

mogasens_csv

Generated by Doxygen 1.8.17

1 Namespace Index	1
1.1 Namespace List	1
2 Hierarchical Index	3
2.1 Class Hierarchy	3
3 Class Index	5
3.1 Class List	5
4 File Index	7
4.1 File List	7
5 Namespace Documentation	11
5.1 cl Namespace Reference	11
5.1.1 Typedef Documentation	12
5.1.1.1 column_type	12
5.1.1.2 Expected	13
5.1.2 Enumeration Type Documentation	13
5.1.2.1 Channel	13
5.1.2.2 Column	13
5.1.2.3 CsvFileKind	14
5.1.2.4 Sensor	14
5.1.3 Function Documentation	14
5.1.3.1 CL_SPECIALIZE_COL_TRAITS() [1/11]	14
5.1.3.2 CL_SPECIALIZE_COL_TRAITS() [2/11]	14
5.1.3.3 CL_SPECIALIZE_COL_TRAITS() [3/11]	15
5.1.3.4 CL_SPECIALIZE_COL_TRAITS() [4/11]	15
5.1.3.5 CL_SPECIALIZE_COL_TRAITS() [5/11]	15
5.1.3.6 CL_SPECIALIZE_COL_TRAITS() [6/11]	15
5.1.3.7 CL_SPECIALIZE_COL_TRAITS() [7/11]	15
5.1.3.8 CL_SPECIALIZE_COL_TRAITS() [8/11]	15
5.1.3.9 CL_SPECIALIZE_COL_TRAITS() [9/11]	16
5.1.3.10 CL_SPECIALIZE_COL_TRAITS() [10/11]	16
5.1.3.11 CL_SPECIALIZE_COL_TRAITS() [11/11]	16
5.1.3.12 dataSetAccessor()	16
5.1.3.13 dos2unix()	16
5.1.3.14 isAccelerometer()	17
5.1.3.15 isGyroscope()	18
5.1.3.16 operator<<() [1/4]	18
5.1.3.17 operator<<() [2/4]	18
5.1.3.18 operator<<() [3/4]	19
5.1.3.19 operator<<() [4/4]	20
5.1.3.20 readCsvFile()	20
5.1.3.21 s2n()	21

5.1.3.22 threshold()	21
5.1.3.23 to_string()	22
5.1.3.24 useUnbufferedIo()	22
5.1.4 Variable Documentation	22
5.1.4.1 accelerometerThreshold	23
5.1.4.2 channelCount	23
5.1.4.3 channels	23
5.1.4.4 column_index	23
5.1.4.5 data_set_accessor_v	23
5.1.4.6 gyroscopeThreshold	24
5.1.4.7 sensors	24
5.2 cl::fs Namespace Reference	24
5.2.1 Enumeration Type Documentation	25
5.2.1.1 DirectoryListingOption	25
5.2.2 Function Documentation	25
5.2.2.1 directoryListing()	25
5.2.2.2 formatError()	26
5.2.2.3 operator<()	27
5.2.2.4 operator<<()	27
5.2.2.5 operator==()	27
5.2.2.6 utf16ToUtf8()	28
5.2.2.7 utf8ToUtf16()	29
5.3 cm Namespace Reference	29
5.3.1 Enumeration Type Documentation	31
5.3.1.1 DataSetIdentifier	31
5.3.1.2 Imu	31
5.3.2 Function Documentation	32
5.3.2.1 closestOne()	32
5.3.2.2 CM_SORTER() [1/4]	33
5.3.2.3 CM_SORTER() [2/4]	33
5.3.2.4 CM_SORTER() [3/4]	33
5.3.2.5 CM_SORTER() [4/4]	33
5.3.2.6 confusionMatrixBestConfigs()	33
5.3.2.7 createSegmentationResults()	34
5.3.2.8 distance()	36
5.3.2.9 distanceScore()	36
5.3.2.10 fetch()	37
5.3.2.11 interpolatedDataSetPaths()	38
5.3.2.12 operator"!="() [1/2]	39
5.3.2.13 operator"!="() [2/2]	39
5.3.2.14 operator<() [1/2]	40
5.3.2.15 operator<() [2/2]	40

5.3.2.16 operator<<() [1/6]	40
5.3.2.17 operator<<() [2/6]	41
5.3.2.18 operator<<() [3/6]	41
5.3.2.19 operator<<() [4/6]	41
5.3.2.20 operator<<() [5/6]	42
5.3.2.21 operator<<() [6/6]	43
5.3.2.22 operator==() [1/2]	43
5.3.2.23 operator==() [2/2]	43
5.3.2.24 orderConfigurationsByQuality()	44
5.3.2.25 pythonOutput()	44
5.3.2.26 segment()	46
5.3.2.27 splitString()	47
5.3.2.28 toDataSetIdentifier()	47
5.3.3 Variable Documentation	48
5.3.3.1 imuCount	48
5.3.3.2 imus	49
5.4 cs Namespace Reference	49
5.4.1 Enumeration Type Documentation	50
5.4.1.1 FilterKind	50
5.4.1.2 SegmentationKind	50
5.4.2 Function Documentation	52
5.4.2.1 CS_SPECIALIZE_DATA_SET_INFO() [1/20]	52
5.4.2.2 CS_SPECIALIZE_DATA_SET_INFO() [2/20]	52
5.4.2.3 CS_SPECIALIZE_DATA_SET_INFO() [3/20]	52
5.4.2.4 CS_SPECIALIZE_DATA_SET_INFO() [4/20]	52
5.4.2.5 CS_SPECIALIZE_DATA_SET_INFO() [5/20]	53
5.4.2.6 CS_SPECIALIZE_DATA_SET_INFO() [6/20]	53
5.4.2.7 CS_SPECIALIZE_DATA_SET_INFO() [7/20]	53
5.4.2.8 CS_SPECIALIZE_DATA_SET_INFO() [8/20]	53
5.4.2.9 CS_SPECIALIZE_DATA_SET_INFO() [9/20]	53
5.4.2.10 CS_SPECIALIZE_DATA_SET_INFO() [10/20]	53
5.4.2.11 CS_SPECIALIZE_DATA_SET_INFO() [11/20]	54
5.4.2.12 CS_SPECIALIZE_DATA_SET_INFO() [12/20]	54
5.4.2.13 CS_SPECIALIZE_DATA_SET_INFO() [13/20]	54
5.4.2.14 CS_SPECIALIZE_DATA_SET_INFO() [14/20]	54
5.4.2.15 CS_SPECIALIZE_DATA_SET_INFO() [15/20]	54
5.4.2.16 CS_SPECIALIZE_DATA_SET_INFO() [16/20]	54
5.4.2.17 CS_SPECIALIZE_DATA_SET_INFO() [17/20]	55
5.4.2.18 CS_SPECIALIZE_DATA_SET_INFO() [18/20]	55
5.4.2.19 CS_SPECIALIZE_DATA_SET_INFO() [19/20]	55
5.4.2.20 CS_SPECIALIZE_DATA_SET_INFO() [20/20]	55
5.4.2.21 logFiles()	55

