

MVCommon

Generated by Doxygen 1.8.16



<b>1 Mantis Vision: MVCommon</b>	<b>1</b>
<b>2 Release Notes</b>	<b>3</b>
<b>3 Hierarchical Index</b>	<b>7</b>
3.1 Class Hierarchy . . . . .	7
<b>4 Data Structure Index</b>	<b>9</b>
4.1 Data Structures . . . . .	9
<b>5 File Index</b>	<b>13</b>
5.1 File List . . . . .	13
<b>6 Data Structure Documentation</b>	<b>15</b>
6.1 MVCommon::AndroidSystemLoggerSink Class Reference . . . . .	15
6.1.1 Detailed Description . . . . .	15
6.1.2 Constructor & Destructor Documentation . . . . .	15
6.1.2.1 AndroidSystemLoggerSink() . . . . .	15
6.1.3 Member Function Documentation . . . . .	16
6.1.3.1 HandleLogEntry() . . . . .	16
6.2 MVCommon::AppleSystemLoggerSink Class Reference . . . . .	16
6.2.1 Detailed Description . . . . .	17
6.2.2 Constructor & Destructor Documentation . . . . .	17
6.2.2.1 AppleSystemLoggerSink() . . . . .	17
6.2.3 Member Function Documentation . . . . .	17
6.2.3.1 HandleLogEntry() . . . . .	17
6.3 MVCommon::BlockingCounter Class Reference . . . . .	18
6.3.1 Detailed Description . . . . .	18
6.3.2 Constructor & Destructor Documentation . . . . .	18
6.3.2.1 BlockingCounter() . . . . .	18
6.3.3 Member Function Documentation . . . . .	19
6.3.3.1 Increment() . . . . .	19
6.3.3.2 operator+=(()) . . . . .	19
6.3.3.3 Value() . . . . .	19
6.3.3.4 WaitUntil() . . . . .	20
6.3.3.5 WaitUntilFor() . . . . .	20
6.3.3.6 WaitUntilValue() . . . . .	20
6.3.3.7 WaitUntilValueFor() . . . . .	21
6.4 MVCommon::BlockingCounterValueEquals Class Reference . . . . .	21
6.4.1 Detailed Description . . . . .	22
6.4.2 Constructor & Destructor Documentation . . . . .	22
6.4.2.1 BlockingCounterValueEquals() . . . . .	22
6.4.3 Member Function Documentation . . . . .	22
6.4.3.1 operator()(()) . . . . .	22

6.5 MVCommon::ByteArray Class Reference	23
6.5.1 Detailed Description	24
6.5.2 Constructor & Destructor Documentation	24
6.5.2.1 ByteArray() [1/5]	24
6.5.2.2 ByteArray() [2/5]	24
6.5.2.3 ByteArray() [3/5]	24
6.5.2.4 ByteArray() [4/5]	25
6.5.2.5 ByteArray() [5/5]	25
6.5.3 Member Function Documentation	25
6.5.3.1 Data()	25
6.5.3.2 operator<=<=() [1/2]	25
6.5.3.3 operator<=<=() [2/2]	26
6.5.3.4 operator=()	26
6.5.3.5 operator>>=()	27
6.5.3.6 operator[]() [1/2]	27
6.5.3.7 operator[]() [2/2]	27
6.5.3.8 Pop() [1/2]	28
6.5.3.9 Pop() [2/2]	28
6.5.3.10 Push() [1/3]	28
6.5.3.11 Push() [2/3]	29
6.5.3.12 Push() [3/3]	29
6.5.3.13 Size()	30
6.5.3.14 Skip()	30
6.5.3.15 Subarray()	30
6.6 MVCommon::ByteArrayHasher Struct Reference	31
6.6.1 Detailed Description	31
6.6.2 Member Function Documentation	31
6.6.2.1 operator()()	31
6.7 MVCommon::CameraParams Struct Reference	31
6.7.1 Detailed Description	33
6.7.2 Constructor & Destructor Documentation	33
6.7.2.1 CameraParams()	33
6.7.3 Member Function Documentation	33
6.7.3.1 DenormalizePoint() [1/2]	33
6.7.3.2 DenormalizePoint() [2/2]	34
6.7.3.3 FromRawBytes()	34
6.7.3.4 NormalizePoint() [1/2]	34
6.7.3.5 NormalizePoint() [2/2]	35
6.7.3.6 ScaleToResolution()	35
6.7.3.7 ToRawBytes()	35
6.7.3.8 ToString()	36
6.7.3.9 UndistortPoint() [1/2]	36

6.7.3.10 UndistortPoint() [2/2]	36
6.7.4 Field Documentation	36
6.7.4.1 distortionC	36
6.8 MVCommon::CameraParamsHasher Struct Reference	37
6.8.1 Detailed Description	37
6.8.2 Member Function Documentation	37
6.8.2.1 operator()	37
6.9 MVCommon::Color Struct Reference	37
6.9.1 Detailed Description	39
6.9.2 Constructor & Destructor Documentation	39
6.9.2.1 Color() [1/3]	39
6.9.2.2 Color() [2/3]	39
6.9.2.3 Color() [3/3]	40
6.9.3 Member Function Documentation	40
6.9.3.1 FromString()	40
6.9.3.2 GetAlpha()	41
6.9.3.3 GetAlphaByte()	41
6.9.3.4 GetBlue()	41
6.9.3.5 GetBlueByte()	41
6.9.3.6 GetGreen()	42
6.9.3.7 GetGreenByte()	42
6.9.3.8 GetRed()	42
6.9.3.9 GetRedByte()	42
6.9.3.10 GetRGBBrightness()	43
6.9.3.11 GetRGBBrightnessByte()	43
6.9.3.12 SetAlpha()	43
6.9.3.13 SetAlphaByte()	43
6.9.3.14 SetBlue()	44
6.9.3.15 SetBlueByte()	44
6.9.3.16 SetGreen()	44
6.9.3.17 SetGreenByte()	44
6.9.3.18 SetRed()	45
6.9.3.19 SetRedByte()	45
6.9.3.20 SetValue() [1/3]	45
6.9.3.21 SetValue() [2/3]	46
6.9.3.22 SetValue() [3/3]	46
6.9.3.23 ToRGB_HTMLString()	46
6.9.3.24 ToString()	47
6.10 MVCommon::ColorHasher Struct Reference	47
6.10.1 Detailed Description	47
6.10.2 Member Function Documentation	47
6.10.2.1 operator()	47

6.11 MVCommon::FileLoggerSink Class Reference	48
6.11.1 Detailed Description	48
6.11.2 Constructor & Destructor Documentation	48
6.11.2.1 FileLoggerSink()	48
6.11.3 Member Function Documentation	49
6.11.3.1 HandleLogEntry()	49
6.12 MVCommon::Guid Struct Reference	49
6.12.1 Detailed Description	50
6.12.2 Constructor & Destructor Documentation	50
6.12.2.1 Guid() [1/2]	50
6.12.2.2 Guid() [2/2]	50
6.12.3 Member Function Documentation	51
6.12.3.1 FromHexString()	51
6.12.3.2 FromRawBytes()	51
6.12.3.3 FromRfc4122()	52
6.12.3.4 IsNil()	52
6.12.3.5 Nil()	53
6.12.3.6 ToHexString()	53
6.12.3.7 ToRawBytes()	53
6.12.3.8 ToRfc4122()	53
6.13 MVCommon::GuidAliasDatabase Class Reference	54
6.13.1 Detailed Description	55
6.13.2 Constructor & Destructor Documentation	55
6.13.2.1 GuidAliasDatabase() [1/2]	55
6.13.2.2 GuidAliasDatabase() [2/2]	55
6.13.3 Member Function Documentation	55
6.13.3.1 AliasRegistered()	55
6.13.3.2 Begin()	56
6.13.3.3 End()	56
6.13.3.4 GetGuidAlias()	56
6.13.3.5 GetGuidWithAlias()	57
6.13.3.6 GuidRegistered()	57
6.13.3.7 RegisterGuidAlias()	57
6.13.3.8 TryGetGuidAlias()	58
6.13.3.9 TryGetGuidWithAlias()	58
6.13.3.10 UnregisterGuidAlias() [1/2]	59
6.13.3.11 UnregisterGuidAlias() [2/2]	59
6.14 MVCommon::GuidAliasDatabaseIterator Class Reference	59
6.14.1 Detailed Description	60
6.14.2 Constructor & Destructor Documentation	60
6.14.2.1 GuidAliasDatabaseIterator() [1/2]	60
6.14.2.2 GuidAliasDatabaseIterator() [2/2]	60

6.14.3 Member Function Documentation	60
6.14.3.1 operator*()	61
6.14.3.2 operator++() [1/2]	61
6.14.3.3 operator++() [2/2]	61
6.15 MVCommon::GuidHasher Struct Reference	61
6.15.1 Detailed Description	62
6.15.2 Member Function Documentation	62
6.15.2.1 operator()()	62
6.16 MVCommon::IBlockingCounterCondition Class Reference	62
6.16.1 Detailed Description	63
6.16.2 Member Function Documentation	63
6.16.2.1 operator()()	63
6.17 MVCommon::ILoggerSink Class Reference	63
6.17.1 Detailed Description	64
6.17.2 Constructor & Destructor Documentation	64
6.17.2.1 ILoggerSink()	64
6.17.3 Member Function Documentation	64
6.17.3.1 AddLogEntry()	64
6.17.3.2 GetLogLevel()	65
6.17.3.3 HandleLogEntry()	65
6.17.3.4 LogLevelToString()	65
6.17.3.5 SetLogLevel()	66
6.17.3.6 TimestampToString()	66
6.18 MVCommon::IThreadPoolJob Class Reference	66
6.18.1 Detailed Description	67
6.18.2 Member Function Documentation	67
6.18.2.1 operator()()	67
6.19 MVCommon::LogEntry Struct Reference	67
6.19.1 Detailed Description	68
6.19.2 Constructor & Destructor Documentation	68
6.19.2.1 LogEntry()	68
6.19.3 Member Function Documentation	68
6.19.3.1 GetLevel()	69
6.19.3.2 GetMessage()	69
6.19.3.3 GetTag()	69
6.19.3.4 GetThreadID()	69
6.19.3.5 GetTimestamp()	70
6.20 MVCommon::Logger Class Reference	70
6.20.1 Detailed Description	71
6.20.2 Constructor & Destructor Documentation	71
6.20.2.1 Logger()	71
6.20.3 Member Function Documentation	71

6.20.3.1 AddLoggerSink()	71
6.20.3.2 GetLogLevel()	71
6.20.3.3 LogMessage() [1/2]	72
6.20.3.4 LogMessage() [2/2]	72
6.20.3.5 RemoveLoggerSink()	73
6.20.3.6 SetLogLevel()	73
6.21 MVCommon::LoggerRegistry Class Reference	73
6.21.1 Detailed Description	74
6.21.2 Member Function Documentation	74
6.21.2.1 GetInstance()	74
6.21.2.2 GetLogger()	74
6.21.2.3 RegisterLogger()	75
6.21.2.4 UnregisterLogger()	75
6.22 MVCommon::Math Class Reference	75
6.22.1 Detailed Description	76
6.22.2 Member Function Documentation	76
6.22.2.1 AlmostEqual() [1/2]	76
6.22.2.2 AlmostEqual() [2/2]	77
6.22.2.3 Clamp()	77
6.23 MVCommon::Matrix4x4d Struct Reference	78
6.23.1 Detailed Description	79
6.23.2 Constructor & Destructor Documentation	79
6.23.2.1 Matrix4x4d() [1/2]	79
6.23.2.2 Matrix4x4d() [2/2]	80
6.23.3 Member Function Documentation	80
6.23.3.1 CreateLookAt()	80
6.23.3.2 CreateOrtographic()	82
6.23.3.3 CreatePerspective()	82
6.23.3.4 CreateRotationAroundAxis()	83
6.23.3.5 CreateRotationFromEulerAnglesZYX()	83
6.23.3.6 CreateRotationFromVersor()	84
6.23.3.7 CreateScale()	84
6.23.3.8 CreateTranslation()	84
6.23.3.9 CreateZero()	85
6.23.3.10 FromRawBytes()	85
6.23.3.11 FromRawElements()	85
6.23.3.12 FromString()	86
6.23.3.13 Inverted()	86
6.23.3.14 operator[]() [1/2]	86
6.23.3.15 operator[]() [2/2]	87
6.23.3.16 RotationTranslationMatrixInverted()	87
6.23.3.17 ToRawBytes()	88



6.23.3.18 ToRawElements()	88
6.23.3.19 ToString()	88
6.23.3.20 Transposed()	88
6.24 MVCommon::Matrix4x4dHasher Struct Reference	89
6.24.1 Detailed Description	89
6.24.2 Member Function Documentation	89
6.24.2.1 operator()()	89
6.25 MVCommon::Matrix4x4f Struct Reference	89
6.25.1 Detailed Description	91
6.25.2 Constructor & Destructor Documentation	91
6.25.2.1 Matrix4x4f() [1/2]	91
6.25.2.2 Matrix4x4f() [2/2]	92
6.25.3 Member Function Documentation	92
6.25.3.1 CreateLookAt()	92
6.25.3.2 CreateOrtographic()	93
6.25.3.3 CreatePerspective()	93
6.25.3.4 CreateRotationAroundAxis()	94
6.25.3.5 CreateRotationFromEulerAnglesZYX()	94
6.25.3.6 CreateRotationFromVersor()	94
6.25.3.7 CreateScale()	95
6.25.3.8 CreateTranslation()	95
6.25.3.9 CreateZero()	95
6.25.3.10 FromRawBytes()	96
6.25.3.11 FromRawElements()	96
6.25.3.12 FromString()	97
6.25.3.13 Inverted()	97
6.25.3.14 operator[]() [1/2]	97
6.25.3.15 operator[]() [2/2]	98
6.25.3.16 RotationTranslationMatrixInverted()	98
6.25.3.17 ToRawBytes()	98
6.25.3.18 ToRawElements()	99
6.25.3.19 ToString()	99
6.25.3.20 Transposed()	99
6.26 MVCommon::Matrix4x4fHasher Struct Reference	100
6.26.1 Detailed Description	100
6.26.2 Member Function Documentation	100
6.26.2.1 operator()()	100
6.27 MVCommon::Pair< TFirst, TSecond > Class Template Reference	100
6.27.1 Detailed Description	101
6.27.2 Constructor & Destructor Documentation	101
6.27.2.1 Pair()	101
6.28 MVCommon::RedirectingLoggerSink Class Reference	102

6.28.1 Detailed Description	102
6.28.2 Constructor & Destructor Documentation	102
6.28.2.1 RedirectingLoggerSink()	102
6.28.3 Member Function Documentation	103
6.28.3.1 HandleLogEntry()	103
6.29 MVCommon::SharedGuidAliasDatabasePtr Class Reference	103
6.29.1 Detailed Description	104
6.29.2 Constructor & Destructor Documentation	104
6.29.2.1 SharedGuidAliasDatabasePtr() [1/3]	104
6.29.2.2 SharedGuidAliasDatabasePtr() [2/3]	104
6.29.2.3 SharedGuidAliasDatabasePtr() [3/3]	104
6.29.2.4 ~SharedGuidAliasDatabasePtr()	106
6.29.3 Member Function Documentation	106
6.29.3.1 Get()	106
6.29.3.2 operator bool()	106
6.29.3.3 operator*()	107
6.29.3.4 operator->()	107
6.29.3.5 operator=() [1/2]	107
6.29.3.6 operator=() [2/2]	107
6.30 MVCommon::SharedLoggerPtr Class Reference	108
6.30.1 Detailed Description	108
6.30.2 Constructor & Destructor Documentation	109
6.30.2.1 SharedLoggerPtr() [1/3]	109
6.30.2.2 SharedLoggerPtr() [2/3]	109
6.30.2.3 SharedLoggerPtr() [3/3]	109
6.30.2.4 ~SharedLoggerPtr()	109
6.30.3 Member Function Documentation	110
6.30.3.1 Get()	110
6.30.3.2 operator bool()	110
6.30.3.3 operator*()	110
6.30.3.4 operator->()	110
6.30.3.5 operator=() [1/2]	110
6.30.3.6 operator=() [2/2]	111
6.31 MVCommon::SharedLoggerSinkPtr Class Reference	111
6.31.1 Detailed Description	112
6.31.2 Constructor & Destructor Documentation	112
6.31.2.1 SharedLoggerSinkPtr() [1/3]	112
6.31.2.2 SharedLoggerSinkPtr() [2/3]	112
6.31.2.3 SharedLoggerSinkPtr() [3/3]	113
6.31.2.4 ~SharedLoggerSinkPtr()	113
6.31.3 Member Function Documentation	113
6.31.3.1 Get()	113

6.31.3.2 operator bool()	113
6.31.3.3 operator*()	114
6.31.3.4 operator->()	114
6.31.3.5 operator=() [1/2]	114
6.31.3.6 operator=() [2/2]	114
6.32 MVCommon::SharedThreadPoolJobPtr Class Reference	115
6.32.1 Detailed Description	115
6.32.2 Constructor & Destructor Documentation	116
6.32.2.1 SharedThreadPoolJobPtr() [1/3]	116
6.32.2.2 SharedThreadPoolJobPtr() [2/3]	116
6.32.2.3 SharedThreadPoolJobPtr() [3/3]	116
6.32.2.4 ~SharedThreadPoolJobPtr()	116
6.32.3 Member Function Documentation	117
6.32.3.1 Get()	117
6.32.3.2 operator bool()	117
6.32.3.3 operator*()	117
6.32.3.4 operator->()	117
6.32.3.5 operator=() [1/2]	117
6.32.3.6 operator=() [2/2]	118
6.33 MVCommon::StdOutLoggerSink Class Reference	118
6.33.1 Detailed Description	119
6.33.2 Constructor & Destructor Documentation	119
6.33.2.1 StdOutLoggerSink()	119
6.33.3 Member Function Documentation	119
6.33.3.1 HandleLogEntry()	119
6.34 MVCommon::String Class Reference	120
6.34.1 Detailed Description	120
6.34.2 Constructor & Destructor Documentation	120
6.34.2.1 String() [1/3]	120
6.34.2.2 String() [2/3]	121
6.34.2.3 String() [3/3]	121
6.34.3 Member Function Documentation	121
6.34.3.1 CStr()	121
6.34.3.2 Length()	122
6.34.3.3 Substr()	122
6.35 MVCommon::StringHasher Struct Reference	122
6.35.1 Detailed Description	122
6.35.2 Member Function Documentation	123
6.35.2.1 operator>()	123
6.36 MVCommon::ThreadPool Class Reference	123
6.36.1 Detailed Description	124
6.36.2 Constructor & Destructor Documentation	124

6.36.2.1 ThreadPool()	124
6.36.2.2 ~ThreadPool()	124
6.36.3 Member Function Documentation	124
6.36.3.1 DoJob()	124
6.36.3.2 GetThreadsCount()	125
6.36.3.3 GetUnoccupiedThreadsCount()	125
6.36.3.4 HasUnoccupiedThreads()	125
6.36.3.5 ResetJobs()	126
6.37 MVCommon::Vector2d Struct Reference	126
6.37.1 Detailed Description	127
6.37.2 Constructor & Destructor Documentation	127
6.37.2.1 Vector2d()	127
6.37.3 Member Function Documentation	127
6.37.3.1 Abs()	127
6.37.3.2 Dot()	128
6.37.3.3 FromRawBytes()	128
6.37.3.4 FromString()	128
6.37.3.5 Inverted()	129
6.37.3.6 Length()	129
6.37.3.7 Normalized()	129
6.37.3.8 operator[]() [1/2]	129
6.37.3.9 operator[]() [2/2]	130
6.37.3.10 ToRawBytes()	130
6.37.3.11 ToString()	131
6.38 MVCommon::Vector2dHasher Struct Reference	131
6.38.1 Detailed Description	131
6.38.2 Member Function Documentation	131
6.38.2.1 operator()()	131
6.39 MVCommon::Vector2f Struct Reference	132
6.39.1 Detailed Description	133
6.39.2 Constructor & Destructor Documentation	133
6.39.2.1 Vector2f()	133
6.39.3 Member Function Documentation	133
6.39.3.1 Abs()	133
6.39.3.2 Dot()	134
6.39.3.3 FromRawBytes()	134
6.39.3.4 FromString()	134
6.39.3.5 Inverted()	135
6.39.3.6 Length()	135
6.39.3.7 Normalized()	135
6.39.3.8 operator[]() [1/2]	135
6.39.3.9 operator[]() [2/2]	136

6.39.3.10 ToRawBytes()	136
6.39.3.11 ToString()	137
6.40 MVCommon::Vector2fHasher Struct Reference	137
6.40.1 Detailed Description	137
6.40.2 Member Function Documentation	137
6.40.2.1 operator>()	137
6.41 MVCommon::Vector3d Struct Reference	138
6.41.1 Detailed Description	139
6.41.2 Constructor & Destructor Documentation	139
6.41.2.1 Vector3d() [1/2]	139
6.41.2.2 Vector3d() [2/2]	139
6.41.3 Member Function Documentation	140
6.41.3.1 Abs()	140
6.41.3.2 Cross()	140
6.41.3.3 Dot()	140
6.41.3.4 FromRawBytes()	141
6.41.3.5 FromString()	141
6.41.3.6 GetXY()	142
6.41.3.7 Inverted()	142
6.41.3.8 Length()	142
6.41.3.9 Normalized()	142
6.41.3.10 operator[]() [1/2]	142
6.41.3.11 operator[]() [2/2]	143
6.41.3.12 ToRawBytes()	143
6.41.3.13 ToString()	144
6.42 MVCommon::Vector3dHasher Struct Reference	144
6.42.1 Detailed Description	144
6.42.2 Member Function Documentation	144
6.42.2.1 operator>()	144
6.43 MVCommon::Vector3f Struct Reference	145
6.43.1 Detailed Description	146
6.43.2 Constructor & Destructor Documentation	146
6.43.2.1 Vector3f() [1/2]	146
6.43.2.2 Vector3f() [2/2]	146
6.43.3 Member Function Documentation	147
6.43.3.1 Abs()	147
6.43.3.2 Cross()	147
6.43.3.3 Dot()	147
6.43.3.4 FromRawBytes()	148
6.43.3.5 FromString()	148
6.43.3.6 GetXY()	149
6.43.3.7 Inverted()	149

6.43.3.8 Length()	149
6.43.3.9 Normalized()	149
6.43.3.10 operator[]() [1/2]	149
6.43.3.11 operator[]() [2/2]	150
6.43.3.12 ToRawBytes()	150
6.43.3.13 ToString()	151
6.44 MVCommon::Vector3f Hasher Struct Reference	151
6.44.1 Detailed Description	151
6.44.2 Member Function Documentation	151
6.44.2.1 operator()()	151
6.45 MVCommon::Vector4d Struct Reference	152
6.45.1 Detailed Description	153
6.45.2 Constructor & Destructor Documentation	153
6.45.2.1 Vector4d() [1/2]	153
6.45.2.2 Vector4d() [2/2]	153
6.45.3 Member Function Documentation	154
6.45.3.1 Abs()	154
6.45.3.2 Dot()	154
6.45.3.3 FromRawBytes()	154
6.45.3.4 FromString()	155
6.45.3.5 GetXYZ()	155
6.45.3.6 Inverted()	156
6.45.3.7 Length()	156
6.45.3.8 Normalized()	156
6.45.3.9 operator[]() [1/2]	156
6.45.3.10 operator[]() [2/2]	157
6.45.3.11 ToRawBytes()	157
6.45.3.12 ToString()	157
6.46 MVCommon::Vector4d Hasher Struct Reference	158
6.46.1 Detailed Description	158
6.46.2 Member Function Documentation	158
6.46.2.1 operator()()	158
6.47 MVCommon::Vector4f Struct Reference	159
6.47.1 Detailed Description	160
6.47.2 Constructor & Destructor Documentation	160
6.47.2.1 Vector4f() [1/2]	160
6.47.2.2 Vector4f() [2/2]	160
6.47.3 Member Function Documentation	161
6.47.3.1 Abs()	161
6.47.3.2 Dot()	161
6.47.3.3 FromRawBytes()	161
6.47.3.4 FromString()	162

6.47.3.5 GetXYZ()	162
6.47.3.6 Inverted()	162
6.47.3.7 Length()	163
6.47.3.8 Normalized()	163
6.47.3.9 operator[]() [1/2]	163
6.47.3.10 operator[]() [2/2]	164
6.47.3.11 ToRawBytes()	164
6.47.3.12 ToString()	164
6.48 MVCommon::Vector4fHasher Struct Reference	165
6.48.1 Detailed Description	165
6.48.2 Member Function Documentation	165
6.48.2.1 operator()()	165
6.49 MVCommon::VersionInfo Struct Reference	165
6.49.1 Detailed Description	166
6.49.2 Constructor & Destructor Documentation	166
6.49.2.1 VersionInfo()	166
6.49.3 Member Function Documentation	166
6.49.3.1 ToString()	167
6.49.4 Field Documentation	167
6.49.4.1 major	167
6.49.4.2 minor	167
6.49.4.3 patch	167
6.50 MVCommon::VersionInfoHasher Struct Reference	167
6.50.1 Detailed Description	168
6.50.2 Member Function Documentation	168
6.50.2.1 operator()()	168
6.51 MVCommon::Versord Struct Reference	168
6.51.1 Detailed Description	169
6.51.2 Member Function Documentation	169
6.51.2.1 CreateRotationAroundAxis()	169
6.51.2.2 CreateRotationFromEulerAnglesZYZ()	170
6.51.2.3 CreateRotationFromMatrix()	170
6.51.2.4 FromElementsVector()	170
6.51.2.5 FromRawBytes()	171
6.51.2.6 FromRawElements()	171
6.51.2.7 FromString()	172
6.51.2.8 Inverted()	172
6.51.2.9 ToElementsVector()	173
6.51.2.10 ToEulerAnglesZYZ()	173
6.51.2.11 ToRawBytes()	173
6.51.2.12 ToRawElements()	173
6.51.2.13 ToString()	174

6.52 MVCommon::VersordHasher Struct Reference	174
6.52.1 Detailed Description	174
6.52.2 Member Function Documentation	174
6.52.2.1 operator>()	174
6.53 MVCommon::Versorf Struct Reference	175
6.53.1 Detailed Description	176
6.53.2 Member Function Documentation	176
6.53.2.1 CreateRotationAroundAxis()	176
6.53.2.2 CreateRotationFromEulerAnglesZYX()	176
6.53.2.3 CreateRotationFromMatrix()	177
6.53.2.4 FromElementsVector()	177
6.53.2.5 FromRawBytes()	177
6.53.2.6 FromRawElements()	178
6.53.2.7 FromString()	178
6.53.2.8 Inverted()	179
6.53.2.9 ToElementsVector()	179
6.53.2.10 ToEulerAnglesZYX()	179
6.53.2.11 ToRawBytes()	179
6.53.2.12 ToRawElements()	180
6.53.2.13 ToString()	180
6.54 MVCommon::VersorfHasher Struct Reference	180
6.54.1 Detailed Description	181
6.54.2 Member Function Documentation	181
6.54.2.1 operator>()	181
6.55 MVCommon::WeakLoggerPtr Class Reference	181
6.55.1 Detailed Description	182
6.55.2 Constructor & Destructor Documentation	182
6.55.2.1 WeakLoggerPtr() [1/3]	182
6.55.2.2 WeakLoggerPtr() [2/3]	182
6.55.2.3 WeakLoggerPtr() [3/3]	182
6.55.3 Member Function Documentation	183
6.55.3.1 Expired()	183
6.55.3.2 Lock()	183
6.55.3.3 operator=() [1/2]	183
6.55.3.4 operator=() [2/2]	184
<b>7 File Documentation</b>	<b>185</b>
7.1 public/MVCommon/CUtil.h File Reference	185
7.1.1 Macro Definition Documentation	185
7.1.1.1 MV_VALUE_TO_STR	185
7.2 public/MVCommon/guid/GuidGenerator.h File Reference	185
7.2.1 Function Documentation	186



---

7.2.1.1 GenerateGuid() [1/2]	186
7.2.1.2 GenerateGuid() [2/2]	186
7.3 public/MVCommon/logger/LoggerLogLevel.h File Reference	186
7.3.1 Enumeration Type Documentation	187
7.3.1.1 LoggerLogLevel	187
7.4 public/MVCommon/logger/LogLevel.h File Reference	187
7.4.1 Enumeration Type Documentation	187
7.4.1.1 LogLevel	188
7.5 public/MVCommon/MVCommonVersion.h File Reference	189
7.6 public/MVCommon/Utils/VersionInfo.h File Reference	189
<b>Index</b>	<b>191</b>



# Chapter 1

## Mantis Vision: MVCommon

A collection of common utilities and services.

### Table of Contents

- [Release Notes](#)

### Supported Platforms

Currently the module is ported to these platforms:

- Windows (x64),
- Linux (x64, arm64)
- MacOS (x64),
- Android (armeabi-v7a, arm64-v8a),
- iOS (arm64) and
- LuminOS.



## Chapter 2

# Release Notes

### 1.2.0

Initial version (as extracted from Mvx2 framework)

#### Module

- **1.2.0\_M1** | introduced versioning to MVCommon libraries (in the form of [MVCommon::VersionInfo](#) class and [MVCommonVersion.h](#) file)
- **1.2.0\_M2** | added [CUtil.h](#) file with C preprocessor utility macros

#### Build support

- **1.2.0\_BS1** | added MVCommon's own MVCommonConfig.cmake file for cmake support
- **1.2.0\_BS2** | added MVCommonNet's own MVCommonNetConfig.cmake and MVCommonNet\_iOSConfig.cmake files for cmake support

#### Documentation

- **1.2.0\_D1** | added MVCommon's own 'release notes' section to its documentation
- **1.2.0\_D2** | switched documentation from xml-style comments to doxygen-style comments

### 2.0.0

#### Module

- **2.0.0\_M1** | made default constructor and destructor of `NonAssignable` class protected, as there shall not exist objects of `NonAssignable` class itself
- **2.0.0\_M2** | updated `libjpeg-turbo` 3rdparty dependency to version 2.0.2

## Build support

- **2.0.0\_BS1** | Android and LuminOS libraries size reduced by ~90%
- **2.0.0\_BS2** | android API level raised from 19 to 21
- **2.0.0\_BS3** | Linux and MacOS binaries do not consist of a versioned library file and a version-neutral symlink file anymore - the library file itself has version-neutral name

## 3.0.0

### Module

- **3.0.0\_M1** | introduced a protected `MVCommon::NativeObjectHolder::m_nativeObjectLock` field into the `MVCommon::NativeObjectHolder` instances in `MVCommonNet` to allow its derivatives to lock the held native object during asynchronous operations on them
- **3.0.0\_M2** | fixed a bug of `MVCommon::NetLoggerSink` in `MVCommonNet` which prevented parallel logging via `.Net` sinks
- **3.0.0\_M3** | fixed `MVCommon::BlockingCounter::WaitUntilValue()` and `MVCommon::BlockingCounter::WaitUntilValueFor()` which could miss a target counter value in case the counter's value was repeatedly updated too fast, and thus not ending the waiting
- **3.0.0\_M4** | extended `MVCommon::BlockingCounter` with a support for blocking the execution until an arbitrary condition on the counter's value is passed:
  - introduced interface `MVCommon::IBlockingCounterCondition` and its derivative `MVCommon::BlockingCounterValueEquals`
  - introduced `MVCommon::BlockingCounter::WaitUntil()` and `MVCommon::BlockingCounter::WaitUntilFor()` functions,
  - introduced `MVCommon::NetBlockingCounterCondition` into `MVCommonNet`
- **3.0.0\_M5** | refactored `MVCommon::ThreadPool` :
  - moved and renamed the `MVCommon/legacy/MVCommon/concurrency/ThreadPool.hpp` header file to `MVCommon/utils/threadpool/ThreadPool.h`,
  - replaced the `MVCommon::ThreadPool::Job` signature of jobs by an `MVCommon::IThreadPoolJob` interface,
  - introduced an `MVCommon::ThreadPool::WaitForAnUnoccupiedThread()` function,
  - refactored the implementation to support the new features,
  - introduced a `.Net` version of `MVCommon::ThreadPool` into `MVCommonNet`,
  - added a documentation section for the API
- **3.0.0\_M6** | fixed `MVCommon::FileHelper::OpenFileReadOnly()` and `MVCommon::FileHelper::OpenFileForWriting()` utility functions which prevented concurrent reading from a file that is already open for writing or reading (affects only windows)

## Build support

- **3.0.0\_BS1** | CMake minimal required version increased from 3.9 to 3.14
  - updated `MVCommonConfig.cmake`, `MVCommonNetConfig.cmake` and `MVCommonNet_iOSConfig.cmake` scripts and their dependencies

## 4.0.0

### Module

- **4.0.0\_M1** | upgraded multiple internal dependencies with possible effect on:
  - [MVCommon::GuidAliasDatabase](#)
  - [MVCommon::Logger](#) and [MVCommon::ILoggerSink](#)
  - [MVCommon::FileHelper](#)

### Build support

- **4.0.0\_BS1** | from now on the windows libraries are compiled using msvc compiler version 142 (VS 2019)
- **4.0.0\_BS2** | upgraded `cmake/toolchains/ios.cmake` toolchain file used for building for iOS platform

### Documentation

- **4.0.0\_D1** | introduced PDF documentation as an alternative to the HTML one:
  - `doc/MVCommon.pdf`
  - `doc/MVCommonNet.pdf`





## Chapter 3

# Hierarchical Index

### 3.1 Class Hierarchy

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

MVCommon::ByteArray	23
MVCommon::ByteArrayHasher	31
MVCommon::CameraParams	31
MVCommon::CameraParamsHasher	37
MVCommon::Color	37
MVCommon::ColorHasher	47
MVCommon::Guid	49
MVCommon::GuidAliasDatabase	54
MVCommon::GuidAliasDatabaseIterator	59
MVCommon::GuidHasher	61
MVCommon::IBlockingCounterCondition	62
MVCommon::BlockingCounterValueEquals	21
MVCommon::IThreadPoolJob	66
MVCommon::LogEntry	67
MVCommon::Math	75
MVCommon::Matrix4x4d	78
MVCommon::Matrix4x4dHasher	89
MVCommon::Matrix4x4f	89
MVCommon::Matrix4x4fHasher	100
NonAssignable	
MVCommon::BlockingCounter	18
MVCommon::ILoggerSink	63
MVCommon::AndroidSystemLoggerSink	15
MVCommon::AppleSystemLoggerSink	16
MVCommon::FileLoggerSink	48
MVCommon::RedirectingLoggerSink	102
MVCommon::StdOutLoggerSink	118
MVCommon::Logger	70
MVCommon::LoggerRegistry	73
MVCommon::ThreadPool	123
MVCommon::Pair< TFirst, TSecond >	100
MVCommon::SharedGuidAliasDatabasePtr	103
MVCommon::SharedLoggerPtr	108
MVCommon::SharedLoggerSinkPtr	111
MVCommon::SharedThreadPoolJobPtr	115

MVCommon::String . . . . .	120
MVCommon::StringHasher . . . . .	122
MVCommon::Vector2d . . . . .	126
MVCommon::Vector2dHasher . . . . .	131
MVCommon::Vector2f . . . . .	132
MVCommon::Vector2fHasher . . . . .	137
MVCommon::Vector3d . . . . .	138
MVCommon::Vector3dHasher . . . . .	144
MVCommon::Vector3f . . . . .	145
MVCommon::Vector3fHasher . . . . .	151
MVCommon::Vector4d . . . . .	152
MVCommon::Vector4dHasher . . . . .	158
MVCommon::Vector4f . . . . .	159
MVCommon::Vector4fHasher . . . . .	165
MVCommon::VersionInfo . . . . .	165
MVCommon::VersionInfoHasher . . . . .	167
MVCommon::Versord . . . . .	168
MVCommon::VersordHasher . . . . .	174
MVCommon::Versorf . . . . .	175
MVCommon::VersorfHasher . . . . .	180
MVCommon::WeakLoggerPtr . . . . .	181

## Chapter 4

# Data Structure Index

### 4.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">MVCommon::AndroidSystemLoggerSink</a>	
A logger sink implementation for logging log messages via Android system logging facility . . .	15
<a href="#">MVCommon::AppleSystemLoggerSink</a>	
A logger sink implementation for logging log messages via Apple system logging facility . . .	16
<a href="#">MVCommon::BlockingCounter</a>	
A counter with a feature of blocking a thread until the counter has a specific value . . .	18
<a href="#">MVCommon::BlockingCounterValueEquals</a>	
A counter condition for checking equality of its value with a target value . . .	21
<a href="#">MVCommon::ByteArray</a>	
An array of bytes . . .	23
<a href="#">MVCommon::ByteArrayHasher</a>	
A hasher for <a href="#">ByteArray</a> objects so they can be used in unordered collections . . .	31
<a href="#">MVCommon::CameraParams</a>	
A data structure containing intrinsic and extrinsic parameters of cameras . . .	31
<a href="#">MVCommon::CameraParamsHasher</a>	
A hasher for <a href="#">CameraParams</a> objects so they can be used in unordered collections . . .	37
<a href="#">MVCommon::Color</a>	
An RGBA color . . .	37
<a href="#">MVCommon::ColorHasher</a>	
A hasher for <a href="#">Color</a> objects so they can be used in unordered collections . . .	47
<a href="#">MVCommon::FileLoggerSink</a>	
A logger sink implementation for logging into a file . . .	48
<a href="#">MVCommon::Guid</a>	
A globally-unique identifier implementation . . .	49
<a href="#">MVCommon::GuidAliasDatabase</a>	
A database of <a href="#">Guid</a> aliases . . .	54
<a href="#">MVCommon::GuidAliasDatabaseIterator</a>	
An iterator over elements of <a href="#">GuidAliasDatabase</a> collections . . .	59
<a href="#">MVCommon::GuidHasher</a>	
A hasher for <a href="#">Guid</a> objects so they can be used in unordered collections . . .	61
<a href="#">MVCommon::IBlockingCounterCondition</a>	
An interface of conditions usable with blocking counters . . .	62
<a href="#">MVCommon::ILoggerSink</a>	
An interface of logger sinks . . .	63
<a href="#">MVCommon::IThreadPoolJob</a>	
An interface of thread pool jobs . . .	66

