalVariable
 - arcs::aubo_scope::GlobalVariable, 268, 269
- GREATER_THAN
 - arcs::aubo_scope::UntilNode, 582
 - arcs::aubo_scope::WaitNode, 627
- GRIP_ACTION
 - arcs::aubo_scope::GripperNode, 289
- GripperNode
 - arcs::aubo_scope::GripperNode, 290, 291
- Ground
 - arcs::aubo_scope::RenderInterface, 464
- HaltNode
 - arcs::aubo_scope::HaltNode, 296, 297
- hasCoordinate
 - arcs::aubo_scope::RenderInterface, 473
- hashCode
 - arcs::aubo_scope::Feature, 216
 - arcs::aubo_scope::Io, 329
 - arcs::aubo_scope::Payload, 374
 - arcs::aubo_scope::TCP, 547
 - arcs::aubo_scope::Variable, 608
 - arcs::aubo_scope::Waypoint, 638
- hasPointCloud
 - arcs::aubo_scope::RenderInterface, 474
- hasWaypoint
 - arcs::aubo_scope::RenderInterface, 474
- HOLE_SEARCH
 - arcs::aubo_scope::AssemblyNode, 111

- ifCondition
 - arcs::aubo_scope::ScriptWriter, 508
- IfNode
 - arcs::aubo_scope::IfNode, 299, 300
- ifNotCondition
 - arcs::aubo_scope::ScriptWriter, 508
- IGNORE_ACTIVE_TCP
 - arcs::aubo_scope::MoveNode, 358
- IMPERIAL
 - arcs::aubo_scope, 82
- include Directory Reference, 62
- include/aubo_caps Directory Reference, 55
- include/aubo_caps/class_forward.h, 659, 661
- include/aubo_caps/contribution Directory Reference, 57
- include/aubo_caps/contribution/contribution_provider.h, 662, 663
- include/aubo_caps/contribution/driver Directory Reference, 60
- include/aubo_caps/contribution/driver/gripper Directory Reference, 62
- include/aubo_caps/contribution/driver/gripper/capability Directory Reference, 57
- include/aubo_caps/contribution/driver/gripper/capability/force_capability.h, 664, 665
- include/aubo_caps/contribution/driver/gripper/capability/pose_capability.h, 667, 668
- include/aubo_caps/contribution/driver/gripper/capability/register_capabilities.h, 669, 670
- include/aubo_caps/contribution/driver/gripper/capability/robot_capabilities.h, 672, 673
- include/aubo_caps/contribution/driver/gripper/capability/robot_capabilities.h, 674, 675
- include/aubo_caps/contribution/driver/gripper/grip_includes.h, 677, 678
- include/aubo_caps/contribution/driver/gripper/gripper_capabilities.h, 679, 681
- include/aubo_caps/contribution/driver/gripper/gripper_capabilities.h, 681, 683
- include/aubo_caps/contribution/driver/gripper/gripper_capabilities.h, 686, 687
- include/aubo_caps/contribution/driver/gripper/release_capabilities.h, 690, 691
- include/aubo_caps/contribution/driver/screw_drive Directory Reference, 67
- include/aubo_caps/contribution/driver/screw_drive_capabilities.h, 692, 694
- include/aubo_caps/contribution/driver/screw_drive_capabilities.h, 694, 696
- include/aubo_caps/contribution/driver/screw_drive_capabilities.h, 702, 703
- include/aubo_caps/contribution/driver/screw_drive_capabilities.h, 705, 707
- include/aubo_caps/contribution/installation Directory Reference, 63
- include/aubo_caps/contribution/installation/installation_creation_controllers.h, 709, 710
- include/aubo_caps/contribution/installation/installation_node_controllers.h, 711, 712
- include/aubo_caps/contribution/installation/installation_node_controllers.h, 712, 713
- include/aubo_caps/contribution/installation/installation_node_controllers.h, 714, 715
- include/aubo_caps/contribution/installation/installation_node_services.h, 716, 717
- include/aubo_caps/contribution/installation/installation_node_services.h, 719, 721
- include/aubo_caps/contribution/program Directory Reference, 65
- include/aubo_caps/contribution/program/program_api_provider.h, 721, 722
- include/aubo_caps/contribution/program/program_contribution_controllers.h, 724, 725
- include/aubo_caps/contribution/program/program_creation_controllers.h, 727, 728
- include/aubo_caps/contribution/program/program_node_contribution_controllers.h, 729, 730
- include/aubo_caps/contribution/program/program_node_services.h, 732
- include/aubo_caps/contribution/program/program_node_view.h, 734, 735
- include/aubo_caps/contribution/program/program_node_view_api_provider.h, 736, 737
- include/aubo_caps/contribution/program/program_node_view_api_provider.h, 738, 739
- include/aubo_caps/domain/data Directory Reference, 58
- include/aubo_caps/domain/data/data_model.h, 742, 743
- include/aubo_caps/domain/device Directory Reference, 58
- include/aubo_caps/domain/device/device.h, 752, 754
- include/aubo_caps/domain/device/device_manager.h, 755, 757
- include/aubo_caps/domain/device/device_registration_manager.h, 758, 759
- include/aubo_caps/domain/device/gripper_device.h, 760, 762
- include/aubo_caps/domain/extension_info.h, 762, 764
- include/aubo_caps/provider/feature Directory Reference, 61
- include/aubo_caps/provider/feature/base_feature.h, 765, 766
- include/aubo_caps/provider/feature/feature.h, 766, 767
- include/aubo_caps/provider/feature/feature_contribution_model.h, 769, 771
- include/aubo_caps/provider/feature/feature_line.h, 772, 773
- include/aubo_caps/provider/feature/feature_model.h, 774, 775
- include/aubo_caps/provider/feature/feature_plane.h, 776, 777