5.4.2.22 operator"!="	56
5.4.2.23 operator<<() [1/3]	57
5.4.2.24 operator<<() [2/3]	57
5.4.2.25 operator<<() [3/3]	57
5.4.2.26 operator==()	58
5.4.2.27 PL_DEFINE_EXCEPTION_TYPE()	58
5.4.2.28 repetitionCount()	58
5.4.3 Variable Documentation	59
5.4.3.1 logPath	59
5.4.3.2 oldLogPath	59
5.5 ctg Namespace Reference	60
5.5.1 Function Documentation	60
5.5.1.1 aboveThreshold()	60
5.5.1.2 averageComparisonValueCalculator()	61
5.5.1.3 halfMaximumComparisonValueCalculator()	62
5.5.1.4 isRelevant()	62
5.5.1.5 percentageOf()	63
5.5.1.6 runAboveThreshold()	64
5.6 fmc Namespace Reference	64
5.6.1 Function Documentation	65
5.6.1.1 adjustHardwareTimestamp()	65
5.6.1.2 convertToUnixLineEndings()	65
5.6.1.3 createBackupFile()	66
5.6.1.4 deleteNonBoschSensors()	67
5.6.1.5 deleteOutOfBoundsValues()	67
5.6.1.6 removeZerosFromField()	67
5.6.1.7 restoreFromBackup()	68
5.6.1.8 writeFile()	68
6 Class Documentation	71
6.1 cm::Configuration::Builder Class Reference	71
6.1.1 Detailed Description	71
6.1.2 Constructor & Destructor Documentation	72
6.1.2.1 Builder()	72
6.1.3 Member Function Documentation	72
6.1.3.1 build()	72
6.1.3.2 deleteTooClose()	73
6.1.3.3 deleteTooLowVariance()	74
6.1.3.4 filterKind()	75
6.1.3.5 imu()	76
6.1.3.6 segmentationKind()	77
6.1.3.7 skipWindow()	78

6.1.3.8 <code>windowSize()</code>	79
6.2 <code>cl::col_traits< Col ></code> Struct Template Reference	80
6.2.1 Detailed Description	80
6.3 <code>cm::Configuration</code> Class Reference	81
6.3.1 Detailed Description	82
6.3.2 Constructor & Destructor Documentation	82
6.3.2.1 <code>Configuration()</code>	82
6.3.3 Member Function Documentation	82
6.3.3.1 <code>createFilePath()</code>	83
6.3.3.2 <code>deleteTooClose()</code>	83
6.3.3.3 <code>deleteTooCloseOptions()</code>	84
6.3.3.4 <code>deleteTooLowVariance()</code>	84
6.3.3.5 <code>deleteTooLowVarianceOptions()</code>	85
6.3.3.6 <code>filterKind()</code>	85
6.3.3.7 <code>filterKindOptions()</code>	86
6.3.3.8 <code>importSegmentationPoints()</code>	86
6.3.3.9 <code>imu()</code>	87
6.3.3.10 <code>imuOptions()</code>	88
6.3.3.11 <code>isInitialized()</code>	88
6.3.3.12 <code>segmentationKind()</code>	89
6.3.3.13 <code>segmentationKindOptions()</code>	89
6.3.3.14 <code>serializeSegmentationPoints()</code>	89
6.3.3.15 <code>skipWindow()</code>	90
6.3.3.16 <code>skipWindowOptions()</code>	91
6.3.3.17 <code>windowSize()</code>	91
6.3.3.18 <code>windowSizeOptions()</code>	92
6.3.4 Friends And Related Function Documentation	92
6.3.4.1 <code>Builder</code>	92
6.3.4.2 <code>operator"!="</code>	92
6.3.4.3 <code>operator<<</code>	93
6.3.4.4 <code>operator==</code>	93
6.3.4.5 <code>std::hash< Configuration ></code>	94
6.4 <code>cm::ConfigWithDistanceScore</code> Struct Reference	94
6.4.1 Detailed Description	94
6.4.2 Constructor & Destructor Documentation	94
6.4.2.1 <code>ConfigWithDistanceScore()</code>	95
6.4.3 Member Data Documentation	95
6.4.3.1 <code>config</code>	95
6.4.3.2 <code>distScore</code>	95
6.5 <code>cm::ConfigWithTotalConfusionMatrix</code> Struct Reference	95
6.5.1 Detailed Description	96
6.5.2 Constructor & Destructor Documentation	96

6.5.2.1 ConfigWithTotalConfusionMatrix() [1/2]	96
6.5.2.2 ConfigWithTotalConfusionMatrix() [2/2]	96
6.5.3 Member Data Documentation	96
6.5.3.1 config	96
6.5.3.2 matrix	97
6.6 cm::ConfusionMatrix Class Reference	97
6.6.1 Detailed Description	97
6.6.2 Member Typedef Documentation	97
6.6.2.1 this_type	97
6.6.3 Constructor & Destructor Documentation	98
6.6.3.1 ConfusionMatrix()	98
6.6.4 Member Function Documentation	98
6.6.4.1 falseNegatives()	98
6.6.4.2 falsePositives()	98
6.6.4.3 incrementFalseNegatives()	98
6.6.4.4 incrementFalsePositives()	99
6.6.4.5 incrementTrueNegatives()	99
6.6.4.6 incrementTruePositives()	99
6.6.4.7 operator+=()	100
6.6.4.8 totalCount()	100
6.6.4.9 trueNegatives()	100
6.6.4.10 truePositives()	100
6.7 cm::CsvFileInfo Class Reference	101
6.7.1 Detailed Description	101
6.7.2 Constructor & Destructor Documentation	101
6.7.2.1 CsvFileInfo()	101
6.7.3 Member Function Documentation	101
6.7.3.1 hardwareTimestamps()	102
6.8 cs::CsvLineBuilder Class Reference	102
6.8.1 Detailed Description	103
6.8.2 Member Typedef Documentation	103
6.8.2.1 this_type	103
6.8.3 Constructor & Destructor Documentation	103
6.8.3.1 CsvLineBuilder()	103
6.8.4 Member Function Documentation	103
6.8.4.1 build()	104
6.8.4.2 dataSet()	104
6.8.4.3 deleteLowVariance()	105
6.8.4.4 deleteTooClose()	106
6.8.4.5 filter()	107
6.8.4.6 isOld()	108
6.8.4.7 kind()	109