<a href="#">MVCommon::LogEntry</a>	
A log entry data structure . . . . .	67
<a href="#">MVCommon::Logger</a>	
A logger . . . . .	70
<a href="#">MVCommon::LoggerRegistry</a>	
A global registry of loggers . . . . .	73
<a href="#">MVCommon::Math</a>	
A utility class for math operations . . . . .	75
<a href="#">MVCommon::Matrix4x4d</a>	
A 4x4 matrix with double-precision floating-point values . . . . .	78
<a href="#">MVCommon::Matrix4x4dHasher</a>	
A hasher for <a href="#">Matrix4x4d</a> objects so they can be used in unordered collections . . . . .	89
<a href="#">MVCommon::Matrix4x4f</a>	
A 4x4 matrix with single-precision floating-point values . . . . .	89
<a href="#">MVCommon::Matrix4x4fHasher</a>	
A hasher for <a href="#">Matrix4x4f</a> objects so they can be used in unordered collections . . . . .	100
<a href="#">MVCommon::Pair&lt; TFirst, TSecond &gt;</a>	
A pair of values . . . . .	100
<a href="#">MVCommon::RedirectingLoggerSink</a>	
A logger sink implementation for redirecting log messages to another logger . . . . .	102
<a href="#">MVCommon::SharedGuidAliasDatabasePtr</a>	
A shared smart-pointer to a guid alias database . . . . .	103
<a href="#">MVCommon::SharedLoggerPtr</a>	
A shared smart-pointer to a logger . . . . .	108
<a href="#">MVCommon::SharedLoggerSinkPtr</a>	
A shared smart-pointer to a logger sink . . . . .	111
<a href="#">MVCommon::SharedThreadPoolJobPtr</a>	
A shared smart-pointer to a thread pool job . . . . .	115
<a href="#">MVCommon::StdOutLoggerSink</a>	
A logger sink implementation for logging into a standard output . . . . .	118
<a href="#">MVCommon::String</a>	
A string implementation . . . . .	120
<a href="#">MVCommon::StringHasher</a>	
A hasher for <a href="#">String</a> objects so they can be used in unordered collections . . . . .	122
<a href="#">MVCommon::ThreadPool</a>	
A pool of threads . . . . .	123
<a href="#">MVCommon::Vector2d</a>	
A 2-dimensional vector with double-precision floating-point values . . . . .	126
<a href="#">MVCommon::Vector2dHasher</a>	
A hasher for <a href="#">Vector2d</a> objects so they can be used in unordered collections . . . . .	131
<a href="#">MVCommon::Vector2f</a>	
A 2-dimensional vector with single-precision floating-point values . . . . .	132
<a href="#">MVCommon::Vector2fHasher</a>	
A hasher for <a href="#">Vector2f</a> objects so they can be used in unordered collections . . . . .	137
<a href="#">MVCommon::Vector3d</a>	
A 3-dimensional vector with double-precision floating-point values . . . . .	138
<a href="#">MVCommon::Vector3dHasher</a>	
A hasher for <a href="#">Vector3d</a> objects so they can be used in unordered collections . . . . .	144
<a href="#">MVCommon::Vector3f</a>	
A 3-dimensional vector with single-precision floating-point values . . . . .	145
<a href="#">MVCommon::Vector3fHasher</a>	
A hasher for <a href="#">Vector3f</a> objects so they can be used in unordered collections . . . . .	151
<a href="#">MVCommon::Vector4d</a>	
A 4-dimensional vector with double-precision floating-point values . . . . .	152
<a href="#">MVCommon::Vector4dHasher</a>	
A hasher for <a href="#">Vector4d</a> objects so they can be used in unordered collections . . . . .	158
<a href="#">MVCommon::Vector4f</a>	
A 4-dimensional vector with single-precision floating-point values . . . . .	159

<a href="#">MVCommon::Vector4fHasher</a>	
A hasher for <a href="#">Vector4f</a> objects so they can be used in unordered collections	165
<a href="#">MVCommon::VersionInfo</a>	
A structure holding module version information	165
<a href="#">MVCommon::VersionInfoHasher</a>	
A hasher for <a href="#">VersionInfo</a> objects so they can be used in unordered collections	167
<a href="#">MVCommon::Versord</a>	
A rotational quaternion (i.e. versor) with double-precision floating-point values	168
<a href="#">MVCommon::VersordHasher</a>	
A hasher for <a href="#">Versord</a> objects so they can be used in unordered collections	174
<a href="#">MVCommon::Versorf</a>	
A rotational quaternion (i.e. versor) with single-precision floating-point values	175
<a href="#">MVCommon::VersorfHasher</a>	
A hasher for <a href="#">Versorf</a> objects so they can be used in unordered collections	180
<a href="#">MVCommon::WeakLoggerPtr</a>	
A weak smart-pointer to a logger	181



## Chapter 5

# File Index

### 5.1 File List

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

public/MVCommon/CUtil.h	185
public/MVCommon/Memory.h	??
public/MVCommon/MVCommonAPI.h	??
public/MVCommon/MVCommonVersion.h	189
public/MVCommon/PlatformDef.h	??
public/MVCommon/data/CameraParams.h	??
public/MVCommon/data/Color.h	??
public/MVCommon/guid/Guid.h	??
public/MVCommon/guid/GuidAliasDatabase.h	??
public/MVCommon/guid/GuidAliasDatabaseIterator.h	??
public/MVCommon/guid/GuidGenerator.h	185
public/MVCommon/guid/SharedGuidAliasDatabasePtr.h	??
public/MVCommon/logger/ILoggerSink.h	??
public/MVCommon/logger/LogEntry.h	??
public/MVCommon/logger/Logger.h	??
public/MVCommon/logger/LoggerLogLevel.h	186
public/MVCommon/logger/LoggerRegistry.h	??
public/MVCommon/logger/LogLevel.h	187
public/MVCommon/logger/SharedLoggerPtr.h	??
public/MVCommon/logger/SharedLoggerSinkPtr.h	??
public/MVCommon/logger/WeakLoggerPtr.h	??
public/MVCommon/logger/sinks/AndroidSystemLoggerSink.h	??
public/MVCommon/logger/sinks/AppleSystemLoggerSink.h	??
public/MVCommon/logger/sinks/FileLoggerSink.h	??
public/MVCommon/logger/sinks/RedirectingLoggerSink.h	??
public/MVCommon/logger/sinks/StdOutLoggerSink.h	??
public/MVCommon/math/Math.h	??
public/MVCommon/math/Matrix4x4d.h	??
public/MVCommon/math/Matrix4x4f.h	??
public/MVCommon/math/Vector2d.h	??
public/MVCommon/math/Vector2f.h	??
public/MVCommon/math/Vector3d.h	??
public/MVCommon/math/Vector3f.h	??
public/MVCommon/math/Vector4d.h	??
public/MVCommon/math/Vector4f.h	??

public/MVCommon/math/ <b>Versord.h</b>	??
public/MVCommon/math/ <b>Versorf.h</b>	??
public/MVCommon/settings/ <b>HostSettings.h</b>	??
public/MVCommon/utils/ <b>ByteArray.h</b>	??
public/MVCommon/utils/ <b>Pair.h</b>	??
public/MVCommon/utils/ <b>String.h</b>	??
public/MVCommon/utils/ <a href="#">VersionInfo.h</a>	189
public/MVCommon/utils/blockingcounter/ <b>BlockingCounter.h</b>	??
public/MVCommon/utils/blockingcounter/ <b>BlockingCounterValueEquals.h</b>	??
public/MVCommon/utils/blockingcounter/ <b>IBlockingCounterCondition.h</b>	??
public/MVCommon/utils/threadpool/ <b>IThreadPoolJob.h</b>	??
public/MVCommon/utils/threadpool/ <b>SharedThreadPoolJobPtr.h</b>	??
public/MVCommon/utils/threadpool/ <b>ThreadPool.h</b>	??



## Chapter 6

# Data Structure Documentation

### 6.1 MVCommon::AndroidSystemLoggerSink Class Reference

A logger sink implementation for logging log messages via Android system logging facility.

```
#include <AndroidSystemLoggerSink.h>
```

Inherits [MVCommon::ILoggerSink](#).

#### Public Member Functions

- MVCOMMON\_API [AndroidSystemLoggerSink](#) ([LoggerLogLevel](#) logLevel=LoggerLogLevel::LLL\_VERBOSE)  
*A constructor.*
- MVCOMMON\_API [~AndroidSystemLoggerSink](#) ()  
*A destructor.*

#### Protected Member Functions

- virtual void [HandleLogEntry](#) ([LogEntry](#) const &logEntry) override  
*A callback executed when a new log entry is added.*

#### Additional Inherited Members

##### 6.1.1 Detailed Description

A logger sink implementation for logging log messages via Android system logging facility.

In case the sink is instantiated on a non-Android platform, log messages are not handled at all.

##### 6.1.2 Constructor & Destructor Documentation

###### 6.1.2.1 AndroidSystemLoggerSink()

```
MVCOMMON_API MVCommon::AndroidSystemLoggerSink::AndroidSystemLoggerSink (  
    LoggerLogLevel logLevel = LoggerLogLevel::LLL\_VERBOSE )
```

A constructor.

**Parameters**

<i>logLevel</i>	an initial log level (default value -> all log messages are processed)
-----------------	--

**6.1.3 Member Function Documentation****6.1.3.1 HandleLogEntry()**

```
virtual void MVCommon::AndroidSystemLoggerSink::HandleLogEntry (
    LogEntry const & logEntry ) [override], [protected], [virtual]
```

A callback executed when a new log entry is added.

**Parameters**

<i>logEntry</i>	a new log entry
-----------------	-----------------

Implements [MVCommon::ILoggerSink](#).

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

- public/MVCommon/logger/sinks/AndroidSystemLoggerSink.h

**6.2 MVCommon::AppleSystemLoggerSink Class Reference**

A logger sink implementation for logging log messages via Apple system logging facility.

```
#include <AppleSystemLoggerSink.h>
```

Inherits [MVCommon::ILoggerSink](#).

**Public Member Functions**

- MVCOMMON\_API [AppleSystemLoggerSink](#) ([LoggerLogLevel](#) logLevel=LoggerLogLevel::LLL\_VERBOSE)  
*A constructor.*
- MVCOMMON\_API [~AppleSystemLoggerSink](#) ()  
*A destructor.*

**Protected Member Functions**

- virtual void [HandleLogEntry](#) ([LogEntry](#) const &logEntry) override  
*A callback executed when a new log entry is added.*

## Additional Inherited Members

### 6.2.1 Detailed Description

A logger sink implementation for logging log messages via Apple system logging facility.

In case the sink is instantiated on a non-Apple platform (MacOS, iOS, ...), log messages are not handled at all.

### 6.2.2 Constructor & Destructor Documentation

#### 6.2.2.1 AppleSystemLoggerSink()

```
MVCOMMON_API MVCommon::AppleSystemLoggerSink::AppleSystemLoggerSink (
    LoggerLogLevel logLevel = LoggerLogLevel::LLL\_VERBOSE )
```

A constructor.

##### Parameters

<i>logLevel</i>	an initial log level (default value -> all log messages are processed)
-----------------	--

### 6.2.3 Member Function Documentation

#### 6.2.3.1 HandleLogEntry()

```
virtual void MVCommon::AppleSystemLoggerSink::HandleLogEntry (
    LogEntry const & logEntry ) [override], [protected], [virtual]
```

A callback executed when a new log entry is added.

##### Parameters

<i>logEntry</i>	a new log entry
-----------------	-----------------

Implements [MVCommon::ILoggerSink](#).

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

- public/MVCommon/logger/sinks/AppleSystemLoggerSink.h

## 6.3 MVCommon::BlockingCounter Class Reference

A counter with a feature of blocking a thread until the counter has a specific value.

```
#include <BlockingCounter.h>
```

Inherits NonAssignable.

### Public Member Functions

- MVCOMMON\_API [BlockingCounter](#) (int32\_t initialValue=0, int32\_t waitersCountHint=1)  
*A constructor.*
- MVCOMMON\_API [~BlockingCounter](#) ()  
*A destructor.*
- MVCOMMON\_API void [Increment](#) (int32\_t change=1)  
*Increments the counter by a given value.*
- MVCOMMON\_API int32\_t [Value](#) () const  
*Gets current value of the counter.*
- MVCOMMON\_API int32\_t [WaitUntilValue](#) (int32\_t targetValue) const  
*Blocks current thread until the counter reaches given value.*
- MVCOMMON\_API int32\_t [WaitUntilValueFor](#) (int32\_t targetValue, uint64\_t milliseconds) const  
*Blocks current thread until the counter reaches given value or until a timeout expires.*
- MVCOMMON\_API int32\_t [WaitUntil](#) (IBlockingCounterCondition &condition) const  
*Blocks current thread until the counter's value is accepted by a condition.*
- MVCOMMON\_API int32\_t [WaitUntilFor](#) (IBlockingCounterCondition &condition, uint64\_t milliseconds) const  
*Blocks current thread until the counter's value is accepted by a condition or until a timeout expires.*
- MVCOMMON\_API [BlockingCounter](#) & [operator+=](#) (int32\_t change)  
*Increments the counter by a given value.*

### 6.3.1 Detailed Description

A counter with a feature of blocking a thread until the counter has a specific value.

### 6.3.2 Constructor & Destructor Documentation

#### 6.3.2.1 BlockingCounter()

```
MVCOMMON_API MVCommon::BlockingCounter::BlockingCounter (
    int32_t initialValue = 0,
    int32_t waitersCountHint = 1 )
```

A constructor.

## Parameters

<i>initialValue</i>	an initial value of the counter
<i>waitersCountHint</i>	a hint about expected count of waiting threads - it allows an optimization of internal memory allocations made per each waiting call in cases when count of parallel waiters can be predicted. Special value 0 will result in allocations made every time, and <code>negative</code> hint value results in no deallocations (and thus maximum reusability of the memory) during the entire lifetime of the counter.

### 6.3.3 Member Function Documentation

#### 6.3.3.1 Increment()

```
MVCOMMON_API void MVCommon::BlockingCounter::Increment (
    int32_t change = 1 )
```

Increments the counter by a given value.

## Parameters

<i>change</i>	a change to increase the counter's value by (may be negative)
---------------	---

#### 6.3.3.2 operator+=()

```
MVCOMMON_API BlockingCounter& MVCommon::BlockingCounter::operator+= (
    int32_t change )
```

Increments the counter by a given value.

## Parameters

<i>change</i>	a change to increase the counter's value by (may be negative)
---------------	---

## Returns

this counter

#### 6.3.3.3 Value()

```
MVCOMMON_API int32_t MVCommon::BlockingCounter::Value ( ) const
```

Gets current value of the counter.

**Returns**

counter's value

**6.3.3.4 WaitUntil()**

```
MVCOMMON_API int32_t MVCommon::BlockingCounter::WaitUntil (
    IBlockingCounterCondition & condition ) const
```

Blocks current thread until the counter's value is accepted by a condition.

**Parameters**

<i>condition</i>	a condition that must pass in order to unblock the thread
------------------	---

**Returns**

the value which was accepted by the condition

**6.3.3.5 WaitUntilFor()**

```
MVCOMMON_API int32_t MVCommon::BlockingCounter::WaitUntilFor (
    IBlockingCounterCondition & condition,
    uint64_t milliseconds ) const
```

Blocks current thread until the counter's value is accepted by a condition or until a timeout expires.

**Parameters**

<i>condition</i>	a condition that must pass in order to unblock the thread
<i>milliseconds</i>	a timeout (in milliseconds) after which the current thread is unblocked at the latest

**Returns**

the value accepted by the condition when the counter reaches it before the timeout expires, current counter's value otherwise

**6.3.3.6 WaitUntilValue()**

```
MVCOMMON_API int32_t MVCommon::BlockingCounter::WaitUntilValue (
    int32_t targetValue ) const
```

Blocks current thread until the counter reaches given value.

**Parameters**

<i>targetValue</i>	a value the counter has to reach in order to unblock the thread
--------------------	---

**Returns**

the target value

**6.3.3.7 WaitUntilValueFor()**

```
MVCOMMON_API int32_t MVCommon::BlockingCounter::WaitUntilValueFor (
    int32_t targetValue,
    uint64_t milliseconds ) const
```

Blocks current thread until the counter reaches given value or until a timeout expires.

**Parameters**

<i>targetValue</i>	a value the counter has to reach in order to unblock the thread
<i>milliseconds</i>	a timeout (in milliseconds) after which the current thread is unblocked at the latest

**Returns**

the target value when the counter reaches it before the timeout expires, current counter's value otherwise

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

- public/MVCommon/utils/blockingcounter/BlockingCounter.h

**6.4 MVCommon::BlockingCounterValueEquals Class Reference**

A counter condition for checking equality of its value with a target value.

```
#include <BlockingCounterValueEquals.h>
```

Inherits [MVCommon::IBlockingCounterCondition](#).

**Public Member Functions**

- MVCOMMON\_API [BlockingCounterValueEquals](#) (int32\_t targetValue)  
*A constructor.*
- virtual MVCOMMON\_API bool [operator\(\)](#) (int32\_t value) override  
*An operator executed for checking the condition with a value.*

### 6.4.1 Detailed Description

A counter condition for checking equality of its value with a target value.

### 6.4.2 Constructor & Destructor Documentation

#### 6.4.2.1 BlockingCounterValueEquals()

```
MVCOMMON_API MVCommon::BlockingCounterValueEquals::BlockingCounterValueEquals (
    int32_t targetValue )
```

A constructor.

##### Parameters

<i>targetValue</i>	a target value
--------------------	----------------

### 6.4.3 Member Function Documentation

#### 6.4.3.1 operator>()

```
virtual MVCOMMON_API bool MVCommon::BlockingCounterValueEquals::operator() (
    int32_t value ) [override], [virtual]
```

An operator executed for checking the condition with a value.

##### Parameters

<i>value</i>	a value the condition is checked with
--------------	---------------------------------------

##### Returns

true in case the value is equal to the target value, false otherwise

Implements [MVCommon::IBlockingCounterCondition](#).

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

- public/MVCommon/utils/blockingcounter/BlockingCounterValueEquals.h



## 6.5 MVCommon::ByteArray Class Reference

An array of bytes.

```
#include <ByteArray.h>
```

### Public Member Functions

- MVCOMMON\_API [ByteArray](#) ()  
*A constructor.*
- MVCOMMON\_API [ByteArray](#) (uint8\_t const \*data, size\_t size)  
*A constructor.*
- MVCOMMON\_API [ByteArray](#) (uint8\_t byte, size\_t count=1)  
*A constructor.*
- MVCOMMON\_API [ByteArray](#) ([ByteArray](#) const &other)  
*A copy constructor.*
- MVCOMMON\_API [ByteArray](#) ([ByteArray](#) &&other)  
*A move constructor.*
- MVCOMMON\_API [~ByteArray](#) ()  
*A destructor.*
- MVCOMMON\_API const uint8\_t \* [Data](#) () const  
*Gets a pointer to the array's internal continuous memory.*
- MVCOMMON\_API size\_t [Size](#) () const  
*Gets size of the array.*
- MVCOMMON\_API void [Clear](#) ()  
*Empties the array.*
- MVCOMMON\_API [ByteArray](#) & [Push](#) ([ByteArray](#) const &other)  
*Pushes another array of bytes to the end of this array.*
- MVCOMMON\_API [ByteArray](#) & [Push](#) (uint8\_t byte, size\_t count=1)  
*Pushes an array of the same byte to the end of this array.*
- MVCOMMON\_API [ByteArray](#) & [Push](#) (uint8\_t const \*data, size\_t size)  
*Pushes data to the end of this array.*
- MVCOMMON\_API uint8\_t [Pop](#) ()  
*Pops and removes a single byte from the front of the array.*
- MVCOMMON\_API [ByteArray](#) [Pop](#) (size\_t count)  
*Pops an array of bytes from the front of the array.*
- MVCOMMON\_API void [Skip](#) (size\_t count=1)  
*Skips an array of bytes from the front of the array.*
- MVCOMMON\_API [ByteArray](#) [Subarray](#) (size\_t startPos=0, size\_t count=1)  
*Creates a subarray of bytes from the array, not removing the bytes from the original array.*
- MVCOMMON\_API uint8\_t & [operator\[\]](#) (size\_t pos)  
*Accesses a specific byte in the array.*
- MVCOMMON\_API const uint8\_t & [operator\[\]](#) (size\_t pos) const  
*Accesses a specific byte in the array.*
- MVCOMMON\_API [ByteArray](#) & [operator=](#) ([ByteArray](#) other)  
*An assignment operator.*
- MVCOMMON\_API [ByteArray](#) & [operator<<=](#) ([ByteArray](#) const &other)  
*Pushes another array of bytes to the end of this array.*
- MVCOMMON\_API [ByteArray](#) & [operator<<=](#) (uint8\_t byte)  
*Pushes a byte to the end of this array.*
- MVCOMMON\_API [ByteArray](#) & [operator>>=](#) (uint8\_t &byte)  
*Extracts a byte from the front of the array.*

### 6.5.1 Detailed Description

An array of bytes.

The implementation maintains a continuous array (vector) of bytes (`uint8_t`), which is resized when necessary and under specific conditions for maximum efficiency. The array provides operations for pushing bytes to the end of the array and for popping them from the array's front, behaving thus like a queue. The difference from `std::queue` is that the array's internal storage is continuous.

### 6.5.2 Constructor & Destructor Documentation

#### 6.5.2.1 `ByteArray()` [1/5]

```
MVCOMMON_API MVCommon::ByteArray::ByteArray ( )
```

A constructor.

Creates an empty array of bytes.

#### 6.5.2.2 `ByteArray()` [2/5]

```
MVCOMMON_API MVCommon::ByteArray::ByteArray (
    uint8_t const * data,
    size_t size )
```

A constructor.

##### Parameters

<i>data</i>	a pointer to data to initialize the array with
<i>size</i>	a size of the data to initialize the array with

Creates an array of bytes initialized with the given data.

#### 6.5.2.3 `ByteArray()` [3/5]

```
MVCOMMON_API MVCommon::ByteArray::ByteArray (
    uint8_t byte,
    size_t count = 1 )
```

A constructor.

##### Parameters

<i>byte</i>	a byte to initialize the array with
<i>count</i>	a count of bytes to initialize the array with

Creates an array of bytes containing the given amount of the same byte.

#### 6.5.2.4 ByteArray() [4/5]

```
MVCOMMON_API MVCommon::ByteArray::ByteArray (
    ByteArray const & other )
```

A copy constructor.

##### Parameters

<i>other</i>	an array to make a copy of
--------------	----------------------------

#### 6.5.2.5 ByteArray() [5/5]

```
MVCOMMON_API MVCommon::ByteArray::ByteArray (
    ByteArray && other )
```

A move constructor.

##### Parameters

<i>other</i>	an array to move
--------------	------------------

### 6.5.3 Member Function Documentation

#### 6.5.3.1 Data()

```
MVCOMMON_API const uint8_t* MVCommon::ByteArray::Data ( ) const
```

Gets a pointer to the array's internal continuous memory.

##### Returns

a pointer to the array's memory

The call may return different pointers at different times, for example when some bytes were already popped. The returned pointer always points to the next byte that would be popped if such a call took place.

#### 6.5.3.2 operator<<=() [1/2]

```
MVCOMMON_API ByteArray& MVCommon::ByteArray::operator<<= (
    ByteArray const & other )
```

Pushes another array of bytes to the end of this array.

**Parameters**

<i>other</i>	an array to push to this array
--------------	--------------------------------

**Returns**

this array

**6.5.3.3 operator<<=()** [2/2]

```
MVCOMMON_API ByteArray& MVCommon::ByteArray::operator<<= (
    uint8_t byte )
```

Pushes a byte to the end of this array.

**Parameters**

<i>byte</i>	a byte to push
-------------	----------------

**Returns**

this array

**6.5.3.4 operator=()**

```
MVCOMMON_API ByteArray& MVCommon::ByteArray::operator= (
    ByteArray other )
```

An assignment operator.

Replaces the array's content by a copy of another array's content.

**Parameters**

<i>other</i>	an array to copy the content from
--------------	-----------------------------------

**Returns**

this array

### 6.5.3.5 operator>>=()

```
MVCOMMON_API ByteArray& MVCommon::ByteArray::operator>>= (
    uint8_t & byte )
```

Extracts a byte from the front of the array.

#### Parameters

<i>byte</i>	a reference to byte to extract into
-------------	-------------------------------------

#### Returns

this array

#### Exceptions

<i>std::runtime_error</i>	raised when there are no data available in the array
---------------------------	--

### 6.5.3.6 operator[]() [1/2]

```
MVCOMMON_API uint8_t& MVCommon::ByteArray::operator[] (
    size_t pos )
```

Accesses a specific byte in the array.

#### Parameters

<i>pos</i>	an index of the byte to access
------------	--------------------------------

#### Returns

a reference to the byte of the array

No bounds checking is performed.

### 6.5.3.7 operator[]() [2/2]

```
MVCOMMON_API const uint8_t& MVCommon::ByteArray::operator[] (
    size_t pos ) const
```

Accesses a specific byte in the array.

#### Parameters

<i>pos</i>	an index of the byte to access
------------	--------------------------------

**Returns**

a reference to the byte of the array

No bounds checking is performed.

**6.5.3.8 Pop() [1/2]**

```
MVCOMMON_API uint8_t MVCommon::ByteArray::Pop ( )
```

Pops and removes a single byte from the front of the array.

**Returns**

the front byte

**Exceptions**

<code>std::runtime_error</code>	raised when there are no data available in the array
---------------------------------	--

**6.5.3.9 Pop() [2/2]**

```
MVCOMMON_API ByteArray MVCommon::ByteArray::Pop (
    size_t count )
```

Pops an array of bytes from the front of the array.

**Parameters**

<code>count</code>	a count of bytes to pop
--------------------	-------------------------

**Returns**

the array of bytes popped from the front

The call always succeeds, even when there is not enough bytes in the array. The returned array will in such case contain less bytes than requested.

**6.5.3.10 Push() [1/3]**

```
MVCOMMON_API ByteArray& MVCommon::ByteArray::Push (
    ByteArray const & other )
```

Pushes another array of bytes to the end of this array.

## Parameters

<i>other</i>	an array to push to this array
--------------	--------------------------------

## Returns

this array

**6.5.3.11 Push()** [2/3]

```
MVCOMMON_API ByteArray& MVCommon::ByteArray::Push (
    uint8_t byte,
    size_t count = 1 )
```

Pushes an array of the same byte to the end of this array.

## Parameters

<i>byte</i>	a byte to push
<i>count</i>	a count of bytes to push

## Returns

this array

**6.5.3.12 Push()** [3/3]

```
MVCOMMON_API ByteArray& MVCommon::ByteArray::Push (
    uint8_t const * data,
    size_t size )
```

Pushes data to the end of this array.

## Parameters

<i>data</i>	a pointer to data to push
<i>size</i>	a size of the data to push

## Returns

this array

#### 6.5.3.13 Size()

```
MVCOMMON_API size_t MVCommon::ByteArray::Size ( ) const
```

Gets size of the array.

##### Returns

array's size

#### 6.5.3.14 Skip()

```
MVCOMMON_API void MVCommon::ByteArray::Skip (
    size_t count = 1 )
```

Skips an array of bytes from the front of the array.

##### Parameters

<i>count</i>	a count of bytes to skip
--------------	--------------------------

The call always succeeds, even when there is not enough bytes in the array.

#### 6.5.3.15 Subarray()

```
MVCOMMON_API ByteArray MVCommon::ByteArray::Subarray (
    size_t startPos = 0,
    size_t count = 1 )
```

Creates a subarray of bytes from the array, not removing the bytes from the original array.

##### Parameters

<i>startPos</i>	a position of the first byte
<i>count</i>	a count of bytes

##### Returns

the subarray of bytes

The call always succeeds, even when there is not enough bytes in the original array or when the starting position is outside of the valid range. The returned array will in such case contain less bytes than requested or even no bytes at all.

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

- public/MVCommon/Utils/ByteArray.h



## 6.6 MVCommon::ByteArrayHasher Struct Reference

A hasher for [ByteArray](#) objects so they can be used in unordered collections.

```
#include <ByteArray.h>
```

### Public Member Functions

- MVCOMMON\_API size\_t [operator\(\)](#) ([ByteArray](#) const &byteArray) const  
*Calculates a hash value from the object.*

#### 6.6.1 Detailed Description

A hasher for [ByteArray](#) objects so they can be used in unordered collections.

#### 6.6.2 Member Function Documentation

##### 6.6.2.1 operator()()

```
MVCOMMON_API size_t MVCommon::ByteArrayHasher::operator() (
    ByteArray const & byteArray ) const
```

Calculates a hash value from the object.

##### Parameters

<i>byteArray</i>	an object to calculate the hash value of
------------------	--

##### Returns

hash value of the object

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

- public/MVCommon/Utils/ByteArray.h

## 6.7 MVCommon::CameraParams Struct Reference

A data structure containing intrinsic and extrinsic parameters of cameras.

```
#include <CameraParams.h>
```

## Public Member Functions

- MVCOMMON\_API [CameraParams](#) (uint32\_t width=0, uint32\_t height=0, [Vector2f](#) F=[Vector2f](#)(1.0f, 1.0f))  
*A constructor.*
- MVCOMMON\_API [~CameraParams](#) ()  
*A destructor.*
- MVCOMMON\_API [String ToString](#) () const  
*Converts the camera params into a human-readable string.*
- MVCOMMON\_API void [ToRawBytes](#) ([ByteArray](#) &bytes) const  
*Serializes the camera params into a byte array.*
- MVCOMMON\_API void [NormalizePoint](#) ([Vector2f](#) &point) const  
*Normalizes a point coordinates using focal lengths and principal point offsets of camera.*
- MVCOMMON\_API void [NormalizePoint](#) ([Vector3f](#) &point) const  
*Normalizes a point x and y coordinates using focal lengths and principal point offsets of camera.*
- MVCOMMON\_API void [DenormalizePoint](#) ([Vector2f](#) &point) const  
*Denormalizes a point coordinates using focal lengths and principal point offsets of camera.*
- MVCOMMON\_API void [DenormalizePoint](#) ([Vector3f](#) &point) const  
*Denormalizes a point x and y coordinates using focal lengths and principal point offsets of camera.*
- MVCOMMON\_API void [UndistortPoint](#) ([Vector2f](#) &point) const  
*Transforms a point coordinates to compensate camera lens distortion.*
- MVCOMMON\_API void [UndistortPoint](#) ([Vector3f](#) &point) const  
*Transforms a point x and y coordinates to compensate camera lens distortion.*
- MVCOMMON\_API [CameraParams ScaleToResolution](#) (uint32\_t targetWidth, uint32\_t targetHeight) const  
*Creates a new camera params with focal lengths and principal point offsets scaled for a target resolution.*

## Static Public Member Functions

- static MVCOMMON\_API [CameraParams FromRawBytes](#) ([ByteArray](#) &bytes, bool consumeBytes=false)  
*Deserializes camera params from a byte array.*

## Data Fields

- uint32\_t [width](#)  
*A width (in pixels).*
- uint32\_t [height](#)  
*A height (in pixels).*
- [Vector2f](#) F  
*Focal lengths (in pixels) in x and y directions.*
- [Vector2f](#) C  
*Principal point offsets (in pixels) in x and y directions.*
- [Vector2f](#) distortionC  
*Principal point offsets for distortion operations (in pixels) in x and y directions.*
- float [distortion](#) [5]  
*Distortion coefficients.*
- [Vector3f](#) translation  
*Translation offset of the origin in the camera's coordinate system.*
- [Matrix4x4f](#) rotation  
*Rotation matrix offset of the origin in the camera's coordinate system.*

## Static Public Attributes

- static const MVCOMMON\_API size\_t [RAW\\_BYTES\\_SIZE](#)  
*A count of bytes the camera params requires in a serialized form.*

### 6.7.1 Detailed Description

A data structure containing intrinsic and extrinsic parameters of cameras.

### 6.7.2 Constructor & Destructor Documentation

#### 6.7.2.1 CameraParams()

```
MVCOMMON_API MVCommon::CameraParams::CameraParams (
    uint32_t width = 0,
    uint32_t height = 0,
    Vector2f F = Vector2f(1.0f, 1.0f) )
```

A constructor.

Parameters

<i>width</i>	a width (in pixels)
<i>height</i>	a height (in pixels)
<i>F</i>	focal lengths (in pixels) in x and y directions

Constructs the camera params with default values - principal point offsets equal to [width/2, height/2], and distortion, translation and rotation set to identities.

### 6.7.3 Member Function Documentation

#### 6.7.3.1 DenormalizePoint() [1/2]

```
MVCOMMON_API void MVCommon::CameraParams::DenormalizePoint (
    Vector2f & point ) const
```

Denormalizes a point coordinates using focal lengths and principal point offsets of camera.

Parameters

<i>point</i>	a point to denormalize
--------------	------------------------

### 6.7.3.2 DenormalizePoint() [2/2]

```
MVCOMMON_API void MVCommon::CameraParams::DenormalizePoint (
    Vector3f & point ) const
```

Denormalizes a point x and y coordinates using focal lengths and principal point offsets of camera.

#### Parameters

<i>point</i>	a point to denormalize
--------------	------------------------

### 6.7.3.3 FromRawBytes()

```
static MVCOMMON_API CameraParams MVCommon::CameraParams::FromRawBytes (
    ByteArray & bytes,
    bool consumeBytes = false ) [static]
```

Deserializes camera params from a byte array.

#### Parameters

<i>bytes</i>	an array of camera params bytes
<i>consumeBytes</i>	determines whether bytes of the camera params shall be removed from the array

#### Returns

camera params

#### Exceptions

<i>std::invalid_argument</i>	raised when there are not enough bytes in the array
------------------------------	---

### 6.7.3.4 NormalizePoint() [1/2]

```
MVCOMMON_API void MVCommon::CameraParams::NormalizePoint (
    Vector2f & point ) const
```

Normalizes a point coordinates using focal lengths and principal point offsets of camera.

#### Parameters

<i>point</i>	a point to normalize
--------------	----------------------

### 6.7.3.5 NormalizePoint() [2/2]

```
MVCOMMON_API void MVCommon::CameraParams::NormalizePoint (  
    Vector3f & point ) const
```

Normalizes a point x and y coordinates using focal lengths and principal point offsets of camera.

#### Parameters

<i>point</i>	a point to normalize
--------------	----------------------

### 6.7.3.6 ScaleToResolution()

```
MVCOMMON_API CameraParams MVCommon::CameraParams::ScaleToResolution (  
    uint32_t targetWidth,  
    uint32_t targetHeight ) const
```

Creates a new camera params with focal lengths and principal point offsets scaled for a target resolution.

#### Parameters

<i>targetWidth</i>	target width
<i>targetHeight</i>	target height

#### Returns

a new camera params

Vertical (y) and horizontal (x) elements are scaled independently. Distortion, translation and rotation are preserved in the new camera params.

### 6.7.3.7 ToRawBytes()

```
MVCOMMON_API void MVCommon::CameraParams::ToRawBytes (  
    ByteArray & bytes ) const
```

Serializes the camera params into a byte array.

#### Parameters

<i>bytes</i>	a byte array to serialize into
--------------	--------------------------------

### 6.7.3.8 ToString()

```
MVCOMMON_API String MVCommon::CameraParams::ToString ( ) const
```

Converts the camera params into a human-readable string.

#### Returns

the camera params string

### 6.7.3.9 UndistortPoint() [1/2]

```
MVCOMMON_API void MVCommon::CameraParams::UndistortPoint (
    Vector2f & point ) const
```

Transforms a point coordinates to compensate camera lens distortion.

#### Parameters

<i>point</i>	a point to undistort
--------------	----------------------

### 6.7.3.10 UndistortPoint() [2/2]

```
MVCOMMON_API void MVCommon::CameraParams::UndistortPoint (
    Vector3f & point ) const
```

Transforms a point x and y coordinates to compensate camera lens distortion.

#### Parameters

<i>point</i>	a point to undistort
--------------	----------------------

## 6.7.4 Field Documentation

### 6.7.4.1 distortionC

```
Vector2f MVCommon::CameraParams::distortionC
```

Principal point offsets for distortion operations (in pixels) in x and y directions.

Its value may be slightly different than the value of [C](#).