- [777, 778](#)
- [include/aubo_caps/domain/feature/feature_point.h,](#)
[779, 780](#)
- [include/aubo_caps/domain/feature/feature_tool.h,](#)
[781, 782](#)
- [include/aubo_caps/domain/function](#) Directory
Reference, [61](#)
- [include/aubo_caps/domain/function/function.h,](#)
[782, 784](#)
- [include/aubo_caps/domain/function/function_model.h,](#)
[785, 786](#)
- [include/aubo_caps/domain/installation](#) Directory
Reference, [63](#)
- [include/aubo_caps/domain/installation/installation_api.h,](#)
[788, 789](#)
- [include/aubo_caps/domain/io](#) Directory Refer-
ence, [64](#)
- [include/aubo_caps/domain/io/analog_io.h,](#) [791,](#)
[793](#)
- [include/aubo_caps/domain/io/boolean_register.h,](#)
[794, 796](#)
- [include/aubo_caps/domain/io/digital_io.h,](#) [796,](#)
[798](#)
- [include/aubo_caps/domain/io/double_register.h,](#)
[799, 801](#)
- [include/aubo_caps/domain/io/ethernet_ip.h,](#) [801,](#)
[802](#)
- [include/aubo_caps/domain/io/integer_register.h,](#)
[804, 805](#)
- [include/aubo_caps/domain/io/io.h,](#) [805, 806](#)
- [include/aubo_caps/domain/io/io_model.h,](#) [808,](#)
[809](#)
- [include/aubo_caps/domain/io/modbus_io.h,](#) [811,](#)
[813](#)
- [include/aubo_caps/domain/io/profinet.h,](#) [815, 816](#)
- [include/aubo_caps/domain/io/register.h,](#) [818, 820](#)
- [include/aubo_caps/domain/payload](#) Directory
Reference, [65](#)
- [include/aubo_caps/domain/payload/payload.h,](#)
[820, 821](#)
- [include/aubo_caps/domain/payload/payload_contribution_model.h,](#)
[824, 825](#)
- [include/aubo_caps/domain/payload/payload_model.h,](#)
[830, 832](#)
- [include/aubo_caps/domain/program](#) Directory
Reference, [66](#)
- [include/aubo_caps/domain/program/nodes](#) Direc-
tory Reference, [64](#)
- [include/aubo_caps/domain/program/nodes/builtin](#)
Directory Reference, [56](#)
- [include/aubo_caps/domain/program/nodes/builtin/assignment_node.h,](#)
[833, 835](#)
- [include/aubo_caps/domain/program/nodes/builtin/assignment_node.h,](#)
[835, 837](#)
- [include/aubo_caps/domain/program/nodes/builtin/break_node.h,](#)
[838, 840](#)
- [include/aubo_caps/domain/program/nodes/builtin/call_node.h,](#)
[840, 842](#)
- [include/aubo_caps/domain/program/nodes/builtin/case_node.h,](#)
[842, 844](#)
- [include/aubo_caps/domain/program/nodes/builtin/circle_move_node.h,](#)
[844, 846](#)
- [include/aubo_caps/domain/program/nodes/builtin/comment_node.h,](#)
[846, 848](#)
- [include/aubo_caps/domain/program/nodes/builtin/conveyor_track_node.h,](#)
[848, 850](#)
- [include/aubo_caps/domain/program/nodes/builtin/default_case_node.h,](#)
[851, 853](#)
- [include/aubo_caps/domain/program/nodes/builtin/direction_node.h,](#)
[853, 855](#)
- [include/aubo_caps/domain/program/nodes/builtin/else_if_node.h,](#)
[856, 859](#)
- [include/aubo_caps/domain/program/nodes/builtin/else_node.h,](#)
[859, 861](#)
- [include/aubo_caps/domain/program/nodes/builtin/folder_node.h,](#)
[861, 863](#)
- [include/aubo_caps/domain/program/nodes/builtin/force_node.h,](#)
[863, 865](#)
- [include/aubo_caps/domain/program/nodes/builtin/force_option_node.h,](#)
[866, 868](#)
- [include/aubo_caps/domain/program/nodes/builtin/gripper_node.h,](#)
[868, 870](#)
- [include/aubo_caps/domain/program/nodes/builtin/halt_node.h,](#)
[872, 874](#)
- [include/aubo_caps/domain/program/nodes/builtin/if_node.h,](#)
[874, 876](#)
- [include/aubo_caps/domain/program/nodes/builtin/loop_node.h,](#)
[876, 879](#)
- [include/aubo_caps/domain/program/nodes/builtin/move_node.h,](#)
[880, 882](#)
- [include/aubo_caps/domain/program/nodes/builtin/offline_track_node.h,](#)
[885, 887](#)
- [include/aubo_caps/domain/program/nodes/builtin/pallet_node.h,](#)
[887, 889](#)
- [include/aubo_caps/domain/program/nodes/builtin/popup_node.h,](#)
[889, 891](#)
- [include/aubo_caps/domain/program/nodes/builtin/program_home_node.h,](#)
[891, 894](#)
- [include/aubo_caps/domain/program/nodes/builtin/record_track_node.h,](#)
[894, 896](#)
- [include/aubo_caps/domain/program/nodes/builtin/screw_driving_node.h,](#)
[896, 898](#)
- [include/aubo_caps/domain/program/nodes/builtin/script_node.h,](#)
[898, 900](#)
- [include/aubo_caps/domain/program/nodes/builtin/seek_node.h,](#)
[900, 902](#)
- [include/aubo_caps/domain/program/nodes/builtin/set_node.h,](#)
[902, 904](#)
- [include/aubo_caps/domain/program/nodes/builtin/set_payload_node.h,](#)
[906, 909](#)
- [include/aubo_caps/domain/program/nodes/builtin/sub_prog_node.h,](#)
[910, 912](#)
- [include/aubo_caps/domain/program/nodes/builtin/switch_node.h,](#)
[912, 914](#)
- [include/aubo_caps/domain/program/nodes/builtin/thread_node.h,](#)
[914, 916](#)

- include/aubo_caps/domain/program/nodes/builtin/timer_node.h, 1002, 1003
916, 918
- include/aubo_caps/domain/program/nodes/builtin/until_node.h, 1004, 1005
918, 920
- include/aubo_caps/domain/program/nodes/builtin/visual_location_node.h, 924, 926