6.8.4.8 repetitions()	110
6.8.4.9 segmentationPoints()	111
6.8.4.10 sensor()	112
6.8.4.11 skipWindow()	113
6.8.4.12 windowSize()	114
6.9 cl::data_set_accessor< Chan > Struct Template Reference	115
6.9.1 Detailed Description	115
6.10 cs::data_set_info< Tag > Struct Template Reference	116
6.10.1 Detailed Description	116
6.11 cl::DataPoint Class Reference	116
6.11.1 Detailed Description	116
6.11.2 Constructor & Destructor Documentation	117
6.11.2.1 DataPoint()	117
6.11.3 Member Function Documentation	117
6.11.3.1 channel()	117
6.11.3.2 fileName()	118
6.11.3.3 sensor()	118
6.11.3.4 time()	119
6.11.3.5 value()	119
6.11.4 Friends And Related Function Documentation	119
6.11.4.1 operator<<	120
6.12 cl::DataSet Class Reference	120
6.12.1 Detailed Description	121
6.12.2 Member Typedef Documentation	121
6.12.2.1 ChannelAccessor	121
6.12.2.2 size_type	121
6.12.3 Member Function Documentation	121
6.12.3.1 accelerometerAverage()	121
6.12.3.2 accelerometerMaximum()	122
6.12.3.3 accelerometerX()	122
6.12.3.4 accelerometerY()	123
6.12.3.5 accelerometerZ()	123
6.12.3.6 create()	124
6.12.3.7 extractId()	125
6.12.3.8 fileName()	125
6.12.3.9 gyroscopeAverage()	126
6.12.3.10 gyroscopeMaximum()	127
6.12.3.11 gyroscopeX()	127
6.12.3.12 gyroscopeY()	128
6.12.3.13 gyroscopeZ()	128
6.12.3.14 hardwareTimestamp()	129
6.12.3.15 rowCount()	129

6.12.3.16 time()	129
6.12.3.17 trigger()	130
6.13 cl::Error Class Reference	130
6.13.1 Detailed Description	131
6.13.2 Member Enumeration Documentation	131
6.13.2.1 Kind	131
6.13.3 Constructor & Destructor Documentation	131
6.13.3.1 Error()	131
6.13.4 Member Function Documentation	132
6.13.4.1 file()	132
6.13.4.2 function()	132
6.13.4.3 kind()	133
6.13.4.4 line()	133
6.13.4.5 message()	133
6.13.4.6 raise()	134
6.13.4.7 to_string()	134
6.13.5 Friends And Related Function Documentation	134
6.13.5.1 operator<<	134
6.14 cl::Exception Class Reference	134
6.14.1 Detailed Description	135
6.14.2 Member Typedef Documentation	135
6.14.2.1 base_type	135
6.14.3 Constructor & Destructor Documentation	136
6.14.3.1 Exception() [1/2]	136
6.14.3.2 Exception() [2/2]	136
6.14.4 Member Function Documentation	136
6.14.4.1 file()	136
6.14.4.2 function()	137
6.14.4.3 line()	137
6.15 cl::fs::File Class Reference	137
6.15.1 Detailed Description	138
6.15.2 Constructor & Destructor Documentation	138
6.15.2.1 File()	138
6.15.3 Member Function Documentation	139
6.15.3.1 copyTo()	139
6.15.3.2 create()	140
6.15.3.3 exists()	140
6.15.3.4 moveTo()	141
6.15.3.5 path()	142
6.15.3.6 remove()	142
6.15.3.7 size()	143
6.16 cl::fs::FileStream Class Reference	143

6.16.1 Detailed Description	144
6.16.2 Member Typedef Documentation	144
6.16.2.1 this_type	144
6.16.3 Member Enumeration Documentation	144
6.16.3.1 OpenMode	144
6.16.4 Constructor & Destructor Documentation	145
6.16.4.1 FileStream()	145
6.16.4.2 ~FileStream()	145
6.16.5 Member Function Documentation	145
6.16.5.1 create()	145
6.16.5.2 operator=()	146
6.16.5.3 PL_NONCOPYABLE()	147
6.16.5.4 readAll()	147
6.16.5.5 write()	147
6.17 std::hash<::cl::fs::Path > Struct Reference	148
6.17.1 Detailed Description	148
6.17.2 Member Function Documentation	148
6.17.2.1 operator()()	148
6.18 std::hash<::cm::Configuration > Struct Reference	148
6.18.1 Detailed Description	148
6.18.2 Member Function Documentation	148
6.18.2.1 operator()()	149
6.19 cs::LogInfo Class Reference	149
6.19.1 Detailed Description	150
6.19.2 Constructor & Destructor Documentation	150
6.19.2.1 LogInfo()	150
6.19.3 Member Function Documentation	150
6.19.3.1 create()	150
6.19.3.2 deleteLowVariance()	151
6.19.3.3 deleteTooClose()	151
6.19.3.4 filterKind()	152
6.19.3.5 isInitialized()	152
6.19.3.6 logFilePath()	152
6.19.3.7 segmentationKind()	153
6.19.3.8 sensor()	153
6.19.3.9 skipWindow()	153
6.19.3.10 windowSize()	154
6.19.4 Friends And Related Function Documentation	154
6.19.4.1 operator"!="	154
6.19.4.2 operator<<	154
6.19.4.3 operator==	155
6.19.5 Member Data Documentation	155

6.19.5.1 invalidSensor	155
6.20 cs::LogLine Class Reference	155
6.20.1 Detailed Description	156
6.20.2 Member Function Documentation	156
6.20.2.1 fileName()	156
6.20.2.2 filePath()	157
6.20.2.3 parse()	157
6.20.2.4 segmentationPointCount()	158
6.20.2.5 sensor()	158
6.20.3 Member Data Documentation	159
6.20.3.1 invalidSensor	159
6.21 cm::ManualSegmentationPoint Class Reference	159
6.21.1 Detailed Description	160
6.21.2 Constructor & Destructor Documentation	160
6.21.2.1 ManualSegmentationPoint()	160
6.21.3 Member Function Documentation	161
6.21.3.1 asMilliseconds()	161
6.21.3.2 convertToHardwareTimestamps()	161
6.21.3.3 frame()	162
6.21.3.4 hour()	163
6.21.3.5 minute()	164
6.21.3.6 readCsvFile()	164
6.21.3.7 second()	165
6.21.4 Friends And Related Function Documentation	165
6.21.4.1 operator"!="	165
6.21.4.2 operator<<	166
6.21.4.3 operator==	166
6.22 cl::fs::Path Class Reference	167
6.22.1 Detailed Description	167
6.22.2 Constructor & Destructor Documentation	168
6.22.2.1 Path() [1/3]	168
6.22.2.2 Path() [2/3]	168
6.22.2.3 Path() [3/3]	168
6.22.3 Member Function Documentation	168
6.22.3.1 exists()	169
6.22.3.2 isDirectory()	169
6.22.3.3 isFile()	170
6.22.3.4 str()	171
6.22.4 Friends And Related Function Documentation	171
6.22.4.1 operator<	171
6.22.4.2 operator<<	172
6.22.4.3 operator==	172