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

- public/MVCommon/data/CameraParams.h

## 6.8 MVCommon::CameraParamsHasher Struct Reference

A hasher for [CameraParams](#) objects so they can be used in unordered collections.

```
#include <CameraParams.h>
```

### Public Member Functions

- MVCOMMON\_API size\_t [operator\(\)](#) ([CameraParams](#) const &cameraParams) const  
*Calculates a hash value from the object.*

#### 6.8.1 Detailed Description

A hasher for [CameraParams](#) objects so they can be used in unordered collections.

#### 6.8.2 Member Function Documentation

##### 6.8.2.1 operator()()

```
MVCOMMON_API size_t MVCommon::CameraParamsHasher::operator() (
    CameraParams const & cameraParams ) const
```

Calculates a hash value from the object.

##### Parameters

<i>cameraParams</i>	an object to calculate the hash value of
---------------------	--

##### Returns

hash value of the object

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

- public/MVCommon/data/CameraParams.h

## 6.9 MVCommon::Color Struct Reference

An RGBA color.

```
#include <Color.h>
```

## Public Member Functions

- MVCOMMON\_API [Color](#) ()  
*A constructor of the black color.*
- MVCOMMON\_API [Color](#) (uint8\_t redByte, uint8\_t greenByte, uint8\_t blueByte, uint8\_t alphaByte=255)  
*A constructor.*
- MVCOMMON\_API [Color](#) (float red, float green, float blue, float alpha=1.0f)  
*A constructor.*
- MVCOMMON\_API [Color](#) ([Vector4f](#) const &color)  
*A constructor.*
- MVCOMMON\_API [~Color](#) ()  
*A destructor.*
- MVCOMMON\_API [String ToString](#) () const  
*Converts the color into a human-readable string.*
- MVCOMMON\_API [String ToRGB\\_HTMLString](#) () const  
*Converts the RGB part of the color into a HTML hexadecimal string in format #rrggbb.*
- MVCOMMON\_API uint8\_t [GetRedByte](#) () const  
*Returns a red element byte value.*
- MVCOMMON\_API uint8\_t [GetGreenByte](#) () const  
*Returns a green element byte value.*
- MVCOMMON\_API uint8\_t [GetBlueByte](#) () const  
*Returns a blue element byte value.*
- MVCOMMON\_API uint8\_t [GetAlphaByte](#) () const  
*Returns an alpha element byte value.*
- MVCOMMON\_API float [GetRed](#) () const  
*Returns a red element value in range <0.0, 1.0>.*
- MVCOMMON\_API float [GetGreen](#) () const  
*Returns a green element value in range <0.0, 1.0>.*
- MVCOMMON\_API float [GetBlue](#) () const  
*Returns a blue element value in range <0.0, 1.0>.*
- MVCOMMON\_API float [GetAlpha](#) () const  
*Returns an alpha element value in range <0.0, 1.0>.*
- MVCOMMON\_API void [SetValue](#) (uint8\_t redByte, uint8\_t greenByte, uint8\_t blueByte, uint8\_t alpha↵  
Byte=255)  
*Sets value of the color.*
- MVCOMMON\_API void [SetValue](#) (float red, float green, float blue, float alpha=1.0f)  
*Sets value of the color.*
- MVCOMMON\_API void [SetValue](#) ([Vector4f](#) const &color)  
*Sets value of the color.*
- MVCOMMON\_API void [SetRedByte](#) (uint8\_t redByte)  
*Sets red element byte value.*
- MVCOMMON\_API void [SetGreenByte](#) (uint8\_t greenByte)  
*Sets green element byte value.*
- MVCOMMON\_API void [SetBlueByte](#) (uint8\_t blueByte)  
*Sets blue element byte value.*
- MVCOMMON\_API void [SetAlphaByte](#) (uint8\_t alphaByte)  
*Sets alpha element byte value.*
- MVCOMMON\_API void [SetRed](#) (float red)  
*Sets red element value in range <0.0, 1.0>.*
- MVCOMMON\_API void [SetGreen](#) (float green)  
*Sets green element value in range <0.0, 1.0>.*



- MVCOMMON\_API void [SetBlue](#) (float blue)  
*Sets blue element value in range <0.0, 1.0>.*
- MVCOMMON\_API void [SetAlpha](#) (float alpha)  
*Sets alpha element value in range <0.0, 1.0>.*
- MVCOMMON\_API uint8\_t [GetRGBBrightnessByte](#) () const  
*Calculates an RGB brightness byte value of the color.*
- MVCOMMON\_API float [GetRGBBrightness](#) () const  
*Calculates an RGB brightness value of the color in range <0.0, 1.0>.*

## Static Public Member Functions

- static MVCOMMON\_API [Color FromString](#) (String const &str)  
*Creates a color from a human-readable string.*

### 6.9.1 Detailed Description

An RGBA color.

### 6.9.2 Constructor & Destructor Documentation

#### 6.9.2.1 Color() [1/3]

```
MVCOMMON_API MVCommon::Color::Color (
    uint8_t redByte,
    uint8_t greenByte,
    uint8_t blueByte,
    uint8_t alphaByte = 255 )
```

A constructor.

##### Parameters

<i>redByte</i>	a red element byte value
<i>greenByte</i>	a green element byte value
<i>blueByte</i>	a blue element byte value
<i>alphaByte</i>	an alpha element byte value

#### 6.9.2.2 Color() [2/3]

```
MVCOMMON_API MVCommon::Color::Color (
    float red,
```

```
float green,
float blue,
float alpha = 1.0f )
```

A constructor.

#### Parameters

<i>red</i>	a red element value in range <0.0, 1.0>
<i>green</i>	a green element value in range <0.0, 1.0>
<i>blue</i>	a blue element value in range <0.0, 1.0>
<i>alpha</i>	an alpha element value in range <0.0, 1.0>

### 6.9.2.3 Color() [3/3]

```
MVCOMMON_API MVCommon::Color::Color (
    Vector4f const & color )
```

A constructor.

#### Parameters

<i>color</i>	a vector containing color element values in range <0.0, 1.0> (x -> red, y -> green, z -> blue, w -> alpha)
--------------	--

## 6.9.3 Member Function Documentation

### 6.9.3.1 FromString()

```
static MVCOMMON_API Color MVCommon::Color::FromString (
    String const & str ) [static]
```

Creates a color from a human-readable string.

#### Parameters

<i>str</i>	a color string
------------	----------------

#### Returns

a color

### 6.9.3.2 GetAlpha()

```
MVCOMMON_API float MVCommon::Color::GetAlpha ( ) const
```

Returns an alpha element value in range <0.0, 1.0>.

#### Returns

an alpha element value

### 6.9.3.3 GetAlphaByte()

```
MVCOMMON_API uint8_t MVCommon::Color::GetAlphaByte ( ) const
```

Returns an alpha element byte value.

#### Returns

an alpha element byte value

### 6.9.3.4 GetBlue()

```
MVCOMMON_API float MVCommon::Color::GetBlue ( ) const
```

Returns a blue element value in range <0.0, 1.0>.

#### Returns

a blue element value

### 6.9.3.5 GetBlueByte()

```
MVCOMMON_API uint8_t MVCommon::Color::GetBlueByte ( ) const
```

Returns a blue element byte value.

#### Returns

a blue element byte value

#### 6.9.3.6 GetGreen()

```
MVCOMMON_API float MVCommon::Color::GetGreen ( ) const
```

Returns a green element value in range <0.0, 1.0>.

##### Returns

a green element value

#### 6.9.3.7 GetGreenByte()

```
MVCOMMON_API uint8_t MVCommon::Color::GetGreenByte ( ) const
```

Returns a green element byte value.

##### Returns

a green element byte value

#### 6.9.3.8 GetRed()

```
MVCOMMON_API float MVCommon::Color::GetRed ( ) const
```

Returns a red element value in range <0.0, 1.0>.

##### Returns

a red element value

#### 6.9.3.9 GetRedByte()

```
MVCOMMON_API uint8_t MVCommon::Color::GetRedByte ( ) const
```

Returns a red element byte value.

##### Returns

a red element byte value

#### 6.9.3.10 GetRGBBrightness()

```
MVCOMMON_API float MVCommon::Color::GetRGBBrightness ( ) const
```

Calculates an RGB brightness value of the color in range <0.0, 1.0>.

##### Returns

an RGB brightness value

#### 6.9.3.11 GetRGBBrightnessByte()

```
MVCOMMON_API uint8_t MVCommon::Color::GetRGBBrightnessByte ( ) const
```

Calculates an RGB brightness byte value of the color.

##### Returns

an RGB brightness byte value

#### 6.9.3.12 SetAlpha()

```
MVCOMMON_API void MVCommon::Color::SetAlpha (
    float alpha )
```

Sets alpha element value in range <0.0, 1.0>.

##### Parameters

<i>alpha</i>	a new alpha element value
--------------	---------------------------

#### 6.9.3.13 SetAlphaByte()

```
MVCOMMON_API void MVCommon::Color::SetAlphaByte (
    uint8_t alphaByte )
```

Sets alpha element byte value.

##### Parameters

<i>alphaByte</i>	a new alpha element byte value
------------------	--------------------------------

#### 6.9.3.14 SetBlue()

```
MVCOMMON_API void MVCommon::Color::SetBlue (
    float blue )
```

Sets blue element value in range <0.0, 1.0>.

##### Parameters

<i>blue</i>	a new blue element value
-------------	--------------------------

#### 6.9.3.15 SetBlueByte()

```
MVCOMMON_API void MVCommon::Color::SetBlueByte (
    uint8_t blueByte )
```

Sets blue element byte value.

##### Parameters

<i>blueByte</i>	a new blue element byte value
-----------------	-------------------------------

#### 6.9.3.16 SetGreen()

```
MVCOMMON_API void MVCommon::Color::SetGreen (
    float green )
```

Sets green element value in range <0.0, 1.0>.

##### Parameters

<i>green</i>	a new green element value
--------------	---------------------------

#### 6.9.3.17 SetGreenByte()

```
MVCOMMON_API void MVCommon::Color::SetGreenByte (
    uint8_t greenByte )
```

Sets green element byte value.

## Parameters

<i>greenByte</i>	a new green element byte value
------------------	--------------------------------

**6.9.3.18 SetRed()**

```
MVCOMMON_API void MVCommon::Color::SetRed (
    float red )
```

Sets red element value in range <0.0, 1.0>.

## Parameters

<i>red</i>	a new red element value
------------	-------------------------

**6.9.3.19 SetRedByte()**

```
MVCOMMON_API void MVCommon::Color::SetRedByte (
    uint8_t redByte )
```

Sets red element byte value.

## Parameters

<i>redByte</i>	a new red element byte value
----------------	------------------------------

**6.9.3.20 SetValue()** [1/3]

```
MVCOMMON_API void MVCommon::Color::SetValue (
    float red,
    float green,
    float blue,
    float alpha = 1.0f )
```

Sets value of the color.

## Parameters

<i>red</i>	a new red element value in range <0.0, 1.0>
<i>green</i>	a new green element value in range <0.0, 1.0>
<i>blue</i>	a new blue element value in range <0.0, 1.0>
<i>alpha</i>	a new alpha element value in range <0.0, 1.0>

**6.9.3.21 SetValue() [2/3]**

```
MVCOMMON_API void MVCommon::Color::SetValue (
    uint8_t redByte,
    uint8_t greenByte,
    uint8_t blueByte,
    uint8_t alphaByte = 255 )
```

Sets value of the color.

**Parameters**

<i>redByte</i>	a new red element byte value
<i>greenByte</i>	a new green element byte value
<i>blueByte</i>	a new blue element byte value
<i>alphaByte</i>	a new alpha element byte value

**6.9.3.22 SetValue() [3/3]**

```
MVCOMMON_API void MVCommon::Color::SetValue (
    Vector4f const & color )
```

Sets value of the color.

**Parameters**

<i>color</i>	a vector containing new color element values in range <0.0, 1.0> (x -> red, y -> green, z -> blue, w -> alpha)
--------------	--

**6.9.3.23 ToRGB\_HTMLString()**

```
MVCOMMON_API String MVCommon::Color::ToRGB_HTMLString ( ) const
```

Converts the RGB part of the color into a HTML hexadecimal string in format #rrggbb.

**Returns**

the RGB HTML hexadecimal string



### 6.9.3.24 ToString()

```
MVCOMMON_API String MVCommon::Color::ToString ( ) const
```

Converts the color into a human-readable string.

#### Returns

the color string

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

- public/MVCommon/data/Color.h

## 6.10 MVCommon::ColorHasher Struct Reference

A hasher for [Color](#) objects so they can be used in unordered collections.

```
#include <Color.h>
```

### Public Member Functions

- MVCOMMON\_API size\_t [operator\(\)](#) ([Color](#) const &color) const  
*Calculates a hash value from the object.*

### 6.10.1 Detailed Description

A hasher for [Color](#) objects so they can be used in unordered collections.

### 6.10.2 Member Function Documentation

#### 6.10.2.1 operator()()

```
MVCOMMON_API size_t MVCommon::ColorHasher::operator() (
    Color const & color ) const
```

Calculates a hash value from the object.

#### Parameters

<i>color</i>	an object to calculate the hash value of
--------------	--

**Returns**

hash value of the object

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

- public/MVCommon/data/Color.h

## 6.11 MVCommon::FileLoggerSink Class Reference

A logger sink implementation for logging into a file.

```
#include <FileLoggerSink.h>
```

Inherits [MVCommon::ILoggerSink](#).

### Public Member Functions

- MVCOMMON\_API [FileLoggerSink](#) ([MVCommon::String](#) const &path, [LoggerLogLevel](#) logLevel=LoggerLogLevel::LLL\_VERBOSE)  
*A constructor.*
- MVCOMMON\_API [~FileLoggerSink](#) ()  
*A destructor.*

### Protected Member Functions

- virtual void [HandleLogEntry](#) ([LogEntry](#) const &logEntry) override  
*A callback executed when a new log entry is added.*

### Additional Inherited Members

#### 6.11.1 Detailed Description

A logger sink implementation for logging into a file.

#### 6.11.2 Constructor & Destructor Documentation

##### 6.11.2.1 FileLoggerSink()

```
MVCOMMON_API MVCommon::FileLoggerSink::FileLoggerSink (
    MVCommon::String const & path,
    LoggerLogLevel logLevel = LoggerLogLevel::LLL\_VERBOSE )
```

A constructor.

## Parameters

<i>path</i>	a path of the file
<i>logLevel</i>	an initial log level (default value -> all log messages are processed)

### 6.11.3 Member Function Documentation

#### 6.11.3.1 HandleLogEntry()

```
virtual void MVCommon::FileLoggerSink::HandleLogEntry (
    LogEntry const & logEntry ) [override], [protected], [virtual]
```

A callback executed when a new log entry is added.

## Parameters

<i>logEntry</i>	a new log entry
-----------------	-----------------

Implements [MVCommon::ILoggerSink](#).

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

- public/MVCommon/logger/sinks/FileLoggerSink.h

## 6.12 MVCommon::Guid Struct Reference

A globally-unique identifier implementation.

```
#include <Guid.h>
```

### Public Member Functions

- MVCOMMON\_API [Guid](#) ()  
*A constructor of a [Guid](#) with all bytes set to 0.*
- MVCOMMON\_API [Guid](#) ([Guid](#) const &other)  
*A copy constructor.*
- MVCOMMON\_API [Guid](#) ([Guid](#) &&other)  
*A move constructor.*
- virtual MVCOMMON\_API ~[Guid](#) ()  
*A destructor.*
- MVCOMMON\_API [String ToHexString](#) () const  
*Formats the [Guid](#) to hexadecimal 00000000-0000-0000-0000-000000000000 format.*
- MVCOMMON\_API void [ToRawBytes](#) ([ByteArray](#) &bytes) const  
*Formats the [Guid](#) into a raw bytes array.*
- MVCOMMON\_API void [ToRfc4122](#) ([ByteArray](#) &bytes) const  
*Formats the [Guid](#) into RFC 4122 format.*
- MVCOMMON\_API bool [IsNil](#) () const  
*Checks whether the [Guid](#) is a Nil [Guid](#) (with all bytes set to 0).*

## Static Public Member Functions

- static MVCOMMON\_API [Guid Nil](#) ()  
*Constructs a new Nil [Guid](#) (with all bytes set to 0).*
- static MVCOMMON\_API [Guid FromHexString](#) (String const &str)  
*Parses a string in hexadecimal format 00000000-0000-0000-0000-000000000000 into a [Guid](#).*
- static MVCOMMON\_API [Guid FromRawBytes](#) (ByteArray &bytes, bool consumeBytes=false)  
*Constructs a [Guid](#) using a raw bytes array (must contain 16 elements).*
- static MVCOMMON\_API [Guid FromRfc4122](#) (ByteArray &bytes, bool consumeBytes=false)  
*Constructs a [Guid](#) using an array of bytes in RFC 4122 format (must contain 16 elements).*

## Static Public Attributes

- static const MVCOMMON\_API size\_t [RAW\\_BYTES\\_SIZE](#)  
*A constant indicating the size of raw bytes of the [Guid](#).*
- static const MVCOMMON\_API size\_t [RFC4122\\_BYTES\\_SIZE](#)  
*A constant indicating the size of bytes in RFC 4122 format of the [Guid](#).*

### 6.12.1 Detailed Description

A globally-unique identifier implementation.

### 6.12.2 Constructor & Destructor Documentation

#### 6.12.2.1 [Guid\(\)](#) [1/2]

```
MVCOMMON_API MVCommon::Guid::Guid (
    Guid const & other )
```

A copy constructor.

##### Parameters

<i>other</i>	a <a href="#">Guid</a> to make a copy of
--------------	--

#### 6.12.2.2 [Guid\(\)](#) [2/2]

```
MVCOMMON_API MVCommon::Guid::Guid (
    Guid && other )
```

A move constructor.

## Parameters

<i>other</i>	a <a href="#">Guid</a> to move
--------------	--------------------------------

## 6.12.3 Member Function Documentation

### 6.12.3.1 FromHexString()

```
static MVCOMMON_API Guid MVCommon::Guid::FromHexString (  
    String const & str ) [static]
```

Parses a string in hexadecimal format 00000000-0000-0000-0000-000000000000 into a [Guid](#).

## Parameters

<i>str</i>	a string to parse
------------	-------------------

## Returns

a [Guid](#)

The input string must be at least 32 characters long (i.e. 32 hexa characters). It can optionally contain an opening and a closing bracket ('{' and '}') and 4 hyphens on specific positions of the string.

## Exceptions

<i>std::invalid_argument</i>	raised when the format of the string is invalid and can not be parsed
------------------------------	---

### 6.12.3.2 FromRawBytes()

```
static MVCOMMON_API Guid MVCommon::Guid::FromRawBytes (  
    ByteArray & bytes,  
    bool consumeBytes = false ) [static]
```

Constructs a [Guid](#) using a raw bytes array (must contain 16 elements).

## Parameters

<i>bytes</i>	an array of 16 bytes
<i>consumeBytes</i>	an indication whether the bytes of the array shall be consumed

**Returns**

a [Guid](#)

**Exceptions**

<code>std::invalid_argument</code>	raised when there are not enough bytes in the array
------------------------------------	---

**6.12.3.3 FromRfc4122()**

```
static MVCOMMON_API Guid MVCommon::Guid::FromRfc4122 (
    ByteArray & bytes,
    bool consumeBytes = false ) [static]
```

Constructs a [Guid](#) using an array of bytes in RFC 4122 format (must contain 16 elements).

**Parameters**

<i>bytes</i>	an array of 16 bytes in RFC 4122 format
<i>consumeBytes</i>	an indication whether the bytes of the array shall be consumed

**Returns**

a [Guid](#)

**Exceptions**

<code>std::invalid_argument</code>	raised when there are not enough bytes in the array
------------------------------------	---

**6.12.3.4 IsNil()**

```
MVCOMMON_API bool MVCommon::Guid::IsNil ( ) const
```

Checks whether the [Guid](#) is a Nil [Guid](#) (with all bytes set to 0).

**Returns**

true when the [Guid](#) is a Nil [Guid](#)

#### 6.12.3.5 Nil()

```
static MVCOMMON_API Guid MVCommon::Guid::Nil ( ) [static]
```

Constructs a new Nil [Guid](#) (with all bytes set to 0).

##### Returns

a Nil [Guid](#)

#### 6.12.3.6 ToHexString()

```
MVCOMMON_API String MVCommon::Guid::ToHexString ( ) const
```

Formats the [Guid](#) to hexadecimal 00000000-0000-0000-0000-000000000000 format.

##### Returns

a string of 36 characters (32 for hexa characters and 4 for hyphens)

#### 6.12.3.7 ToRawBytes()

```
MVCOMMON_API void MVCommon::Guid::ToRawBytes (
    ByteArray & bytes ) const
```

Formats the [Guid](#) into a raw bytes array.

##### Parameters

<i>bytes</i>	an array to store 16 raw bytes in
--------------	-----------------------------------

#### 6.12.3.8 ToRfc4122()

```
MVCOMMON_API void MVCommon::Guid::ToRfc4122 (
    ByteArray & bytes ) const
```

Formats the [Guid](#) into RFC 4122 format.

##### Parameters

<i>bytes</i>	an array to store 16 raw bytes in RFC 4122 format in
--------------	--

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

- public/MVCommon/guid/Guid.h

## 6.13 MVCommon::GuidAliasDatabase Class Reference

A database of [Guid](#) aliases.

```
#include <GuidAliasDatabase.h>
```

### Public Types

- typedef [GuidAliasDatabaseIterator](#) [Iterator](#)  
*An alternative type name declaration for [GuidAliasDatabaseIterator](#).*

### Public Member Functions

- MVCOMMON\_API [GuidAliasDatabase](#) ()  
*A constructor.*
- MVCOMMON\_API [GuidAliasDatabase](#) ([GuidAliasDatabase](#) const &other)  
*A copy constructor.*
- MVCOMMON\_API [GuidAliasDatabase](#) ([GuidAliasDatabase](#) &&other)  
*A move constructor.*
- virtual MVCOMMON\_API ~[GuidAliasDatabase](#) ()  
*A destructor.*
- MVCOMMON\_API void [RegisterGuidAlias](#) ([MVCommon::Guid](#) const &guid, [MVCommon::String](#) const &alias)  
*Registers a new [Guid](#) alias.*
- MVCOMMON\_API void [UnregisterGuidAlias](#) ([MVCommon::Guid](#) const &guid)  
*Unregisters a [Guid](#) alias.*
- MVCOMMON\_API void [UnregisterGuidAlias](#) ([MVCommon::String](#) const &alias)  
*Unregisters a [Guid](#) alias.*
- MVCOMMON\_API bool [TryGetGuidAlias](#) ([MVCommon::Guid](#) const &guid, [MVCommon::String](#) &alias) const  
*Tries to get an alias registered for a given [Guid](#).*
- MVCOMMON\_API bool [TryGetGuidWithAlias](#) ([MVCommon::String](#) const &alias, [MVCommon::Guid](#) &guid) const  
*Tries to get a [Guid](#) with an alias registered.*
- MVCOMMON\_API [MVCommon::String](#) [GetGuidAlias](#) ([MVCommon::Guid](#) const &guid, [MVCommon::String](#) const &fallbackAlias="") const  
*Gets an alias registered for a given [Guid](#).*
- MVCOMMON\_API [MVCommon::Guid](#) [GetGuidWithAlias](#) ([MVCommon::String](#) const &alias, [MVCommon::Guid](#) const &fallbackGuid=[MVCommon::Guid::Nil](#)()) const  
*Gets a [Guid](#) with an alias registered.*
- MVCOMMON\_API bool [GuidRegistered](#) ([MVCommon::Guid](#) const &guid) const  
*Checks whether a [Guid](#) has already an alias registered.*
- MVCOMMON\_API bool [AliasRegistered](#) ([MVCommon::String](#) const &alias) const  
*Checks whether there already is a [Guid](#) with an alias registered.*
- MVCOMMON\_API [Iterator](#) [Begin](#) () const  
*Returns an iterator to the first entry of the database.*
- MVCOMMON\_API [Iterator](#) [End](#) () const  
*Returns an iterator to the last entry of the database.*



### 6.13.1 Detailed Description

A database of [Guid](#) aliases.

The database keeps pairs of [Guid](#) and [String](#) alias objects and provides fast bi-directional mapping between them. Each [Guid](#) can only have a single alias assigned and each alias can only be assigned to a single [Guid](#).

### 6.13.2 Constructor & Destructor Documentation

#### 6.13.2.1 GuidAliasDatabase() [1/2]

```
MVCOMMON_API MVCommon::GuidAliasDatabase::GuidAliasDatabase (
    GuidAliasDatabase const & other )
```

A copy constructor.

Parameters

<i>other</i>	a database to make a copy of
--------------	------------------------------

#### 6.13.2.2 GuidAliasDatabase() [2/2]

```
MVCOMMON_API MVCommon::GuidAliasDatabase::GuidAliasDatabase (
    GuidAliasDatabase && other )
```

A move constructor.

Parameters

<i>other</i>	a database to move
--------------	--------------------

### 6.13.3 Member Function Documentation

#### 6.13.3.1 AliasRegistered()

```
MVCOMMON_API bool MVCommon::GuidAliasDatabase::AliasRegistered (
    MVCommon::String const & alias ) const
```

Checks whether there already is a [Guid](#) with an alias registered.

**Parameters**

<i>alias</i>	an alias to check
--------------	-------------------

**Returns**

true in case there already is a [Guid](#) registered with the alias

**6.13.3.2 Begin()**

```
MVCOMMON_API Iterator MVCommon::GuidAliasDatabase::Begin ( ) const
```

Returns an iterator to the first entry of the database.

**Returns**

an iterator

The returned iterator is equal to [End\(\)](#) iterator when the database is empty.

**6.13.3.3 End()**

```
MVCOMMON_API Iterator MVCommon::GuidAliasDatabase::End ( ) const
```

Returns an iterator to the last entry of the database.

**Returns**

an iterator

**6.13.3.4 GetGuidAlias()**

```
MVCOMMON_API MVCommon::String MVCommon::GuidAliasDatabase::GetGuidAlias (
    MVCommon::Guid const & guid,
    MVCommon::String const & fallbackAlias = "" ) const
```

Gets an alias registered for a given [Guid](#).

**Parameters**

<i>guid</i>	a <a href="#">Guid</a> to get the alias for
<i>fallbackAlias</i>	a string returned in case there is no alias registered for the <a href="#">Guid</a>

**Returns**

an alias of the [Guid](#) or the fallback string in case there is none

**6.13.3.5 GetGuidWithAlias()**

```
MVCOMMON_API MVCommon::Guid MVCommon::GuidAliasDatabase::GetGuidWithAlias (
    MVCommon::String const & alias,
    MVCommon::Guid const & fallbackGuid = MVCommon::Guid::Nil() ) const
```

Gets a [Guid](#) with an alias registered.

**Parameters**

<i>alias</i>	an alias to look a <a href="#">Guid</a> registered with for
<i>fallbackGuid</i>	a <a href="#">Guid</a> returned in case there is no <a href="#">Guid</a> registered with the alias

**Returns**

a [Guid](#) registered with the alias or fallback [Guid](#) in case there is none

**6.13.3.6 GuidRegistered()**

```
MVCOMMON_API bool MVCommon::GuidAliasDatabase::GuidRegistered (
    MVCommon::Guid const & guid ) const
```

Checks whether a [Guid](#) has already an alias registered.

**Parameters**

<i>guid</i>	a <a href="#">Guid</a> to check
-------------	---------------------------------

**Returns**

true in case the [Guid](#) already has an alias registered

**6.13.3.7 RegisterGuidAlias()**

```
MVCOMMON_API void MVCommon::GuidAliasDatabase::RegisterGuidAlias (
    MVCommon::Guid const & guid,
    MVCommon::String const & alias )
```

Registers a new [Guid](#) alias.

## Parameters

<i>guid</i>	a <a href="#">Guid</a> to register alias for
<i>alias</i>	an alias of the <a href="#">Guid</a>

## Exceptions

<i>std::invalid_argument</i>	raised when there already is a different alias registered for the given <a href="#">Guid</a> or the alias is already registered with another <a href="#">Guid</a>
------------------------------	---

### 6.13.3.8 TryGetGuidAlias()

```
MVCOMMON_API bool MVCommon::GuidAliasDatabase::TryGetGuidAlias (
    MVCommon::Guid const & guid,
    MVCommon::String & alias ) const
```

Tries to get an alias registered for a given [Guid](#).

## Parameters

<i>guid</i>	a <a href="#">Guid</a> to get the alias for
<i>alias</i>	a target to store the alias to

## Returns

true in case there is an alias registered for the [Guid](#)

### 6.13.3.9 TryGetGuidWithAlias()

```
MVCOMMON_API bool MVCommon::GuidAliasDatabase::TryGetGuidWithAlias (
    MVCommon::String const & alias,
    MVCommon::Guid & guid ) const
```

Tries to get a [Guid](#) with an alias registered.

## Parameters

<i>alias</i>	an alias to look a <a href="#">Guid</a> registered with for
<i>guid</i>	a target to store the <a href="#">Guid</a> to

## Returns

true in case there is a [Guid](#) registered with the alias

**6.13.3.10 UnregisterGuidAlias() [1/2]**

```
MVCOMMON_API void MVCommon::GuidAliasDatabase::UnregisterGuidAlias (
    MVCommon::Guid const & guid )
```

Unregisters a [Guid](#) alias.

**Parameters**

<i>guid</i>	a <a href="#">Guid</a> to unregister alias of
-------------	---

**6.13.3.11 UnregisterGuidAlias() [2/2]**

```
MVCOMMON_API void MVCommon::GuidAliasDatabase::UnregisterGuidAlias (
    MVCommon::String const & alias )
```

Unregisters a [Guid](#) alias.

**Parameters**

<i>alias</i>	an alias to unregister
--------------	------------------------

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

- public/MVCommon/guid/GuidAliasDatabase.h

**6.14 MVCommon::GuidAliasDatabaseIterator Class Reference**

An iterator over elements of [GuidAliasDatabase](#) collections.

```
#include <GuidAliasDatabaseIterator.h>
```

**Public Types**

- using [ValueType](#) = [Pair](#)< [Guid](#) const, [String](#) const >  
A type of iterated-over elements of [GuidAliasDatabase](#) collection.

**Public Member Functions**

- MVCOMMON\_API [GuidAliasDatabaseIterator](#) ([GuidAliasDatabaseIterator](#) const &other)  
A copy constructor.
- MVCOMMON\_API [GuidAliasDatabaseIterator](#) ([GuidAliasDatabaseIterator](#) &&other)  
A move constructor.
- virtual MVCOMMON\_API ~[GuidAliasDatabaseIterator](#) ()

- *A destructor.*
- MVCOMMON\_API [GuidAliasDatabaseIterator](#) & operator++ ()
- *A prefix incrementation operator.*
- MVCOMMON\_API [GuidAliasDatabaseIterator](#) operator++ (int)
- *A postfix incrementation operator.*
- MVCOMMON\_API [ValueType](#) operator\* () const
- *Dereferences the iterator.*

### 6.14.1 Detailed Description

An iterator over elements of [GuidAliasDatabase](#) collections.

### 6.14.2 Constructor & Destructor Documentation

#### 6.14.2.1 [GuidAliasDatabaseIterator](#)() [1/2]

```
MVCOMMON_API MVCommon::GuidAliasDatabaseIterator::GuidAliasDatabaseIterator (
    GuidAliasDatabaseIterator const & other )
```

A copy constructor.

##### Parameters

<i>other</i>	an iterator to make a copy of
--------------	-------------------------------

#### 6.14.2.2 [GuidAliasDatabaseIterator](#)() [2/2]

```
MVCOMMON_API MVCommon::GuidAliasDatabaseIterator::GuidAliasDatabaseIterator (
    GuidAliasDatabaseIterator && other )
```

A move constructor.

##### Parameters

<i>other</i>	an iterator to move
--------------	---------------------

### 6.14.3 Member Function Documentation

### 6.14.3.1 operator\*()

```
MVCOMMON_API ValueType MVCommon::GuidAliasDatabaseIterator::operator* ( ) const
```

Dereferences the iterator.

#### Returns

a pair of [Guid](#) and its alias

### 6.14.3.2 operator++() [1/2]

```
MVCOMMON_API GuidAliasDatabaseIterator& MVCommon::GuidAliasDatabaseIterator::operator++ ( )
```

A prefix incrementation operator.

Moves the iterator to the next element and returns this updated iterator.

#### Returns

this iterator after it was updated

### 6.14.3.3 operator++() [2/2]

```
MVCOMMON_API GuidAliasDatabaseIterator MVCommon::GuidAliasDatabaseIterator::operator++ (
    int )
```

A postfix incrementation operator.

Moves the iterator to the next element, but returns the original iterator.

#### Returns

the original unupdated iterator

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

- public/MVCommon/guid/GuidAliasDatabaseIterator.h

## 6.15 MVCommon::GuidHasher Struct Reference

A hasher for [Guid](#) objects so they can be used in unordered collections.

```
#include <Guid.h>
```

## Public Member Functions

- MVCOMMON\_API size\_t [operator\(\)](#) ([Guid](#) const &guid) const  
*Calculates a hash value from the object.*

### 6.15.1 Detailed Description

A hasher for [Guid](#) objects so they can be used in unordered collections.

### 6.15.2 Member Function Documentation

#### 6.15.2.1 [operator\(\)](#)()

```
MVCOMMON_API size_t MVCommon::GuidHasher::operator() (
    Guid const & guid ) const
```

Calculates a hash value from the object.

#### Parameters

<i>guid</i>	an object to calculate the hash value of
-------------	--

#### Returns

hash value of the object

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

- public/MVCommon/guid/Guid.h

## 6.16 MVCommon::IBlockingCounterCondition Class Reference

An interface of conditions usable with blocking counters.

```
#include <IBlockingCounterCondition.h>
```

Inherited by [MVCommon::BlockingCounterValueEquals](#).

## Public Member Functions

- virtual MVCOMMON\_API [~IBlockingCounterCondition](#) ()  
*A destructor.*
- virtual MVCOMMON\_API bool [operator\(\)](#) (int32\_t value)=0  
*An operator executed for checking the condition with a value.*



### 6.16.1 Detailed Description

An interface of conditions usable with blocking counters.

### 6.16.2 Member Function Documentation

#### 6.16.2.1 operator()

```
virtual MVCOMMON_API bool MVCommon::IBlockingCounterCondition::operator() (
    int32_t value ) [pure virtual]
```

An operator executed for checking the condition with a value.

##### Parameters

<i>value</i>	a value the condition is checked with
--------------	---------------------------------------

##### Returns

true in case the condition passes with the value, false otherwise

Implemented in [MVCommon::BlockingCounterValueEquals](#).

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

- public/MVCommon/utils/blockingcounter/IBlockingCounterCondition.h

## 6.17 MVCommon::ILoggerSink Class Reference

An interface of logger sinks.

```
#include <ILoggerSink.h>
```

Inherits NonAssignable.

Inherited by [MVCommon::AndroidSystemLoggerSink](#), [MVCommon::AppleSystemLoggerSink](#), [MVCommon::FileLoggerSink](#), [MVCommon::RedirectingLoggerSink](#), and [MVCommon::StdOutLoggerSink](#).

### Public Member Functions

- MVCOMMON\_API [ILoggerSink](#) ([LoggerLogLevel](#) logLevel=LoggerLogLevel::LLL\_VERBOSE)  
*A constructor.*
- virtual MVCOMMON\_API [~ILoggerSink](#) ()  
*A destructor.*
- MVCOMMON\_API void [SetLogLevel](#) ([LoggerLogLevel](#) logLevel)  
*Changes log level of the sink for log messages filtering.*
- MVCOMMON\_API [LoggerLogLevel](#) [GetLogLevel](#) () const  
*Returns current log level of the sink.*
- void [AddLogEntry](#) ([LogEntry](#) const &logEntry)  
*Adds a new log entry to the sink.*

## Static Public Member Functions

- static MVCOMMON\_API [MVCommon::String](#) [TimestampToString](#) ([LogEntry::Timestamp](#) timestamp, bool includeDate=true)  
*A default formatter of timestamps usable in logger sink implementations.*
- static MVCOMMON\_API [MVCommon::String](#) [LogLevelToString](#) ([LogLevel](#) level, bool shortVersion=false)  
*A default formatter of log levels usable in logger sink implementations.*

## Protected Member Functions

- virtual MVCOMMON\_API void [HandleLogEntry](#) ([LogEntry](#) const &logEntry)=0  
*A callback executed when a new log entry is added.*

### 6.17.1 Detailed Description

An interface of logger sinks.

When a logger sink is attached to a logger, it will receive all log messages logged via that logger, assuming that the log message is not filtered by the sink's own log level setting.

### 6.17.2 Constructor & Destructor Documentation

#### 6.17.2.1 ILoggerSink()

```
MVCOMMON_API MVCommon::ILoggerSink::ILoggerSink (
    LoggerLogLevel logLevel = LoggerLogLevel::LLL\_VERBOSE )
```

A constructor.

Parameters

<i>logLevel</i>	an initial log level (default value -> all log messages are processed)
-----------------	--

### 6.17.3 Member Function Documentation

#### 6.17.3.1 AddLogEntry()

```
void MVCommon::ILoggerSink::AddLogEntry (
    LogEntry const & logEntry )
```

Adds a new log entry to the sink.

## Parameters

<i>logEntry</i>	a new log entry
-----------------	-----------------

A log level of the entry is compared with log level of the sink so entries with lesser log levels than the sink's log level are ignored.