- include/aubo_caps/domain/program/nodes/builtin/wait_node.h, 1007, 1009
926, 928
- include/aubo_caps/domain/program/nodes/builtin/waypoint_node.h, 1010, 1012
931, 933
- include/aubo_caps/domain/program/nodes/program_node.h, 1013, 1015
937, 938
- include/aubo_caps/domain/program/nodes/program_node_factory.h, 1015, 1017
939, 942
- include/aubo_caps/domain/program/program_api.h, 1018, 1020
945, 946
- include/aubo_caps/domain/program/program_model.h, 1021, 1022
947, 948
- include/aubo_caps/domain/program/structure Directory Reference, 68
- include/aubo_caps/domain/program/structure/tree_node.h, 1024, 1025
949, 950
- include/aubo_caps/domain/script Directory Reference, 67
- include/aubo_caps/domain/script/script_writer.h, 955, 956
- include/aubo_caps/domain/system_api.h, 958, 960
- include/aubo_caps/domain/tcp Directory Reference, 68
- include/aubo_caps/domain/tcp/tcp.h, 966, 968
- include/aubo_caps/domain/tcp/tcp_contribution_model.h, 969, 970
- include/aubo_caps/domain/tcp/tcp_model.h, 972, 973
- include/aubo_caps/domain/undoredo Directory Reference, 69
- include/aubo_caps/domain/undoredo/undo_redo_manager.h, 975, 977
- include/aubo_caps/domain/user_interface Directory Reference, 69
- include/aubo_caps/domain/user_interface/input_validation.h, 978, 979
- include/aubo_caps/domain/user_interface/keyboard_input_handler.h, 981, 982
- include/aubo_caps/domain/user_interface/popup_installation_node_view.h, 984, 985
- include/aubo_caps/domain/user_interface/render_interface.h, 986, 987
- include/aubo_caps/domain/user_interface/user_interaction_model.h, 995, 997
- include/aubo_caps/domain/user_interface/user_interface_api.h, 1000, 1001
- include/aubo_caps/domain/value Directory Reference, 70
- include/aubo_caps/domain/value/expression Directory Reference, 60
- include/aubo_caps/domain/value/expression/expression.h, 1002, 1003
- include/aubo_caps/domain/value/value_factory_provider.h, 1004, 1005
- include/aubo_caps/domain/variable Directory
include/aubo_caps/domain/variable/feature_variable.h,
include/aubo_caps/domain/variable/global_variable.h,
include/aubo_caps/domain/variable/persisted_variable.h,
include/aubo_caps/domain/variable/variable.h,
include/aubo_caps/domain/variable/variable_factory.h,
include/aubo_caps/domain/variable/variable_model.h,
include/aubo_caps/domain/waypoint Directory
Reference, 71
include/aubo_caps/domain/waypoint/waypoint.h,
include/aubo_caps/domain/waypoint/waypoint_model.h,
1027, 1028
- increaseIndent
arcs::aubo_scope::ScriptWriter, 509
- INCREMENT_VARIABLE
arcs::aubo_scope::SetNode, 519
- incrementVariable
arcs::aubo_scope::ScriptWriter, 509
- INPUT
arcs::aubo_scope::WaypointNode, 648
- INPUT_BOOL
arcs::aubo_scope::AssignmentNode, 115
- INPUT_FLOAT
arcs::aubo_scope::AssignmentNode, 115
- INPUT_INTEGER
arcs::aubo_scope::AssignmentNode, 115
- INPUT_STRING
arcs::aubo_scope::AssignmentNode, 115
- InputMode
arcs::aubo_scope::AssignmentNode, 115
- InsertChildAfter
arcs::aubo_scope::TreeNode, 568
- InsertChildBefore
arcs::aubo_scope::TreeNode, 569
- Installation
WidgetHandle, 720
- InstallationApi, 40
arcs::aubo_scope::InstallationApi, 306, 307
- InstallationApiProvider
arcs::aubo_scope::InstallationApiProvider,
308, 309
- InstallationCreationContext
arcs::aubo_scope::InstallationCreationContext,

- 312, 313
- INTEGER
 - arcs::aubo_scope::Io, 326
- IntegerRangeValidator
 - arcs::aubo_scope::IntegerRangeValidator, 320
- interface_
 - arcs::aubo_scope::AnalogIoFilter, 104
 - arcs::aubo_scope::DigitalIoFilter, 181
- InterfaceType
 - arcs::aubo_scope::Io, 325
- Io
 - arcs::aubo_scope::Io, 326, 327
- IO_INPUT
 - arcs::aubo_scope::UntilNode, 582
- IOInputSelectionType
 - arcs::aubo_scope::UntilNode, 583
- IoModel
 - arcs::aubo_scope::IoModel, 333, 334
- IoType
 - arcs::aubo_scope::Io, 326
- is_input_
 - arcs::aubo_scope::AnalogIoFilter, 104
 - arcs::aubo_scope::DigitalIoFilter, 181
 - arcs::aubo_scope::EtherNetIPFilter, 209
 - arcs::aubo_scope::ModbusIoFilter, 355
 - arcs::aubo_scope::ProfinetFilter, 407
- is_max_inclusive_
 - arcs::aubo_scope::DoubleRangeValidator, 191
- is_min_inclusive_
 - arcs::aubo_scope::DoubleRangeValidator, 191
- isCurrent
 - arcs::aubo_scope::AnalogIo, 102
- isDefined
 - arcs::aubo_scope::Feature, 216
 - arcs::aubo_scope::ProgramNodeContribution, 437
 - arcs::aubo_scope::Waypoint, 638
- isEvaluateContinously
 - arcs::aubo_scope::LoopNode, 347
- isGripDetectionEnabled
 - arcs::aubo_scope::GripActionParameters, 272
- isInProgramTree
 - arcs::aubo_scope::TreeNode, 570
- isInput
 - arcs::aubo_scope::Io, 329
- isJoggable
 - arcs::aubo_scope::Feature, 216
- isPopupCountZero
 - arcs::aubo_scope::PopupManager, 394
- isRealRobot
 - SystemApi, 52
- isReleaseDetectionEnabled
 - arcs::aubo_scope::ReleaseActionParameters, 461
- isResolvable
 - arcs::aubo_scope::Device, 172
 - arcs::aubo_scope::Feature, 217
 - arcs::aubo_scope::Io, 329
 - arcs::aubo_scope::Payload, 375
 - arcs::aubo_scope::PersistedVariable, 391
 - arcs::aubo_scope::TCP, 547
- isSet