6.23 cl::Process Class Reference	173
6.23.1 Detailed Description	173
6.23.2 Member Typedef Documentation	173
6.23.2.1 this_type	173
6.23.3 Constructor & Destructor Documentation	173
6.23.3.1 Process()	173
6.23.3.2 ~Process()	174
6.23.4 Member Function Documentation	174
6.23.4.1 create()	174
6.23.4.2 file() [1/2]	174
6.23.4.3 file() [2/2]	174
6.23.4.4 operator=()	174
6.23.4.5 PL_NONCOPYABLE()	175
7 File Documentation	177
7.1 compare_segmentation/CMakeLists.txt File Reference	177
7.1.1 Function Documentation	177
7.1.1.1 set()	177
7.2 compare_segmentation/test/CMakeLists.txt File Reference	177
7.2.1 Function Documentation	177
7.2.1.1 include()	178
7.3 counting/CMakeLists.txt File Reference	178
7.3.1 Function Documentation	178
7.3.1.1 set()	178
7.4 counting/test/CMakeLists.txt File Reference	178
7.4.1 Function Documentation	178
7.4.1.1 include()	178
7.5 csv_lib/CMakeLists.txt File Reference	179
7.5.1 Function Documentation	179
7.5.1.1 set()	179
7.6 csv_lib/test/CMakeLists.txt File Reference	179
7.6.1 Function Documentation	179
7.6.1.1 include()	179
7.7 fix_csv/CMakeLists.txt File Reference	180
7.7.1 Function Documentation	180
7.7.1.1 set()	180
7.8 fix_csv/test/CMakeLists.txt File Reference	180
7.8.1 Function Documentation	180
7.8.1.1 include()	180
7.9 confusion_matrix/CMakeLists.txt File Reference	181
7.9.1 Function Documentation	181
7.9.1.1 set()	181

7.10 confusion_matrix/test/CMakeLists.txt File Reference	181
7.10.1 Function Documentation	181
7.10.1.1 include()	181
7.11 compare_segmentation/include/csv_line.hpp File Reference	182
7.12 compare_segmentation/include/data_set_info.hpp File Reference	183
7.12.1 Macro Definition Documentation	184
7.12.1.1 CS_SPECIALIZE_DATA_SET_INFO	184
7.13 compare_segmentation/include/filter_kind.hpp File Reference	185
7.14 compare_segmentation/include/log_files.hpp File Reference	186
7.15 compare_segmentation/include/log_info.hpp File Reference	186
7.16 compare_segmentation/include/log_line.hpp File Reference	187
7.17 compare_segmentation/include/paths.hpp File Reference	188
7.18 compare_segmentation/include/segmentation_kind.hpp File Reference	189
7.19 compare_segmentation/src/csv_line.cpp File Reference	190
7.20 compare_segmentation/src/data_set_info.cpp File Reference	191
7.21 compare_segmentation/src/filter_kind.cpp File Reference	191
7.22 compare_segmentation/src/log_files.cpp File Reference	192
7.23 compare_segmentation/src/log_info.cpp File Reference	193
7.24 compare_segmentation/src/log_line.cpp File Reference	194
7.25 compare_segmentation/src/main.cpp File Reference	194
7.25.1 Function Documentation	195
7.25.1.1 main()	195
7.26 compare_segmentation/test/main.cpp File Reference	196
7.26.1 Function Documentation	197
7.26.1.1 main()	197
7.27 counting/src/main.cpp File Reference	197
7.27.1 Function Documentation	198
7.27.1.1 main()	198
7.28 counting/test/main.cpp File Reference	199
7.28.1 Function Documentation	200
7.28.1.1 main()	200
7.29 csv_lib/test/main.cpp File Reference	200
7.29.1 Function Documentation	201
7.29.1.1 main()	201
7.30 fix_csv/src/main.cpp File Reference	201
7.30.1 Function Documentation	202
7.30.1.1 main()	202
7.31 fix_csv/test/main.cpp File Reference	202
7.31.1 Function Documentation	203
7.31.1.1 main()	203
7.32 confusion_matrix/src/main.cpp File Reference	203
7.32.1 Macro Definition Documentation	204

7.32.1.1 SORT_PRINT	204
7.32.2 Function Documentation	204
7.32.2.1 main()	204
7.33 confusion_matrix/test/main.cpp File Reference	205
7.33.1 Function Documentation	206
7.33.1.1 main()	206
7.34 compare_segmentation/src/segmentation_kind.cpp File Reference	206
7.35 compare_segmentation/test/csv_line_test.cpp File Reference	207
7.35.1 Function Documentation	208
7.35.1.1 TEST()	208
7.36 compare_segmentation/test/data_set_info_test.cpp File Reference	208
7.36.1 Function Documentation	208
7.36.1.1 TEST()	209
7.37 compare_segmentation/test/log_files_test.cpp File Reference	209
7.37.1 Function Documentation	209
7.37.1.1 TEST() [1/3]	210
7.37.1.2 TEST() [2/3]	210
7.37.1.3 TEST() [3/3]	211
7.38 compare_segmentation/test/log_info_test.cpp File Reference	211
7.38.1 Function Documentation	212
7.38.1.1 TEST() [1/19]	212
7.38.1.2 TEST() [2/19]	212
7.38.1.3 TEST() [3/19]	213
7.38.1.4 TEST() [4/19]	213
7.38.1.5 TEST() [5/19]	214
7.38.1.6 TEST() [6/19]	214
7.38.1.7 TEST() [7/19]	215
7.38.1.8 TEST() [8/19]	215
7.38.1.9 TEST() [9/19]	216
7.38.1.10 TEST() [10/19]	216
7.38.1.11 TEST() [11/19]	217
7.38.1.12 TEST() [12/19]	217
7.38.1.13 TEST() [13/19]	218
7.38.1.14 TEST() [14/19]	218
7.38.1.15 TEST() [15/19]	219
7.38.1.16 TEST() [16/19]	219
7.38.1.17 TEST() [17/19]	220
7.38.1.18 TEST() [18/19]	220
7.38.1.19 TEST() [19/19]	221
7.39 compare_segmentation/test/log_line_test.cpp File Reference	221
7.39.1 Function Documentation	221
7.39.1.1 TEST() [1/4]	222