### 6.17.3.2 GetLogLevel()

```
MVCOMMON_API LoggerLogLevel MVCommon::ILoggerSink::GetLogLevel ( ) const
```

Returns current log level of the sink.

## Returns

sink's log level

### 6.17.3.3 HandleLogEntry()

```
virtual MVCOMMON_API void MVCommon::ILoggerSink::HandleLogEntry (
    LogEntry const & logEntry ) [protected], [pure virtual]
```

A callback executed when a new log entry is added.

## Parameters

<i>logEntry</i>	a new log entry
-----------------	-----------------

Implemented in [MVCommon::FileLoggerSink](#), [MVCommon::RedirectingLoggerSink](#), [MVCommon::AndroidSystemLoggerSink](#), [MVCommon::AppleSystemLoggerSink](#), and [MVCommon::StdOutLoggerSink](#).

### 6.17.3.4 LogLevelToString()

```
static MVCOMMON_API MVCommon::String MVCommon::ILoggerSink::LogLevelToString (
    LogLevel level,
    bool shortVersion = false ) [static]
```

A default formatter of log levels usable in logger sink implementations.

## Parameters

<i>level</i>	a log level to format
<i>shortVersion</i>	indicates whether a short (single character) or a long (whole level name) version shall be used

**Returns**

a formatted log level

**6.17.3.5 SetLogLevel()**

```
MVCOMMON_API void MVCommon::ILoggerSink::SetLogLevel (
    LoggerLogLevel logLevel )
```

Changes log level of the sink for log messages filtering.

**Parameters**

<i>logLevel</i>	a new log level
-----------------	-----------------

**6.17.3.6 TimestampToString()**

```
static MVCOMMON_API MVCommon::String MVCommon::ILoggerSink::TimestampToString (
    LogEntry::Timestamp timestamp,
    bool includeDate = true ) [static]
```

A default formatter of timestamps usable in logger sink implementations.

A format of the resulting string is "1900-Jan-01 00:00:00.000" when date is included and "00:00:00.000" when date is ommited.

**Parameters**

<i>timestamp</i>	a timestamp to format
<i>includeDate</i>	indicates whether a date shall be included in the formatted string

**Returns**

a formatted timestamp

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

- public/MVCommon/logger/ILoggerSink.h

**6.18 MVCommon::IThreadPoolJob Class Reference**

An interface of thread pool jobs.

```
#include <IThreadPoolJob.h>
```

## Public Member Functions

- virtual MVCOMMON\_API [~IThreadPoolJob](#) ()  
*A destructor.*
- virtual MVCOMMON\_API void [operator\(\)](#) (uint32\_t threadID)=0  
*The job-executing operator.*

### 6.18.1 Detailed Description

An interface of thread pool jobs.

### 6.18.2 Member Function Documentation

#### 6.18.2.1 [operator\(\)](#)

```
virtual MVCOMMON_API void MVCommon::IThreadPoolJob::operator() (
    uint32_t threadID ) [pure virtual]
```

The job-executing operator.

#### Parameters

<i>threadID</i>	an ID of the thread that executes the job
-----------------	---

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

- public/MVCommon/utils/threadpool/IThreadPoolJob.h

## 6.19 MVCommon::LogEntry Struct Reference

A log entry data structure.

```
#include <LogEntry.h>
```

## Public Types

- typedef uint64\_t [Timestamp](#)  
*A type of log entry timestamps.*
- typedef [MVCommon::String](#) [ThreadID](#)  
*A type of log entry thread IDs.*

## Public Member Functions

- `LogEntry` (`Timestamp` timestamp, `ThreadID` threadID, `LogLevel` level, `MVCommon::String` const &tag, `MVCommon::String` const &message)  
*A constructor.*
- `MVCOMMON_API LogLevel GetLevel ()` const  
*Returns level of the log entry.*
- `MVCOMMON_API MVCommon::String GetTag ()` const  
*Returns tag of the log entry.*
- `MVCOMMON_API MVCommon::String GetMessage ()` const  
*Returns message of the log entry.*
- `MVCOMMON_API Timestamp GetTimestamp ()` const  
*Returns timestamp of the log entry.*
- `MVCOMMON_API ThreadID GetThreadID ()` const  
*Returns originating thread ID of the log entry.*

### 6.19.1 Detailed Description

A log entry data structure.

### 6.19.2 Constructor & Destructor Documentation

#### 6.19.2.1 LogEntry()

```
MVCommon::LogEntry::LogEntry (
    Timestamp timestamp,
    ThreadID threadID,
    LogLevel level,
    MVCommon::String const & tag,
    MVCommon::String const & message )
```

A constructor.

#### Parameters

<i>timestamp</i>	a timestamp (as number of milliseconds since the epoch - 1970-01-01 00:00:00.000)
<i>threadID</i>	a thread ID
<i>level</i>	a log level
<i>tag</i>	a log tag
<i>message</i>	a log message

### 6.19.3 Member Function Documentation

### 6.19.3.1 GetLevel()

```
MVCOMMON_API LogLevel MVCommon::LogEntry::GetLevel ( ) const
```

Returns level of the log entry.

#### Returns

log entry level

### 6.19.3.2 GetMessage()

```
MVCOMMON_API MVCommon::String MVCommon::LogEntry::GetMessage ( ) const
```

Returns message of the log entry.

#### Returns

log entry message

### 6.19.3.3 GetTag()

```
MVCOMMON_API MVCommon::String MVCommon::LogEntry::GetTag ( ) const
```

Returns tag of the log entry.

#### Returns

log entry tag

### 6.19.3.4 GetThreadID()

```
MVCOMMON_API ThreadID MVCommon::LogEntry::GetThreadID ( ) const
```

Returns originating thread ID of the log entry.

#### Returns

log entry thread ID

### 6.19.3.5 GetTimestamp()

```
MVCOMMON_API Timestamp MVCommon::LogEntry::GetTimestamp ( ) const
```

Returns timestamp of the log entry.

Timestamp is stored as number of milliseconds since the epoch - 1970-01-01 00:00:00.000.

#### Returns

log entry timestamp

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

- public/MVCommon/logger/LogEntry.h

## 6.20 MVCommon::Logger Class Reference

A logger.

```
#include <Logger.h>
```

Inherits NonAssignable.

### Public Member Functions

- MVCOMMON\_API [Logger](#) ([LoggerLogLevel](#) logLevel=[LoggerLogLevel::LLL\\_DEBUG](#))  
*A constructor.*
- MVCOMMON\_API [~Logger](#) ()  
*A destructor.*
- MVCOMMON\_API void [SetLogLevel](#) ([LoggerLogLevel](#) logLevel)  
*Changes log level of the logger for log messages filtering.*
- MVCOMMON\_API [LoggerLogLevel](#) [GetLogLevel](#) () const  
*Returns current log level of the logger.*
- MVCOMMON\_API void [AddLoggerSink](#) ([SharedLoggerSinkPtr](#) spLoggerSink)  
*Registers a logger sink.*
- MVCOMMON\_API void [RemoveLoggerSink](#) ([SharedLoggerSinkPtr](#) spLoggerSink)  
*Unregisters a logger sink.*
- MVCOMMON\_API void [RemoveAllLoggerSinks](#) ()  
*Unregisters all logger sinks.*
- MVCOMMON\_API void [LogMessage](#) ([LogLevel](#) level, char const \*tag, char const \*format,...)  
*Logs a new message.*
- MVCOMMON\_API void [LogMessage](#) ([LogEntry::Timestamp](#) timestamp, [LogEntry::ThreadID](#) threadID, [LogLevel](#) level, char const \*tag, char const \*format,...)  
*Logs a new message.*



## 6.20.1 Detailed Description

A logger.

A logger receives requests for messages logging, filters them according to its log level setting and asynchronously pushes them to all attached logger sinks for customized handling.

## 6.20.2 Constructor & Destructor Documentation

### 6.20.2.1 Logger()

```
MVCOMMON_API MVCommon::Logger::Logger (
    LoggerLogLevel logLevel = LoggerLogLevel::LLL_DEBUG )
```

A constructor.

#### Parameters

<i>logLevel</i>	an initial log level (default value -> debug and higher level messages are processed)
-----------------	---

## 6.20.3 Member Function Documentation

### 6.20.3.1 AddLoggerSink()

```
MVCOMMON_API void MVCommon::Logger::AddLoggerSink (
    SharedLoggerSinkPtr spLoggerSink )
```

Registers a logger sink.

#### Parameters

<i>spLoggerSink</i>	a logger sink to register
---------------------	---------------------------

### 6.20.3.2 GetLogLevel()

```
MVCOMMON_API LoggerLogLevel MVCommon::Logger::GetLogLevel ( ) const
```

Returns current log level of the logger.

**Returns**

logger's log level

**6.20.3.3 LogMessage() [1/2]**

```
MVCOMMON_API void MVCommon::Logger::LogMessage (
    LogEntry::Timestamp timestamp,
    LogEntry::ThreadID threadID,
    LogLevel level,
    char const * tag,
    char const * format,
    ... )
```

Logs a new message.

**Parameters**

<i>timestamp</i>	a timestamp of the log message (as number of milliseconds since the epoch - 1970-01-01 00:00:00.000)
<i>threadID</i>	a thread ID of the log message
<i>level</i>	a level of the log message
<i>tag</i>	a tag of the log message
<i>format</i>	a C-style formatting directive of the message in case there are additional arguments

A log level of the entry is compared with log level of the sink so entries with lesser log levels than the sink's log level are ignored.

**6.20.3.4 LogMessage() [2/2]**

```
MVCOMMON_API void MVCommon::Logger::LogMessage (
    LogLevel level,
    char const * tag,
    char const * format,
    ... )
```

Logs a new message.

**Parameters**

<i>level</i>	a level of the log message
<i>tag</i>	a tag of the log message
<i>format</i>	a C-style formatting directive of the message in case there are additional arguments

A log level of the entry is compared with log level of the sink so entries with lesser log levels than the sink's log level are ignored.

Current time and the caller's thread is used for the log message.

### 6.20.3.5 RemoveLoggerSink()

```
MVCOMMON_API void MVCommon::Logger::RemoveLoggerSink (
    SharedLoggerSinkPtr spLoggerSink )
```

Unregisters a logger sink.

#### Parameters

<i>spLoggerSink</i>	a logger sink to unregister
---------------------	-----------------------------

### 6.20.3.6 SetLogLevel()

```
MVCOMMON_API void MVCommon::Logger::SetLogLevel (
    LoggerLogLevel logLevel )
```

Changes log level of the logger for log messages filtering.

#### Parameters

<i>logLevel</i>	a new log level
-----------------	-----------------

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

- public/MVCommon/logger/Logger.h

## 6.21 MVCommon::LoggerRegistry Class Reference

A global registry of loggers.

```
#include <LoggerRegistry.h>
```

Inherits NonAssignable.

### Public Member Functions

- MVCOMMON\_API [~LoggerRegistry](#) ()  
*A destructor.*

- MVCOMMON\_API void [RegisterLogger](#) (MVCommon::String const &loggerAlias, [SharedLoggerPtr](#) sp↔  
Logger)  
*Registers a logger instance to the registry.*
- MVCOMMON\_API void [UnregisterLogger](#) (MVCommon::String const &loggerAlias)  
*Unregisters a logger registered with an alias from the registry.*
- MVCOMMON\_API [SharedLoggerPtr](#) [GetLogger](#) (MVCommon::String const &loggerAlias) const  
*Returns a logger registered with an alias.*
- MVCOMMON\_API void [ClearRegistry](#) ()  
*Clears the registry - removes all registered loggers.*

## Static Public Member Functions

- static MVCOMMON\_API [LoggerRegistry](#) & [GetInstance](#) ()  
*Returns the only instance of the registry.*

### 6.21.1 Detailed Description

A global registry of loggers.

Serves as a global accessor to [Logger](#) instances in cases where direct access is not possible.

### 6.21.2 Member Function Documentation

#### 6.21.2.1 GetInstance()

```
static MVCOMMON_API LoggerRegistry& MVCommon::LoggerRegistry::GetInstance ( ) [static]
```

Returns the only instance of the registry.

#### Returns

the registry instance

#### 6.21.2.2 GetLogger()

```
MVCOMMON_API SharedLoggerPtr MVCommon::LoggerRegistry::GetLogger (
    MVCommon::String const & loggerAlias ) const
```

Returns a logger registered with an alias.

## Parameters

<i>loggerAlias</i>	an alias the logger to return is supposed to be registered with
--------------------	---

## Returns

a logger with the alias or nullptr if there is none

**6.21.2.3 RegisterLogger()**

```
MVCOMMON_API void MVCommon::LoggerRegistry::RegisterLogger (
    MVCommon::String const & loggerAlias,
    SharedLoggerPtr spLogger )
```

Registers a logger instance to the registry.

## Parameters

<i>loggerAlias</i>	an alias to register the logger with
<i>spLogger</i>	a logger to register

Replaces the previous logger registered with the same alias in case there was one.

**6.21.2.4 UnregisterLogger()**

```
MVCOMMON_API void MVCommon::LoggerRegistry::UnregisterLogger (
    MVCommon::String const & loggerAlias )
```

Unregisters a logger registered with an alias from the registry.

## Parameters

<i>loggerAlias</i>	an alias to unregister a logger registered with
--------------------	---

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

- public/MVCommon/logger/LoggerRegistry.h

**6.22 MVCommon::Math Class Reference**

A utility class for math operations.

```
#include <Math.h>
```

## Static Public Member Functions

- static MVCOMMON\_API bool [AlmostEqual](#) (float val1, float val2, float precision=[SINGLE\\_EPSILON](#))  
*Compares two single-precision floating-point values with a tolerance.*
- static MVCOMMON\_API bool [AlmostEqual](#) (double val1, double val2, double precision=[DOUBLE\\_EPSILON](#))  
*Compares two double-precision floating-point values with a tolerance.*
- template<class T >  
static T [Clamp](#) (T const &value, T const &min, T const &max)  
*Clamps a value to a minimum-maximum range.*

## Static Public Attributes

- static const MVCOMMON\_API float [SINGLE\\_EPSILON](#)  
*A smallest possible difference between two single-precision floating-point values.*
- static const MVCOMMON\_API double [DOUBLE\\_EPSILON](#)  
*A smallest possible difference between two double-precision floating-point values.*

### 6.22.1 Detailed Description

A utility class for math operations.

### 6.22.2 Member Function Documentation

#### 6.22.2.1 AlmostEqual() [1/2]

```
static MVCOMMON_API bool MVCommon::Math::AlmostEqual (
    double val1,
    double val2,
    double precision = DOUBLE\_EPSILON ) [static]
```

Compares two double-precision floating-point values with a tolerance.

#### Parameters

<i>val1</i>	a value to compare
<i>val2</i>	a value to compare
<i>precision</i>	a required precision

#### Returns

true in case the difference between the two values is less or equal than a very small value (epsilon)

### 6.22.2.2 AlmostEqual() [2/2]

```
static MVCOMMON_API bool MVCommon::Math::AlmostEqual (
    float val1,
    float val2,
    float precision = SINGLE_EPSILON ) [static]
```

Compares two single-precision floating-point values with a tolerance.

#### Parameters

<i>val1</i>	a value to compare
<i>val2</i>	a value to compare
<i>precision</i>	a required precision

#### Returns

true in case the difference between the two values is less or equal than a very small value (epsilon)

### 6.22.2.3 Clamp()

```
template<class T >
static T MVCommon::Math::Clamp (
    T const & value,
    T const & min,
    T const & max ) [inline], [static]
```

Clamps a value to a minimum-maximum range.

#### Template Parameters

<i>T</i>	a type of the values
----------	----------------------

#### Parameters

<i>value</i>	a value to clamp
<i>min</i>	a minimum value
<i>max</i>	a maximum value

#### Returns

a clamped value (in the given range)

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

- public/MVCommon/math/Math.h

## 6.23 MVCommon::Matrix4x4d Struct Reference

A 4x4 matrix with double-precision floating-point values.

```
#include <Matrix4x4d.h>
```

### Public Member Functions

- MVCOMMON\_API [Matrix4x4d](#) ()  
*A constructor of an identity matrix (with all elements on main diagonal set to 1 and the rest set to 0).*
- MVCOMMON\_API [Matrix4x4d](#) (double a00, double a01, double a02, double a03, double a10, double a11, double a12, double a13, double a20, double a21, double a22, double a23, double a30, double a31, double a32, double a33)  
*A constructor.*
- MVCOMMON\_API [Matrix4x4d](#) ([Vector4d](#) const &row0, [Vector4d](#) const &row1, [Vector4d](#) const &row2, [Vector4d](#) const &row3)  
*A constructor.*
- MVCOMMON\_API [String ToString](#) () const  
*Converts the matrix into a human-readable string.*
- MVCOMMON\_API void [ToRawBytes](#) ([ByteArray](#) &bytes) const  
*Serializes the matrix into a byte array.*
- MVCOMMON\_API void [ToRawElements](#) (double \*elements) const  
*Serializes the matrix into an elements array.*
- MVCOMMON\_API [Matrix4x4d Transposed](#) () const  
*Creates a transposed matrix.*
- MVCOMMON\_API [Matrix4x4d Inverted](#) (bool &ok) const  
*Creates an inverted matrix.*
- MVCOMMON\_API [Matrix4x4d RotationTranslationMatrixInverted](#) () const  
*Creates an inverted matrix of a rotation-translation matrix.*
- MVCOMMON\_API [Vector4d](#) & [operator\[\]](#) (size\_t pos)  
*Accesses a specific row in the matrix.*
- MVCOMMON\_API const [Vector4d](#) & [operator\[\]](#) (size\_t pos) const  
*Accesses a specific row in the matrix.*

### Static Public Member Functions

- static MVCOMMON\_API [Matrix4x4d FromString](#) ([String](#) const &str)  
*Creates a matrix from a human-readable string.*
- static MVCOMMON\_API [Matrix4x4d FromRawBytes](#) ([ByteArray](#) &bytes, bool consumeBytes=false)  
*Deserializes matrix from a byte array.*
- static MVCOMMON\_API [Matrix4x4d FromRawElements](#) (double const \*elements)  
*Deserializes matrix from an elements array.*
- static MVCOMMON\_API [Matrix4x4d CreateZero](#) ()  
*Creates a matrix with all elements set to zero.*
- static MVCOMMON\_API [Matrix4x4d CreateTranslation](#) ([Vector3d](#) const &translation)  
*Creates a translation matrix.*
- static MVCOMMON\_API [Matrix4x4d CreateScale](#) ([Vector3d](#) const &scale)  
*Creates a scaling matrix.*
- static MVCOMMON\_API [Matrix4x4d CreateRotationFromEulerAnglesZYX](#) ([Vector3d](#) const &eulerAngles)



- Creates a rotation matrix from Euler angles (in degrees) in eulerAngles.z -> eulerAngles.y -> eulerAngles.x order.*
- static MVCOMMON\_API [Matrix4x4d CreateRotationAroundAxis](#) ([Vector3d](#) const &axis, double angle)
- Creates a rotation matrix from axis of rotation and an angle (in degrees).*
- static MVCOMMON\_API [Matrix4x4d CreateRotationFromVersor](#) ([Versord](#) const &versor)
- Creates a rotation matrix from a versor.*
- static MVCOMMON\_API [Matrix4x4d CreateOrtographic](#) (double left, double right, double bottom, double top, double near, double far)
- Creates a matrix for ortographic projection.*
- static MVCOMMON\_API [Matrix4x4d CreatePerspective](#) (double fieldOfView, double aspectRatio, double near, double far)
- Creates a matrix for perspective projection.*
- static MVCOMMON\_API [Matrix4x4d CreateLookAt](#) ([Vector3d](#) const &eyePosition, [Vector3d](#) const &center←Point, [Vector3d](#) const &upDirection)
- Creates a viewing transformation matrix.*

## Static Public Attributes

- static const MVCOMMON\_API size\_t [RAW\\_BYTES\\_SIZE](#)
- A count of bytes the matrix requires in a serialized form.*

### 6.23.1 Detailed Description

A 4x4 matrix with double-precision floating-point values.

### 6.23.2 Constructor & Destructor Documentation

#### 6.23.2.1 [Matrix4x4d\(\)](#) [1/2]

```
MVCOMMON_API MVCommon::Matrix4x4d::Matrix4x4d (
    double a00,
    double a01,
    double a02,
    double a03,
    double a10,
    double a11,
    double a12,
    double a13,
    double a20,
    double a21,
    double a22,
    double a23,
    double a30,
    double a31,
    double a32,
    double a33 )
```

A constructor.

## Parameters

<i>a00</i>	an m[0][0] value
<i>a01</i>	an m[0][1] value
<i>a02</i>	an m[0][2] value
<i>a03</i>	an m[0][3] value
<i>a10</i>	an m[1][0] value
<i>a11</i>	an m[1][1] value
<i>a12</i>	an m[1][2] value
<i>a13</i>	an m[1][3] value
<i>a20</i>	an m[2][0] value
<i>a21</i>	an m[2][1] value
<i>a22</i>	an m[2][2] value
<i>a23</i>	an m[2][3] value
<i>a30</i>	an m[3][0] value
<i>a31</i>	an m[3][1] value
<i>a32</i>	an m[3][2] value
<i>a33</i>	an m[3][3] value

**6.23.2.2 Matrix4x4d()** [2/2]

```
MVCOMMON_API MVCommon::Matrix4x4d::Matrix4x4d (
    Vector4d const & row0,
    Vector4d const & row1,
    Vector4d const & row2,
    Vector4d const & row3 )
```

A constructor.

## Parameters

<i>row0</i>	a row 0
<i>row1</i>	a row 1
<i>row2</i>	a row 2
<i>row3</i>	a row 3

**6.23.3 Member Function Documentation****6.23.3.1 CreateLookAt()**

```
static MVCOMMON_API Matrix4x4d MVCommon::Matrix4x4d::CreateLookAt (
    Vector3d const & eyePosition,
```

```
Vector3d const & centerPoint,  
Vector3d const & upDirection ) [static]
```

Creates a viewing transformation matrix.

**Parameters**

<i>eyePosition</i>	a position of the viewing camera
<i>centerPoint</i>	a point the camera is looking at
<i>upDirection</i>	an up-direction of the viewing camera

**Returns**

a viewing transformation matrix

The resulting 'look-at' matrix follows the same principle as OpenGL's `gluLookAt` utility function: the matrix maps the reference (center) point to the negative z axis and the eye position to the origin. similarly, the up direction projected onto the viewing plane is mapped to the positive y axis. The UP vector must not be parallel to the line of sight from the eye position to the reference point.

**6.23.3.2 CreateOrtographic()**

```
static MVCOMMON_API Matrix4x4d MVCommon::Matrix4x4d::CreateOrtographic (
    double left,
    double right,
    double bottom,
    double top,
    double near,
    double far ) [static]
```

Creates a matrix for ortographic projection.

**Parameters**

<i>left</i>	a left coordinate
<i>right</i>	a right coordinate
<i>bottom</i>	a bottom coordinate
<i>top</i>	a top coordinate
<i>near</i>	a near coordinate
<i>far</i>	a far coordinate

**Returns**

an ortographic-projection matrix

**6.23.3.3 CreatePerspective()**

```
static MVCOMMON_API Matrix4x4d MVCommon::Matrix4x4d::CreatePerspective (
    double fieldOfView,
    double aspectRatio,
    double near,
    double far ) [static]
```

Creates a matrix for perspective projection.

## Parameters

<i>fieldOfView</i>	a field of view (in degrees)
<i>aspectRatio</i>	an aspect ratio
<i>near</i>	a near coordinate
<i>far</i>	a far coordinate

## Returns

a perspective-projection matrix

**6.23.3.4 CreateRotationAroundAxis()**

```
static MVCOMMON_API Matrix4x4d MVCommon::Matrix4x4d::CreateRotationAroundAxis (
    Vector3d const & axis,
    double angle ) [static]
```

Creates a rotation matrix from axis of rotation and an angle (in degrees).

## Parameters

<i>axis</i>	an axis of rotation
<i>angle</i>	an angle

## Returns

a rotation matrix

**6.23.3.5 CreateRotationFromEulerAnglesZYX()**

```
static MVCOMMON_API Matrix4x4d MVCommon::Matrix4x4d::CreateRotationFromEulerAnglesZYX (
    Vector3d const & eulerAngles ) [static]
```

Creates a rotation matrix from Euler angles (in degrees) in eulerAngles.z -> eulerAngles.y -> eulerAngles.x order.

## Parameters

<i>eulerAngles</i>	Euler angles of Z-Y-X rotation
--------------------	--------------------------------

## Returns

a rotation matrix

#### 6.23.3.6 CreateRotationFromVersor()

```
static MVCOMMON_API Matrix4x4d MVCommon::Matrix4x4d::CreateRotationFromVersor (
    Versord const & versor ) [static]
```

Creates a rotation matrix from a versor.

##### Parameters

<i>versor</i>	a versor describing rotation
---------------	------------------------------

##### Returns

a rotation matrix

#### 6.23.3.7 CreateScale()

```
static MVCOMMON_API Matrix4x4d MVCommon::Matrix4x4d::CreateScale (
    Vector3d const & scale ) [static]
```

Creates a scaling matrix.

##### Parameters

<i>scale</i>	a vector describing the scale
--------------	-------------------------------

##### Returns

a scaling matrix

#### 6.23.3.8 CreateTranslation()

```
static MVCOMMON_API Matrix4x4d MVCommon::Matrix4x4d::CreateTranslation (
    Vector3d const & translation ) [static]
```

Creates a translation matrix.

##### Parameters

<i>translation</i>	a vector describing the translation
--------------------	-------------------------------------

##### Returns

a translation matrix

### 6.23.3.9 CreateZero()

```
static MVCOMMON_API Matrix4x4d MVCommon::Matrix4x4d::CreateZero ( ) [static]
```

Creates a matrix with all elements set to zero.

#### Returns

a zero matrix

### 6.23.3.10 FromRawBytes()

```
static MVCOMMON_API Matrix4x4d MVCommon::Matrix4x4d::FromRawBytes (
    ByteArray & bytes,
    bool consumeBytes = false ) [static]
```

Deserializes matrix from a byte array.

#### Parameters

<i>bytes</i>	an array of matrix bytes
<i>consumeBytes</i>	determines whether bytes of the matrix shall be removed from the array

#### Returns

a matrix

#### Exceptions

<i>std::invalid_argument</i>	raised when there are not enough bytes in the array
------------------------------	---

### 6.23.3.11 FromRawElements()

```
static MVCOMMON_API Matrix4x4d MVCommon::Matrix4x4d::FromRawElements (
    double const * elements ) [static]
```

Deserializes matrix from an elements array.

#### Parameters

<i>elements</i>	an array of 4x4 elements
-----------------	--------------------------

**Returns**

a matrix

**6.23.3.12 FromString()**

```
static MVCOMMON_API Matrix4x4d MVCommon::Matrix4x4d::FromString (
    String const & str ) [static]
```

Creates a matrix from a human-readable string.

**Parameters**

<i>str</i>	a matrix string
------------	-----------------

**Returns**

a matrix

**6.23.3.13 Inverted()**

```
MVCOMMON_API Matrix4x4d MVCommon::Matrix4x4d::Inverted (
    bool & ok ) const
```

Creates an inverted matrix.

**Parameters**

<i>ok</i>	indicates whether the inverted matrix was successfully created, since it is not always possible to create one
-----------	---

**Returns**

an inverted matrix

**6.23.3.14 operator[]() [1/2]**

```
MVCOMMON_API Vector4d& MVCommon::Matrix4x4d::operator[] (
    size_t pos )
```

Accesses a specific row in the matrix.



## Parameters

<i>pos</i>	an index of the row to access
------------	-------------------------------

## Returns

a reference to the row of the matrix

## Exceptions

<i>std::out_of_range</i>	raised when the index of the row is out of range (0-3)
--------------------------	--

**6.23.3.15 operator[]()** [2/2]

```
MVCOMMON_API const Vector4d& MVCommon::Matrix4x4d::operator[] (
    size_t pos ) const
```

Accesses a specific row in the matrix.

## Parameters

<i>pos</i>	an index of the row to access
------------	-------------------------------

## Returns

a reference to the row of the matrix

## Exceptions

<i>std::out_of_range</i>	raised when the index of the row is out of range (0-3)
--------------------------	--

**6.23.3.16 RotationTranslationMatrixInverted()**

```
MVCOMMON_API Matrix4x4d MVCommon::Matrix4x4d::RotationTranslationMatrixInverted ( ) const
```

Creates an inverted matrix of a rotation-translation matrix.

It is always possible to create an inverted matrix of a rotation-translation matrix and the algorithm is much simpler and more effective than generic inversion algorithm. However, it is up to user to know what matrices he calls the function on

- the function assumes the matrix is a rotation-translation matrix.

## Returns

an inverted matrix

### 6.23.3.17 ToRawBytes()

```
MVCOMMON_API void MVCommon::Matrix4x4d::ToRawBytes (
    ByteArray & bytes ) const
```

Serializes the matrix into a byte array.

#### Parameters

<i>bytes</i>	a byte array to serialize into
--------------	--------------------------------

### 6.23.3.18 ToRawElements()

```
MVCOMMON_API void MVCommon::Matrix4x4d::ToRawElements (
    double * elements ) const
```

Serializes the matrix into an elements array.

#### Parameters

<i>elements</i>	an array of 4x4 elements
-----------------	--------------------------

### 6.23.3.19 ToString()

```
MVCOMMON_API String MVCommon::Matrix4x4d::ToString ( ) const
```

Converts the matrix into a human-readable string.

#### Returns

the matrix string

### 6.23.3.20 Transposed()

```
MVCOMMON_API Matrix4x4d MVCommon::Matrix4x4d::Transposed ( ) const
```

Creates a transposed matrix.

#### Returns

a transposed matrix

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

- public/MVCommon/math/Matrix4x4d.h

## 6.24 MVCommon::Matrix4x4dHasher Struct Reference

A hasher for [Matrix4x4d](#) objects so they can be used in unordered collections.

```
#include <Matrix4x4d.h>
```

### Public Member Functions

- MVCOMMON\_API size\_t [operator\(\)](#) ([Matrix4x4d](#) const &matrix) const  
*Calculates a hash value from the object.*

#### 6.24.1 Detailed Description

A hasher for [Matrix4x4d](#) objects so they can be used in unordered collections.

#### 6.24.2 Member Function Documentation

##### 6.24.2.1 [operator\(\)](#)()

```
MVCOMMON_API size_t MVCommon::Matrix4x4dHasher::operator() (
    Matrix4x4d const & matrix ) const
```

Calculates a hash value from the object.

##### Parameters

<i>matrix</i>	an object to calculate the hash value of
---------------	--

##### Returns

hash value of the object

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

- public/MVCommon/math/Matrix4x4d.h

## 6.25 MVCommon::Matrix4x4f Struct Reference

A 4x4 matrix with single-precision floating-point values.

```
#include <Matrix4x4f.h>
```

## Public Member Functions

- MVCOMMON\_API [Matrix4x4f](#) ()  
*A constructor of an identity matrix (with all elements on main diagonal set to 1 and the rest set to 0).*
- MVCOMMON\_API [Matrix4x4f](#) (float a00, float a01, float a02, float a03, float a10, float a11, float a12, float a13, float a20, float a21, float a22, float a23, float a30, float a31, float a32, float a33)  
*A constructor.*
- MVCOMMON\_API [Matrix4x4f](#) ([Vector4f](#) const &row0, [Vector4f](#) const &row1, [Vector4f](#) const &row2, [Vector4f](#) const &row3)  
*A constructor.*
- MVCOMMON\_API [String ToString](#) () const  
*Converts the matrix into a human-readable string.*
- MVCOMMON\_API void [ToRawBytes](#) ([ByteArray](#) &bytes) const  
*Serializes the matrix into a byte array.*
- MVCOMMON\_API void [ToRawElements](#) (float \*elements) const  
*Serializes the matrix into an elements array.*
- MVCOMMON\_API [Matrix4x4f Transposed](#) () const  
*Creates a transposed matrix.*
- MVCOMMON\_API [Matrix4x4f Inverted](#) (bool &ok) const  
*Creates an inverted matrix.*
- MVCOMMON\_API [Matrix4x4f RotationTranslationMatrixInverted](#) () const  
*Creates an inverted matrix of a rotation-translation matrix.*
- MVCOMMON\_API [Vector4f](#) & [operator\[\]](#) (size\_t pos)  
*Accesses a specific row in the matrix.*
- MVCOMMON\_API const [Vector4f](#) & [operator\[\]](#) (size\_t pos) const  
*Accesses a specific row in the matrix.*

## Static Public Member Functions

- static MVCOMMON\_API [Matrix4x4f FromString](#) ([String](#) const &str)  
*Creates a matrix from a human-readable string.*
- static MVCOMMON\_API [Matrix4x4f FromRawBytes](#) ([ByteArray](#) &bytes, bool consumeBytes=false)  
*Deserializes matrix from a byte array.*
- static MVCOMMON\_API [Matrix4x4f FromRawElements](#) (float const \*elements)  
*Deserializes matrix from an elements array.*
- static MVCOMMON\_API [Matrix4x4f CreateZero](#) ()  
*Creates a matrix with all elements set to zero.*
- static MVCOMMON\_API [Matrix4x4f CreateTranslation](#) ([Vector3f](#) const &translation)  
*Creates a translation matrix.*
- static MVCOMMON\_API [Matrix4x4f CreateScale](#) ([Vector3f](#) const &scale)  
*Creates a scaling matrix.*
- static MVCOMMON\_API [Matrix4x4f CreateRotationFromEulerAnglesZYX](#) ([Vector3f](#) const &eulerAngles)  
*Creates a rotation matrix from Euler angles (in degrees) in eulerAngles.z -> eulerAngles.y -> eulerAngles.x order.*
- static MVCOMMON\_API [Matrix4x4f CreateRotationAroundAxis](#) ([Vector3f](#) const &axis, float angle)  
*Creates a rotation matrix from axis of rotation and an angle (in degrees).*
- static MVCOMMON\_API [Matrix4x4f CreateRotationFromVersor](#) ([Versor](#) const &versor)  
*Creates a rotation matrix from a versor.*
- static MVCOMMON\_API [Matrix4x4f CreateOrtographic](#) (float left, float right, float bottom, float top, float near, float far)  
*Creates a matrix for ortographic projection.*

- static MVCOMMON\_API [Matrix4x4f CreatePerspective](#) (float fieldOfView, float aspectRatio, float near, float far)  
*Creates a matrix for perspective projection.*
- static MVCOMMON\_API [Matrix4x4f CreateLookAt](#) ([Vector3f](#) const &eyePosition, [Vector3f](#) const &centerPoint, [Vector3f](#) const &upDirection)  
*Creates a viewing transformation matrix.*

## Static Public Attributes

- static const MVCOMMON\_API size\_t [RAW\\_BYTES\\_SIZE](#)  
*A count of bytes the matrix requires in a serialized form.*

### 6.25.1 Detailed Description

A 4x4 matrix with single-precision floating-point values.

### 6.25.2 Constructor & Destructor Documentation

#### 6.25.2.1 [Matrix4x4f\(\)](#) [1/2]

```
MVCOMMON_API MVCommon::Matrix4x4f::Matrix4x4f (
    float a00,
    float a01,
    float a02,
    float a03,
    float a10,
    float a11,
    float a12,
    float a13,
    float a20,
    float a21,
    float a22,
    float a23,
    float a30,
    float a31,
    float a32,
    float a33 )
```

A constructor.

#### Parameters

<i>a00</i>	an m[0][0] value
<i>a01</i>	an m[0][1] value
<i>a02</i>	an m[0][2] value
<i>a03</i>	an m[0][3] value
<i>a10</i>	an m[1][0] value
<i>a11</i>	an m[1][1] value

## Parameters

<i>a12</i>	an m[1][2] value
<i>a13</i>	an m[1][3] value
<i>a20</i>	an m[2][0] value
<i>a21</i>	an m[2][1] value
<i>a22</i>	an m[2][2] value
<i>a23</i>	an m[2][3] value
<i>a30</i>	an m[3][0] value
<i>a31</i>	an m[3][1] value
<i>a32</i>	an m[3][2] value
<i>a33</i>	an m[3][3] value

**6.25.2.2 Matrix4x4f() [2/2]**

```
MVCOMMON_API MVCommon::Matrix4x4f::Matrix4x4f (
    Vector4f const & row0,
    Vector4f const & row1,
    Vector4f const & row2,
    Vector4f const & row3 )
```

A constructor.

## Parameters

<i>row0</i>	a row 0
<i>row1</i>	a row 1
<i>row2</i>	a row 2
<i>row3</i>	a row 3

**6.25.3 Member Function Documentation****6.25.3.1 CreateLookAt()**

```
static MVCOMMON_API Matrix4x4f MVCommon::Matrix4x4f::CreateLookAt (
    Vector3f const & eyePosition,
    Vector3f const & centerPoint,
    Vector3f const & upDirection ) [static]
```

Creates a viewing transformation matrix.

## Parameters

<i>eyePosition</i>	a position of the viewing camera
<i>centerPoint</i>	a point the camera is looking at
<i>upDirection</i>	an up-direction of the viewing camera

**Returns**

a viewing transformation matrix

The resulting 'look-at' matrix follows the same principle as OpenGL's `gluLookAt` utility function: the matrix maps the reference (center) point to the negative z axis and the eye position to the origin. similarly, the up direction projected onto the viewing plane is mapped to the positive y axis. The UP vector must not be parallel to the line of sight from the eye position to the reference point.