 - arcs::aubo_scope::DataModel, 157
- isSlave
 - arcs::aubo_scope::EtherNetIP, 207
 - arcs::aubo_scope::ModbusIo, 353
 - arcs::aubo_scope::Profinet, 405
- isSuppressed
 - arcs::aubo_scope::TreeNode, 570
- isUsable
 - arcs::aubo_scope::Payload, 375
- isValid
 - arcs::aubo_scope::DoubleRangeValidator, 191
 - arcs::aubo_scope::InputValidator, 303
 - arcs::aubo_scope::IntegerRangeValidator, 321
 - arcs::aubo_scope::StringLengthValidator, 533
 - arcs::aubo_scope::UIntegerRangeValidator, 575
- isVariable
 - arcs::aubo_scope::Feature, 217
- isVoltage
 - arcs::aubo_scope::AnalogIo, 103
- JOINT_MOTION
 - arcs::aubo_scope::WaypointNode, 647
- KeyboardManager
 - arcs::aubo_scope::KeyboardManager, 338, 339
- killThread
 - arcs::aubo_scope::ScriptWriter, 509
- label
 - arcs::aubo_scope::IoConfigure, 331
- LESS_THAN
 - arcs::aubo_scope::UntilNode, 582
 - arcs::aubo_scope::WaitNode, 627
- lineFeed
 - arcs::aubo_scope::ScriptWriter, 509
- LINKIO
 - arcs::aubo_scope::Io, 326
- LOAD
 - arcs::aubo_scope::InstallationCreationContext, 312
 - arcs::aubo_scope::ProgramCreationContext, 422
- locateDescendantTreeNode
 - arcs::aubo_scope::TreeNode, 570
- LoopNode
 - arcs::aubo_scope::LoopNode, 345, 346
- LOOSEN
 - arcs::aubo_scope, 82
- MASS_0KG
 - arcs::aubo_scope::Payload, 372
- MASS_AND_CENTER_OF_GRAVITY_EXCEEDED
 - arcs::aubo_scope::Payload, 372
- max_length_

- arcs::aubo_scope::StringLengthValidator, [534](#)
- max_value_
 - arcs::aubo_scope::DoubleRangeValidator, [191](#)
 - arcs::aubo_scope::IntegerRangeValidator, [321](#)
 - arcs::aubo_scope::UIntegerRangeValidator, [575](#)
- maxs
 - arcs::aubo_scope::IoConfigure, [331](#)
- MESSAGE
 - arcs::aubo_scope::PopupNode, [397](#)
- METRIC
 - arcs::aubo_scope, [82](#)
- min_length_
 - arcs::aubo_scope::StringLengthValidator, [534](#)
- min_value_
 - arcs::aubo_scope::DoubleRangeValidator, [192](#)
 - arcs::aubo_scope::IntegerRangeValidator, [321](#)
 - arcs::aubo_scope::UIntegerRangeValidator, [575](#)
- mins
 - arcs::aubo_scope::IoConfigure, [331](#)
- MODBUS
 - arcs::aubo_scope::Io, [326](#)
- ModbusIo
 - arcs::aubo_scope::ModbusIo, [351](#), [352](#)
- ModbusIoFilter
 - arcs::aubo_scope::ModbusIoFilter, [354](#)
- MOTION
 - arcs::aubo_scope::ForceNode, [252](#)
- MotionType
 - arcs::aubo_scope::WaypointNode, [647](#)
- MOVE_J
 - arcs::aubo_scope::MoveNode, [357](#)
- MOVE_L
 - arcs::aubo_scope::MoveNode, [357](#)
- MOVE_S
 - arcs::aubo_scope::MoveNode, [357](#)
- MOVE_TCP
 - arcs::aubo_scope::MoveNode, [358](#)
- MoveNode
 - arcs::aubo_scope::MoveNode, [358](#), [359](#)
- names
 - arcs::aubo_scope::IoConfigure, [331](#)
- NEW
 - arcs::aubo_scope::InstallationCreationContext, [312](#)
 - arcs::aubo_scope::ProgramCreationContext, [422](#)
- NEW_WORKPIECE
 - arcs::aubo_scope::ConveyorTrackingNode, [145](#)
- NO_ACTION
 - arcs::aubo_scope::SetNode, [518](#)
- NO_BLEND
 - arcs::aubo_scope::UntilNode, [581](#)
 - arcs::aubo_scope::WaypointNode, [647](#)
- NO_OUTPUT
 - arcs::aubo_scope::SetNode, [519](#)
- NO_TYPE
 - arcs::aubo_scope::SetNode, [518](#)
 - arcs::aubo_scope::WaitNode, [627](#)
- NO_WAIT
 - arcs::aubo_scope::WaitNode, [627](#)
- NO_WARNING
 - arcs::aubo_scope::Payload, [372](#)
- NodeCreationType
 - arcs::aubo_scope::InstallationCreationContext, [311](#)
 - arcs::aubo_scope::ProgramCreationContext, [421](#)
- NONE
 - arcs::aubo_scope::AssemblyNode, [111](#)
- None
 - arcs::aubo_scope::RenderInterface, [464](#)
- NONE_CONFIG_TYPE
 - arcs::aubo_scope::UntilNode, [582](#)
- NONE_INPUT_SELECT
 - arcs::aubo_scope::UntilNode, [583](#)
- note
 - arcs::aubo_scope::ScriptWriter, [509](#)
- notifyPopupHidden
 - arcs::aubo_scope::PopupManager, [394](#)
- notifyPopupShown
 - arcs::aubo_scope::PopupManager, [394](#)
- num
 - arcs::aubo_scope::IoConfigure, [332](#)
- Obstacle
 - arcs::aubo_scope::RenderInterface, [465](#)
- OfflineTrackNode
 - arcs::aubo_scope::OfflineTrackNode, [365](#), [366](#)
- openView
 - arcs::aubo_scope::InstallationNodeContribution, [315](#)
 - arcs::aubo_scope::ProgramNodeContribution, [437](#)
- OPERATOR
 - arcs::aubo_scope::AssignmentNode, [115](#)
- operator()
 - arcs::aubo_scope::AnalogIoFilter, [104](#)
 - arcs::aubo_scope::DigitalIoFilter, [181](#)
 - arcs::aubo_scope::EtherNetIPFilter, [209](#)
 - arcs::aubo_scope::FeatureDefinedFilter, [223](#)
 - arcs::aubo_scope::FeatureJoggableFilter, [223](#)
 - arcs::aubo_scope::FeatureVariableFilter, [246](#)
 - arcs::aubo_scope::GlobalVariableFilter, [270](#)
 - arcs::aubo_scope::InputIoFilter, [302](#)
 - arcs::aubo_scope::ModbusIoFilter, [355](#)
 - arcs::aubo_scope::OutputIoFilter, [367](#)
 - arcs::aubo_scope::PersistedVariableFilter, [392](#)
 - arcs::aubo_scope::ProfinetFilter, [407](#)
- OUTPUT
 - arcs::aubo_scope::SetNode, [519](#)
- OutputSelectionType
 - arcs::aubo_scope::SetNode, [519](#)
- PalletNode

- arcs::aubo_scope::PalletNode, [369](#), [370](#)
- ParametersWarning
 - arcs::aubo_scope::Payload, [371](#)
- ParameterType