7.39.1.2 TEST() [2/4]	222
7.39.1.3 TEST() [3/4]	223
7.39.1.4 TEST() [4/4]	223
7.40 confusion_matrix/include/closest_one.hpp File Reference	223
7.41 confusion_matrix/include/configuration.hpp File Reference	224
7.42 confusion_matrix/include/confusion_matrix.hpp File Reference	226
7.43 confusion_matrix/include/confusion_matrix_best_configs.hpp File Reference	227
7.43.1 Macro Definition Documentation	228
7.43.1.1 CM_SORTER	228
7.44 confusion_matrix/include/create_segmentation_results.hpp File Reference	228
7.45 confusion_matrix/include/csv_file_info.hpp File Reference	229
7.46 confusion_matrix/include/data_set_identifier.hpp File Reference	230
7.46.1 Macro Definition Documentation	231
7.46.1.1 CM_DATA_SET_IDENTIFIER	232
7.46.1.2 CM_DATA_SET_IDENTIFIER_X	232
7.47 confusion_matrix/include/distance.hpp File Reference	232
7.48 confusion_matrix/include/distance_score.hpp File Reference	233
7.49 confusion_matrix/include/fetch.hpp File Reference	234
7.50 confusion_matrix/include/imu.hpp File Reference	235
7.50.1 Macro Definition Documentation	236
7.50.1.1 CM_IMU	237
7.50.1.2 CM_IMU_X [1/3]	237
7.50.1.3 CM_IMU_X [2/3]	237
7.50.1.4 CM_IMU_X [3/3]	237
7.51 confusion_matrix/include/interpolated_data_set_paths.hpp File Reference	238
7.52 confusion_matrix/include/manual_segmentation_point.hpp File Reference	239
7.53 confusion_matrix/include/order_configurations_by_quality.hpp File Reference	240
7.54 confusion_matrix/include/python_output.hpp File Reference	241
7.55 confusion_matrix/include/segment.hpp File Reference	242
7.56 confusion_matrix/include/split_string.hpp File Reference	243
7.57 confusion_matrix/src/closest_one.cpp File Reference	243
7.58 confusion_matrix/src/configuration.cpp File Reference	244
7.58.1 Macro Definition Documentation	245
7.58.1.1 CM_ENSURE_CONTAINS	245
7.58.1.2 CM_ENSURE_HAS_VALUE	245
7.59 confusion_matrix/src/confusion_matrix.cpp File Reference	246
7.60 confusion_matrix/src/confusion_matrix_best_configs.cpp File Reference	246
7.61 confusion_matrix/src/create_segmentation_results.cpp File Reference	247
7.62 confusion_matrix/src/csv_file_info.cpp File Reference	248
7.63 confusion_matrix/src/data_set_identifier.cpp File Reference	248
7.63.1 Macro Definition Documentation	249
7.63.1.1 CM_DATA_SET_IDENTIFIER_X	249

7.63.1.2 DSI	249
7.64 confusion_matrix/src/distance.cpp File Reference	249
7.65 confusion_matrix/src/distance_score.cpp File Reference	250
7.66 confusion_matrix/src/imu.cpp File Reference	250
7.66.1 Macro Definition Documentation	251
7.66.1.1 CM_IMU_X	251
7.67 confusion_matrix/src/interpolated_data_set_paths.cpp File Reference	252
7.68 confusion_matrix/src/manual_segmentation_point.cpp File Reference	252
7.68.1 Macro Definition Documentation	253
7.68.1.1 DSI	253
7.69 confusion_matrix/src/order_configurations_by_quality.cpp File Reference	253
7.70 confusion_matrix/src/python_output.cpp File Reference	254
7.70.1 Macro Definition Documentation	255
7.70.1.1 CM_DEV_NULL	255
7.70.1.2 CM_SEGMENTOR	255
7.71 confusion_matrix/src/segment.cpp File Reference	255
7.72 confusion_matrix/src/split_string.cpp File Reference	256
7.73 confusion_matrix/test/data_set_identifier_test.cpp File Reference	257
7.73.1 Macro Definition Documentation	257
7.73.1.1 DSI	257
7.73.2 Function Documentation	257
7.73.2.1 TEST()	258
7.74 confusion_matrix/test/interpolated_data_set_paths_test.cpp File Reference	258
7.74.1 Function Documentation	258
7.74.1.1 TEST()	259
7.75 confusion_matrix/test/manual_segmentation_point_test.cpp File Reference	259
7.75.1 Macro Definition Documentation	260
7.75.1.1 DSI	260
7.75.2 Function Documentation	260
7.75.2.1 TEST() [1/11]	260
7.75.2.2 TEST() [2/11]	261
7.75.2.3 TEST() [3/11]	261
7.75.2.4 TEST() [4/11]	261
7.75.2.5 TEST() [5/11]	261
7.75.2.6 TEST() [6/11]	261
7.75.2.7 TEST() [7/11]	262
7.75.2.8 TEST() [8/11]	262
7.75.2.9 TEST() [9/11]	262
7.75.2.10 TEST() [10/11]	262
7.75.2.11 TEST() [11/11]	262
7.76 confusion_matrix/test/segment_test.cpp File Reference	263
7.76.1 Macro Definition Documentation	263