**6.25.3.2 CreateOrtographic()**

```
static MVCOMMON_API Matrix4x4f MVCommon::Matrix4x4f::CreateOrtographic (
    float left,
    float right,
    float bottom,
    float top,
    float near,
    float far ) [static]
```

Creates a matrix for ortographic projection.

**Parameters**

<i>left</i>	a left coordinate
<i>right</i>	a right coordinate
<i>bottom</i>	a bottom coordinate
<i>top</i>	a top coordinate
<i>near</i>	a near coordinate
<i>far</i>	a far coordinate

**Returns**

an ortographic-projection matrix

**6.25.3.3 CreatePerspective()**

```
static MVCOMMON_API Matrix4x4f MVCommon::Matrix4x4f::CreatePerspective (
    float fieldOfView,
    float aspectRatio,
    float near,
    float far ) [static]
```

Creates a matrix for perspective projection.

**Parameters**

<i>fieldOfView</i>	a field of view (in degrees)
<i>aspectRatio</i>	an aspect ratio
<i>near</i>	a near coordinate
<i>far</i>	a far coordinate

**Returns**

a perspective-projection matrix

**6.25.3.4 CreateRotationAroundAxis()**

```
static MVCOMMON_API Matrix4x4f MVCommon::Matrix4x4f::CreateRotationAroundAxis (
    Vector3f const & axis,
    float angle ) [static]
```

Creates a rotation matrix from axis of rotation and an angle (in degrees).

**Parameters**

<i>axis</i>	an axis of rotation
<i>angle</i>	an angle

**Returns**

a rotation matrix

**6.25.3.5 CreateRotationFromEulerAnglesZYX()**

```
static MVCOMMON_API Matrix4x4f MVCommon::Matrix4x4f::CreateRotationFromEulerAnglesZYX (
    Vector3f const & eulerAngles ) [static]
```

Creates a rotation matrix from Euler angles (in degrees) in eulerAngles.z -> eulerAngles.y -> eulerAngles.x order.

**Parameters**

<i>eulerAngles</i>	Euler angles of Z-Y-X rotation
--------------------	--------------------------------

**Returns**

a rotation matrix

**6.25.3.6 CreateRotationFromVersor()**

```
static MVCOMMON_API Matrix4x4f MVCommon::Matrix4x4f::CreateRotationFromVersor (
    Versorf const & versor ) [static]
```

Creates a rotation matrix from a versor.



**Parameters**

<i>versor</i>	a versor describing rotation
---------------	------------------------------

**Returns**

a rotation matrix

**6.25.3.7 CreateScale()**

```
static MVCOMMON_API Matrix4x4f MVCommon::Matrix4x4f::CreateScale (
    Vector3f const & scale ) [static]
```

Creates a scaling matrix.

**Parameters**

<i>scale</i>	a vector describing the scale
--------------	-------------------------------

**Returns**

a scaling matrix

**6.25.3.8 CreateTranslation()**

```
static MVCOMMON_API Matrix4x4f MVCommon::Matrix4x4f::CreateTranslation (
    Vector3f const & translation ) [static]
```

Creates a translation matrix.

**Parameters**

<i>translation</i>	a vector describing the translation
--------------------	-------------------------------------

**Returns**

a translation matrix

**6.25.3.9 CreateZero()**

```
static MVCOMMON_API Matrix4x4f MVCommon::Matrix4x4f::CreateZero ( ) [static]
```

Creates a matrix with all elements set to zero.

**Returns**

a zero matrix

**6.25.3.10 FromRawBytes()**

```
static MVCOMMON_API Matrix4x4f MVCommon::Matrix4x4f::FromRawBytes (
    ByteArray & bytes,
    bool consumeBytes = false ) [static]
```

Deserializes matrix from a byte array.

**Parameters**

<i>bytes</i>	an array of matrix bytes
<i>consumeBytes</i>	determines whether bytes of the matrix shall be removed from the array

**Returns**

a matrix

**Exceptions**

<i>std::invalid_argument</i>	raised when there are not enough bytes in the array
------------------------------	---

**6.25.3.11 FromRawElements()**

```
static MVCOMMON_API Matrix4x4f MVCommon::Matrix4x4f::FromRawElements (
    float const * elements ) [static]
```

Deserializes matrix from an elements array.

**Parameters**

<i>elements</i>	an array of 4x4 elements
-----------------	--------------------------

**Returns**

a matrix

### 6.25.3.12 FromString()

```
static MVCOMMON_API Matrix4x4f MVCommon::Matrix4x4f::FromString (
    String const & str ) [static]
```

Creates a matrix from a human-readable string.

#### Parameters

<i>str</i>	a matrix string
------------	-----------------

#### Returns

a matrix

### 6.25.3.13 Inverted()

```
MVCOMMON_API Matrix4x4f MVCommon::Matrix4x4f::Inverted (
    bool & ok ) const
```

Creates an inverted matrix.

#### Parameters

<i>ok</i>	indicates whether the inverted matrix was successfully created, since it is not always possible to create one
-----------	---

#### Returns

an inverted matrix

### 6.25.3.14 operator[]() [1/2]

```
MVCOMMON_API Vector4f& MVCommon::Matrix4x4f::operator[] (
    size_t pos )
```

Accesses a specific row in the matrix.

#### Parameters

<i>pos</i>	an index of the row to access
------------	-------------------------------

#### Returns

a reference to the row of the matrix

**Exceptions**

<code>std::out_of_range</code>	raised when the index of the row is out of range (0-3)
--------------------------------	--

**6.25.3.15 operator[]()** [2/2]

```
MVCOMMON_API const Vector4f& MVCommon::Matrix4x4f::operator[] (
    size_t pos ) const
```

Accesses a specific row in the matrix.

**Parameters**

<i>pos</i>	an index of the row to access
------------	-------------------------------

**Returns**

a reference to the row of the matrix

**Exceptions**

<code>std::out_of_range</code>	raised when the index of the row is out of range (0-3)
--------------------------------	--

**6.25.3.16 RotationTranslationMatrixInverted()**

```
MVCOMMON_API Matrix4x4f MVCommon::Matrix4x4f::RotationTranslationMatrixInverted ( ) const
```

Creates an inverted matrix of a rotation-translation matrix.

It is always possible to create an inverted matrix of a rotation-translation matrix and the algorithm is much simpler and more effective than generic inversion algorithm. However, it is up to user to know what matrices he calls the function on

- the function assumes the matrix is a rotation-translation matrix.

**Returns**

an inverted matrix

**6.25.3.17 ToRawBytes()**

```
MVCOMMON_API void MVCommon::Matrix4x4f::ToRawBytes (
    ByteArray & bytes ) const
```

Serializes the matrix into a byte array.

## Parameters

<i>bytes</i>	a byte array to serialize into
--------------	--------------------------------

### 6.25.3.18 ToRawElements()

```
MVCOMMON_API void MVCommon::Matrix4x4f::ToRawElements (
    float * elements ) const
```

Serializes the matrix into an elements array.

## Parameters

<i>elements</i>	an array of 4x4 elements
-----------------	--------------------------

### 6.25.3.19 ToString()

```
MVCOMMON_API String MVCommon::Matrix4x4f::ToString ( ) const
```

Converts the matrix into a human-readable string.

## Returns

the matrix string

### 6.25.3.20 Transposed()

```
MVCOMMON_API Matrix4x4f MVCommon::Matrix4x4f::Transposed ( ) const
```

Creates a transposed matrix.

## Returns

a transposed matrix

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

- public/MVCommon/math/Matrix4x4f.h

## 6.26 MVCommon::Matrix4x4fHasher Struct Reference

A hasher for [Matrix4x4f](#) objects so they can be used in unordered collections.

```
#include <Matrix4x4f.h>
```

### Public Member Functions

- MVCOMMON\_API size\_t [operator\(\)](#) ([Matrix4x4f](#) const &matrix) const  
*Calculates a hash value from the object.*

#### 6.26.1 Detailed Description

A hasher for [Matrix4x4f](#) objects so they can be used in unordered collections.

#### 6.26.2 Member Function Documentation

##### 6.26.2.1 operator()

```
MVCOMMON_API size_t MVCommon::Matrix4x4fHasher::operator() (
    Matrix4x4f const & matrix ) const
```

Calculates a hash value from the object.

##### Parameters

<i>matrix</i>	an object to calculate the hash value of
---------------	--

##### Returns

hash value of the object

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

- public/MVCommon/math/Matrix4x4f.h

## 6.27 MVCommon::Pair< TFirst, TSecond > Class Template Reference

A pair of values.

```
#include <Pair.h>
```

## Public Types

- using `FirstType` = TFirst  
*A type of pair's first element.*
- using `SecondType` = TSecond  
*A type of pair's second element.*

## Public Member Functions

- `Pair` (TFirst const &`first`, TSecond const &`second`)  
*A constructor.*
- virtual `~Pair` ()  
*A destructor.*

## Data Fields

- TFirst `first`  
*A first value.*
- TSecond `second`  
*A second value.*

### 6.27.1 Detailed Description

```
template<typename TFirst, typename TSecond>
class MVCommon::Pair< TFirst, TSecond >
```

A pair of values.

#### Template Parameters

<i>TFirst</i>	a type of pair's first element
<i>TSecond</i>	a type of pair's second element

### 6.27.2 Constructor & Destructor Documentation

#### 6.27.2.1 Pair()

```
template<typename TFirst , typename TSecond >
MVCommon::Pair< TFirst, TSecond >::Pair (
    TFirst const & first,
    TSecond const & second ) [inline]
```

A constructor.

## Parameters

<i>first</i>	a first value
<i>second</i>	a second value

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

- public/MVCommon/utis/Pair.h

## 6.28 MVCommon::RedirectingLoggerSink Class Reference

A logger sink implementation for redirecting log messages to another logger.

```
#include <RedirectingLoggerSink.h>
```

Inherits [MVCommon::ILoggerSink](#).

### Public Member Functions

- MVCOMMON\_API [RedirectingLoggerSink](#) ([SharedLoggerPtr](#) spLogger, [LoggerLogLevel](#) logLevel=Logger↵LogLevel::LLL\_VERBOSE)  
*A constructor.*
- MVCOMMON\_API [~RedirectingLoggerSink](#) ()  
*A destructor.*

### Protected Member Functions

- virtual void [HandleLogEntry](#) ([LogEntry](#) const &logEntry) override  
*A callback executed when a new log entry is added.*

### Additional Inherited Members

#### 6.28.1 Detailed Description

A logger sink implementation for redirecting log messages to another logger.

All properties of log messages (timestamp, thread ID, ...) received from a logger are preserved when redirected to the target logger.

#### 6.28.2 Constructor & Destructor Documentation

##### 6.28.2.1 RedirectingLoggerSink()

```
MVCOMMON_API MVCommon::RedirectingLoggerSink::RedirectingLoggerSink (
    SharedLoggerPtr spLogger,
    LoggerLogLevel logLevel = LoggerLogLevel::LLL\_VERBOSE )
```

A constructor.



## Parameters

<i>spLogger</i>	a logger to redirect log messages to
<i>logLevel</i>	an initial log level (default value -> all log messages are processed)

## 6.28.3 Member Function Documentation

### 6.28.3.1 HandleLogEntry()

```
virtual void MVCommon::RedirectingLoggerSink::HandleLogEntry (
    LogEntry const & logEntry ) [override], [protected], [virtual]
```

A callback executed when a new log entry is added.

## Parameters

<i>logEntry</i>	a new log entry
-----------------	-----------------

Implements [MVCommon::ILoggerSink](#).

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

- public/MVCommon/logger/sinks/RedirectingLoggerSink.h

## 6.29 MVCommon::SharedGuidAliasDatabasePtr Class Reference

A shared smart-pointer to a guid alias database.

```
#include <SharedGuidAliasDatabasePtr.h>
```

### Public Member Functions

- MVCOMMON\_API [SharedGuidAliasDatabasePtr](#) ()  
*A constructor.*
- MVCOMMON\_API [SharedGuidAliasDatabasePtr](#) ([GuidAliasDatabase](#) \*pGuidAliasDatabase)  
*A constructor.*
- MVCOMMON\_API [SharedGuidAliasDatabasePtr](#) ([SharedGuidAliasDatabasePtr](#) const &other)  
*A copy-constructor.*
- MVCOMMON\_API [~SharedGuidAliasDatabasePtr](#) ()  
*A destructor.*
- MVCOMMON\_API [SharedGuidAliasDatabasePtr](#) & operator= ([SharedGuidAliasDatabasePtr](#) const &other)  
*Makes the pointer point to a guid alias database pointed to by the other pointer.*
- MVCOMMON\_API [SharedGuidAliasDatabasePtr](#) & operator= ([GuidAliasDatabase](#) \*pGuidAliasDatabase)

- *Makes the pointer point to a guid alias database.*  
MVCOMMON\_API `operator bool` () const
- *Converts the pointer to a boolean value.*  
MVCOMMON\_API `GuidAliasDatabase & operator*` () const
- *'Indirection' operator.*  
MVCOMMON\_API `GuidAliasDatabase * operator->` () const
- *'Dereference' operator.*  
MVCOMMON\_API `GuidAliasDatabase * Get` () const
- *Returns a raw pointer to the pointed-to guid alias database.*

## 6.29.1 Detailed Description

A shared smart-pointer to a guid alias database.

Allows sharing of the same guid alias database object by multiple owners and automatically destroys guid alias database objects when no more pointers point to them.

## 6.29.2 Constructor & Destructor Documentation

### 6.29.2.1 SharedGuidAliasDatabasePtr() [1/3]

```
MVCOMMON_API MVCommon::SharedGuidAliasDatabasePtr::SharedGuidAliasDatabasePtr ( )
```

A constructor.

Initializes the pointer with nullptr.

### 6.29.2.2 SharedGuidAliasDatabasePtr() [2/3]

```
MVCOMMON_API MVCommon::SharedGuidAliasDatabasePtr::SharedGuidAliasDatabasePtr (
    GuidAliasDatabase * pGuidAliasDatabase )
```

A constructor.

Parameters

<i>pGuidAliasDatabase</i>	a guid alias database to share a pointer to
---------------------------	---

### 6.29.2.3 SharedGuidAliasDatabasePtr() [3/3]

```
MVCOMMON_API MVCommon::SharedGuidAliasDatabasePtr::SharedGuidAliasDatabasePtr (
    SharedGuidAliasDatabasePtr const & other )
```

A copy-constructor.

## Parameters

<i>other</i>	other pointer to share a pointed-to guid alias database with
--------------	--

**6.29.2.4 ~SharedGuidAliasDatabasePtr()**

```
MVCOMMON_API MVCommon::SharedGuidAliasDatabasePtr::~~SharedGuidAliasDatabasePtr ( )
```

A destructor.

Destroys the pointed-to guid alias database if this was the last pointer pointing to it.

**6.29.3 Member Function Documentation****6.29.3.1 Get()**

```
MVCOMMON_API GuidAliasDatabase* MVCommon::SharedGuidAliasDatabasePtr::Get ( ) const
```

Returns a raw pointer to the pointed-to guid alias database.

**Returns**

a raw pointer to the pointed-to guid alias database

**6.29.3.2 operator bool()**

```
MVCOMMON_API MVCommon::SharedGuidAliasDatabasePtr::operator bool ( ) const
```

Converts the pointer to a boolean value.

**Returns**

true in case the pointed-to guid alias database is not null

### 6.29.3.3 operator\*()

```
MVCOMMON_API GuidAliasDatabase& MVCommon::SharedGuidAliasDatabasePtr::operator* ( ) const
```

'Indirection' operator.

#### Returns

a reference to the pointed-to guid alias database

### 6.29.3.4 operator->()

```
MVCOMMON_API GuidAliasDatabase* MVCommon::SharedGuidAliasDatabasePtr::operator-> ( ) const
```

'Dereference' operator.

#### Returns

a raw pointer to the pointed-to guid alias database

### 6.29.3.5 operator=( ) [1/2]

```
MVCOMMON_API SharedGuidAliasDatabasePtr& MVCommon::SharedGuidAliasDatabasePtr::operator= (
    GuidAliasDatabase * pGuidAliasDatabase )
```

Makes the pointer point to a guid alias database.

Destroys previously pointed-to guid alias database if this was the last pointer pointing to it.

#### Parameters

<i>pGuidAliasDatabase</i>	a guid alias database to point to
---------------------------	-----------------------------------

#### Returns

the pointer itself

### 6.29.3.6 operator=( ) [2/2]

```
MVCOMMON_API SharedGuidAliasDatabasePtr& MVCommon::SharedGuidAliasDatabasePtr::operator= (
    SharedGuidAliasDatabasePtr const & other )
```

Makes the pointer point to a guid alias database pointed to by the *other* pointer.

Destroys previously pointed-to guid alias database if this was the last pointer pointing to it.

## Parameters

<i>other</i>	other pointer to share a pointed-to guid alias database with
--------------	--

## Returns

the pointer itself

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

- public/MVCommon/guid/SharedGuidAliasDatabasePtr.h

## 6.30 MVCommon::SharedLoggerPtr Class Reference

A shared smart-pointer to a logger.

```
#include <SharedLoggerPtr.h>
```

### Public Member Functions

- MVCOMMON\_API [SharedLoggerPtr](#) ()  
*A constructor.*
- MVCOMMON\_API [SharedLoggerPtr](#) ([Logger](#) \*pLogger)  
*A constructor.*
- MVCOMMON\_API [SharedLoggerPtr](#) ([SharedLoggerPtr](#) const &other)  
*A copy-constructor.*
- MVCOMMON\_API [~SharedLoggerPtr](#) ()  
*A destructor.*
- MVCOMMON\_API [SharedLoggerPtr](#) & [operator=](#) ([SharedLoggerPtr](#) const &other)  
*Makes the pointer point to a logger pointed to by the *other* pointer.*
- MVCOMMON\_API [SharedLoggerPtr](#) & [operator=](#) ([Logger](#) \*pLogger)  
*Makes the pointer point to a logger.*
- MVCOMMON\_API [operator bool](#) () const  
*Converts the pointer to a boolean value.*
- MVCOMMON\_API [Logger](#) & [operator\\*](#) () const  
*'Indirection' operator.*
- MVCOMMON\_API [Logger](#) \* [operator->](#) () const  
*'Dereference' operator.*
- MVCOMMON\_API [Logger](#) \* [Get](#) () const  
*Returns a raw pointer to the pointed-to logger.*

### 6.30.1 Detailed Description

A shared smart-pointer to a logger.

Allows sharing of the same logger object by multiple owners and automatically destroys logger objects when no more pointers point to them.

## 6.30.2 Constructor & Destructor Documentation

### 6.30.2.1 SharedLoggerPtr() [1/3]

```
MVCOMMON_API MVCommon::SharedLoggerPtr::SharedLoggerPtr ( )
```

A constructor.

Initializes the pointer with nullptr.

### 6.30.2.2 SharedLoggerPtr() [2/3]

```
MVCOMMON_API MVCommon::SharedLoggerPtr::SharedLoggerPtr (
    Logger * pLogger )
```

A constructor.

Parameters

<i>pLogger</i>	a logger to share a pointer to
----------------	--------------------------------

### 6.30.2.3 SharedLoggerPtr() [3/3]

```
MVCOMMON_API MVCommon::SharedLoggerPtr::SharedLoggerPtr (
    SharedLoggerPtr const & other )
```

A copy-constructor.

Parameters

<i>other</i>	other pointer to share a pointed-to logger with
--------------	---

### 6.30.2.4 ~SharedLoggerPtr()

```
MVCOMMON_API MVCommon::SharedLoggerPtr::~~SharedLoggerPtr ( )
```

A destructor.

Destroys the pointed-to logger if this was the last pointer pointing to it.

### 6.30.3 Member Function Documentation

#### 6.30.3.1 Get()

```
MVCOMMON_API Logger* MVCommon::SharedLoggerPtr::Get ( ) const
```

Returns a raw pointer to the pointed-to logger.

**Returns**

a raw pointer to the pointed-to logger

#### 6.30.3.2 operator bool()

```
MVCOMMON_API MVCommon::SharedLoggerPtr::operator bool ( ) const
```

Converts the pointer to a boolean value.

**Returns**

true in case the pointed-to logger is not null

#### 6.30.3.3 operator\*()

```
MVCOMMON_API Logger& MVCommon::SharedLoggerPtr::operator* ( ) const
```

'Indirection' operator.

**Returns**

a reference to the pointed-to logger

#### 6.30.3.4 operator->()

```
MVCOMMON_API Logger* MVCommon::SharedLoggerPtr::operator-> ( ) const
```

'Dereference' operator.

**Returns**

a raw pointer to the pointed-to logger

#### 6.30.3.5 operator=( ) [1/2]

```
MVCOMMON_API SharedLoggerPtr& MVCommon::SharedLoggerPtr::operator= (   
    Logger * pLogger )
```

Makes the pointer point to a logger.

Destroys previously pointed-to logger if this was the last pointer pointing to it.



## Parameters

<i>pLogger</i>	a logger to point to
----------------	----------------------

## Returns

the pointer itself

**6.30.3.6 operator=()** [2/2]

```
MVCOMMON_API SharedLoggerPtr& MVCommon::SharedLoggerPtr::operator= (
    SharedLoggerPtr const & other )
```

Makes the pointer point to a logger pointed to by the *other* pointer.

Destroys previously pointed-to logger if this was the last pointer pointing to it.

## Parameters

<i>other</i>	other pointer to share a pointed-to logger with
--------------	---

## Returns

the pointer itself

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

- public/MVCommon/logger/SharedLoggerPtr.h

**6.31 MVCommon::SharedLoggerSinkPtr Class Reference**

A shared smart-pointer to a logger sink.

```
#include <SharedLoggerSinkPtr.h>
```

**Public Member Functions**

- MVCOMMON\_API [SharedLoggerSinkPtr](#) ()  
*A constructor.*
- MVCOMMON\_API [SharedLoggerSinkPtr](#) (ILoggerSink \*pLoggerSink)  
*A constructor.*
- MVCOMMON\_API [SharedLoggerSinkPtr](#) (SharedLoggerSinkPtr const &other)  
*A copy-constructor.*
- MVCOMMON\_API [~SharedLoggerSinkPtr](#) ()

*A destructor.*

- MVCOMMON\_API [SharedLoggerSinkPtr](#) & [operator=](#) ([SharedLoggerSinkPtr](#) const &other)

*Makes the pointer point to a logger sink pointed to by the `other` pointer.*

- MVCOMMON\_API [SharedLoggerSinkPtr](#) & [operator=](#) ([ILoggerSink](#) \*pLoggerSink)

*Makes the pointer point to a logger sink.*

- MVCOMMON\_API [operator bool](#) () const

*Converts the pointer to a boolean value.*

- MVCOMMON\_API [ILoggerSink](#) & [operator\\*](#) () const

*'Indirection' operator.*

- MVCOMMON\_API [ILoggerSink](#) \* [operator->](#) () const

*'Dereference' operator.*

- MVCOMMON\_API [ILoggerSink](#) \* [Get](#) () const

*Returns a raw pointer to the pointed-to logger sink.*

### 6.31.1 Detailed Description

A shared smart-pointer to a logger sink.

Allows sharing of the same logger sink object by multiple owners and automatically destroys logger sink objects when no more pointers point to them.

### 6.31.2 Constructor & Destructor Documentation

#### 6.31.2.1 SharedLoggerSinkPtr() [1/3]

```
MVCOMMON_API MVCommon::SharedLoggerSinkPtr::SharedLoggerSinkPtr ( )
```

A constructor.

Initializes the pointer with nullptr.

#### 6.31.2.2 SharedLoggerSinkPtr() [2/3]

```
MVCOMMON_API MVCommon::SharedLoggerSinkPtr::SharedLoggerSinkPtr (
    ILoggerSink * pLoggerSink )
```

A constructor.

Parameters

<i>pLoggerSink</i>	a logger sink to share a pointer to
--------------------	-------------------------------------

### 6.31.2.3 SharedLoggerSinkPtr() [3/3]

```
MVCOMMON_API MVCommon::SharedLoggerSinkPtr::SharedLoggerSinkPtr (
    SharedLoggerSinkPtr const & other )
```

A copy-constructor.

#### Parameters

<i>other</i>	other pointer to share a pointed-to logger sink with
--------------	--

### 6.31.2.4 ~SharedLoggerSinkPtr()

```
MVCOMMON_API MVCommon::SharedLoggerSinkPtr::~~SharedLoggerSinkPtr ( )
```

A destructor.

Destroys the pointed-to logger sink if this was the last pointer pointing to it.

## 6.31.3 Member Function Documentation

### 6.31.3.1 Get()

```
MVCOMMON_API ILoggerSink* MVCommon::SharedLoggerSinkPtr::Get ( ) const
```

Returns a raw pointer to the pointed-to logger sink.

#### Returns

a raw pointer to the pointed-to logger sink

### 6.31.3.2 operator bool()

```
MVCOMMON_API MVCommon::SharedLoggerSinkPtr::operator bool ( ) const
```

Converts the pointer to a boolean value.

#### Returns

true in case the pointed-to logger sink is not null

**6.31.3.3 operator\*()**

```
MVCOMMON_API ILoggerSink& MVCommon::SharedLoggerSinkPtr::operator* ( ) const
```

'Indirection' operator.

**Returns**

a reference to the pointed-to logger sink

**6.31.3.4 operator->()**

```
MVCOMMON_API ILoggerSink* MVCommon::SharedLoggerSinkPtr::operator-> ( ) const
```

'Dereference' operator.

**Returns**

a raw pointer to the pointed-to logger sink

**6.31.3.5 operator=() [1/2]**

```
MVCOMMON_API SharedLoggerSinkPtr& MVCommon::SharedLoggerSinkPtr::operator= (
    ILoggerSink * pLoggerSink )
```

Makes the pointer point to a logger sink.

Destroys previously pointed-to logger sink if this was the last pointer pointing to it.

**Parameters**

<i>pLoggerSink</i>	a logger sink to point to
--------------------	---------------------------

**Returns**

the pointer itself

**6.31.3.6 operator=() [2/2]**

```
MVCOMMON_API SharedLoggerSinkPtr& MVCommon::SharedLoggerSinkPtr::operator= (
    SharedLoggerSinkPtr const & other )
```

Makes the pointer point to a logger sink pointed to by the *other* pointer.

Destroys previously pointed-to logger sink if this was the last pointer pointing to it.

## Parameters

<i>other</i>	other pointer to share a pointed-to logger sink with
--------------	--

## Returns

the pointer itself

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

- public/MVCommon/logger/SharedLoggerSinkPtr.h

## 6.32 MVCommon::SharedThreadPoolJobPtr Class Reference

A shared smart-pointer to a thread pool job.

```
#include <SharedThreadPoolJobPtr.h>
```

### Public Member Functions

- MVCOMMON\_API [SharedThreadPoolJobPtr](#) ()  
*A constructor.*
- MVCOMMON\_API [SharedThreadPoolJobPtr](#) (IThreadPoolJob \*pJob)  
*A constructor.*
- MVCOMMON\_API [SharedThreadPoolJobPtr](#) (SharedThreadPoolJobPtr const &other)  
*A copy-constructor.*
- MVCOMMON\_API [~SharedThreadPoolJobPtr](#) ()  
*A destructor.*
- MVCOMMON\_API [SharedThreadPoolJobPtr](#) & operator= (SharedThreadPoolJobPtr const &other)  
*Makes the pointer point to a thread pool job pointed to by the *other* pointer.*
- MVCOMMON\_API [SharedThreadPoolJobPtr](#) & operator= (IThreadPoolJob \*pJob)  
*Makes the pointer point to a thread pool job.*
- MVCOMMON\_API [operator bool](#) () const  
*Converts the pointer to a boolean value.*
- MVCOMMON\_API IThreadPoolJob & operator\* () const  
*'Indirection' operator.*
- MVCOMMON\_API IThreadPoolJob \* operator-> () const  
*'Dereference' operator.*
- MVCOMMON\_API IThreadPoolJob \* Get () const  
*Returns a raw pointer to the pointed-to thread pool job.*

### 6.32.1 Detailed Description

A shared smart-pointer to a thread pool job.

Allows sharing of the same job object by multiple owners and automatically destroys job objects when no more pointers point to them.

## 6.32.2 Constructor & Destructor Documentation

### 6.32.2.1 SharedThreadPoolJobPtr() [1/3]

```
MVCOMMON_API MVCommon::SharedThreadPoolJobPtr::SharedThreadPoolJobPtr ( )
```

A constructor.

Initializes the pointer with nullptr.

### 6.32.2.2 SharedThreadPoolJobPtr() [2/3]

```
MVCOMMON_API MVCommon::SharedThreadPoolJobPtr::SharedThreadPoolJobPtr (
    IThreadPoolJob * pJob )
```

A constructor.

Parameters

<i>pJob</i>	a thread pool job to share a pointer to
-------------	---

### 6.32.2.3 SharedThreadPoolJobPtr() [3/3]

```
MVCOMMON_API MVCommon::SharedThreadPoolJobPtr::SharedThreadPoolJobPtr (
    SharedThreadPoolJobPtr const & other )
```

A copy-constructor.

Parameters

<i>other</i>	other pointer to share a pointed-to thread pool job with
--------------	--

### 6.32.2.4 ~SharedThreadPoolJobPtr()

```
MVCOMMON_API MVCommon::SharedThreadPoolJobPtr::~~SharedThreadPoolJobPtr ( )
```

A destructor.

Destroys the pointed-to thread pool job if this was the last pointer pointing to it.

### 6.32.3 Member Function Documentation

#### 6.32.3.1 Get()

```
MVCOMMON_API IThreadPoolJob* MVCommon::SharedThreadPoolJobPtr::Get ( ) const
```

Returns a raw pointer to the pointed-to thread pool job.

**Returns**

a raw pointer to the pointed-to job

#### 6.32.3.2 operator bool()

```
MVCOMMON_API MVCommon::SharedThreadPoolJobPtr::operator bool ( ) const
```

Converts the pointer to a boolean value.

**Returns**

true in case the pointed-to thread pool job is not null

#### 6.32.3.3 operator\*()

```
MVCOMMON_API IThreadPoolJob& MVCommon::SharedThreadPoolJobPtr::operator* ( ) const
```

'Indirection' operator.

**Returns**

a reference to the pointed-to thread pool job

#### 6.32.3.4 operator->()

```
MVCOMMON_API IThreadPoolJob* MVCommon::SharedThreadPoolJobPtr::operator-> ( ) const
```

'Dereference' operator.

**Returns**

a raw pointer to the pointed-to thread pool job

#### 6.32.3.5 operator=() [1/2]

```
MVCOMMON_API SharedThreadPoolJobPtr& MVCommon::SharedThreadPoolJobPtr::operator= (
    IThreadPoolJob * pJob )
```

Makes the pointer point to a thread pool job.

Destroys previously pointed-to thread pool job if this was the last pointer pointing to it.

**Parameters**

<i>pJob</i>	a thread pool job to point to
-------------	-------------------------------

**Returns**

the pointer itself

**6.32.3.6 operator=()** [2/2]

```
MVCOMMON_API SharedThreadPoolJobPtr& MVCommon::SharedThreadPoolJobPtr::operator= (
    SharedThreadPoolJobPtr const & other )
```

Makes the pointer point to a thread pool job pointed to by the *other* pointer.

Destroys previously pointed-to thread pool job if this was the last pointer pointing to it.

**Parameters**

<i>other</i>	other pointer to share a pointed-to thread pool job with
--------------	--

**Returns**

the pointer itself

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

- public/MVCommon/utils/threadpool/SharedThreadPoolJobPtr.h

**6.33 MVCommon::StdOutLoggerSink Class Reference**

A logger sink implementation for logging into a standard output.

```
#include <StdOutLoggerSink.h>
```

Inherits [MVCommon::ILoggerSink](#).

**Public Member Functions**

- MVCOMMON\_API [StdOutLoggerSink](#) (bool colorizeByLevel=false, [LoggerLogLevel](#) logLevel=LoggerLog↵  
Level::LLL\_VERBOSE)  
A constructor.
- MVCOMMON\_API [~StdOutLoggerSink](#) ()  
A destructor.



## Protected Member Functions

- virtual void [HandleLogEntry](#) ([LogEntry](#) const &logEntry) override  
*A callback executed when a new log entry is added.*

## Additional Inherited Members

### 6.33.1 Detailed Description

A logger sink implementation for logging into a standard output.

### 6.33.2 Constructor & Destructor Documentation

#### 6.33.2.1 StdOutLoggerSink()

```
MVCOMMON_API MVCommon::StdOutLoggerSink::StdOutLoggerSink (
    bool colorizeByLevel = false,
    LoggerLogLevel logLevel = LoggerLogLevel::LLL\_VERBOSE )
```

A constructor.

##### Parameters

<i>colorizeByLevel</i>	determines whether log messages shall be colorized based on their level - actual behaviour depends on the console used
<i>logLevel</i>	an initial log level (default value -> all log messages are processed)

### 6.33.3 Member Function Documentation

#### 6.33.3.1 HandleLogEntry()

```
virtual void MVCommon::StdOutLoggerSink::HandleLogEntry (
    LogEntry const & logEntry ) [override], [protected], [virtual]
```

A callback executed when a new log entry is added.

##### Parameters

<i>logEntry</i>	a new log entry
-----------------	-----------------

Implements [MVCommon::ILoggerSink](#).

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

- public/MVCommon/logger/sinks/StdOutLoggerSink.h

## 6.34 MVCommon::String Class Reference

A string implementation.

```
#include <String.h>
```

### Public Member Functions

- MVCOMMON\_API [String](#) (char const \*cString="")  
*A constructor.*
- MVCOMMON\_API [String](#) ([String](#) const &other)  
*A copy constructor.*
- MVCOMMON\_API [String](#) ([String](#) &&other)  
*A move constructor.*
- MVCOMMON\_API [~String](#) ()  
*A destructor.*
- MVCOMMON\_API const char \* [CStr](#) () const  
*Gets a C string.*
- MVCOMMON\_API size\_t [Length](#) () const  
*Gets length of the string.*
- MVCOMMON\_API [String Substr](#) (size\_t pos=0, size\_t len=-1) const  
*Generates a substring of the string.*

### 6.34.1 Detailed Description

A string implementation.

Manages lifetime of a string.

### 6.34.2 Constructor & Destructor Documentation

#### 6.34.2.1 [String\(\)](#) [1/3]

```
MVCOMMON_API MVCommon::String::String (
    char const * cString = "" )
```

A constructor.

## Parameters

<i>cString</i>	a C string
----------------	------------

**6.34.2.2 String()** [2/3]

```
MVCOMMON_API MVCommon::String::String (
    String const & other )
```

A copy constructor.

## Parameters

<i>other</i>	a string to make a copy of
--------------	----------------------------

**6.34.2.3 String()** [3/3]

```
MVCOMMON_API MVCommon::String::String (
    String && other )
```

A move constructor.

## Parameters

<i>other</i>	a string to move
--------------	------------------

**6.34.3 Member Function Documentation****6.34.3.1 CStr()**

```
MVCOMMON_API const char* MVCommon::String::CStr ( ) const
```

Gets a C string.

## Returns

a pointer to the C string

### 6.34.3.2 Length()

```
MVCOMMON_API size_t MVCommon::String::Length ( ) const
```

Gets length of the string.

#### Returns

string's length

### 6.34.3.3 Substr()

```
MVCOMMON_API String MVCommon::String::Substr (
    size_t pos = 0,
    size_t len = -1 ) const
```

Generates a substring of the string.

#### Parameters

<i>pos</i>	a starting position of the string to generate the substring from
<i>len</i>	a length of the substring (special value -1 means the rest of the original string)

#### Returns

the string's substring

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

- public/MVCommon/Utils/String.h

## 6.35 MVCommon::StringHasher Struct Reference

A hasher for [String](#) objects so they can be used in unordered collections.

```
#include <String.h>
```

### Public Member Functions

- MVCOMMON\_API size\_t [operator\(\)](#) ([String](#) const &string) const  
*Calculates a hash value from the object.*

### 6.35.1 Detailed Description

A hasher for [String](#) objects so they can be used in unordered collections.

## 6.35.2 Member Function Documentation

### 6.35.2.1 operator()()

```
MVCOMMON_API size_t MVCommon::StringHasher::operator() (
    String const & string ) const
```

Calculates a hash value from the object.

#### Parameters

<i>string</i>	an object to calculate the hash value of
---------------	--

#### Returns

hash value of the object

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

- public/MVCommon/Utils/String.h

## 6.36 MVCommon::ThreadPool Class Reference

A pool of threads.

```
#include <ThreadPool.h>
```

Inherits NonAssignable.

### Public Member Functions

- MVCOMMON\_API [ThreadPool](#) (uint32\_t threadsCount=1, int32\_t waitersForUnoccupiedThreadsCount↔ Hint=1)  
*A constructor.*
- MVCOMMON\_API [~ThreadPool](#) ()  
*A destructor.*
- MVCOMMON\_API uint32\_t [GetThreadsCount](#) () const  
*Gets the threads count.*
- MVCOMMON\_API bool [DoJob](#) (SharedThreadPoolJobPtr spJob)  
*Instructs the pool to execute a job on an unoccupied thread.*
- MVCOMMON\_API bool [HasUnoccupiedThreads](#) () const  
*Determines whether there are unoccupied threads available in the pool.*
- MVCOMMON\_API uint32\_t [GetUnoccupiedThreadsCount](#) () const  
*Determines a count of unoccupied threads in the pool.*
- MVCOMMON\_API void [WaitForAnUnoccupiedThread](#) () const  
*Blocks current thread until there is at least one unoccupied thread in the pool.*
- MVCOMMON\_API void [ResetJobs](#) ()  
*Resets all jobs and threads executed by the pool.*

### 6.36.1 Detailed Description

A pool of threads.