 - arcs::aubo_scope::DirectionNode, [184](#)
- Payload
 - arcs::aubo_scope::Payload, [372](#), [373](#)
- PAYLOAD_SELECTION
 - arcs::aubo_scope::GripperNode, [289](#)
 - arcs::aubo_scope::SetPayloadNode, [526](#)
- PayloadContributionModel
 - arcs::aubo_scope::PayloadContributionModel, [377](#), [378](#)
- PayloadModel
 - arcs::aubo_scope::PayloadModel, [385](#), [386](#)
- PersistedVariable
 - arcs::aubo_scope::PersistedVariable, [390](#), [391](#)
- POINT
 - arcs::aubo_scope::ForceNode, [252](#)
- Point
 - arcs::aubo_scope::RenderInterface, [464](#)
- PointCloud
 - arcs::aubo_scope::RenderInterface, [465](#)
- popupCount
 - arcs::aubo_scope::PopupManager, [394](#)
- PopupManager
 - arcs::aubo_scope::PopupManager, [393](#), [394](#)
- PopupNode
 - arcs::aubo_scope::PopupNode, [398](#), [399](#)
- PositionDefinition
 - arcs::aubo_scope::WaypointNode, [647](#)
- postAction
 - arcs::aubo_scope::ProgramNodeContribution, [437](#)
- PREVIOUS_WORKPIECE
 - arcs::aubo_scope::ConveyorTrackingNode, [145](#)
- PROFINET
 - arcs::aubo_scope::Io, [326](#)
- Profinet
 - arcs::aubo_scope::Profinet, [403](#), [404](#)
- ProfinetFilter
 - arcs::aubo_scope::ProfinetFilter, [407](#)
- ProgramApi, [41](#)
 - arcs::aubo_scope::ProgramApi, [410](#), [411](#)
 - getProgramModel, [42](#)
 - getUndoRedoManager, [42](#)
 - getWaypointModel, [42](#)
- ProgramApiProvider
 - arcs::aubo_scope::ProgramApiProvider, [412](#), [413](#)
- ProgramContributionConfiguration
 - arcs::aubo_scope::ProgramContributionConfiguration, [416](#), [417](#)
- ProgramCreationContext
 - arcs::aubo_scope::ProgramCreationContext, [422](#), [423](#)
- ProgramHomeNode
 - arcs::aubo_scope::ProgramHomeNode, [426](#), [427](#)
- ProgramModel
 - arcs::aubo_scope::ProgramModel, [429](#), [430](#)
- ProgramNode
 - arcs::aubo_scope::ProgramNode, [433](#), [434](#)
- ProgramNodeFactory
 - arcs::aubo_scope::ProgramNodeFactory, [440](#), [441](#)
- range_func_
 - arcs::aubo_scope::DoubleRangeValidator, [192](#)
 - arcs::aubo_scope::IntegerRangeValidator, [321](#)
 - arcs::aubo_scope::StringLengthValidator, [534](#)
 - arcs::aubo_scope::UIntegerRangeValidator, [575](#)
- range_func_with_flags_
 - arcs::aubo_scope::DoubleRangeValidator, [192](#)
- RATED_MASS_EXCEEDED
 - arcs::aubo_scope::Payload, [372](#)
- REACHED_WAYPOINT
 - arcs::aubo_scope::UntilNode, [582](#)
- RealRobot
 - arcs::aubo_scope::RenderInterface, [464](#)
- recordChanges
 - arcs::aubo_scope::UndoRedoManager, [578](#)
- RecordTrackNode
 - arcs::aubo_scope::RecordTrackNode, [454](#), [455](#)
- registerCodeEditorKeyboard
 - arcs::aubo_scope::KeyboardManager, [339](#)
- registerDoubleKeyboard
 - arcs::aubo_scope::KeyboardManager, [339](#)
- registerDriveScrewNotOKCapability
 - arcs::aubo_scope::ScrewdriverConfiguration, [482](#)
- registerDriveScrewOKCapability
 - arcs::aubo_scope::ScrewdriverConfiguration, [484](#)
- registerExpressionKeyboard
 - arcs::aubo_scope::KeyboardManager, [340](#)
- registerFeedScrewCapability
 - arcs::aubo_scope::ScrewdriverConfiguration, [485](#)
- registerGripDetectedCapability
 - arcs::aubo_scope::GripperConfiguration, [276](#)
- registerGripperContribution
 - arcs::aubo_scope::DeviceRegistrationManager, [174](#)
- registerGrippingForceCapability
 - arcs::aubo_scope::GripperConfiguration, [277](#)
- registerGrippingVacuumCapability
 - arcs::aubo_scope::GripperConfiguration, [278](#)
- registerIntegerKeyboard
 - arcs::aubo_scope::KeyboardManager, [340](#)
- registerIPAddressKeyboard
 - arcs::aubo_scope::KeyboardManager, [340](#)
- registerNumberKeyboard
 - arcs::aubo_scope::KeyboardManager, [340](#)
- registerOperationTypeCapability

- arcs::aubo_scope::ScrewdriverConfiguration, 485
- registerPasswordKeyboard
 - arcs::aubo_scope::KeyboardManager, 340
- registerPositiveDoubleKeyboard
 - arcs::aubo_scope::KeyboardManager, 341
- registerPositiveIntegerKeyboard
 - arcs::aubo_scope::KeyboardManager, 341
- registerPrepareToStartScrewdriverCapability
 - arcs::aubo_scope::ScrewdriverConfiguration, 486
- registerProgramSelectionCapability
 - arcs::aubo_scope::ScrewdriverConfiguration, 486
- registerReleaseDetectedCapability
 - arcs::aubo_scope::GripperConfiguration, 279
- registerScrewDriverContribution
 - arcs::aubo_scope::DeviceRegistrationManager, 175
- registerScrewdriverReadyCapability
 - arcs::aubo_scope::ScrewdriverConfiguration, 487
- registerSpeedCapability
 - arcs::aubo_scope::GripperConfiguration, 280
- registerStandardKeyboard
 - arcs::aubo_scope::KeyboardManager, 341
- registerStringKeyboard
 - arcs::aubo_scope::KeyboardManager, 342
- registerWidthCapability
 - arcs::aubo_scope::GripperConfiguration, 281
- RELATIVE_POSITION
 - arcs::aubo_scope::WaypointNode, 647
- RelativeOffsetType
 - arcs::aubo_scope::WaypointNode, 648
- RelativePositionType
 - arcs::aubo_scope::WaypointNode, 648
- RELEASE_ACTION
 - arcs::aubo_scope::GripperNode, 289
- remove
 - arcs::aubo_scope::DataModel, 157
- removeChild