7.76.1.1 EXPECT_SEGMENTATION_POINTS	263
7.76.2 Function Documentation	263
7.76.2.1 TEST()	264
7.77 confusion_matrix/test/split_string_test.cpp File Reference	264
7.77.1 Function Documentation	265
7.77.1.1 TEST()	265
7.78 counting/include/above_threshold.hpp File Reference	266
7.79 counting/include/average_comparison_value_calculator.hpp File Reference	267
7.80 counting/include/half_maximum_comparison_value_calculator.hpp File Reference	267
7.81 counting/include/is_relevant.hpp File Reference	268
7.82 counting/include/percentage_of.hpp File Reference	269
7.83 counting/include/run_above_threshold.hpp File Reference	270
7.84 counting/src/above_threshold.cpp File Reference	271
7.84.1 Macro Definition Documentation	272
7.84.1.1 CL_CHANNEL_X	272
7.84.2 Variable Documentation	272
7.84.2.1 channel	273
7.84.2.2 channelAccessor	273
7.85 counting/src/average_comparison_value_calculator.cpp File Reference	273
7.86 counting/src/half_maximum_comparison_value_calculator.cpp File Reference	274
7.87 counting/src/run_above_threshold.cpp File Reference	274
7.88 counting/test/above_threshold_test.cpp File Reference	275
7.88.1 Macro Definition Documentation	275
7.88.1.1 EXPECT_LONG_DOUBLE_EQ	276
7.88.2 Function Documentation	276
7.88.2.1 TEST()	276
7.89 counting/test/percentage_of_test.cpp File Reference	277
7.89.1 Macro Definition Documentation	277
7.89.1.1 EXPECT_LONG_DOUBLE_EQ	277
7.89.2 Function Documentation	277
7.89.2.1 TEST()	278
7.90 csv_lib/include/cl/channel.hpp File Reference	278
7.90.1 Macro Definition Documentation	279
7.90.1.1 CL_CHANNEL	280
7.90.1.2 CL_CHANNEL_X [1/4]	280
7.90.1.3 CL_CHANNEL_X [2/4]	280
7.90.1.4 CL_CHANNEL_X [3/4]	280
7.90.1.5 CL_CHANNEL_X [4/4]	281
7.91 csv_lib/include/cl/column.hpp File Reference	281
7.91.1 Macro Definition Documentation	282
7.91.1.1 CL_SPECIALIZE_COL_TRAITS	283
7.92 csv_lib/include/cl/data_point.hpp File Reference	283

7.93 csv_lib/include/cl/data_set.hpp File Reference	284
7.94 csv_lib/include/cl/dos2unix.hpp File Reference	285
7.95 csv_lib/include/cl/error.hpp File Reference	286
7.95.1 Macro Definition Documentation	286
7.95.1.1 CL_ERROR_KIND	287
7.95.1.2 CL_ERROR_KIND_X	287
7.95.1.3 CL_UNEXPECTED	287
7.96 csv_lib/include/cl/exception.hpp File Reference	287
7.96.1 Macro Definition Documentation	288
7.96.1.1 CL_THROW	288
7.96.1.2 CL_THROW_FMT	288
7.97 csv_lib/include/cl/fs/directory_listing.hpp File Reference	289
7.98 csv_lib/include/cl/fs/file.hpp File Reference	290
7.99 csv_lib/include/cl/fs/file_stream.hpp File Reference	290
7.100 csv_lib/include/cl/fs/path.hpp File Reference	291
7.101 csv_lib/include/cl/fs/separator.hpp File Reference	292
7.101.1 Macro Definition Documentation	293
7.101.1.1 CL_FS_SEPARATOR	293
7.102 csv_lib/include/cl/fs/windows.hpp File Reference	293
7.102.1 Detailed Description	294
7.103 csv_lib/include/cl/process.hpp File Reference	295
7.104 csv_lib/include/cl/read_csv_file.hpp File Reference	295
7.105 csv_lib/include/cl/s2n.hpp File Reference	296
7.106 csv_lib/include/cl/sensor.hpp File Reference	297
7.106.1 Macro Definition Documentation	298
7.106.1.1 CL_SENSOR	298
7.106.1.2 CL_SENSOR_X [1/2]	299
7.106.1.3 CL_SENSOR_X [2/2]	299
7.107 csv_lib/include/cl/to_string.hpp File Reference	299
7.108 csv_lib/include/cl/use_unbuffered_io.hpp File Reference	300
7.109 csv_lib/src/cl/channel.cpp File Reference	300
7.109.1 Macro Definition Documentation	301
7.109.1.1 CL_CHANNEL_X [1/2]	301
7.109.1.2 CL_CHANNEL_X [2/2]	301
7.110 csv_lib/src/cl/data_point.cpp File Reference	302
7.110.1 Function Documentation	302
7.110.1.1 channel()	302
7.110.1.2 fileName()	303
7.110.1.3 sensor()	303
7.110.1.4 time()	304
7.110.1.5 value()	304
7.111 csv_lib/src/cl/data_set.cpp File Reference	305

7.112 csv_lib/src/cl/dos2unix.cpp File Reference	305
7.113 csv_lib/src/cl/error.cpp File Reference	306
7.113.1 Macro Definition Documentation	306
7.113.1.1 CL_ERROR_KIND_X	307
7.114 csv_lib/src/cl/exception.cpp File Reference	307
7.115 csv_lib/src/cl/fs/directory_listing.cpp File Reference	307
7.116 csv_lib/src/cl/fs/file.cpp File Reference	308
7.117 csv_lib/src/cl/fs/file_stream.cpp File Reference	308
7.118 csv_lib/src/cl/fs/path.cpp File Reference	309
7.119 csv_lib/src/cl/fs/windows.cpp File Reference	310
7.120 csv_lib/src/cl/process.cpp File Reference	310
7.121 csv_lib/src/cl/read_csv_file.cpp File Reference	311
7.122 csv_lib/src/cl/sensor.cpp File Reference	312
7.122.1 Macro Definition Documentation	312
7.122.1.1 CL_SENSOR_X	312
7.123 csv_lib/src/cl/use_unbuffered_io.cpp File Reference	313
7.124 csv_lib/test/channel_test.cpp File Reference	313
7.124.1 Function Documentation	314
7.124.1.1 TEST() [1/4]	314
7.124.1.2 TEST() [2/4]	314
7.124.1.3 TEST() [3/4]	314
7.124.1.4 TEST() [4/4]	315
7.125 csv_lib/test/column_test.cpp File Reference	315
7.125.1 Function Documentation	316
7.125.1.1 TEST() [1/2]	316
7.125.1.2 TEST() [2/2]	316
7.126 csv_lib/test/data_point_test.cpp File Reference	317
7.126.1 Function Documentation	317
7.126.1.1 TEST() [1/2]	317
7.126.1.2 TEST() [2/2]	318
7.126.2 Variable Documentation	318
7.126.2.1 dp	318
7.127 csv_lib/test/data_set_test.cpp File Reference	319
7.127.1 Macro Definition Documentation	319
7.127.1.1 EXPECT_LONG_DOUBLE_EQ	319
7.127.2 Function Documentation	319
7.127.2.1 TEST() [1/4]	320
7.127.2.2 TEST() [2/4]	320
7.127.2.3 TEST() [3/4]	321
7.127.2.4 TEST() [4/4]	322
7.128 csv_lib/test/directory_listing_test.cpp File Reference	323
7.128.1 Function Documentation	324