The pool maintains a fixed-size collection of threads usable for executing jobs. It hides the details about creation and maintenance of threads and about dispatching of jobs to them, allowing a user to focus on the jobs themselves.

### 6.36.2 Constructor & Destructor Documentation

#### 6.36.2.1 ThreadPool()

```
MVCOMMON_API MVCommon::ThreadPool::ThreadPool (
    uint32_t threadsCount = 1,
    int32_t waitersForUnoccupiedThreadsCountHint = 1 )
```

A constructor.

Instantiates the threads.

##### Parameters

<i>threadsCount</i>	a count of threads the pool maintains
<i>waitersForUnoccupiedThreadsCountHint</i>	a hint about expected count of threads calling <a href="#">WaitForAnUnoccupiedThread</a> - it allows an optimization of internal memory allocations made per each waiting call in cases when count of parallel waiters can be predicted. Special value 0 will result in allocations made every time, and negative hint value results in no deallocations (and thus maximum reusability of the memory) during the entire lifetime of the thread pool.

#### 6.36.2.2 ~ThreadPool()

```
MVCOMMON_API MVCommon::ThreadPool::~~ThreadPool ( )
```

A destructor.

Shuts down all maintained threads and waits until they complete their execution.

### 6.36.3 Member Function Documentation

#### 6.36.3.1 DoJob()

```
MVCOMMON_API bool MVCommon::ThreadPool::DoJob (
    SharedThreadPoolJobPtr spJob )
```

Instructs the pool to execute a job on an unoccupied thread.

**Parameters**

<i>spJob</i>	a job to execute
--------------	------------------

**Returns**

true in case there is an unoccupied thread that will execute the job, false otherwise

**6.36.3.2 GetThreadsCount()**

```
MVCOMMON_API uint32_t MVCommon::ThreadPool::GetThreadsCount ( ) const
```

Gets the threads count.

**Returns**

the count of threads in the pool

**6.36.3.3 GetUnoccupiedThreadsCount()**

```
MVCOMMON_API uint32_t MVCommon::ThreadPool::GetUnoccupiedThreadsCount ( ) const
```

Determines a count of unoccupied threads in the pool.

**Returns**

a count of currently unoccupied threads

**6.36.3.4 HasUnoccupiedThreads()**

```
MVCOMMON_API bool MVCommon::ThreadPool::HasUnoccupiedThreads ( ) const
```

Determines whether there are unoccupied threads available in the pool.

**Returns**

true if there is at least one unoccupied thread, false otherwise

### 6.36.3.5 ResetJobs()

```
MVCOMMON_API void MVCommon::ThreadPool::ResetJobs ( )
```

Resets all jobs and threads executed by the pool.

Waits until all jobs are completed, shuts down all the threads and reinitializes them.

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

- public/MVCommon/utils/threadpool/ThreadPool.h

## 6.37 MVCommon::Vector2d Struct Reference

A 2-dimensional vector with double-precision floating-point values.

```
#include <Vector2d.h>
```

### Public Member Functions

- MVCOMMON\_API [Vector2d](#) ()  
*A default constructor.*
- MVCOMMON\_API [Vector2d](#) (double *x*, double *y*)  
*A constructor.*
- MVCOMMON\_API [String ToString](#) () const  
*Converts the vector into a human-readable string.*
- MVCOMMON\_API void [ToRawBytes](#) ([ByteArray](#) &bytes) const  
*Serializes the vector into a byte array.*
- MVCOMMON\_API double [Length](#) () const  
*Gets a length of the vector.*
- MVCOMMON\_API [Vector2d Inverted](#) () const  
*Creates a vector with inverted dimensions (1/x).*
- MVCOMMON\_API [Vector2d Normalized](#) () const  
*Creates a normalized vector (with length equal to 1).*
- MVCOMMON\_API [Vector2d Abs](#) () const  
*Creates a vector with dimensions with absolute values of the original vector.*
- MVCOMMON\_API double & [operator\[\]](#) (size\_t pos)  
*Accesses vector dimension value via index.*
- MVCOMMON\_API const double & [operator\[\]](#) (size\_t pos) const  
*Accesses vector dimension value via index.*

### Static Public Member Functions

- static MVCOMMON\_API [Vector2d FromString](#) ([String](#) const &str)  
*Creates a vector from a human-readable string.*
- static MVCOMMON\_API [Vector2d FromRawBytes](#) ([ByteArray](#) &bytes, bool consumeBytes=false)  
*Deserializes vector from a byte array.*
- static MVCOMMON\_API double [Dot](#) ([Vector2d](#) const &lhs, [Vector2d](#) const &rhs)  
*Calculates a dot product of two vectors.*



## Data Fields

- double [x](#)  
*An x coordinate.*
- double [y](#)  
*A y coordinate.*

## Static Public Attributes

- static const MVCOMMON\_API size\_t [RAW\\_BYTES\\_SIZE](#)  
*A count of bytes the vector requires in a serialized form.*

### 6.37.1 Detailed Description

A 2-dimensional vector with double-precision floating-point values.

### 6.37.2 Constructor & Destructor Documentation

#### 6.37.2.1 Vector2d()

```
MVCOMMON_API MVCommon::Vector2d::Vector2d (  
    double x,  
    double y )
```

A constructor.

#### Parameters

<i>x</i>	an x coordinate
<i>y</i>	a y coordinate

### 6.37.3 Member Function Documentation

#### 6.37.3.1 Abs()

```
MVCOMMON_API Vector2d MVCommon::Vector2d::Abs ( ) const
```

Creates a vector with dimensions with absolute values of the original vector.

#### Returns

a vector with absolute-valued dimensions

### 6.37.3.2 Dot()

```
static MVCOMMON_API double MVCommon::Vector2d::Dot (  
    Vector2d const & lhs,  
    Vector2d const & rhs ) [static]
```

Calculates a dot product of two vectors.

#### Parameters

<i>lhs</i>	a first vector-operand
<i>rhs</i>	a second vector-operand

#### Returns

a dot product

### 6.37.3.3 FromRawBytes()

```
static MVCOMMON_API Vector2d MVCommon::Vector2d::FromRawBytes (  
    ByteArray & bytes,  
    bool consumeBytes = false ) [static]
```

Deserializes vector from a byte array.

#### Parameters

<i>bytes</i>	an array of vector bytes
<i>consumeBytes</i>	determines whether bytes of the vector shall be removed from the array

#### Returns

a vector

#### Exceptions

<i>std::invalid_argument</i>	raised when there are not enough bytes in the array
------------------------------	---

### 6.37.3.4 FromString()

```
static MVCOMMON_API Vector2d MVCommon::Vector2d::FromString (  
    String const & str ) [static]
```

Creates a vector from a human-readable string.

**Parameters**

<i>str</i>	a vector string
------------	-----------------

**Returns**

a vector

**6.37.3.5 Inverted()**

```
MVCOMMON_API Vector2d MVCommon::Vector2d::Inverted ( ) const
```

Creates a vector with inverted dimensions (1/x).

**Returns**

an inverted vector

**6.37.3.6 Length()**

```
MVCOMMON_API double MVCommon::Vector2d::Length ( ) const
```

Gets a length of the vector.

**Returns**

vector's length

**6.37.3.7 Normalized()**

```
MVCOMMON_API Vector2d MVCommon::Vector2d::Normalized ( ) const
```

Creates a normalized vector (with length equal to 1).

Returns an unchanged vector in case its length is equal to 0.

**Returns**

a normalized vector

**6.37.3.8 operator[]() [1/2]**

```
MVCOMMON_API double& MVCommon::Vector2d::operator[] (
    size_t pos )
```

Accesses vector dimension value via index.

**Parameters**

<i>pos</i>	an index of the dimension to access
------------	-------------------------------------

**Returns**

a reference to the dimension value

**Exceptions**

<i>std::out_of_range</i>	raised when index is out of range (0-1)
--------------------------	---

**6.37.3.9 operator[]() [2/2]**

```
MVCOMMON_API const double& MVCommon::Vector2d::operator[] (
    size_t pos ) const
```

Accesses vector dimension value via index.

**Parameters**

<i>pos</i>	an index of the dimension to access
------------	-------------------------------------

**Returns**

a reference to the dimension value

**Exceptions**

<i>std::out_of_range</i>	raised when index is out of range (0-1)
--------------------------	---

**6.37.3.10 ToRawBytes()**

```
MVCOMMON_API void MVCommon::Vector2d::ToRawBytes (
    ByteArray & bytes ) const
```

Serializes the vector into a byte array.

**Parameters**

<i>bytes</i>	a byte array to serialize into
--------------	--------------------------------

### 6.37.3.11 ToString()

```
MVCOMMON_API String MVCommon::Vector2d::ToString ( ) const
```

Converts the vector into a human-readable string.

#### Returns

the vector string

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

- public/MVCommon/math/Vector2d.h

## 6.38 MVCommon::Vector2dHasher Struct Reference

A hasher for [Vector2d](#) objects so they can be used in unordered collections.

```
#include <Vector2d.h>
```

### Public Member Functions

- MVCOMMON\_API size\_t [operator\(\)](#) ([Vector2d](#) const &vector) const  
*Calculates a hash value from the object.*

### 6.38.1 Detailed Description

A hasher for [Vector2d](#) objects so they can be used in unordered collections.

### 6.38.2 Member Function Documentation

#### 6.38.2.1 operator()()

```
MVCOMMON_API size_t MVCommon::Vector2dHasher::operator() (
    Vector2d const & vector ) const
```

Calculates a hash value from the object.

#### Parameters

<i>vector</i>	an object to calculate the hash value of
---------------	--

**Returns**

hash value of the object

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

- public/MVCommon/math/Vector2d.h

## 6.39 MVCommon::Vector2f Struct Reference

A 2-dimensional vector with single-precision floating-point values.

```
#include <Vector2f.h>
```

### Public Member Functions

- MVCOMMON\_API [Vector2f](#) ()  
*A default constructor.*
- MVCOMMON\_API [Vector2f](#) (float x, float y)  
*A constructor.*
- MVCOMMON\_API [String ToString](#) () const  
*Converts the vector into a human-readable string.*
- MVCOMMON\_API void [ToRawBytes](#) ([ByteArray](#) &bytes) const  
*Serializes the vector into a byte array.*
- MVCOMMON\_API float [Length](#) () const  
*Gets a length of the vector.*
- MVCOMMON\_API [Vector2f Inverted](#) () const  
*Creates a vector with inverted dimensions (1/x).*
- MVCOMMON\_API [Vector2f Normalized](#) () const  
*Creates a normalized vector (with length equal to 1).*
- MVCOMMON\_API [Vector2f Abs](#) () const  
*Creates a vector with dimensions with absolute values of the original vector.*
- MVCOMMON\_API float & [operator\[\]](#) (size\_t pos)  
*Accesses vector dimension value via index.*
- MVCOMMON\_API const float & [operator\[\]](#) (size\_t pos) const  
*Accesses vector dimension value via index.*

### Static Public Member Functions

- static MVCOMMON\_API [Vector2f FromString](#) ([String](#) const &str)  
*Creates a vector from a human-readable string.*
- static MVCOMMON\_API [Vector2f FromRawBytes](#) ([ByteArray](#) &bytes, bool consumeBytes=false)  
*Deserializes vector from a byte array.*
- static MVCOMMON\_API float [Dot](#) ([Vector2f](#) const &lhs, [Vector2f](#) const &rhs)  
*Calculates a dot product of two vectors.*

## Data Fields

- float [x](#)  
*An x coordinate.*
- float [y](#)  
*A y coordinate.*

## Static Public Attributes

- static const MVCOMMON\_API size\_t [RAW\\_BYTES\\_SIZE](#)  
*A count of bytes the vector requires in a serialized form.*

### 6.39.1 Detailed Description

A 2-dimensional vector with single-precision floating-point values.

### 6.39.2 Constructor & Destructor Documentation

#### 6.39.2.1 Vector2f()

```
MVCOMMON_API MVCommon::Vector2f::Vector2f (  
    float x,  
    float y )
```

A constructor.

##### Parameters

<i>x</i>	an x coordinate
<i>y</i>	a y coordinate

### 6.39.3 Member Function Documentation

#### 6.39.3.1 Abs()

```
MVCOMMON_API Vector2f MVCommon::Vector2f::Abs ( ) const
```

Creates a vector with dimensions with absolute values of the original vector.

##### Returns

a vector with absolute-valued dimensions

### 6.39.3.2 Dot()

```
static MVCOMMON_API float MVCommon::Vector2f::Dot (
    Vector2f const & lhs,
    Vector2f const & rhs ) [static]
```

Calculates a dot product of two vectors.

#### Parameters

<i>lhs</i>	a first vector-operand
<i>rhs</i>	a second vector-operand

#### Returns

a dot product

### 6.39.3.3 FromRawBytes()

```
static MVCOMMON_API Vector2f MVCommon::Vector2f::FromRawBytes (
    ByteArray & bytes,
    bool consumeBytes = false ) [static]
```

Deserializes vector from a byte array.

#### Parameters

<i>bytes</i>	an array of vector bytes
<i>consumeBytes</i>	determines whether bytes of the vector shall be removed from the array

#### Returns

a vector

#### Exceptions

<i>std::invalid_argument</i>	raised when there are not enough bytes in the array
------------------------------	---

### 6.39.3.4 FromString()

```
static MVCOMMON_API Vector2f MVCommon::Vector2f::FromString (
    String const & str ) [static]
```

Creates a vector from a human-readable string.



**Parameters**

<i>str</i>	a vector string
------------	-----------------

**Returns**

a vector

**6.39.3.5 Inverted()**

```
MVCOMMON_API Vector2f MVCommon::Vector2f::Inverted ( ) const
```

Creates a vector with inverted dimensions (1/x).

**Returns**

an inverted vector

**6.39.3.6 Length()**

```
MVCOMMON_API float MVCommon::Vector2f::Length ( ) const
```

Gets a length of the vector.

**Returns**

vector's length

**6.39.3.7 Normalized()**

```
MVCOMMON_API Vector2f MVCommon::Vector2f::Normalized ( ) const
```

Creates a normalized vector (with length equal to 1).

Returns an unchanged vector in case its length is equal to 0.

**Returns**

a normalized vector

**6.39.3.8 operator[]() [1/2]**

```
MVCOMMON_API float& MVCommon::Vector2f::operator[] (
    size_t pos )
```

Accesses vector dimension value via index.

**Parameters**

<i>pos</i>	an index of the dimension to access
------------	-------------------------------------

**Returns**

a reference to the dimension value

**Exceptions**

<i>std::out_of_range</i>	raised when index is out of range (0-1)
--------------------------	---

**6.39.3.9 operator[]() [2/2]**

```
MVCOMMON_API const float& MVCommon::Vector2f::operator[] (
    size_t pos ) const
```

Accesses vector dimension value via index.

**Parameters**

<i>pos</i>	an index of the dimension to access
------------	-------------------------------------

**Returns**

a reference to the dimension value

**Exceptions**

<i>std::out_of_range</i>	raised when index is out of range (0-1)
--------------------------	---

**6.39.3.10 ToRawBytes()**

```
MVCOMMON_API void MVCommon::Vector2f::ToRawBytes (
    ByteArray & bytes ) const
```

Serializes the vector into a byte array.

**Parameters**

<i>bytes</i>	a byte array to serialize into
--------------	--------------------------------

### 6.39.3.11 ToString()

```
MVCOMMON_API String MVCommon::Vector2f::ToString ( ) const
```

Converts the vector into a human-readable string.

#### Returns

the vector string

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

- public/MVCommon/math/Vector2f.h

## 6.40 MVCommon::Vector2fHasher Struct Reference

A hasher for [Vector2f](#) objects so they can be used in unordered collections.

```
#include <Vector2f.h>
```

### Public Member Functions

- MVCOMMON\_API size\_t [operator\(\)](#) ([Vector2f](#) const &vector) const  
*Calculates a hash value from the object.*

### 6.40.1 Detailed Description

A hasher for [Vector2f](#) objects so they can be used in unordered collections.

### 6.40.2 Member Function Documentation

#### 6.40.2.1 operator()()

```
MVCOMMON_API size_t MVCommon::Vector2fHasher::operator() (
    Vector2f const & vector ) const
```

Calculates a hash value from the object.

#### Parameters

<i>vector</i>	an object to calculate the hash value of
---------------	--

**Returns**

hash value of the object

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

- public/MVCommon/math/Vector2f.h

## 6.41 MVCommon::Vector3d Struct Reference

A 3-dimensional vector with double-precision floating-point values.

```
#include <Vector3d.h>
```

### Public Member Functions

- MVCOMMON\_API [Vector3d](#) ()  
*A default constructor.*
- MVCOMMON\_API [Vector3d](#) (double [x](#), double [y](#), double [z](#))  
*A constructor.*
- MVCOMMON\_API [Vector3d](#) ([Vector2d](#) const &vector2, double [z](#)=0.0)  
*A constructor.*
- MVCOMMON\_API [String ToString](#) () const  
*Converts the vector into a human-readable string.*
- MVCOMMON\_API void [ToRawBytes](#) ([ByteArray](#) &bytes) const  
*Serializes the vector into a byte array.*
- MVCOMMON\_API double [Length](#) () const  
*Gets a length of the vector.*
- MVCOMMON\_API [Vector3d Inverted](#) () const  
*Creates a vector with inverted dimensions (1/x).*
- MVCOMMON\_API [Vector3d Normalized](#) () const  
*Creates a normalized vector (with length equal to 1).*
- MVCOMMON\_API [Vector3d Abs](#) () const  
*Creates a vector with dimensions with absolute values of the original vector.*
- MVCOMMON\_API [Vector2d GetXY](#) () const  
*Extracts x and y coordinates as a 2-dimensional vector.*
- MVCOMMON\_API double & [operator\[\]](#) (size\_t pos)  
*Accesses vector dimension value via index.*
- MVCOMMON\_API const double & [operator\[\]](#) (size\_t pos) const  
*Accesses vector dimension value via index.*

### Static Public Member Functions

- static MVCOMMON\_API [Vector3d FromString](#) ([String](#) const &str)  
*Creates a vector from a human-readable string.*
- static MVCOMMON\_API [Vector3d FromRawBytes](#) ([ByteArray](#) &bytes, bool consumeBytes=false)  
*Deserializes vector from a byte array.*
- static MVCOMMON\_API double [Dot](#) ([Vector3d](#) const &lhs, [Vector3d](#) const &rhs)  
*Calculates a dot product of two vectors.*
- static MVCOMMON\_API [Vector3d Cross](#) ([Vector3d](#) const &lhs, [Vector3d](#) const &rhs)  
*Calculates a cross product of two vectors.*

## Data Fields

- double [x](#)  
*An x coordinate.*
- double [y](#)  
*A y coordinate.*
- double [z](#)  
*A z coordinate.*

## Static Public Attributes

- static const MVCOMMON\_API size\_t [RAW\\_BYTES\\_SIZE](#)  
*A count of bytes the vector requires in a serialized form.*

### 6.41.1 Detailed Description

A 3-dimensional vector with double-precision floating-point values.

### 6.41.2 Constructor & Destructor Documentation

#### 6.41.2.1 Vector3d() [1/2]

```
MVCOMMON_API MVCommon::Vector3d::Vector3d (  
    double x,  
    double y,  
    double z )
```

A constructor.

##### Parameters

<a href="#">x</a>	an x coordinate
<a href="#">y</a>	a y coordinate
<a href="#">z</a>	a z coordinate

#### 6.41.2.2 Vector3d() [2/2]

```
MVCOMMON_API MVCommon::Vector3d::Vector3d (  
    Vector2d const & vector2,  
    double z = 0.0 )
```

A constructor.

**Parameters**

<i>vector2</i>	a 2-dimensional vector whose x and y coordinates will be grabbed
<i>z</i>	a z coordinate

### 6.41.3 Member Function Documentation

#### 6.41.3.1 Abs()

```
MVCOMMON_API Vector3d MVCommon::Vector3d::Abs ( ) const
```

Creates a vector with dimensions with absolute values of the original vector.

**Returns**

a vector with absolute-valued dimensions

#### 6.41.3.2 Cross()

```
static MVCOMMON_API Vector3d MVCommon::Vector3d::Cross (
    Vector3d const & lhs,
    Vector3d const & rhs ) [static]
```

Calculates a cross product of two vectors.

**Parameters**

<i>lhs</i>	a left-hand-side vector-operand
<i>rhs</i>	a right-hand-side vector-operand

**Returns**

a vector representing the cross product

#### 6.41.3.3 Dot()

```
static MVCOMMON_API double MVCommon::Vector3d::Dot (
    Vector3d const & lhs,
    Vector3d const & rhs ) [static]
```

Calculates a dot product of two vectors.

## Parameters

<i>lhs</i>	a first vector-operand
<i>rhs</i>	a second vector-operand

## Returns

a dot product

### 6.41.3.4 FromRawBytes()

```
static MVCOMMON_API Vector3d MVCommon::Vector3d::FromRawBytes (  
    ByteArray & bytes,  
    bool consumeBytes = false ) [static]
```

Deserializes vector from a byte array.

## Parameters

<i>bytes</i>	an array of vector bytes
<i>consumeBytes</i>	determines whether bytes of the vector shall be removed from the array

## Returns

a vector

## Exceptions

<i>std::invalid_argument</i>	raised when there are not enough bytes in the array
------------------------------	---

### 6.41.3.5 FromString()

```
static MVCOMMON_API Vector3d MVCommon::Vector3d::FromString (  
    String const & str ) [static]
```

Creates a vector from a human-readable string.

## Parameters

<i>str</i>	a vector string
------------	-----------------

**Returns**

a vector

**6.41.3.6 GetXY()**

```
MVCOMMON_API Vector2d MVCommon::Vector3d::GetXY ( ) const
```

Extracts x and y coordinates as a 2-dimensional vector.

**Returns**

a 2-dimensional vector

**6.41.3.7 Inverted()**

```
MVCOMMON_API Vector3d MVCommon::Vector3d::Inverted ( ) const
```

Creates a vector with inverted dimensions (1/x).

**Returns**

an inverted vector

**6.41.3.8 Length()**

```
MVCOMMON_API double MVCommon::Vector3d::Length ( ) const
```

Gets a length of the vector.

**Returns**

vector's length

**6.41.3.9 Normalized()**

```
MVCOMMON_API Vector3d MVCommon::Vector3d::Normalized ( ) const
```

Creates a normalized vector (with length equal to 1).

Returns an unchanged vector in case its length is equal to 0.

**Returns**

a normalized vector

**6.41.3.10 operator[]() [1/2]**

```
MVCOMMON_API double& MVCommon::Vector3d::operator[] (
    size_t pos )
```

Accesses vector dimension value via index.



## Parameters

<i>pos</i>	an index of the dimension to access
------------	-------------------------------------

## Returns

a reference to the dimension value

## Exceptions

<i>std::out_of_range</i>	raised when index is out of range (0-2)
--------------------------	---

**6.41.3.11 operator[]() [2/2]**

```
MVCOMMON_API const double& MVCommon::Vector3d::operator[] (
    size_t pos ) const
```

Accesses vector dimension value via index.

## Parameters

<i>pos</i>	an index of the dimension to access
------------	-------------------------------------

## Returns

a reference to the dimension value

## Exceptions

<i>std::out_of_range</i>	raised when index is out of range (0-2)
--------------------------	---

**6.41.3.12 ToRawBytes()**

```
MVCOMMON_API void MVCommon::Vector3d::ToRawBytes (
    ByteArray & bytes ) const
```

Serializes the vector into a byte array.

## Parameters

<i>bytes</i>	a byte array to serialize into
--------------	--------------------------------

### 6.41.3.13 ToString()

```
MVCOMMON_API String MVCommon::Vector3d::ToString ( ) const
```

Converts the vector into a human-readable string.

#### Returns

the vector string

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

- public/MVCommon/math/Vector3d.h

## 6.42 MVCommon::Vector3dHasher Struct Reference

A hasher for [Vector3d](#) objects so they can be used in unordered collections.

```
#include <Vector3d.h>
```

### Public Member Functions

- MVCOMMON\_API size\_t [operator\(\)](#) ([Vector3d](#) const &vector) const  
*Calculates a hash value from the object.*

#### 6.42.1 Detailed Description

A hasher for [Vector3d](#) objects so they can be used in unordered collections.

#### 6.42.2 Member Function Documentation

##### 6.42.2.1 operator()()

```
MVCOMMON_API size_t MVCommon::Vector3dHasher::operator() (
    Vector3d const & vector ) const
```

Calculates a hash value from the object.

#### Parameters

<i>vector</i>	an object to calculate the hash value of
---------------	--

**Returns**

hash value of the object

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

- public/MVCommon/math/Vector3d.h

## 6.43 MVCommon::Vector3f Struct Reference

A 3-dimensional vector with single-precision floating-point values.

```
#include <Vector3f.h>
```

### Public Member Functions

- MVCOMMON\_API [Vector3f](#) ()  
*A default constructor.*
- MVCOMMON\_API [Vector3f](#) (float [x](#), float [y](#), float [z](#))  
*A constructor.*
- MVCOMMON\_API [Vector3f](#) ([Vector2f](#) const &vector2, float [z](#)=0.0f)  
*A constructor.*
- MVCOMMON\_API [String](#) [ToString](#) () const  
*Converts the vector into a human-readable string.*
- MVCOMMON\_API void [ToRawBytes](#) ([ByteArray](#) &bytes) const  
*Serializes the vector into a byte array.*
- MVCOMMON\_API float [Length](#) () const  
*Gets a length of the vector.*
- MVCOMMON\_API [Vector3f](#) [Inverted](#) () const  
*Creates a vector with inverted dimensions (1/x).*
- MVCOMMON\_API [Vector3f](#) [Normalized](#) () const  
*Creates a normalized vector (with length equal to 1).*
- MVCOMMON\_API [Vector3f](#) [Abs](#) () const  
*Creates a vector with dimensions with absolute values of the original vector.*
- MVCOMMON\_API [Vector2f](#) [GetX](#) () const  
*Extracts x and y coordinates as a 2-dimensional vector.*
- MVCOMMON\_API float & [operator\[\]](#) (size\_t pos)  
*Accesses vector dimension value via index.*
- MVCOMMON\_API const float & [operator\[\]](#) (size\_t pos) const  
*Accesses vector dimension value via index.*

### Static Public Member Functions

- static MVCOMMON\_API [Vector3f](#) [FromString](#) ([String](#) const &str)  
*Creates a vector from a human-readable string.*
- static MVCOMMON\_API [Vector3f](#) [FromRawBytes](#) ([ByteArray](#) &bytes, bool consumeBytes=false)  
*Deserializes vector from a byte array.*
- static MVCOMMON\_API float [Dot](#) ([Vector3f](#) const &lhs, [Vector3f](#) const &rhs)  
*Calculates a dot product of two vectors.*
- static MVCOMMON\_API [Vector3f](#) [Cross](#) ([Vector3f](#) const &lhs, [Vector3f](#) const &rhs)  
*Calculates a cross product of two vectors.*

## Data Fields

- float [x](#)  
*An x coordinate.*
- float [y](#)  
*A y coordinate.*
- float [z](#)  
*A z coordinate.*

## Static Public Attributes

- static const MVCOMMON\_API size\_t [RAW\\_BYTES\\_SIZE](#)  
*A count of bytes the vector requires in a serialized form.*

### 6.43.1 Detailed Description

A 3-dimensional vector with single-precision floating-point values.

### 6.43.2 Constructor & Destructor Documentation

#### 6.43.2.1 [Vector3f\(\)](#) [1/2]

```
MVCOMMON_API MVCommon::Vector3f::Vector3f (
    float x,
    float y,
    float z )
```

A constructor.

#### Parameters

<a href="#">x</a>	an x coordinate
<a href="#">y</a>	a y coordinate
<a href="#">z</a>	a z coordinate

#### 6.43.2.2 [Vector3f\(\)](#) [2/2]

```
MVCOMMON_API MVCommon::Vector3f::Vector3f (
    Vector2f const & vector2,
    float z = 0.0f )
```

A constructor.

## Parameters

<i>vector2</i>	a 2-dimensional vector whose x and y coordinates will be grabbed
<i>z</i>	a z coordinate

### 6.43.3 Member Function Documentation

#### 6.43.3.1 Abs()

```
MVCOMMON_API Vector3f MVCommon::Vector3f::Abs ( ) const
```

Creates a vector with dimensions with absolute values of the original vector.

## Returns

a vector with absolute-valued dimensions

#### 6.43.3.2 Cross()

```
static MVCOMMON_API Vector3f MVCommon::Vector3f::Cross (
    Vector3f const & lhs,
    Vector3f const & rhs ) [static]
```

Calculates a cross product of two vectors.

## Parameters

<i>lhs</i>	a left-hand-side vector-operand
<i>rhs</i>	a right-hand-side vector-operand

## Returns

a vector representing the cross product

#### 6.43.3.3 Dot()

```
static MVCOMMON_API float MVCommon::Vector3f::Dot (
    Vector3f const & lhs,
    Vector3f const & rhs ) [static]
```

Calculates a dot product of two vectors.

## Parameters

<i>lhs</i>	a first vector-operand
<i>rhs</i>	a second vector-operand

## Returns

a dot product

### 6.43.3.4 FromRawBytes()

```
static MVCOMMON_API Vector3f MVCommon::Vector3f::FromRawBytes (  
    ByteArray & bytes,  
    bool consumeBytes = false ) [static]
```

Deserializes vector from a byte array.

## Parameters

<i>bytes</i>	an array of vector bytes
<i>consumeBytes</i>	determines whether bytes of the vector shall be removed from the array

## Returns

a vector

## Exceptions

<i>std::invalid_argument</i>	raised when there are not enough bytes in the array
------------------------------	---

### 6.43.3.5 FromString()

```
static MVCOMMON_API Vector3f MVCommon::Vector3f::FromString (  
    String const & str ) [static]
```

Creates a vector from a human-readable string.

## Parameters

<i>str</i>	a vector string
------------	-----------------

**Returns**

a vector

**6.43.3.6 GetXY()**

```
MVCOMMON_API Vector2f MVCommon::Vector3f::GetXY ( ) const
```

Extracts x and y coordinates as a 2-dimensional vector.

**Returns**

a 2-dimensional vector

**6.43.3.7 Inverted()**

```
MVCOMMON_API Vector3f MVCommon::Vector3f::Inverted ( ) const
```

Creates a vector with inverted dimensions (1/x).

**Returns**

an inverted vector

**6.43.3.8 Length()**

```
MVCOMMON_API float MVCommon::Vector3f::Length ( ) const
```

Gets a length of the vector.

**Returns**

vector's length

**6.43.3.9 Normalized()**

```
MVCOMMON_API Vector3f MVCommon::Vector3f::Normalized ( ) const
```

Creates a normalized vector (with length equal to 1).

Returns an unchanged vector in case its length is equal to 0.

**Returns**

a normalized vector

**6.43.3.10 operator[]() [1/2]**

```
MVCOMMON_API float& MVCommon::Vector3f::operator[] (
    size_t pos )
```

Accesses vector dimension value via index.

**Parameters**

<i>pos</i>	an index of the dimension to access
------------	-------------------------------------

**Returns**

a reference to the dimension value

**Exceptions**

<i>std::out_of_range</i>	raised when index is out of range (0-2)
--------------------------	---

**6.43.3.11 operator[]() [2/2]**

```
MVCOMMON_API const float& MVCommon::Vector3f::operator[] (
    size_t pos ) const
```

Accesses vector dimension value via index.

**Parameters**

<i>pos</i>	an index of the dimension to access
------------	-------------------------------------

**Returns**

a reference to the dimension value

**Exceptions**

<i>std::out_of_range</i>	raised when index is out of range (0-2)
--------------------------	---

**6.43.3.12 ToRawBytes()**

```
MVCOMMON_API void MVCommon::Vector3f::ToRawBytes (
    ByteArray & bytes ) const
```

Serializes the vector into a byte array.

**Parameters**

<i>bytes</i>	a byte array to serialize into
--------------	--------------------------------



### 6.43.3.13 ToString()

```
MVCOMMON_API String MVCommon::Vector3f::ToString ( ) const
```

Converts the vector into a human-readable string.

#### Returns

the vector string

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

- public/MVCommon/math/Vector3f.h

## 6.44 MVCommon::Vector3fHasher Struct Reference

A hasher for [Vector3f](#) objects so they can be used in unordered collections.

```
#include <Vector3f.h>
```

### Public Member Functions

- MVCOMMON\_API size\_t [operator\(\)](#) ([Vector3f](#) const &vector) const  
*Calculates a hash value from the object.*

### 6.44.1 Detailed Description

A hasher for [Vector3f](#) objects so they can be used in unordered collections.

### 6.44.2 Member Function Documentation

#### 6.44.2.1 operator()()

```
MVCOMMON_API size_t MVCommon::Vector3fHasher::operator() (
    Vector3f const & vector ) const
```

Calculates a hash value from the object.

#### Parameters

<i>vector</i>	an object to calculate the hash value of
---------------	--

**Returns**

hash value of the object

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

- public/MVCommon/math/Vector3f.h

## 6.45 MVCommon::Vector4d Struct Reference

A 4-dimensional vector with double-precision floating-point values.

```
#include <Vector4d.h>
```

### Public Member Functions

- MVCOMMON\_API [Vector4d](#) ()  
*A default constructor.*
- MVCOMMON\_API [Vector4d](#) (double [x](#), double [y](#), double [z](#), double [w](#))  
*A constructor.*
- MVCOMMON\_API [Vector4d](#) ([Vector3d](#) const &vector3, double [w](#)=0.0)  
*A constructor.*
- MVCOMMON\_API [String ToString](#) () const  
*Converts the vector into a human-readable string.*
- MVCOMMON\_API void [ToRawBytes](#) ([ByteArray](#) &bytes) const  
*Serializes the vector into a byte array.*
- MVCOMMON\_API double [Length](#) () const  
*Gets a length of the vector.*
- MVCOMMON\_API [Vector4d Inverted](#) () const  
*Creates a vector with inverted dimensions (1/x).*
- MVCOMMON\_API [Vector4d Normalized](#) () const  
*Creates a normalized vector (with length equal to 1).*
- MVCOMMON\_API [Vector4d Abs](#) () const  
*Creates a vector with dimensions with absolute values of the original vector.*
- MVCOMMON\_API [Vector3d GetXYZ](#) () const  
*Extracts x, y and z coordinates as a 3-dimensional vector.*
- MVCOMMON\_API double & [operator\[\]](#) (size\_t pos)  
*Accesses vector dimension value via index.*
- MVCOMMON\_API const double & [operator\[\]](#) (size\_t pos) const  
*Accesses vector dimension value via index.*

### Static Public Member Functions

- static MVCOMMON\_API [Vector4d FromString](#) ([String](#) const &str)  
*Creates a vector from a human-readable string.*
- static MVCOMMON\_API [Vector4d FromRawBytes](#) ([ByteArray](#) &bytes, bool consumeBytes=false)  
*Deserializes vector from a byte array.*
- static MVCOMMON\_API double [Dot](#) ([Vector4d](#) const &lhs, [Vector4d](#) const &rhs)  
*Calculates a dot product of two vectors.*

## Data Fields

- double [x](#)  
*An x coordinate.*
- double [y](#)  
*A y coordinate.*
- double [z](#)  
*A z coordinate.*
- double [w](#)  
*A w coordinate.*

## Static Public Attributes

- static const MVCOMMON\_API size\_t [RAW\\_BYTES\\_SIZE](#)  
*A count of bytes the vector requires in a serialized form.*

### 6.45.1 Detailed Description

A 4-dimensional vector with double-precision floating-point values.

### 6.45.2 Constructor & Destructor Documentation

#### 6.45.2.1 Vector4d() [1/2]

```
MVCOMMON_API MVCommon::Vector4d::Vector4d (
    double x,
    double y,
    double z,
    double w )
```

A constructor.

##### Parameters

<i>x</i>	an x coordinate
<i>y</i>	a y coordinate
<i>z</i>	a z coordinate
<i>w</i>	a w coordinate

#### 6.45.2.2 Vector4d() [2/2]

```
MVCOMMON_API MVCommon::Vector4d::Vector4d (
    Vector3d const & vector3,
    double w = 0.0 )
```

A constructor.

#### Parameters

<i>vector3</i>	a 3-dimensional vector whose x, y and z coordinates will be grabbed
<i>w</i>	a w coordinate

## 6.45.3 Member Function Documentation

### 6.45.3.1 Abs()

```
MVCOMMON_API Vector4d MVCommon::Vector4d::Abs ( ) const
```

Creates a vector with dimensions with absolute values of the original vector.

#### Returns

a vector with absolute-valued dimensions

### 6.45.3.2 Dot()

```
static MVCOMMON_API double MVCommon::Vector4d::Dot (
    Vector4d const & lhs,
    Vector4d const & rhs ) [static]
```

Calculates a dot product of two vectors.