 - arcs::aubo_scope::TreeNode, 571
- removeCoordinate
 - arcs::aubo_scope::RenderInterface, 475
- removeCube
 - arcs::aubo_scope::RenderInterface, 475
- removeFeature
 - arcs::aubo_scope::FeatureContributionModel, 221
- removeFunction
 - arcs::aubo_scope::FunctionModel, 264
- removeLine
 - arcs::aubo_scope::RenderInterface, 475
- removeMesh
 - arcs::aubo_scope::RenderInterface, 476
- removePayload
 - arcs::aubo_scope::PayloadContributionModel, 381
- removePlane
 - arcs::aubo_scope::RenderInterface, 476
- removePointCloud
 - arcs::aubo_scope::RenderInterface, 476
- removeTcp
 - arcs::aubo_scope::TcpContributionModel, 551
- removeWaypoint
 - arcs::aubo_scope::RenderInterface, 477
 - arcs::aubo_scope::WaypointModel, 643
- renameFeature
 - arcs::aubo_scope::FeatureContributionModel, 221
- renameWaypoint
 - arcs::aubo_scope::WaypointModel, 643
- RenderInterface
 - arcs::aubo_scope::RenderInterface, 465, 466
- requestUserToMoveJoint
 - arcs::aubo_scope::UserInteraction, 595, 596
- requestUserToMoveLine
 - arcs::aubo_scope::UserInteraction, 597
- RESERVE_QUEUE
 - arcs::aubo_scope::ConveyorTrackingNode, 144
- returnMethod
 - arcs::aubo_scope::ScriptWriter, 510
- RobotMovementCallback
 - arcs::aubo_scope, 81
- RobotPositionCallback
 - arcs::aubo_scope, 81
- runThread
 - arcs::aubo_scope::ScriptWriter, 510
- safety_functions
 - arcs::aubo_scope::IoConfigure, 332
- safety_functions_fixed
 - arcs::aubo_scope::IoConfigure, 332
- SafetyPlane
 - arcs::aubo_scope::RenderInterface, 465
- ScrewdriverOperationType
 - arcs::aubo_scope, 81
- ScrewdrivingNode
 - arcs::aubo_scope::ScrewdrivingNode, 496, 497
- ScriptNode
 - arcs::aubo_scope::ScriptNode, 499, 500
- ScriptWriter
 - arcs::aubo_scope::ScriptWriter, 503, 504
- SeekNode
 - arcs::aubo_scope::SeekNode, 514, 515
- sequences
 - arcs::aubo_scope::IoConfigure, 332
- set
 - arcs::aubo_scope::DataModel, 158–166
- setAllowBreakpointOnChildNodesInSubtree
 - arcs::aubo_scope::ProgramContributionConfiguration, 417
- setAllowBreakpointOnNode
 - arcs::aubo_scope::ProgramContributionConfiguration, 417
- setAllowInsertOnlyIntoNodes

- arcs::aubo_scope::ProgramContributionConfiguration, 417
- setAllowStartFromChildNodesInSubtree
 - arcs::aubo_scope::ProgramContributionConfiguration, 418
- setAllowStartFromNode
 - arcs::aubo_scope::ProgramContributionConfiguration, 418
- setAnalogInput
 - arcs::aubo_scope::WaitNode, 630
- setAnalogInputValueToProceed
 - arcs::aubo_scope::UntilNode, 587
- setAnalogOutput
 - arcs::aubo_scope::SetNode, 523
- setAngularDeceleration
 - arcs::aubo_scope::UntilNode, 587
- setAxis
 - arcs::aubo_scope::DirectionNode, 187
- setBlend
 - arcs::aubo_scope::MoveNode, 360
- setBlendParameterType
 - arcs::aubo_scope::UntilNode, 588
- setBlendTrack
 - arcs::aubo_scope::RenderInterface, 477
- setBoolRegister
 - arcs::aubo_scope::WaitNode, 631
- setBoolValueToSet
 - arcs::aubo_scope::SetNode, 523
- setBoolValueToWaitFor
 - arcs::aubo_scope::WaitNode, 632
- setCartesianDeceleration
 - arcs::aubo_scope::UntilNode, 588
- setCenterOfGravity
 - arcs::aubo_scope::SetPayloadNode, 529
- setChildrenAllowed
 - arcs::aubo_scope::ProgramContributionConfiguration, 418
- setChildrenAllowedOnly
 - arcs::aubo_scope::ProgramContributionConfiguration, 419
- setChildSequenceLocked
 - arcs::aubo_scope::TreeNode, 571
- setComment
 - arcs::aubo_scope::CommentNode, 140
- setCompareOperator
 - arcs::aubo_scope::UntilNode, 588
 - arcs::aubo_scope::WaitNode, 632
- setConfigType
 - arcs::aubo_scope::AssignmentNode, 117
 - arcs::aubo_scope::GripperNode, 292
 - arcs::aubo_scope::LoopNode, 347
 - arcs::aubo_scope::MoveNode, 361
 - arcs::aubo_scope::SetNode, 523
 - arcs::aubo_scope::SetPayloadNode, 529
 - arcs::aubo_scope::UntilNode, 588
 - arcs::aubo_scope::WaitNode, 632
- setCurrent
 - arcs::aubo_scope::SetNode, 523
- setCustomDecelerationParameterType
 - arcs::aubo_scope::UntilNode, 588
- setDecelerationParameters
 - arcs::aubo_scope::UntilNode, 588
- setDecelerationParameterType
 - arcs::aubo_scope::UntilNode, 589
- setDeprecated
 - arcs::aubo_scope::ProgramContributionConfiguration, 419
- setDigitalInput
 - arcs::aubo_scope::WaitNode, 632
- setDigitalInputValueToProceed
 - arcs::aubo_scope::UntilNode, 589
- setDirectionSelectionType
 - arcs::aubo_scope::DirectionNode, 187
- setDisplayOrderId
 - arcs::aubo_scope::ProgramContributionConfiguration, 419
- setDistance
 - arcs::aubo_scope::UntilNode, 589
- setDoubleRegisterInput
 - arcs::aubo_scope::WaitNode, 633
- setEvaluateContinously
 - arcs::aubo_scope::LoopNode, 347
- setExpression
 - arcs::aubo_scope::AssignmentNode, 117
 - arcs::aubo_scope::DirectionNode, 187
 - arcs::aubo_scope::ElseIfNode, 198
 - arcs::aubo_scope::IfNode, 300
 - arcs::aubo_scope::LoopNode, 347
 - arcs::aubo_scope::SetNode, 523
- setExpressionConfig
 - arcs::aubo_scope::UntilNode, 590
- setExpressionInputConfig
 - arcs::aubo_scope::WaitNode, 633
- setFeature
 - arcs::aubo_scope::DirectionNode, 187