7.128.1.1 TEST() [1/3]	324
7.128.1.2 TEST() [2/3]	324
7.128.1.3 TEST() [3/3]	325
7.129 csv_lib/test/error_test.cpp File Reference	325
7.129.1 Function Documentation	326
7.129.1.1 TEST() [1/4]	326
7.129.1.2 TEST() [2/4]	326
7.129.1.3 TEST() [3/4]	326
7.129.1.4 TEST() [4/4]	326
7.129.2 Variable Documentation	326
7.129.2.1 error	327
7.130 csv_lib/test/exception_test.cpp File Reference	327
7.130.1 Function Documentation	327
7.130.1.1 TEST()	327
7.131 csv_lib/test/read_csv_file_test.cpp File Reference	328
7.131.1 Function Documentation	328
7.131.1.1 TEST() [1/2]	328
7.131.1.2 TEST() [2/2]	329
7.132 csv_lib/test/s2n_test.cpp File Reference	329
7.132.1 Function Documentation	330
7.132.1.1 TEST() [1/3]	330
7.132.1.2 TEST() [2/3]	330
7.132.1.3 TEST() [3/3]	331
7.133 csv_lib/test/sensor_test.cpp File Reference	331
7.133.1 Function Documentation	332
7.133.1.1 TEST() [1/2]	332
7.133.1.2 TEST() [2/2]	332
7.134 csv_lib/test/to_string_test.cpp File Reference	332
7.134.1 Function Documentation	333
7.134.1.1 TEST()	333
7.135 fix_csv/include/adjust_hardware_timestamp.hpp File Reference	333
7.136 fix_csv/include/convert_to_unix_line_endings.hpp File Reference	334
7.137 fix_csv/include/create_backup_file.hpp File Reference	335
7.138 fix_csv/include/delete_non_bosch_sensors.hpp File Reference	336
7.139 fix_csv/include/delete_out_of_bounds_values.hpp File Reference	337
7.140 fix_csv/include/remove_zeros_from_field.hpp File Reference	338
7.141 fix_csv/include/restore_from_backup.hpp File Reference	339
7.142 fix_csv/include/write_file.hpp File Reference	340
7.143 fix_csv/src/adjust_hardware_timestamp.cpp File Reference	341
7.144 fix_csv/src/convert_to_unix_line_endings.cpp File Reference	342
7.145 fix_csv/src/create_backup_file.cpp File Reference	343
7.146 fix_csv/src/delete_non_bosch_sensors.cpp File Reference	343

7.146.1 Macro Definition Documentation	344
7.146.1.1 CL_SENSOR_X	344
7.147 fix_csv/src/delete_out_of_bounds_values.cpp File Reference	344
7.148 fix_csv/src/remove_zeros_from_field.cpp File Reference	345
7.149 fix_csv/src/restore_from_backup.cpp File Reference	345
7.150 fix_csv/src/write_file.cpp File Reference	346
7.151 fix_csv/test/adjust_hardware_timestamp_test.cpp File Reference	347
7.151.1 Function Documentation	347
7.151.1.1 TEST() [1/5]	347
7.151.1.2 TEST() [2/5]	348
7.151.1.3 TEST() [3/5]	348
7.151.1.4 TEST() [4/5]	349
7.151.1.5 TEST() [5/5]	349
7.152 fix_csv/test/remove_zeros_from_field_test.cpp File Reference	349
7.152.1 Function Documentation	350
7.152.1.1 TEST() [1/6]	350
7.152.1.2 TEST() [2/6]	351
7.152.1.3 TEST() [3/6]	351
7.152.1.4 TEST() [4/6]	352
7.152.1.5 TEST() [5/6]	352
7.152.1.6 TEST() [6/6]	353
Index	355

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

cl	11
cl::fs	24
cm	29
cs	49
ctg	60
fmc	64

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

cm::Configuration::Builder	71
cl::col_traits< Col >	80
cm::Configuration	81
cm::ConfigWithDistanceScore	94
cm::ConfigWithTotalConfusionMatrix	95
cm::ConfusionMatrix	97
cm::CsvFileInfo	101
cs::CsvLineBuilder	102
cl::data_set_accessor< Chan >	115
cs::data_set_info< Tag >	116
cl::DataPoint	116
cl::DataSet	120
cl::Error	130
std::exception	
std::runtime_error	
cl::Exception	134
cl::fs::File	137
cl::fs::FileStream	143
std::hash<::cl::fs::Path >	148
std::hash<::cm::Configuration >	148
cs::LogInfo	149
cs::LogLine	155
cm::ManualSegmentationPoint	159
cl::fs::Path	167
cl::Process	173

Chapter 3

Class Index

3.1 Class List

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

<code>cm::Configuration::Builder</code>	Builder type for <code>Configuration</code>	71
<code>cl::col_traits< Col ></code>		80
<code>cm::Configuration</code>	Represents a possible configuration for the Python segmentor	81
<code>cm::ConfigWithDistanceScore</code>		94
<code>cm::ConfigWithTotalConfusionMatrix</code>		95
<code>cm::ConfusionMatrix</code>		97
<code>cm::CsvFileInfo</code>	Type to hold the hardware timestamps of a CSV file	101
<code>cs::CsvLineBuilder</code>	Builder for a CSV line	102
<code>cl::data_set_accessor< Chan ></code>		115
<code>cs::data_set_info< Tag ></code>	Meta function for data set tags	116
<code>cl::DataPoint</code>		116
<code>cl::DataSet</code>		120
<code>cl::Error</code>		130
<code>cl::Exception</code>		134
<code>cl::fs::File</code>	Represents a file	137
<code>cl::fs::FileStream</code>	A binary file stream	143
<code>std::hash<::cl::fs::Path ></code>		148
<code>std::hash<::cm::Configuration ></code>		148
<code>cs::LogInfo</code>	Information about a log file	149
<code>cs::LogLine</code>	A line out of a log file	155
<code>cm::ManualSegmentationPoint</code>	Type used to represent a manual segmentation point	159
<code>cl::fs::Path</code>	A filesystem path	167
<code>cl::Process</code>		173