#### Parameters

<i>lhs</i>	a first vector-operand
<i>rhs</i>	a second vector-operand

#### Returns

a dot product

### 6.45.3.3 FromRawBytes()

```
static MVCOMMON_API Vector4d MVCommon::Vector4d::FromRawBytes (
    ByteArray & bytes,
    bool consumeBytes = false ) [static]
```

Deserializes vector from a byte array.

## Parameters

<i>bytes</i>	an array of vector bytes
<i>consumeBytes</i>	determines whether bytes of the vector shall be removed from the array

## Returns

a vector

## Exceptions

<i>std::invalid_argument</i>	raised when there are not enough bytes in the array
------------------------------	---

### 6.45.3.4 FromString()

```
static MVCOMMON_API Vector4d MVCommon::Vector4d::FromString (  
    String const & str ) [static]
```

Creates a vector from a human-readable string.

## Parameters

<i>str</i>	a vector string
------------	-----------------

## Returns

a vector

### 6.45.3.5 GetXYZ()

```
MVCOMMON_API Vector3d MVCommon::Vector4d::GetXYZ ( ) const
```

Extracts x, y and z coordinates as a 3-dimensional vector.

## Returns

a 3-dimensional vector

#### 6.45.3.6 Inverted()

```
MVCOMMON_API Vector4d MVCommon::Vector4d::Inverted ( ) const
```

Creates a vector with inverted dimensions (1/x).

##### Returns

an inverted vector

#### 6.45.3.7 Length()

```
MVCOMMON_API double MVCommon::Vector4d::Length ( ) const
```

Gets a length of the vector.

##### Returns

vector's length

#### 6.45.3.8 Normalized()

```
MVCOMMON_API Vector4d MVCommon::Vector4d::Normalized ( ) const
```

Creates a normalized vector (with length equal to 1).

Returns an unchanged vector in case its length is equal to 0.

##### Returns

a normalized vector

#### 6.45.3.9 operator[]() [1/2]

```
MVCOMMON_API double& MVCommon::Vector4d::operator[] (
    size_t pos )
```

Accesses vector dimension value via index.

##### Parameters

<i>pos</i>	an index of the dimension to access
------------	-------------------------------------

**Returns**

a reference to the dimension value

**Exceptions**

<code>std::out_of_range</code>	raised when index is out of range (0-3)
--------------------------------	---

**6.45.3.10 operator[]() [2/2]**

```
MVCOMMON_API const double& MVCommon::Vector4d::operator[] (
    size_t pos ) const
```

Accesses vector dimension value via index.

**Parameters**

<code>pos</code>	an index of the dimension to access
------------------	-------------------------------------

**Returns**

a reference to the dimension value

**Exceptions**

<code>std::out_of_range</code>	raised when index is out of range (0-3)
--------------------------------	---

**6.45.3.11 ToRawBytes()**

```
MVCOMMON_API void MVCommon::Vector4d::ToRawBytes (
    ByteArray & bytes ) const
```

Serializes the vector into a byte array.

**Parameters**

<code>bytes</code>	a byte array to serialize into
--------------------	--------------------------------

**6.45.3.12 ToString()**

```
MVCOMMON_API String MVCommon::Vector4d::ToString ( ) const
```

Converts the vector into a human-readable string.

**Returns**

the vector string

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

- public/MVCommon/math/Vector4d.h

## 6.46 MVCommon::Vector4dHasher Struct Reference

A hasher for [Vector4d](#) objects so they can be used in unordered collections.

```
#include <Vector4d.h>
```

### Public Member Functions

- MVCOMMON\_API size\_t [operator\(\)](#) ([Vector4d](#) const &vector) const  
*Calculates a hash value from the object.*

#### 6.46.1 Detailed Description

A hasher for [Vector4d](#) objects so they can be used in unordered collections.

#### 6.46.2 Member Function Documentation

##### 6.46.2.1 operator()()

```
MVCOMMON_API size_t MVCommon::Vector4dHasher::operator() (
    Vector4d const & vector ) const
```

Calculates a hash value from the object.

**Parameters**

<i>vector</i>	an object to calculate the hash value of
---------------	--

**Returns**

hash value of the object

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

- public/MVCommon/math/Vector4d.h



## 6.47 MVCommon::Vector4f Struct Reference

A 4-dimensional vector with single-precision floating-point values.

```
#include <Vector4f.h>
```

### Public Member Functions

- MVCOMMON\_API [Vector4f](#) ()  
*A default constructor.*
- MVCOMMON\_API [Vector4f](#) (float [x](#), float [y](#), float [z](#), float [w](#))  
*A constructor.*
- MVCOMMON\_API [Vector4f](#) ([Vector3f](#) const &vector3, float [w](#)=0.0f)  
*A constructor.*
- MVCOMMON\_API [String ToString](#) () const  
*Converts the vector into a human-readable string.*
- MVCOMMON\_API void [ToRawBytes](#) ([ByteArray](#) &bytes) const  
*Serializes the vector into a byte array.*
- MVCOMMON\_API float [Length](#) () const  
*Gets a length of the vector.*
- MVCOMMON\_API [Vector4f Inverted](#) () const  
*Creates a vector with inverted dimensions (1/x).*
- MVCOMMON\_API [Vector4f Normalized](#) () const  
*Creates a normalized vector (with length equal to 1).*
- MVCOMMON\_API [Vector4f Abs](#) () const  
*Creates a vector with dimensions with absolute values of the original vector.*
- MVCOMMON\_API [Vector3f GetXYZ](#) () const  
*Extracts x, y and z coordinates as a 3-dimensional vector.*
- MVCOMMON\_API float & [operator\[\]](#) (size\_t pos)  
*Accesses vector dimension value via index.*
- MVCOMMON\_API const float & [operator\[\]](#) (size\_t pos) const  
*Accesses vector dimension value via index.*

### Static Public Member Functions

- static MVCOMMON\_API [Vector4f FromString](#) ([String](#) const &str)  
*Creates a vector from a human-readable string.*
- static MVCOMMON\_API [Vector4f FromRawBytes](#) ([ByteArray](#) &bytes, bool consumeBytes=false)  
*Deserializes vector from a byte array.*
- static MVCOMMON\_API float [Dot](#) ([Vector4f](#) const &lhs, [Vector4f](#) const &rhs)  
*Calculates a dot product of two vectors.*

### Data Fields

- float [x](#)  
*An x coordinate.*
- float [y](#)  
*A y coordinate.*
- float [z](#)  
*A z coordinate.*
- float [w](#)  
*A w coordinate.*

## Static Public Attributes

- static const MVCOMMON\_API size\_t [RAW\\_BYTES\\_SIZE](#)  
A count of bytes the vector requires in a serialized form.

### 6.47.1 Detailed Description

A 4-dimensional vector with single-precision floating-point values.

### 6.47.2 Constructor & Destructor Documentation

#### 6.47.2.1 Vector4f() [1/2]

```
MVCOMMON_API MVCommon::Vector4f::Vector4f (
    float x,
    float y,
    float z,
    float w )
```

A constructor.

##### Parameters

<i>x</i>	an x coordinate
<i>y</i>	a y coordinate
<i>z</i>	a z coordinate
<i>w</i>	a w coordinate

#### 6.47.2.2 Vector4f() [2/2]

```
MVCOMMON_API MVCommon::Vector4f::Vector4f (
    Vector3f const & vector3,
    float w = 0.0f )
```

A constructor.

##### Parameters

<i>vector3</i>	a 3-dimensional vector whose x, y and z coordinates will be grabbed
<i>w</i>	a w coordinate

## 6.47.3 Member Function Documentation

### 6.47.3.1 Abs()

```
MVCOMMON_API Vector4f MVCommon::Vector4f::Abs ( ) const
```

Creates a vector with dimensions with absolute values of the original vector.

#### Returns

a vector with absolute-valued dimensions

### 6.47.3.2 Dot()

```
static MVCOMMON_API float MVCommon::Vector4f::Dot (
    Vector4f const & lhs,
    Vector4f const & rhs ) [static]
```

Calculates a dot product of two vectors.

#### Parameters

<i>lhs</i>	a first vector-operand
<i>rhs</i>	a second vector-operand

#### Returns

a dot product

### 6.47.3.3 FromRawBytes()

```
static MVCOMMON_API Vector4f MVCommon::Vector4f::FromRawBytes (
    ByteArray & bytes,
    bool consumeBytes = false ) [static]
```

Deserializes vector from a byte array.

#### Parameters

<i>bytes</i>	an array of vector bytes
<i>consumeBytes</i>	determines whether bytes of the vector shall be removed from the array

**Returns**

a vector

**Exceptions**

<code>std::invalid_argument</code>	raised when there are not enough bytes in the array
------------------------------------	---

**6.47.3.4 FromString()**

```
static MVCOMMON_API Vector4f MVCommon::Vector4f::FromString (
    String const & str ) [static]
```

Creates a vector from a human-readable string.

**Parameters**

<code>str</code>	a vector string
------------------	-----------------

**Returns**

a vector

**6.47.3.5 GetXYZ()**

```
MVCOMMON_API Vector3f MVCommon::Vector4f::GetXYZ ( ) const
```

Extracts x, y and z coordinates as a 3-dimensional vector.

**Returns**

a 3-dimensional vector

**6.47.3.6 Inverted()**

```
MVCOMMON_API Vector4f MVCommon::Vector4f::Inverted ( ) const
```

Creates a vector with inverted dimensions (1/x).

**Returns**

an inverted vector

### 6.47.3.7 Length()

```
MVCOMMON_API float MVCommon::Vector4f::Length ( ) const
```

Gets a length of the vector.

#### Returns

vector's length

### 6.47.3.8 Normalized()

```
MVCOMMON_API Vector4f MVCommon::Vector4f::Normalized ( ) const
```

Creates a normalized vector (with length equal to 1).

Returns an unchanged vector in case its length is equal to 0.

#### Returns

a normalized vector

### 6.47.3.9 operator[]() [1/2]

```
MVCOMMON_API float& MVCommon::Vector4f::operator[] (
    size_t pos )
```

Accesses vector dimension value via index.

#### Parameters

<i>pos</i>	an index of the dimension to access
------------	-------------------------------------

#### Returns

a reference to the dimension value

#### Exceptions

<i>std::out_of_range</i>	raised when index is out of range (0-3)
--------------------------	---

**6.47.3.10 operator[]()** [2/2]

```
MVCOMMON_API const float& MVCommon::Vector4f::operator[] (
    size_t pos ) const
```

Accesses vector dimension value via index.

**Parameters**

<i>pos</i>	an index of the dimension to access
------------	-------------------------------------

**Returns**

a reference to the dimension value

**Exceptions**

<i>std::out_of_range</i>	raised when index is out of range (0-3)
--------------------------	---

**6.47.3.11 ToRawBytes()**

```
MVCOMMON_API void MVCommon::Vector4f::ToRawBytes (
    ByteArray & bytes ) const
```

Serializes the vector into a byte array.

**Parameters**

<i>bytes</i>	a byte array to serialize into
--------------	--------------------------------

**6.47.3.12 ToString()**

```
MVCOMMON_API String MVCommon::Vector4f::ToString ( ) const
```

Converts the vector into a human-readable string.

**Returns**

the vector string

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

- public/MVCommon/math/Vector4f.h

## 6.48 MVCommon::Vector4fHasher Struct Reference

A hasher for [Vector4f](#) objects so they can be used in unordered collections.

```
#include <Vector4f.h>
```

### Public Member Functions

- MVCOMMON\_API size\_t [operator\(\)](#) ([Vector4f](#) const &vector) const  
*Calculates a hash value from the object.*

#### 6.48.1 Detailed Description

A hasher for [Vector4f](#) objects so they can be used in unordered collections.

#### 6.48.2 Member Function Documentation

##### 6.48.2.1 [operator\(\)](#)

```
MVCOMMON_API size_t MVCommon::Vector4fHasher::operator() (  
    Vector4f const & vector ) const
```

Calculates a hash value from the object.

##### Parameters

<a href="#">vector</a>	an object to calculate the hash value of
------------------------	--

##### Returns

hash value of the object

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

- public/MVCommon/math/Vector4f.h

## 6.49 MVCommon::VersionInfo Struct Reference

A structure holding module version information.

```
#include <VersionInfo.h>
```

## Public Member Functions

- MVCOMMON\_API [VersionInfo](#) (uint32\_t [major](#)=0, uint32\_t [minor](#)=0, uint32\_t [patch](#)=0)  
*A constructor.*
- MVCOMMON\_API [MVCommon::String ToString](#) () const  
*Converts the version info into a string with format 'major.minor.patch'.*

## Data Fields

- uint32\_t [major](#)  
*Most-significant version component.*
- uint32\_t [minor](#)  
*Medium-significant version component.*
- uint32\_t [patch](#)  
*Least-significant version component.*

### 6.49.1 Detailed Description

A structure holding module version information.

### 6.49.2 Constructor & Destructor Documentation

#### 6.49.2.1 VersionInfo()

```
MVCOMMON_API MVCommon::VersionInfo::VersionInfo (
    uint32_t major = 0,
    uint32_t minor = 0,
    uint32_t patch = 0 )
```

A constructor.

#### Parameters

<i>major</i>	most-significant version component
<i>minor</i>	medium-significant version component
<i>patch</i>	least-significant version component

### 6.49.3 Member Function Documentation



### 6.49.3.1 ToString()

```
MVCOMMON_API MVCommon::String MVCommon::VersionInfo::ToString ( ) const
```

Converts the version info into a string with format 'major.minor.patch'.

#### Returns

a string containing version

## 6.49.4 Field Documentation

### 6.49.4.1 major

```
uint32_t MVCommon::VersionInfo::major
```

Most-significant version component.

Difference indicates binary-incompatibility.

### 6.49.4.2 minor

```
uint32_t MVCommon::VersionInfo::minor
```

Medium-significant version component.

Increased whenever a new official version is released.

### 6.49.4.3 patch

```
uint32_t MVCommon::VersionInfo::patch
```

Least-significant version component.

Increased whenever an officially released version is patched and re-released.

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

- public/MVCommon/utis/[VersionInfo.h](#)

## 6.50 MVCommon::VersionInfoHasher Struct Reference

A hasher for [VersionInfo](#) objects so they can be used in unordered collections.

```
#include <VersionInfo.h>
```

## Public Member Functions

- MVCOMMON\_API size\_t [operator\(\)](#) ([VersionInfo](#) const &versionInfo) const  
*Calculates a hash value from the object.*

### 6.50.1 Detailed Description

A hasher for [VersionInfo](#) objects so they can be used in unordered collections.

### 6.50.2 Member Function Documentation

#### 6.50.2.1 operator()

```
MVCOMMON_API size_t MVCommon::VersionInfoHasher::operator() (
    VersionInfo const & versionInfo ) const
```

Calculates a hash value from the object.

#### Parameters

<i>versionInfo</i>	an object to calculate the hash value of
--------------------	--

#### Returns

hash value of the object

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

- public/MVCommon/utills/[VersionInfo.h](#)

## 6.51 MVCommon::Versord Struct Reference

A rotational quaternion (i.e. versor) with double-precision floating-point values.

```
#include <Versord.h>
```

## Public Member Functions

- MVCOMMON\_API [Versord](#) ()  
*A constructor of an identity versor (i.e. no rotation).*
- MVCOMMON\_API [String ToString](#) () const  
*Converts the versor into a human-readable string.*

- MVCOMMON\_API void [ToRawBytes](#) (ByteArray &bytes) const  
*Serializes the versor into a byte array.*
- MVCOMMON\_API [Vector4d ToElementsVector](#) () const  
*Converts the versor into a vector with values of versor's internal elements.*
- MVCOMMON\_API void [ToRawElements](#) (double \*elements) const  
*Serializes the versor into an elements array.*
- MVCOMMON\_API [Versord Inverted](#) () const  
*Creates an inverted versor.*
- MVCOMMON\_API [Vector3d ToEulerAnglesZYX](#) () const  
*Converts the versor to Euler angles (in degrees) in z -> y -> x order.*

## Static Public Member Functions

- static MVCOMMON\_API [Versord FromString](#) (String const &str)  
*Creates a versor from a human-readable string.*
- static MVCOMMON\_API [Versord FromRawBytes](#) (ByteArray &bytes, bool consumeBytes=false)  
*Deserializes versor from a byte array.*
- static MVCOMMON\_API [Versord FromElementsVector](#) (Vector4d const &elements)  
*Creates a versor from a vector with values of versor's internal elements.*
- static MVCOMMON\_API [Versord FromRawElements](#) (double const \*elements)  
*Deserializes versor from an elements array.*
- static MVCOMMON\_API [Versord CreateRotationAroundAxis](#) (Vector3d const &axis, double angle)  
*Creates a versor from axis of rotation and an angle (in degrees).*
- static MVCOMMON\_API [Versord CreateRotationFromMatrix](#) (Matrix4x4d const &matrix)  
*Creates a versor from a rotational part of transformation matrix.*
- static MVCOMMON\_API [Versord CreateRotationFromEulerAnglesZYX](#) (Vector3d const &eulerAngles)  
*Creates a versor from Euler angles (in degrees) in eulerAngles.z -> eulerAngles.y -> eulerAngles.x order.*

## Static Public Attributes

- static const MVCOMMON\_API size\_t [RAW\\_BYTES\\_SIZE](#)  
*A count of bytes the versor requires in a serialized form.*

### 6.51.1 Detailed Description

A rotational quaternion (i.e. versor) with double-precision floating-point values.

### 6.51.2 Member Function Documentation

#### 6.51.2.1 CreateRotationAroundAxis()

```
static MVCOMMON_API Versord MVCommon::Versord::CreateRotationAroundAxis (
    Vector3d const & axis,
    double angle ) [static]
```

Creates a versor from axis of rotation and an angle (in degrees).

**Parameters**

<i>axis</i>	an axis of rotation
<i>angle</i>	an angle

**Returns**

a versor

**6.51.2.2 CreateRotationFromEulerAnglesZYX()**

```
static MVCOMMON_API Versord MVCommon::Versord::CreateRotationFromEulerAnglesZYX (  
    Vector3d const & eulerAngles ) [static]
```

Creates a versor from Euler angles (in degrees) in *eulerAngles.z* -> *eulerAngles.y* -> *eulerAngles.x* order.

**Parameters**

<i>eulerAngles</i>	Euler angles of Z-Y-X rotation
--------------------	--------------------------------

**Returns**

a versor

**6.51.2.3 CreateRotationFromMatrix()**

```
static MVCOMMON_API Versord MVCommon::Versord::CreateRotationFromMatrix (  
    Matrix4x4d const & matrix ) [static]
```

Creates a versor from a rotational part of transformation matrix.

**Parameters**

<i>matrix</i>	a matrix to extract the rotation from
---------------	---------------------------------------

**Returns**

a versor

**6.51.2.4 FromElementsVector()**

```
static MVCOMMON_API Versord MVCommon::Versord::FromElementsVector (  
    Vector4d const & elements ) [static]
```

Creates a versor from a vector with values of versor's internal elements.

#### Parameters

<i>elements</i>	a vector with versor's internal elements
-----------------	--

#### Returns

a versor

#### Exceptions

<i>std::invalid_argument</i>	raied when the vector does not represent a rotational quaternion
------------------------------	--

### 6.51.2.5 FromRawBytes()

```
static MVCOMMON_API Versord MVCommon::Versord::FromRawBytes (  
    ByteArray & bytes,  
    bool consumeBytes = false ) [static]
```

Deserializes versor from a byte array.

#### Parameters

<i>bytes</i>	an array of versor bytes
<i>consumeBytes</i>	determines whether bytes of the versor shall be removed from the array

#### Returns

a versor

#### Exceptions

<i>std::invalid_argument</i>	raised when there are not enough bytes in the array or the bytes do not represent a rotational quaternion
------------------------------	---

### 6.51.2.6 FromRawElements()

```
static MVCOMMON_API Versord MVCommon::Versord::FromRawElements (  
    double const * elements ) [static]
```

Deserializes versor from an elements array.

**Parameters**

<i>elements</i>	an array of 4 elements
-----------------	------------------------

**Returns**

a versor

**Exceptions**

<i>std::invalid_argument</i>	raied when the elements do not represent a rotational quaternion
------------------------------	--

**6.51.2.7 FromString()**

```
static MVCOMMON_API Versord MVCommon::Versord::FromString (  
    String const & str ) [static]
```

Creates a versor from a human-readable string.

**Parameters**

<i>str</i>	a versor string
------------	-----------------

**Returns**

a versor

**Exceptions**

<i>std::invalid_argument</i>	raied when the string does not represent a rotational quaternion
------------------------------	--

**6.51.2.8 Inverted()**

```
MVCOMMON_API Versord MVCommon::Versord::Inverted ( ) const
```

Creates an inverted versor.

**Returns**

an inverted versor

### 6.51.2.9 ToElementsVector()

```
MVCOMMON_API Vector4d MVCommon::Versord::ToElementsVector ( ) const
```

Converts the versor into a vector with values of versor's internal elements.

#### Returns

a vector of versor's elements

### 6.51.2.10 ToEulerAnglesZYX()

```
MVCOMMON_API Vector3d MVCommon::Versord::ToEulerAnglesZYX ( ) const
```

Converts the versor to Euler angles (in degrees) in z -> y -> x order.

#### Returns

Euler angles of Z-Y-X rotation

### 6.51.2.11 ToRawBytes()

```
MVCOMMON_API void MVCommon::Versord::ToRawBytes (
    ByteArray & bytes ) const
```

Serializes the versor into a byte array.

#### Parameters

<i>bytes</i>	a byte array to serialize into
--------------	--------------------------------

### 6.51.2.12 ToRawElements()

```
MVCOMMON_API void MVCommon::Versord::ToRawElements (
    double * elements ) const
```

Serializes the versor into an elements array.

#### Parameters

<i>elements</i>	an array of 4 elements
-----------------	------------------------

### 6.51.2.13 ToString()

```
MVCOMMON_API String MVCommon::Versord::ToString ( ) const
```

Converts the versor into a human-readable string.

#### Returns

the versor string

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

- public/MVCommon/math/Versord.h

## 6.52 MVCommon::VersordHasher Struct Reference

A hasher for [Versord](#) objects so they can be used in unordered collections.

```
#include <Versord.h>
```

### Public Member Functions

- MVCOMMON\_API size\_t [operator\(\)](#) ([Versord](#) const &versor) const  
*Calculates a hash value from the object.*

#### 6.52.1 Detailed Description

A hasher for [Versord](#) objects so they can be used in unordered collections.

#### 6.52.2 Member Function Documentation

##### 6.52.2.1 operator()

```
MVCOMMON_API size_t MVCommon::VersordHasher::operator() (  
    Versord const & versor ) const
```

Calculates a hash value from the object.



## Parameters

<i>versorf</i>	an object to calculate the hash value of
----------------	--

## Returns

hash value of the object

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

- public/MVCommon/math/Versord.h

## 6.53 MVCommon::Versorf Struct Reference

A rotational quaternion (i.e. versorf) with single-precision floating-point values.

```
#include <Versorf.h>
```

### Public Member Functions

- MVCOMMON\_API [Versorf](#) ()  
*A constructor of an identity versorf (i.e. no rotation).*
- MVCOMMON\_API [String ToString](#) () const  
*Converts the versorf into a human-readable string.*
- MVCOMMON\_API void [ToRawBytes](#) (ByteArray &bytes) const  
*Serializes the versorf into a byte array.*
- MVCOMMON\_API [Vector4f ToElementsVector](#) () const  
*Converts the versorf into a vector with values of versorf's internal elements.*
- MVCOMMON\_API void [ToRawElements](#) (float \*elements) const  
*Serializes the versorf into an elements array.*
- MVCOMMON\_API [Versorf Inverted](#) () const  
*Creates an inverted versorf.*
- MVCOMMON\_API [Vector3f ToEulerAnglesZYX](#) () const  
*Converts the versorf to Euler angles (in degrees) in z -> y -> x order.*

### Static Public Member Functions

- static MVCOMMON\_API [Versorf FromString](#) (String const &str)  
*Creates a versorf from a human-readable string.*
- static MVCOMMON\_API [Versorf FromRawBytes](#) (ByteArray &bytes, bool consumeBytes=false)  
*Deserializes versorf from a byte array.*
- static MVCOMMON\_API [Versorf FromElementsVector](#) (Vector4f const &elements)  
*Creates a versorf from a vector with values of versorf's internal elements.*
- static MVCOMMON\_API [Versorf FromRawElements](#) (float const \*elements)  
*Deserializes versorf from an elements array.*
- static MVCOMMON\_API [Versorf CreateRotationAroundAxis](#) (Vector3f const &axis, float angle)  
*Creates a versorf from axis of rotation and an angle (in degrees).*
- static MVCOMMON\_API [Versorf CreateRotationFromMatrix](#) (Matrix4x4f const &matrix)  
*Creates a versorf from a rotational part of transformation matrix.*
- static MVCOMMON\_API [Versorf CreateRotationFromEulerAnglesZYX](#) (Vector3f const &eulerAngles)  
*Creates a versorf from Euler angles (in degrees) in eulerAngles.z -> eulerAngles.y -> eulerAngles.x order.*

## Static Public Attributes

- static const MVCOMMON\_API size\_t [RAW\\_BYTES\\_SIZE](#)  
*A count of bytes the versor requires in a serialized form.*

### 6.53.1 Detailed Description

A rotational quaternion (i.e. versor) with single-precision floating-point values.

### 6.53.2 Member Function Documentation

#### 6.53.2.1 CreateRotationAroundAxis()

```
static MVCOMMON_API Versorf MVCommon::Versorf::CreateRotationAroundAxis (  
    Vector3f const & axis,  
    float angle ) [static]
```

Creates a versor from axis of rotation and an angle (in degrees).

##### Parameters

<i>axis</i>	an axis of rotation
<i>angle</i>	an angle

##### Returns

a versor

#### 6.53.2.2 CreateRotationFromEulerAnglesZYX()

```
static MVCOMMON_API Versorf MVCommon::Versorf::CreateRotationFromEulerAnglesZYX (  
    Vector3f const & eulerAngles ) [static]
```

Creates a versor from Euler angles (in degrees) in eulerAngles.z -> eulerAngles.y -> eulerAngles.x order.

##### Parameters

<i>eulerAngles</i>	Euler angles of Z-Y-X rotation
--------------------	--------------------------------

##### Returns

a versor

### 6.53.2.3 CreateRotationFromMatrix()

```
static MVCOMMON_API Versorf MVCommon::Versorf::CreateRotationFromMatrix (
    Matrix4x4f const & matrix ) [static]
```

Creates a versor from a rotational part of transformation matrix.

#### Parameters

<i>matrix</i>	a matrix to extract the rotation from
---------------	---------------------------------------

#### Returns

a versor

### 6.53.2.4 FromElementsVector()

```
static MVCOMMON_API Versorf MVCommon::Versorf::FromElementsVector (
    Vector4f const & elements ) [static]
```

Creates a versor from a vector with values of versor's internal elements.

#### Parameters

<i>elements</i>	a vector with versor's internal elements
-----------------	--

#### Returns

a versor

#### Exceptions

<i>std::invalid_argument</i>	raied when the vector does not represent a rotational quaternion
------------------------------	--

### 6.53.2.5 FromRawBytes()

```
static MVCOMMON_API Versorf MVCommon::Versorf::FromRawBytes (
    ByteArray & bytes,
    bool consumeBytes = false ) [static]
```

Deserializes versor from a byte array.

## Parameters

<i>bytes</i>	an array of versor bytes
<i>consumeBytes</i>	determines whether bytes of the versor shall be removed from the array

## Returns

a versor

## Exceptions

<i>std::invalid_argument</i>	raised when there are not enough bytes in the array or the bytes do not represent a rotational quaternion
------------------------------	---

### 6.53.2.6 FromRawElements()

```
static MVCOMMON_API Versorf MVCommon::Versorf::FromRawElements (  
    float const * elements ) [static]
```

Deserializes versor from an elements array.

## Parameters

<i>elements</i>	an array of 4 elements
-----------------	------------------------

## Returns

a versor

## Exceptions

<i>std::invalid_argument</i>	raied when the elements do not represent a rotational quaternion
------------------------------	--

### 6.53.2.7 FromString()

```
static MVCOMMON_API Versorf MVCommon::Versorf::FromString (  
    String const & str ) [static]
```

Creates a versor from a human-readable string.

## Parameters

<i>str</i>	a versor string
------------	-----------------

**Returns**

a versor

**Exceptions**

<code>std::invalid_argument</code>	raised when the string does not represent a rotational quaternion
------------------------------------	---

**6.53.2.8 Inverted()**

```
MVCOMMON_API Versorf MVCommon::Versorf::Inverted ( ) const
```

Creates an inverted versor.

**Returns**

an inverted versor

**6.53.2.9 ToElementsVector()**

```
MVCOMMON_API Vector4f MVCommon::Versorf::ToElementsVector ( ) const
```

Converts the versor into a vector with values of versor's internal elements.

**Returns**

a vector of versor's elements

**6.53.2.10 ToEulerAnglesZYX()**

```
MVCOMMON_API Vector3f MVCommon::Versorf::ToEulerAnglesZYX ( ) const
```

Converts the versor to Euler angles (in degrees) in z -> y -> x order.

**Returns**

Euler angles of Z-Y-X rotation

**6.53.2.11 ToRawBytes()**

```
MVCOMMON_API void MVCommon::Versorf::ToRawBytes (
    ByteArray & bytes ) const
```

Serializes the versor into a byte array.

**Parameters**

<i>bytes</i>	a byte array to serialize into
--------------	--------------------------------

**6.53.2.12 ToRawElements()**

```
MVCOMMON_API void MVCommon::Versorf::ToRawElements (
    float * elements ) const
```

Serializes the versor into an elements array.

**Parameters**

<i>elements</i>	an array of 4 elements
-----------------	------------------------

**6.53.2.13 ToString()**

```
MVCOMMON_API String MVCommon::Versorf::ToString ( ) const
```

Converts the versor into a human-readable string.

**Returns**

the versor string

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

- public/MVCommon/math/Versorf.h

**6.54 MVCommon::VersorfHasher Struct Reference**

A hasher for [Versorf](#) objects so they can be used in unordered collections.

```
#include <Versorf.h>
```

**Public Member Functions**

- MVCOMMON\_API size\_t [operator\(\)](#) ([Versorf](#) const &versor) const  
*Calculates a hash value from the object.*

### 6.54.1 Detailed Description

A hasher for [Versorf](#) objects so they can be used in unordered collections.

### 6.54.2 Member Function Documentation

#### 6.54.2.1 operator()

```
MVCOMMON_API size_t MVCommon::VersorfHasher::operator() (
    Versorf const & versorf ) const
```

Calculates a hash value from the object.

##### Parameters

<a href="#">versorf</a>	an object to calculate the hash value of
-------------------------	--

##### Returns

hash value of the object

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

- public/MVCommon/math/Versorf.h

## 6.55 MVCommon::WeakLoggerPtr Class Reference

A weak smart-pointer to a logger.

```
#include <WeakLoggerPtr.h>
```

### Public Member Functions

- MVCOMMON\_API [WeakLoggerPtr](#) ()  
*A constructor.*
- MVCOMMON\_API [WeakLoggerPtr](#) ([SharedLoggerPtr](#) spPtr)  
*A constructor.*
- MVCOMMON\_API [WeakLoggerPtr](#) ([WeakLoggerPtr](#) const &other)  
*A copy-constructor.*
- MVCOMMON\_API [~WeakLoggerPtr](#) ()  
*A destructor.*
- MVCOMMON\_API [WeakLoggerPtr](#) & operator= ([WeakLoggerPtr](#) const &other)  
*Makes the pointer point to a logger pointed to by the *other* pointer.*

- MVCOMMON\_API [WeakLoggerPtr](#) & [operator=](#) ([SharedLoggerPtr](#) spPtr)  
*Links the pointer to a shared smart-pointer.*
- MVCOMMON\_API void [Reset](#) ()  
*Resets the pointer to not point to any logger (i.e. nullptr).*
- MVCOMMON\_API bool [Expired](#) () const  
*Determines whether the pointed-to logger still exists, e.g. when there are no more shared smart-pointers pointing to it.*
- MVCOMMON\_API [SharedLoggerPtr](#) [Lock](#) () const  
*Creates a new shared smart-pointer pointing to a logger pointed to by the pointer, increasing thus number of shared smart-pointers pointing to it.*

### 6.55.1 Detailed Description

A weak smart-pointer to a logger.

Allows access to a logger object whose lifetime is maintained by a [SharedLoggerPtr](#) smart-pointers, but does not influence the lifetime itself.

### 6.55.2 Constructor & Destructor Documentation

#### 6.55.2.1 WeakLoggerPtr() [1/3]

```
MVCOMMON_API MVCommon::WeakLoggerPtr::WeakLoggerPtr ( )
```

A constructor.

Initializes the pointer with nullptr.

#### 6.55.2.2 WeakLoggerPtr() [2/3]

```
MVCOMMON_API MVCommon::WeakLoggerPtr::WeakLoggerPtr (
    SharedLoggerPtr spPtr )
```

A constructor.

Parameters

<a href="#">spPtr</a>	a shared smart-pointer to be linked to
-----------------------	--

#### 6.55.2.3 WeakLoggerPtr() [3/3]

```
MVCOMMON_API MVCommon::WeakLoggerPtr::WeakLoggerPtr (
    WeakLoggerPtr const & other )
```



A copy-constructor.

#### Parameters

<i>other</i>	other pointer to share a pointed-to logger with
--------------	---

## 6.55.3 Member Function Documentation

### 6.55.3.1 Expired()

```
MVCOMMON_API bool MVCommon::WeakLoggerPtr::Expired ( ) const
```

Determines whether the pointed-to logger still exists, e.g. when there are no more shared smart-pointers pointing to it.

#### Returns

true if the pointed-to logger does not exist anymore

### 6.55.3.2 Lock()

```
MVCOMMON_API SharedLoggerPtr MVCommon::WeakLoggerPtr::Lock ( ) const
```

Creates a new shared smart-pointer pointing to a logger pointed to by the pointer, increasing thus number of shared smart-pointers pointing to it.

#### Returns

shared smart-pointer pointing to the pointer-to logger or nullptr in case the pointer has expired already

### 6.55.3.3 operator=() [1/2]

```
MVCOMMON_API WeakLoggerPtr& MVCommon::WeakLoggerPtr::operator= (
    SharedLoggerPtr spPtr )
```

Links the pointer to a shared smart-pointer.

#### Parameters

<i>spPtr</i>	a shared smart-pointer to be linked to
--------------	--

**Returns**

the pointer itself

**6.55.3.4 operator=()** [2/2]

```
MVCOMMON_API WeakLoggerPtr& MVCommon::WeakLoggerPtr::operator= (
    WeakLoggerPtr const & other )
```

Makes the pointer point to a logger pointed to by the `other` pointer.

**Parameters**

<i>other</i>	other pointer to share a pointed-to logger with
--------------	---

**Returns**

the pointer itself

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

- public/MVCommon/logger/WeakLoggerPtr.h

## Chapter 7

# File Documentation

## 7.1 public/MVCommon/CUtil.h File Reference

### Macros

- `#define MV_VALUE_TO_STR(x) #x`  
*A macro for converting a value into a string literal.*

### 7.1.1 Macro Definition Documentation

#### 7.1.1.1 MV\_VALUE\_TO\_STR

```
#define MV_VALUE_TO_STR(  
    x ) #x
```

A macro for converting a value into a string literal.

Example:

```
char const * string_literal = MV_VALUE_TO_STR_LITERAL(5);  
// preceding line is processed into:  
// char const * string_literal = "5";
```

## 7.2 public/MVCommon/guid/GuidGenerator.h File Reference

```
#include "Guid.h"
```

### Functions

- MVCOMMON\_API Guid [MVCommon::GuidGenerator::GenerateGuid](#) (Guid const &guidNamespace, String const &seed)  
*Generates a [Guid](#) based on another [Guid](#) (a namespace) and a string seed.*
- MVCOMMON\_API Guid [MVCommon::GuidGenerator::GenerateGuid](#) (String const &seed)  
*Generates a [Guid](#) based on a string seed.*

## 7.2.1 Function Documentation

### 7.2.1.1 GenerateGuid() [1/2]

```
MVCOMMON_API Guid MVCommon::GuidGenerator::GenerateGuid (
    Guid const & guidNamespace,
    String const & seed )
```

Generates a Guid based on another Guid (a namespace) and a string seed.

Using the same Guid namespace and the same seed will always produce the same generated Guid.

#### Parameters

<i>guidNamespace</i>	a Guid in the role of a namespace (ancestor) for the new Guid
<i>seed</i>	a seed for the new Guid generation

#### Returns

generated Guid

### 7.2.1.2 GenerateGuid() [2/2]

```
MVCOMMON_API Guid MVCommon::GuidGenerator::GenerateGuid (
    String const & seed )
```

Generates a Guid based on a string seed.

Using the same seed will always produce the same generated Guid.

#### Parameters

<i>seed</i>	a seed for the new Guid generation
-------------	------------------------------------

#### Returns

generated Guid

## 7.3 public/MVCommon/logger/LoggerLogLevel.h File Reference

```
#include "LogLevel.h"
```

## Enumerations

- enum `MVCommon::LoggerLogLevel` {  
`MVCommon::LLL_SILENT` = 0, `MVCommon::LLL_CRITICAL` = LL\_CRITICAL, `MVCommon::LLL_ERROR` = LL\_ERROR, `MVCommon::LLL_WARNING` = LL\_WARNING,  
`MVCommon::LLL_INFO` = LL\_INFO, `MVCommon::LLL_DEBUG` = LL\_DEBUG, `MVCommon::LLL_VERBOSE` = LL\_VERBOSE }

*An enumeration of logger log levels for filtering log messages.*

### 7.3.1 Enumeration Type Documentation

#### 7.3.1.1 LoggerLogLevel

```
enum MVCommon::LoggerLogLevel
```

An enumeration of logger log levels for filtering log messages.