 - arcs::aubo_scope::MoveNode, 361
- setFeatureSelectionType
 - arcs::aubo_scope::DirectionNode, 187
- setFixedPositionConfig
 - arcs::aubo_scope::WaypointNode, 653
- setFloatValueToSet
 - arcs::aubo_scope::SetNode, 523
- setFloatValueToWaitFor
 - arcs::aubo_scope::WaitNode, 634
- setFolderName
 - arcs::aubo_scope::FolderNode, 250
- setIOInputSelectionType
 - arcs::aubo_scope::UntilNode, 590
- setJointAcceleration
 - arcs::aubo_scope::MoveNode, 361
- setJointMotionParameters
 - arcs::aubo_scope::WaypointNode, 654
- setJointSpeed
 - arcs::aubo_scope::MoveNode, 362
- setLabel
 - arcs::aubo_scope::ScriptWriter, 510

- setLoopCount
 - arcs::aubo_scope::LoopNode, [347](#)
- setLoopCountVariable
 - arcs::aubo_scope::LoopNode, [348](#)
- setMass
 - arcs::aubo_scope::SetPayloadNode, [529](#)
- setMessage
 - arcs::aubo_scope::PopupNode, [399](#)
- setModbusInput
 - arcs::aubo_scope::WaitNode, [634](#)
- setMoveTrack
 - arcs::aubo_scope::RenderInterface, [478](#)
- SetNode
 - arcs::aubo_scope::SetNode, [520](#), [521](#)
- setOperationLogDisplayName
 - arcs::aubo_scope::UserInteraction, [598](#)
- setOutput
 - arcs::aubo_scope::SetNode, [523](#)
- setPayload
 - arcs::aubo_scope::ScriptWriter, [510](#)
- SetPayloadNode
 - arcs::aubo_scope::SetPayloadNode, [527](#), [528](#)
- setPopupCountChangedCallback
 - arcs::aubo_scope::PopupManager, [394](#)
- setPulseTime
 - arcs::aubo_scope::SetNode, [523](#)
- setRenderEmbedded
 - arcs::aubo_scope::RenderInterface, [478](#)
- setRobotIndex
 - arcs::aubo_scope::ScriptWriter, [510](#)
- setSharedMotionParameters
 - arcs::aubo_scope::WaypointNode, [654](#)
- setShowOptions
 - arcs::aubo_scope::RenderInterface, [479](#)
- setTcp
 - arcs::aubo_scope::ScriptWriter, [510](#)
- setTCPSelection
 - arcs::aubo_scope::MoveNode, [362](#)
- setTcpSelectionType
 - arcs::aubo_scope::MoveNode, [362](#)
- setTimeMotionParameters
 - arcs::aubo_scope::WaypointNode, [654](#)
- SettingType
 - arcs::aubo_scope::GripperNode, [289](#)
- setToolAcceleration
 - arcs::aubo_scope::DirectionNode, [187](#)
 - arcs::aubo_scope::MoveNode, [362](#)
- setToolMotionParameters
 - arcs::aubo_scope::WaypointNode, [655](#)
- setToolSpeed
 - arcs::aubo_scope::DirectionNode, [188](#)
 - arcs::aubo_scope::MoveNode, [363](#)
- setType
 - arcs::aubo_scope::DirectionNode, [188](#)
 - arcs::aubo_scope::PopupNode, [399](#)
- setUseJoints
 - arcs::aubo_scope::MoveNode, [363](#)
- setUserInsertable
 - arcs::aubo_scope::ProgramContributionConfiguration, [420](#)
- setValue
 - arcs::aubo_scope::AnalogIo, [103](#)
 - arcs::aubo_scope::DigitalIo, [179](#)
 - arcs::aubo_scope::EtherNetIP, [207](#)
 - arcs::aubo_scope::ModbusIo, [353](#)
 - arcs::aubo_scope::Profinet, [405](#)
- setVariable
 - arcs::aubo_scope::AssignmentNode, [117](#)
- setVariablePositionConfig
 - arcs::aubo_scope::WaypointNode, [655](#), [656](#)
- setVoltage
 - arcs::aubo_scope::SetNode, [524](#)
- setWaitTime
 - arcs::aubo_scope::WaitNode, [634](#)
- setWaypointBlendParameter
 - arcs::aubo_scope::UntilNode, [590](#)
- SHARED
 - arcs::aubo_scope::DirectionNode, [184](#)
- SHARED_BLEND
 - arcs::aubo_scope::WaypointNode, [647](#)
- SHARED_DECELERATION
 - arcs::aubo_scope::UntilNode, [583](#)
- SHARED_MOTION
 - arcs::aubo_scope::DirectionNode, [184](#)
 - arcs::aubo_scope::WaypointNode, [647](#)
- ShowOptions
 - arcs::aubo_scope::RenderInterface, [464](#)
- SIMPLE
 - arcs::aubo_scope::ForceNode, [252](#)
- SINGLE_DIGITAL_PULSE
 - arcs::aubo_scope::SetNode, [519](#)
- sleep
 - arcs::aubo_scope::ScriptWriter, [511](#)
- STANDARD
 - arcs::aubo_scope::Io, [326](#)
- StringLengthValidator
 - arcs::aubo_scope::StringLengthValidator, [532](#)
- SubProgNode
 - arcs::aubo_scope::SubProgNode, [536](#), [537](#)
- SwitchNode
 - arcs::aubo_scope::SwitchNode, [539](#), [540](#)
- sync
 - arcs::aubo_scope::ScriptWriter, [511](#)
- SystemApi, [43](#)
 - arcs::aubo_scope::SystemApi, [543](#), [544](#)
 - getBugfixVersion, [45](#)
 - getBuildNumber, [45](#)
 - getConfigurableDigitalInputConfigure, [45](#)
 - getConfigurableDigitalOutputConfigure, [45](#)
 - getENCConfigure, [45](#)
 - getHomeDefaultPosition, [45](#)
 - getHomePackagePosition, [46](#)
 - getJointDefaultAcceleration, [46](#)
 - getJointDefaultVelocity, [46](#)
 - getJointMaxAcceleration, [46](#)
 - getJointMaxPositions, [46](#)

- getJointMaxVelocity, [47](#)
- getJointMinPositions, [47](#)
- getLocale, [47](#)
- getLocaleForProgrammingLanguage, [47](#)
- getMajorVersion, [47](#)
- getMaxPayloadDistance, [48](#)
- getMaxPayloadMass, [48](#)
- getMinorVersion, [48](#)
- getRobotDhParams, [48](#)
- getSerialNumber, [49](#)
- getStandardAnalogInputConfigure, [49](#)
- getStandardAnalogOutputConfigure, [49](#)
- getStandardDigitalInputConfigure, [49](#)
- getStandardDigitalOutputConfigure, [49](#)
- getStaticLinkInputConfigure, [50](#)
- getStaticLinkOutputConfigure, [50](#)
- getTcpDefaultAcceleration, [50](#)
- getTcpDefaultVelocity, [50](#)
- getTcpMaxAcceleration, [50](#)
- getTcpMaxDistance, [51](#)
- getTcpMaxVelocity, [51](#)
- getToolAnalogInputConfigure, [51](#)
- getToolAnalogOutputConfigure, [51](#)
- getToolDigitalIOConfigure, [51](#)
- getUnitType, [52](#)
- isRealRobot, [52](#)
- TargetRobot
 - arcs::aubo_scope::RenderInterface, [464](#)
- TCP
 - arcs::aubo_scope::TCP, [546](#), [547](#)
- TcpContributionModel
 - arcs::aubo_scope::TcpContributionModel, [549](#), [550](#)
- TcpModel
 - arcs::aubo_scope::TcpModel, [554](#), [555](#)
- TcpSelectionType
 - arcs::aubo_scope::MoveNode, [358](#)
- TEACHING
 - arcs::aubo_scope::WaypointNode, [648](#)
- ThreadNode
 - arcs::aubo_scope::ThreadNode, [559](#), [560](#)
- TIGHTEN
 - arcs::aubo_scope, [82](#)
- TIME
 - arcs::aubo_scope::WaitNode, [627](#)
- TIME_MOTION
 - arcs::aubo_scope::WaypointNode, [647](#)
- TimerNode
 - arcs::aubo_scope::TimerNode, [562](#), [563](#)
- TOOL
 - arcs::aubo_scope::Io, [326](#)
- TOOL_CONTACT
 - arcs::aubo_scope::UntilNode, [582](#)
- TOOL_MOTION
 - arcs::aubo_scope::DirectionNode, [184](#)
 - arcs::aubo_scope::WaypointNode, [647](#)
- ToolPosition
 - arcs::aubo_scope::RenderInterface, [465](#)
- toString
 - arcs::aubo_scope::Expression, [212](#)
 - arcs::aubo_scope::Variable, [608](#)
- Track
 - arcs::aubo_scope::RenderInterface, [464](#)
- traverse
 - arcs::aubo_scope::TreeNode, [572](#)
- TreeNode
 - arcs::aubo_scope::TreeNode, [566](#), [567](#)
- Type
 - arcs::aubo_scope::PopupNode, [397](#)
 - arcs::aubo_scope::Variable, [606](#)
- type_
 - arcs::aubo_scope::EtherNetIPFilter, [209](#)
 - arcs::aubo_scope::ModbusIoFilter, [355](#)
 - arcs::aubo_scope::ProfinetFilter, [407](#)
- types
 - arcs::aubo_scope::IoConfigure, [332](#)
- UIntegerRangeValidator
 - arcs::aubo_scope::UIntegerRangeValidator, [574](#)
- UNDEFINED
 - arcs::aubo_scope::GripperNode, [289](#)
 - arcs::aubo_scope::WaypointNode, [648](#), [649](#)
- UNDEFINED0
 - arcs::aubo_scope::GripperNode, [289](#)
- UndoRedoManager
 - arcs::aubo_scope::UndoRedoManager, [577](#), [578](#)
- UnitType
 - arcs::aubo_scope, [82](#)
- Unknown
 - arcs::aubo_scope::UntilNode, [582](#)
- UNRESOLVED_FEATURE
 - arcs::aubo_scope::DirectionNode, [184](#)
 - arcs::aubo_scope::MoveNode, [358](#)
- UNRESOLVED_OUTPUT
 - arcs::aubo_scope::SetNode, [519](#)
- UntilNode
 - arcs::aubo_scope::UntilNode, [584](#), [585](#)
- updateCapability
 - arcs::aubo_scope::GripForceCapability, [273](#)
 - arcs::aubo_scope::GripVacuumCapability, [293](#)
 - arcs::aubo_scope::SpeedCapability, [530](#)
 - arcs::aubo_scope::WidthCapability, [657](#)
- updateFeature
 - arcs::aubo_scope::FeatureContributionModel, [221](#)
- updatePayload
 - arcs::aubo_scope::PayloadContributionModel, [382](#)
- updateTcp
 - arcs::aubo_scope::TcpContributionModel, [552](#)
- updateValue
 - arcs::aubo_scope::DoubleRangeValidator, [191](#)
 - arcs::aubo_scope::IntegerRangeValidator, [321](#)
 - arcs::aubo_scope::StringLengthValidator, [533](#)

- arcs::aubo_scope::UIntegerRangeValidator, 575
- USE_ACTIVE_TCP
 - arcs::aubo_scope::MoveNode, 358
- UserInteraction
 - arcs::aubo_scope::UserInteraction, 592, 593
- UserInterfaceApi, 52
 - arcs::aubo_scope::UserInterfaceApi, 600, 601
 - getUserInteraction, 53
- VALUE_PERSISTED
 - arcs::aubo_scope::Variable, 606
- ValueFactory
 - arcs::aubo_scope::ValueFactory, 602, 603
- VARIABLE
 - arcs::aubo_scope::WaypointNode, 648
- Variable
 - arcs::aubo_scope::Variable, 607, 608
- VARIABLE_JOINT
 - arcs::aubo_scope::WaypointNode, 649
- VARIABLE_POSE
 - arcs::aubo_scope::WaypointNode, 649
- VARIABLE_POSITION
 - arcs::aubo_scope::WaypointNode, 647
- VariableDefinition
 - arcs::aubo_scope::WaypointNode, 648
- VariableFactory
 - arcs::aubo_scope::VariableFactory, 610, 611
- VariableModel
 - arcs::aubo_scope::VariableModel, 614, 615
- VariableType
 - arcs::aubo_scope::WaypointNode, 649
- View
 - arcs::aubo_scope::RenderInterface, 464
- ViewApiProvider
 - arcs::aubo_scope::ViewApiProvider, 619
- VisualLocalizationProgramNode
 - arcs::aubo_scope::VisualLocalizationProgramNode, 623, 624
- WaitNode
 - arcs::aubo_scope::WaitNode, 628, 629
- WARNING
 - arcs::aubo_scope::PopupNode, 397
- Waypoint
 - arcs::aubo_scope::Waypoint, 636, 637
- WAYPOINT_BLEND
 - arcs::aubo_scope::WaypointNode, 647
- WaypointModel
 - arcs::aubo_scope::WaypointModel, 640, 641
- WaypointNode
 - arcs::aubo_scope::WaypointNode, 649, 650
- whileCondition
 - arcs::aubo_scope::ScriptWriter, 511
- whileNot
 - arcs::aubo_scope::ScriptWriter, 512
- whileTrue
 - arcs::aubo_scope::ScriptWriter, 512
- WidgetHandle
 - arcs::aubo_scope, 81
 - installation_node_view.h, 720
- writeChildren
 - arcs::aubo_scope::ScriptWriter, 512
- X_MINUS
 - arcs::aubo_scope::DirectionNode, 183
- X_PLUS
 - arcs::aubo_scope::DirectionNode, 183
- Y_MINUS
 - arcs::aubo_scope::DirectionNode, 183
- Y_PLUS
 - arcs::aubo_scope::DirectionNode, 183
- Z_MINUS
 - arcs::aubo_scope::DirectionNode, 183
- Z_PLUS
 - arcs::aubo_scope::DirectionNode, 183