Only log messages that are classified with higher log level than the logger is set to are actually logged.

##### Enumerator

LLL_SILENT	No log messages are logged.
LLL_CRITICAL	Only critical log messages are logged.
LLL_ERROR	Only error or higher level log messages are logged.
LLL_WARNING	Only warning and higher level log messages are logged.
LLL_INFO	Only info and higher level log messages are logged.
LLL_DEBUG	Only debug and higher level log messages are logged.
LLL_VERBOSE	All log messages are logged.

## 7.4 public/MVCommon/logger/LogLevel.h File Reference

## Enumerations

- enum `MVCommon::LogLevel` {  
`MVCommon::LL_CRITICAL` = 1, `MVCommon::LL_ERROR` = 2, `MVCommon::LL_WARNING` = 3,  
`MVCommon::LL_INFO` = 4,  
`MVCommon::LL_DEBUG` = 5, `MVCommon::LL_VERBOSE` = 6 }

*An enumeration of log levels.*

### 7.4.1 Enumeration Type Documentation

#### 7.4.1.1 LogLevel

enum `MVCommon::LogLevel`

An enumeration of log levels.

**Enumerator**

LL_CRITICAL	A critical message log level.
LL_ERROR	An error message log level.
LL_WARNING	A warning message log level.
LL_INFO	An info message log level.
LL_DEBUG	A debug message log level.
LL_VERBOSE	A verbose message log level.

## 7.5 public/MVCommon/MVCommonVersion.h File Reference

```
#include "utils/VersionInfo.h"
```

**Macros**

- `#define MVCOMMON_VERSION_MAJOR 4`  
*Current value of the most-significant MVCommon version component.*
- `#define MVCOMMON_VERSION_MINOR 0`  
*Current value of the medium-significant MVCommon version component.*
- `#define MVCOMMON_VERSION_PATCH 0`  
*Current value of the least-significant MVCommon version component.*

**Variables**

- `const VersionInfo MVCommon::MVCOMMON_VERSION = { 4 , 0 , 0 }`  
*An MVCommon version.*

## 7.6 public/MVCommon/utils/VersionInfo.h File Reference

```
#include <MVCommon/MVCommonAPI.h>
#include <MVCommon/Memory.h>
#include "String.h"
```

**Data Structures**

- struct `MVCommon::VersionInfo`  
*A structure holding module version information.*
- struct `MVCommon::VersionInfoHasher`  
*A hasher for `VersionInfo` objects so they can be used in unordered collections.*





# Index

- ~SharedGuidAliasDatabasePtr
  - MVCommon::SharedGuidAliasDatabasePtr, [106](#)
- ~SharedLoggerPtr
  - MVCommon::SharedLoggerPtr, [109](#)
- ~SharedLoggerSinkPtr
  - MVCommon::SharedLoggerSinkPtr, [113](#)
- ~SharedThreadPoolJobPtr
  - MVCommon::SharedThreadPoolJobPtr, [116](#)
- ~ThreadPool
  - MVCommon::ThreadPool, [124](#)
- Abs
  - MVCommon::Vector2d, [127](#)
  - MVCommon::Vector2f, [133](#)
  - MVCommon::Vector3d, [140](#)
  - MVCommon::Vector3f, [147](#)
  - MVCommon::Vector4d, [154](#)
  - MVCommon::Vector4f, [161](#)
- AddLogEntry
  - MVCommon::ILoggerSink, [64](#)
- AddLoggerSink
  - MVCommon::Logger, [71](#)
- AliasRegistered
  - MVCommon::GuidAliasDatabase, [55](#)
- AlmostEqual
  - MVCommon::Math, [76](#)
- AndroidSystemLoggerSink
  - MVCommon::AndroidSystemLoggerSink, [15](#)
- AppleSystemLoggerSink
  - MVCommon::AppleSystemLoggerSink, [17](#)
- Begin
  - MVCommon::GuidAliasDatabase, [56](#)
- BlockingCounter
  - MVCommon::BlockingCounter, [18](#)
- BlockingCounterValueEquals
  - MVCommon::BlockingCounterValueEquals, [22](#)
- ByteArray
  - MVCommon::ByteArray, [24](#), [25](#)
- CameraParams
  - MVCommon::CameraParams, [33](#)
- Clamp
  - MVCommon::Math, [77](#)
- Color
  - MVCommon::Color, [39](#), [40](#)
- CreateLookAt
  - MVCommon::Matrix4x4d, [80](#)
  - MVCommon::Matrix4x4f, [92](#)
- CreateOrtographic
  - MVCommon::Matrix4x4d, [82](#)
  - MVCommon::Matrix4x4f, [93](#)
- CreatePerspective
  - MVCommon::Matrix4x4d, [82](#)
  - MVCommon::Matrix4x4f, [93](#)
- CreateRotationAroundAxis
  - MVCommon::Matrix4x4d, [83](#)
  - MVCommon::Matrix4x4f, [94](#)
  - MVCommon::Versord, [169](#)
  - MVCommon::Versorf, [176](#)
- CreateRotationFromEulerAnglesZYX
  - MVCommon::Matrix4x4d, [83](#)
  - MVCommon::Matrix4x4f, [94](#)
  - MVCommon::Versord, [170](#)
  - MVCommon::Versorf, [176](#)
- CreateRotationFromMatrix
  - MVCommon::Versord, [170](#)
  - MVCommon::Versorf, [177](#)
- CreateRotationFromVersor
  - MVCommon::Matrix4x4d, [83](#)
  - MVCommon::Matrix4x4f, [94](#)
- CreateScale
  - MVCommon::Matrix4x4d, [84](#)
  - MVCommon::Matrix4x4f, [95](#)
- CreateTranslation
  - MVCommon::Matrix4x4d, [84](#)
  - MVCommon::Matrix4x4f, [95](#)
- CreateZero
  - MVCommon::Matrix4x4d, [85](#)
  - MVCommon::Matrix4x4f, [95](#)
- Cross
  - MVCommon::Vector3d, [140](#)
  - MVCommon::Vector3f, [147](#)
- CStr
  - MVCommon::String, [121](#)
- CUtil.h
  - MV\_VALUE\_TO\_STR, [185](#)
- Data
  - MVCommon::ByteArray, [25](#)
- DenormalizePoint
  - MVCommon::CameraParams, [33](#), [34](#)
- distortionC
  - MVCommon::CameraParams, [36](#)
- DoJob
  - MVCommon::ThreadPool, [124](#)
- Dot
  - MVCommon::Vector2d, [127](#)
  - MVCommon::Vector2f, [133](#)
  - MVCommon::Vector3d, [140](#)

- MVCommon::Vector3f, [147](#)
- MVCommon::Vector4d, [154](#)
- MVCommon::Vector4f, [161](#)
- End
  - MVCommon::GuidAliasDatabase, [56](#)
- Expired
  - MVCommon::WeakLoggerPtr, [183](#)
- FileLoggerSink
  - MVCommon::FileLoggerSink, [48](#)
- FromElementsVector
  - MVCommon::Versord, [170](#)
  - MVCommon::Versorf, [177](#)
- FromHexString
  - MVCommon::Guid, [51](#)
- FromRawBytes
  - MVCommon::CameraParams, [34](#)
  - MVCommon::Guid, [51](#)
  - MVCommon::Matrix4x4d, [85](#)
  - MVCommon::Matrix4x4f, [96](#)
  - MVCommon::Vector2d, [128](#)
  - MVCommon::Vector2f, [134](#)
  - MVCommon::Vector3d, [141](#)
  - MVCommon::Vector3f, [148](#)
  - MVCommon::Vector4d, [154](#)
  - MVCommon::Vector4f, [161](#)
  - MVCommon::Versord, [171](#)
  - MVCommon::Versorf, [177](#)
- FromRawElements
  - MVCommon::Matrix4x4d, [85](#)
  - MVCommon::Matrix4x4f, [96](#)
  - MVCommon::Versord, [171](#)
  - MVCommon::Versorf, [178](#)
- FromRfc4122
  - MVCommon::Guid, [52](#)
- FromString
  - MVCommon::Color, [40](#)
  - MVCommon::Matrix4x4d, [86](#)
  - MVCommon::Matrix4x4f, [96](#)
  - MVCommon::Vector2d, [128](#)
  - MVCommon::Vector2f, [134](#)
  - MVCommon::Vector3d, [141](#)
  - MVCommon::Vector3f, [148](#)
  - MVCommon::Vector4d, [155](#)
  - MVCommon::Vector4f, [162](#)
  - MVCommon::Versord, [172](#)
  - MVCommon::Versorf, [178](#)
- GenerateGuid
  - GuidGenerator.h, [186](#)
- Get
  - MVCommon::SharedGuidAliasDatabasePtr, [106](#)
  - MVCommon::SharedLoggerPtr, [110](#)
  - MVCommon::SharedLoggerSinkPtr, [113](#)
  - MVCommon::SharedThreadPoolJobPtr, [117](#)
- GetAlpha
  - MVCommon::Color, [40](#)
- GetAlphaByte
  - MVCommon::Color, [41](#)
- GetBlue
  - MVCommon::Color, [41](#)
- GetBlueByte
  - MVCommon::Color, [41](#)
- GetGreen
  - MVCommon::Color, [41](#)
- GetGreenByte
  - MVCommon::Color, [42](#)
- GetGuidAlias
  - MVCommon::GuidAliasDatabase, [56](#)
- GetGuidWithAlias
  - MVCommon::GuidAliasDatabase, [57](#)
- GetInstance
  - MVCommon::LoggerRegistry, [74](#)
- GetLevel
  - MVCommon::LogEntry, [68](#)
- GetLogger
  - MVCommon::LoggerRegistry, [74](#)
- GetLogLevel
  - MVCommon::ILoggerSink, [65](#)
  - MVCommon::Logger, [71](#)
- GetMessage
  - MVCommon::LogEntry, [69](#)
- GetRed
  - MVCommon::Color, [42](#)
- GetRedByte
  - MVCommon::Color, [42](#)
- GetRGBBrightness
  - MVCommon::Color, [42](#)
- GetRGBBrightnessByte
  - MVCommon::Color, [43](#)
- GetTag
  - MVCommon::LogEntry, [69](#)
- GetThreadId
  - MVCommon::LogEntry, [69](#)
- GetThreadsCount
  - MVCommon::ThreadPool, [125](#)
- GetTimestamp
  - MVCommon::LogEntry, [69](#)
- GetUnoccupiedThreadsCount
  - MVCommon::ThreadPool, [125](#)
- GetX
  - MVCommon::Vector3d, [142](#)
  - MVCommon::Vector3f, [149](#)
- GetXYZ
  - MVCommon::Vector4d, [155](#)
  - MVCommon::Vector4f, [162](#)
- Guid
  - MVCommon::Guid, [50](#)
- GuidAliasDatabase
  - MVCommon::GuidAliasDatabase, [55](#)
- GuidAliasDatabaseIterator
  - MVCommon::GuidAliasDatabaseIterator, [60](#)
- GuidGenerator.h
  - GenerateGuid, [186](#)
- GuidRegistered
  - MVCommon::GuidAliasDatabase, [57](#)

- HandleLogEntry
  - MVCommon::AndroidSystemLoggerSink, [16](#)
  - MVCommon::AppleSystemLoggerSink, [17](#)
  - MVCommon::FileLoggerSink, [49](#)
  - MVCommon::ILoggerSink, [65](#)
  - MVCommon::RedirectingLoggerSink, [103](#)
  - MVCommon::StdOutLoggerSink, [119](#)
- HasUnoccupiedThreads
  - MVCommon::ThreadPool, [125](#)
- ILoggerSink
  - MVCommon::ILoggerSink, [64](#)
- Increment
  - MVCommon::BlockingCounter, [19](#)
- Inverted
  - MVCommon::Matrix4x4d, [86](#)
  - MVCommon::Matrix4x4f, [97](#)
  - MVCommon::Vector2d, [129](#)
  - MVCommon::Vector2f, [135](#)
  - MVCommon::Vector3d, [142](#)
  - MVCommon::Vector3f, [149](#)
  - MVCommon::Vector4d, [155](#)
  - MVCommon::Vector4f, [162](#)
  - MVCommon::Versord, [172](#)
  - MVCommon::Versorf, [179](#)
- IsNil
  - MVCommon::Guid, [52](#)
- Length
  - MVCommon::String, [121](#)
  - MVCommon::Vector2d, [129](#)
  - MVCommon::Vector2f, [135](#)
  - MVCommon::Vector3d, [142](#)
  - MVCommon::Vector3f, [149](#)
  - MVCommon::Vector4d, [156](#)
  - MVCommon::Vector4f, [162](#)
- LL\_CRITICAL
  - LogLevel.h, [189](#)
- LL\_DEBUG
  - LogLevel.h, [189](#)
- LL\_ERROR
  - LogLevel.h, [189](#)
- LL\_INFO
  - LogLevel.h, [189](#)
- LL\_VERBOSE
  - LogLevel.h, [189](#)
- LL\_WARNING
  - LogLevel.h, [189](#)
- LLL\_CRITICAL
  - LoggerLogLevel.h, [187](#)
- LLL\_DEBUG
  - LoggerLogLevel.h, [187](#)
- LLL\_ERROR
  - LoggerLogLevel.h, [187](#)
- LLL\_INFO
  - LoggerLogLevel.h, [187](#)
- LLL\_SILENT
  - LoggerLogLevel.h, [187](#)
- LLL\_VERBOSE
  - LoggerLogLevel.h, [187](#)
- LLL\_WARNING
  - LoggerLogLevel.h, [187](#)
- Lock
  - MVCommon::WeakLoggerPtr, [183](#)
- LogEntry
  - MVCommon::LogEntry, [68](#)
- Logger
  - MVCommon::Logger, [71](#)
- LoggerLogLevel
  - LoggerLogLevel.h, [187](#)
- LoggerLogLevel.h
  - LLL\_CRITICAL, [187](#)
  - LLL\_DEBUG, [187](#)
  - LLL\_ERROR, [187](#)
  - LLL\_INFO, [187](#)
  - LLL\_SILENT, [187](#)
  - LLL\_VERBOSE, [187](#)
  - LLL\_WARNING, [187](#)
  - LoggerLogLevel, [187](#)
- LogLevel
  - LogLevel.h, [187](#)
- LogLevel.h
  - LL\_CRITICAL, [189](#)
  - LL\_DEBUG, [189](#)
  - LL\_ERROR, [189](#)
  - LL\_INFO, [189](#)
  - LL\_VERBOSE, [189](#)
  - LL\_WARNING, [189](#)
  - LogLevel, [187](#)
- LogLevelToString
  - MVCommon::ILoggerSink, [65](#)
- LogMessage
  - MVCommon::Logger, [72](#)
- major
  - MVCommon::VersionInfo, [167](#)
- Matrix4x4d
  - MVCommon::Matrix4x4d, [79](#), [80](#)
- Matrix4x4f
  - MVCommon::Matrix4x4f, [91](#), [92](#)
- minor
  - MVCommon::VersionInfo, [167](#)
- MV\_VALUE\_TO\_STR
  - CUtil.h, [185](#)
- MVCommon::AndroidSystemLoggerSink, [15](#)
  - AndroidSystemLoggerSink, [15](#)
  - HandleLogEntry, [16](#)
- MVCommon::AppleSystemLoggerSink, [16](#)
  - AppleSystemLoggerSink, [17](#)
  - HandleLogEntry, [17](#)
- MVCommon::BlockingCounter, [18](#)
  - BlockingCounter, [18](#)
  - Increment, [19](#)
  - operator+==, [19](#)
  - Value, [19](#)
  - WaitUntil, [20](#)
  - WaitUntilFor, [20](#)
  - WaitUntilValue, [20](#)

- WaitUntilValueFor, 21
- MVCommon::BlockingCounterValueEquals, 21
  - BlockingCounterValueEquals, 22
  - operator(), 22
- MVCommon::ByteArray, 23
  - ByteArray, 24, 25
  - Data, 25
  - operator<=<=, 25, 26
  - operator>>=, 26
  - operator=, 26
  - operator[], 27
  - Pop, 28
  - Push, 28, 29
  - Size, 29
  - Skip, 30
  - Subarray, 30
- MVCommon::ByteArrayHasher, 31
  - operator(), 31
- MVCommon::CameraParams, 31
  - CameraParams, 33
  - DenormalizePoint, 33, 34
  - distortionC, 36
  - FromRawBytes, 34
  - NormalizePoint, 34, 35
  - ScaleToResolution, 35
  - ToRawBytes, 35
  - ToString, 35
  - UndistortPoint, 36
- MVCommon::CameraParamsHasher, 37
  - operator(), 37
- MVCommon::Color, 37
  - Color, 39, 40
  - FromString, 40
  - GetAlpha, 40
  - GetAlphaByte, 41
  - GetBlue, 41
  - GetBlueByte, 41
  - GetGreen, 41
  - GetGreenByte, 42
  - GetRed, 42
  - GetRedByte, 42
  - GetRGBBrightness, 42
  - GetRGBBrightnessByte, 43
  - SetAlpha, 43
  - SetAlphaByte, 43
  - SetBlue, 44
  - SetBlueByte, 44
  - SetGreen, 44
  - SetGreenByte, 44
  - SetRed, 45
  - SetRedByte, 45
  - SetValue, 45, 46
  - ToRGB\_HTMLString, 46
  - ToString, 46
- MVCommon::ColorHasher, 47
  - operator(), 47
- MVCommon::FileLoggerSink, 48
  - FileLoggerSink, 48
- HandleLogEntry, 49
- MVCommon::Guid, 49
  - FromHexString, 51
  - FromRawBytes, 51
  - FromRfc4122, 52
  - Guid, 50
  - IsNil, 52
  - Nil, 52
  - ToHexString, 53
  - ToRawBytes, 53
  - ToRfc4122, 53
- MVCommon::GuidAliasDatabase, 54
  - AliasRegistered, 55
  - Begin, 56
  - End, 56
  - GetGuidAlias, 56
  - GetGuidWithAlias, 57
  - GuidAliasDatabase, 55
  - GuidRegistered, 57
  - RegisterGuidAlias, 57
  - TryGetGuidAlias, 58
  - TryGetGuidWithAlias, 58
  - UnregisterGuidAlias, 58, 59
- MVCommon::GuidAliasDatabaseIterator, 59
  - GuidAliasDatabaseIterator, 60
  - operator\*, 60
  - operator++, 61
- MVCommon::GuidHasher, 61
  - operator(), 62
- MVCommon::IBlockingCounterCondition, 62
  - operator(), 63
- MVCommon::ILoggerSink, 63
  - AddLogEntry, 64
  - GetLogLevel, 65
  - HandleLogEntry, 65
  - ILoggerSink, 64
  - LogLevelToString, 65
  - SetLogLevel, 66
  - TimestampToString, 66
- MVCommon::IThreadPoolJob, 66
  - operator(), 67
- MVCommon::LogEntry, 67
  - GetLevel, 68
  - GetMessage, 69
  - GetTag, 69
  - GetThreadID, 69
  - GetTimestamp, 69
  - LogEntry, 68
- MVCommon::Logger, 70
  - AddLoggerSink, 71
  - GetLogLevel, 71
  - Logger, 71
  - LogMessage, 72
  - RemoveLoggerSink, 73
  - SetLogLevel, 73
- MVCommon::LoggerRegistry, 73
  - GetInstance, 74
  - GetLogger, 74

- RegisterLogger, 75
- UnregisterLogger, 75
- MVCommon::Math, 75
  - AlmostEqual, 76
  - Clamp, 77
- MVCommon::Matrix4x4d, 78
  - CreateLookAt, 80
  - CreateOrtographic, 82
  - CreatePerspective, 82
  - CreateRotationAroundAxis, 83
  - CreateRotationFromEulerAnglesZYX, 83
  - CreateRotationFromVersor, 83
  - CreateScale, 84
  - CreateTranslation, 84
  - CreateZero, 85
  - FromRawBytes, 85
  - FromRawElements, 85
  - FromString, 86
  - Inverted, 86
  - Matrix4x4d, 79, 80
  - operator[], 86, 87
  - RotationTranslationMatrixInverted, 87
  - ToRawBytes, 87
  - ToRawElements, 88
  - ToString, 88
  - Transposed, 88
- MVCommon::Matrix4x4dHasher, 89
  - operator(), 89
- MVCommon::Matrix4x4f, 89
  - CreateLookAt, 92
  - CreateOrtographic, 93
  - CreatePerspective, 93
  - CreateRotationAroundAxis, 94
  - CreateRotationFromEulerAnglesZYX, 94
  - CreateRotationFromVersor, 94
  - CreateScale, 95
  - CreateTranslation, 95
  - CreateZero, 95
  - FromRawBytes, 96
  - FromRawElements, 96
  - FromString, 96
  - Inverted, 97
  - Matrix4x4f, 91, 92
  - operator[], 97, 98
  - RotationTranslationMatrixInverted, 98
  - ToRawBytes, 98
  - ToRawElements, 99
  - ToString, 99
  - Transposed, 99
- MVCommon::Matrix4x4fHasher, 100
  - operator(), 100
- MVCommon::Pair< TFirst, TSecond >, 100
  - Pair, 101
- MVCommon::RedirectingLoggerSink, 102
  - HandleLogEntry, 103
  - RedirectingLoggerSink, 102
- MVCommon::SharedGuidAliasDatabasePtr, 103
  - ~SharedGuidAliasDatabasePtr, 106
- Get, 106
  - operator bool, 106
  - operator\*, 106
  - operator->, 107
  - operator=, 107
  - SharedGuidAliasDatabasePtr, 104
- MVCommon::SharedLoggerPtr, 108
  - ~SharedLoggerPtr, 109
  - Get, 110
  - operator bool, 110
  - operator\*, 110
  - operator->, 110
  - operator=, 110, 111
  - SharedLoggerPtr, 109
- MVCommon::SharedLoggerSinkPtr, 111
  - ~SharedLoggerSinkPtr, 113
  - Get, 113
  - operator bool, 113
  - operator\*, 113
  - operator->, 114
  - operator=, 114
  - SharedLoggerSinkPtr, 112
- MVCommon::SharedThreadPoolJobPtr, 115
  - ~SharedThreadPoolJobPtr, 116
  - Get, 117
  - operator bool, 117
  - operator\*, 117
  - operator->, 117
  - operator=, 117, 118
  - SharedThreadPoolJobPtr, 116
- MVCommon::StdOutLoggerSink, 118
  - HandleLogEntry, 119
  - StdOutLoggerSink, 119
- MVCommon::String, 120
  - CStr, 121
  - Length, 121
  - String, 120, 121
  - Substr, 122
- MVCommon::StringHasher, 122
  - operator(), 123
- MVCommon::ThreadPool, 123
  - ~ThreadPool, 124
  - DoJob, 124
  - GetThreadsCount, 125
  - GetUnoccupiedThreadsCount, 125
  - HasUnoccupiedThreads, 125
  - ResetJobs, 125
  - ThreadPool, 124
- MVCommon::Vector2d, 126
  - Abs, 127
  - Dot, 127
  - FromRawBytes, 128
  - FromString, 128
  - Inverted, 129
  - Length, 129
  - Normalized, 129
  - operator[], 129, 130
  - ToRawBytes, 130

- ToString, [130](#)
- Vector2d, [127](#)
- MVCommon::Vector2dHasher, [131](#)
  - operator(), [131](#)
- MVCommon::Vector2f, [132](#)
  - Abs, [133](#)
  - Dot, [133](#)
  - FromRawBytes, [134](#)
  - FromString, [134](#)
  - Inverted, [135](#)
  - Length, [135](#)
  - Normalized, [135](#)
  - operator[], [135](#), [136](#)
  - ToRawBytes, [136](#)
  - ToString, [136](#)
  - Vector2f, [133](#)
- MVCommon::Vector2fHasher, [137](#)
  - operator(), [137](#)
- MVCommon::Vector3d, [138](#)
  - Abs, [140](#)
  - Cross, [140](#)
  - Dot, [140](#)
  - FromRawBytes, [141](#)
  - FromString, [141](#)
  - GetXY, [142](#)
  - Inverted, [142](#)
  - Length, [142](#)
  - Normalized, [142](#)
  - operator[], [142](#), [143](#)
  - ToRawBytes, [143](#)
  - ToString, [143](#)
  - Vector3d, [139](#)
- MVCommon::Vector3dHasher, [144](#)
  - operator(), [144](#)
- MVCommon::Vector3f, [145](#)
  - Abs, [147](#)
  - Cross, [147](#)
  - Dot, [147](#)
  - FromRawBytes, [148](#)
  - FromString, [148](#)
  - GetXY, [149](#)
  - Inverted, [149](#)
  - Length, [149](#)
  - Normalized, [149](#)
  - operator[], [149](#), [150](#)
  - ToRawBytes, [150](#)
  - ToString, [150](#)
  - Vector3f, [146](#)
- MVCommon::Vector3fHasher, [151](#)
  - operator(), [151](#)
- MVCommon::Vector4d, [152](#)
  - Abs, [154](#)
  - Dot, [154](#)
  - FromRawBytes, [154](#)
  - FromString, [155](#)
  - GetXYZ, [155](#)
  - Inverted, [155](#)
  - Length, [156](#)
  - Normalized, [156](#)
  - operator[], [156](#), [157](#)
  - ToRawBytes, [157](#)
  - ToString, [157](#)
  - Vector4d, [153](#)
- MVCommon::Vector4dHasher, [158](#)
  - operator(), [158](#)
- MVCommon::Vector4f, [159](#)
  - Abs, [161](#)
  - Dot, [161](#)
  - FromRawBytes, [161](#)
  - FromString, [162](#)
  - GetXYZ, [162](#)
  - Inverted, [162](#)
  - Length, [162](#)
  - Normalized, [163](#)
  - operator[], [163](#)
  - ToRawBytes, [164](#)
  - ToString, [164](#)
  - Vector4f, [160](#)
- MVCommon::Vector4fHasher, [165](#)
  - operator(), [165](#)
- MVCommon::VersionInfo, [165](#)
  - major, [167](#)
  - minor, [167](#)
  - patch, [167](#)
  - ToString, [166](#)
  - VersionInfo, [166](#)
- MVCommon::VersionInfoHasher, [167](#)
  - operator(), [168](#)
- MVCommon::Versord, [168](#)
  - CreateRotationAroundAxis, [169](#)
  - CreateRotationFromEulerAnglesZYX, [170](#)
  - CreateRotationFromMatrix, [170](#)
  - FromElementsVector, [170](#)
  - FromRawBytes, [171](#)
  - FromRawElements, [171](#)
  - FromString, [172](#)
  - Inverted, [172](#)
  - ToElementsVector, [172](#)
  - ToEulerAnglesZYX, [173](#)
  - ToRawBytes, [173](#)
  - ToRawElements, [173](#)
  - ToString, [174](#)
- MVCommon::VersordHasher, [174](#)
  - operator(), [174](#)
- MVCommon::Versorf, [175](#)
  - CreateRotationAroundAxis, [176](#)
  - CreateRotationFromEulerAnglesZYX, [176](#)
  - CreateRotationFromMatrix, [177](#)
  - FromElementsVector, [177](#)
  - FromRawBytes, [177](#)
  - FromRawElements, [178](#)
  - FromString, [178](#)
  - Inverted, [179](#)
  - ToElementsVector, [179](#)
  - ToEulerAnglesZYX, [179](#)
  - ToRawBytes, [179](#)

- ToRawElements, 180
- ToString, 180
- MVCommon::VersorfHasher, 180
  - operator(), 181
- MVCommon::WeakLoggerPtr, 181
  - Expired, 183
  - Lock, 183
  - operator=, 183, 184
  - WeakLoggerPtr, 182
- Nil
  - MVCommon::Guid, 52
- Normalized
  - MVCommon::Vector2d, 129
  - MVCommon::Vector2f, 135
  - MVCommon::Vector3d, 142
  - MVCommon::Vector3f, 149
  - MVCommon::Vector4d, 156
  - MVCommon::Vector4f, 163
- NormalizePoint
  - MVCommon::CameraParams, 34, 35
- operator bool
  - MVCommon::SharedGuidAliasDatabasePtr, 106
  - MVCommon::SharedLoggerPtr, 110
  - MVCommon::SharedLoggerSinkPtr, 113
  - MVCommon::SharedThreadPoolJobPtr, 117
- operator<<=
  - MVCommon::ByteArray, 25, 26
- operator>>=
  - MVCommon::ByteArray, 26
- operator\*
  - MVCommon::GuidAliasDatabaseIterator, 60
  - MVCommon::SharedGuidAliasDatabasePtr, 106
  - MVCommon::SharedLoggerPtr, 110
  - MVCommon::SharedLoggerSinkPtr, 113
  - MVCommon::SharedThreadPoolJobPtr, 117
- operator()
  - MVCommon::BlockingCounterValueEquals, 22
  - MVCommon::ByteArrayHasher, 31
  - MVCommon::CameraParamsHasher, 37
  - MVCommon::ColorHasher, 47
  - MVCommon::GuidHasher, 62
  - MVCommon::IBlockingCounterCondition, 63
  - MVCommon::IThreadPoolJob, 67
  - MVCommon::Matrix4x4dHasher, 89
  - MVCommon::Matrix4x4fHasher, 100
  - MVCommon::StringHasher, 123
  - MVCommon::Vector2dHasher, 131
  - MVCommon::Vector2fHasher, 137
  - MVCommon::Vector3dHasher, 144
  - MVCommon::Vector3fHasher, 151
  - MVCommon::Vector4dHasher, 158
  - MVCommon::Vector4fHasher, 165
  - MVCommon::VersionInfoHasher, 168
  - MVCommon::VersordHasher, 174
  - MVCommon::VersorfHasher, 181
- operator++
  - MVCommon::GuidAliasDatabaseIterator, 61
- operator+=
  - MVCommon::BlockingCounter, 19
- operator->
  - MVCommon::SharedGuidAliasDatabasePtr, 107
  - MVCommon::SharedLoggerPtr, 110
  - MVCommon::SharedLoggerSinkPtr, 114
  - MVCommon::SharedThreadPoolJobPtr, 117
- operator=
  - MVCommon::ByteArray, 26
  - MVCommon::SharedGuidAliasDatabasePtr, 107
  - MVCommon::SharedLoggerPtr, 110, 111
  - MVCommon::SharedLoggerSinkPtr, 114
  - MVCommon::SharedThreadPoolJobPtr, 117, 118
  - MVCommon::WeakLoggerPtr, 183, 184
- operator[]
  - MVCommon::ByteArray, 27
  - MVCommon::Matrix4x4d, 86, 87
  - MVCommon::Matrix4x4f, 97, 98
  - MVCommon::Vector2d, 129, 130
  - MVCommon::Vector2f, 135, 136
  - MVCommon::Vector3d, 142, 143
  - MVCommon::Vector3f, 149, 150
  - MVCommon::Vector4d, 156, 157
  - MVCommon::Vector4f, 163
- Pair
  - MVCommon::Pair< TFirst, TSecond >, 101
- patch
  - MVCommon::VersionInfo, 167
- Pop
  - MVCommon::ByteArray, 28
- public/MVCommon/CUtil.h, 185
- public/MVCommon/guid/GuidGenerator.h, 185
- public/MVCommon/logger/LoggerLogLevel.h, 186
- public/MVCommon/logger/LogLevel.h, 187
- public/MVCommon/MVCommonVersion.h, 189
- public/MVCommon/Utils/VersionInfo.h, 189
- Push
  - MVCommon::ByteArray, 28, 29
- RedirectingLoggerSink
  - MVCommon::RedirectingLoggerSink, 102
- RegisterGuidAlias
  - MVCommon::GuidAliasDatabase, 57
- RegisterLogger
  - MVCommon::LoggerRegistry, 75
- RemoveLoggerSink
  - MVCommon::Logger, 73
- ResetJobs
  - MVCommon::ThreadPool, 125
- RotationTranslationMatrixInverted
  - MVCommon::Matrix4x4d, 87
  - MVCommon::Matrix4x4f, 98
- ScaleToResolution
  - MVCommon::CameraParams, 35
- SetAlpha
  - MVCommon::Color, 43
- SetAlphaByte



- MVCommon::Color, [43](#)
- SetBlue
  - MVCommon::Color, [44](#)
- SetBlueByte
  - MVCommon::Color, [44](#)
- SetGreen
  - MVCommon::Color, [44](#)
- SetGreenByte
  - MVCommon::Color, [44](#)
- SetLogLevel
  - MVCommon::ILoggerSink, [66](#)
  - MVCommon::Logger, [73](#)
- SetRed
  - MVCommon::Color, [45](#)
- SetRedByte
  - MVCommon::Color, [45](#)
- SetValue
  - MVCommon::Color, [45](#), [46](#)
- SharedGuidAliasDatabasePtr
  - MVCommon::SharedGuidAliasDatabasePtr, [104](#)
- SharedLoggerPtr
  - MVCommon::SharedLoggerPtr, [109](#)
- SharedLoggerSinkPtr
  - MVCommon::SharedLoggerSinkPtr, [112](#)
- SharedThreadPoolJobPtr
  - MVCommon::SharedThreadPoolJobPtr, [116](#)
- Size
  - MVCommon::ByteArray, [29](#)
- Skip
  - MVCommon::ByteArray, [30](#)
- StdOutLoggerSink
  - MVCommon::StdOutLoggerSink, [119](#)
- String
  - MVCommon::String, [120](#), [121](#)
- Subarray
  - MVCommon::ByteArray, [30](#)
- Substr
  - MVCommon::String, [122](#)
- ThreadPool
  - MVCommon::ThreadPool, [124](#)
- TimestampToString
  - MVCommon::ILoggerSink, [66](#)
- ToElementsVector
  - MVCommon::Versord, [172](#)
  - MVCommon::Versorf, [179](#)
- ToEulerAnglesZYX
  - MVCommon::Versord, [173](#)
  - MVCommon::Versorf, [179](#)
- ToHexString
  - MVCommon::Guid, [53](#)
- ToRawBytes
  - MVCommon::CameraParams, [35](#)
  - MVCommon::Guid, [53](#)
  - MVCommon::Matrix4x4d, [87](#)
  - MVCommon::Matrix4x4f, [98](#)
  - MVCommon::Vector2d, [130](#)
  - MVCommon::Vector2f, [136](#)
  - MVCommon::Vector3d, [143](#)
- MVCommon::Vector3f, [150](#)
- MVCommon::Vector4d, [157](#)
- MVCommon::Vector4f, [164](#)
- MVCommon::Versord, [173](#)
- MVCommon::Versorf, [179](#)
- ToRawElements
  - MVCommon::Matrix4x4d, [88](#)
  - MVCommon::Matrix4x4f, [99](#)
  - MVCommon::Versord, [173](#)
  - MVCommon::Versorf, [180](#)
- ToRfc4122
  - MVCommon::Guid, [53](#)
- ToRGB\_HTMLString
  - MVCommon::Color, [46](#)
- ToString
  - MVCommon::CameraParams, [35](#)
  - MVCommon::Color, [46](#)
  - MVCommon::Matrix4x4d, [88](#)
  - MVCommon::Matrix4x4f, [99](#)
  - MVCommon::Vector2d, [130](#)
  - MVCommon::Vector2f, [136](#)
  - MVCommon::Vector3d, [143](#)
  - MVCommon::Vector3f, [150](#)
  - MVCommon::Vector4d, [157](#)
  - MVCommon::Vector4f, [164](#)
  - MVCommon::VersionInfo, [166](#)
  - MVCommon::Versord, [174](#)
  - MVCommon::Versorf, [180](#)
- Transposed
  - MVCommon::Matrix4x4d, [88](#)
  - MVCommon::Matrix4x4f, [99](#)
- TryGetGuidAlias
  - MVCommon::GuidAliasDatabase, [58](#)
- TryGetGuidWithAlias
  - MVCommon::GuidAliasDatabase, [58](#)
- UndistortPoint
  - MVCommon::CameraParams, [36](#)
- UnregisterGuidAlias
  - MVCommon::GuidAliasDatabase, [58](#), [59](#)
- UnregisterLogger
  - MVCommon::LoggerRegistry, [75](#)
- Value
  - MVCommon::BlockingCounter, [19](#)
- Vector2d
  - MVCommon::Vector2d, [127](#)
- Vector2f
  - MVCommon::Vector2f, [133](#)
- Vector3d
  - MVCommon::Vector3d, [139](#)
- Vector3f
  - MVCommon::Vector3f, [146](#)
- Vector4d
  - MVCommon::Vector4d, [153](#)
- Vector4f
  - MVCommon::Vector4f, [160](#)
- VersionInfo
  - MVCommon::VersionInfo, [166](#)



WaitUntil  
    MVCommon::BlockingCounter, [20](#)  
WaitUntilFor  
    MVCommon::BlockingCounter, [20](#)  
WaitUntilValue  
    MVCommon::BlockingCounter, [20](#)  
WaitUntilValueFor  
    MVCommon::BlockingCounter, [21](#)  
WeakLoggerPtr  
    MVCommon::WeakLoggerPtr, [182](#)