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

compare_segmentation/include/csv_line.hpp	182
compare_segmentation/include/data_set_info.hpp	183
compare_segmentation/include/filter_kind.hpp	185
compare_segmentation/include/log_files.hpp	186
compare_segmentation/include/log_info.hpp	186
compare_segmentation/include/log_line.hpp	187
compare_segmentation/include/paths.hpp	188
compare_segmentation/include/segmentation_kind.hpp	189
compare_segmentation/src/csv_line.cpp	190
compare_segmentation/src/data_set_info.cpp	191
compare_segmentation/src/filter_kind.cpp	191
compare_segmentation/src/log_files.cpp	192
compare_segmentation/src/log_info.cpp	193
compare_segmentation/src/log_line.cpp	194
compare_segmentation/src/main.cpp	194
compare_segmentation/src/segmentation_kind.cpp	206
compare_segmentation/test/csv_line_test.cpp	207
compare_segmentation/test/data_set_info_test.cpp	208
compare_segmentation/test/log_files_test.cpp	209
compare_segmentation/test/log_info_test.cpp	211
compare_segmentation/test/log_line_test.cpp	221
compare_segmentation/test/main.cpp	196
confusion_matrix/include/closest_one.hpp	223
confusion_matrix/include/configuration.hpp	224
confusion_matrix/include/confusion_matrix.hpp	226
confusion_matrix/include/confusion_matrix_best_configs.hpp	227
confusion_matrix/include/create_segmentation_results.hpp	228
confusion_matrix/include/csv_file_info.hpp	229
confusion_matrix/include/data_set_identifier.hpp	230
confusion_matrix/include/distance.hpp	232
confusion_matrix/include/distance_score.hpp	233
confusion_matrix/include/fetch.hpp	234
confusion_matrix/include imu.hpp	235
confusion_matrix/include/interpolated_data_set_paths.hpp	238
confusion_matrix/include/manual_segmentation_point.hpp	239

confusion_matrix/include/order_configurations_by_quality.hpp	240
confusion_matrix/include/python_output.hpp	241
confusion_matrix/include/segment.hpp	242
confusion_matrix/include/split_string.hpp	243
confusion_matrix/src/closest_one.cpp	243
confusion_matrix/src/configuration.cpp	244
confusion_matrix/src/confusion_matrix.cpp	246
confusion_matrix/src/confusion_matrix_best_configs.cpp	246
confusion_matrix/src/create_segmentation_results.cpp	247
confusion_matrix/src/csv_file_info.cpp	248
confusion_matrix/src/data_set_identifier.cpp	248
confusion_matrix/src/distance.cpp	249
confusion_matrix/src/distance_score.cpp	250
confusion_matrix/src imu.cpp	250
confusion_matrix/src/interpolated_data_set_paths.cpp	252
confusion_matrix/src/main.cpp	203
confusion_matrix/src/manual_segmentation_point.cpp	252
confusion_matrix/src/order_configurations_by_quality.cpp	253
confusion_matrix/src/python_output.cpp	254
confusion_matrix/src/segment.cpp	255
confusion_matrix/src/split_string.cpp	256
confusion_matrix/test/data_set_identifier_test.cpp	257
confusion_matrix/test/interpolated_data_set_paths_test.cpp	258
confusion_matrix/test/main.cpp	205
confusion_matrix/test/manual_segmentation_point_test.cpp	259
confusion_matrix/test/segment_test.cpp	263
confusion_matrix/test/split_string_test.cpp	264
counting/include/above_threshold.hpp	266
counting/include/average_comparison_value_calculator.hpp	267
counting/include/half_maximum_comparison_value_calculator.hpp	267
counting/include/is_relevant.hpp	268
counting/include/percentage_of.hpp	269
counting/include/run_above_threshold.hpp	270
counting/src/above_threshold.cpp	271
counting/src/average_comparison_value_calculator.cpp	273
counting/src/half_maximum_comparison_value_calculator.cpp	274
counting/src/main.cpp	197
counting/src/run_above_threshold.cpp	274
counting/test/above_threshold_test.cpp	275
counting/test/main.cpp	199
counting/test/percentage_of_test.cpp	277
csv_lib/include/cl/channel.hpp	278
csv_lib/include/cl/column.hpp	281
csv_lib/include/cl/data_point.hpp	283
csv_lib/include/cl/data_set.hpp	284
csv_lib/include/cl/dos2unix.hpp	285
csv_lib/include/cl/error.hpp	286
csv_lib/include/cl/exception.hpp	287
csv_lib/include/cl/process.hpp	295
csv_lib/include/cl/read_csv_file.hpp	295
csv_lib/include/cl/s2n.hpp	296
csv_lib/include/cl/sensor.hpp	297
csv_lib/include/cl/to_string.hpp	299
csv_lib/include/cl/use_unbuffered_io.hpp	300
csv_lib/include/cl/fs/directory_listing.hpp	289
csv_lib/include/cl/fs/file.hpp	290
csv_lib/include/cl/fs/file_stream.hpp	290
csv_lib/include/cl/fs/path.hpp	291

csv_lib/include/cl/fs/separator.hpp	292
csv_lib/include/cl/fs/windows.hpp	
Contains Microsoft Windows specific functions	293
csv_lib/src/cl/channel.cpp	300
csv_lib/src/cl/data_point.cpp	302
csv_lib/src/cl/data_set.cpp	305
csv_lib/src/cl/dos2unix.cpp	305
csv_lib/src/cl/error.cpp	306
csv_lib/src/cl/exception.cpp	307
csv_lib/src/cl/process.cpp	310
csv_lib/src/cl/read_csv_file.cpp	311
csv_lib/src/cl/sensor.cpp	312
csv_lib/src/cl/use_unbuffered_io.cpp	313
csv_lib/src/cl/fs/directory_listing.cpp	307
csv_lib/src/cl/fs/file.cpp	308
csv_lib/src/cl/fs/file_stream.cpp	308
csv_lib/src/cl/fs/path.cpp	309
csv_lib/src/cl/fs/windows.cpp	310
csv_lib/test/channel_test.cpp	313
csv_lib/test/column_test.cpp	315
csv_lib/test/data_point_test.cpp	317
csv_lib/test/data_set_test.cpp	319
csv_lib/test/directory_listing_test.cpp	323
csv_lib/test/error_test.cpp	325
csv_lib/test/exception_test.cpp	327
csv_lib/test/main.cpp	200
csv_lib/test/read_csv_file_test.cpp	328
csv_lib/test/s2n_test.cpp	329
csv_lib/test/sensor_test.cpp	331
csv_lib/test/to_string_test.cpp	332
fix_csv/include/adjust_hardware_timestamp.hpp	333
fix_csv/include/convert_to_unix_line_endings.hpp	334
fix_csv/include/create_backup_file.hpp	335
fix_csv/include/delete_non_bosch_sensors.hpp	336
fix_csv/include/delete_out_of_bounds_values.hpp	337
fix_csv/include/remove_zeros_from_field.hpp	338
fix_csv/include/restore_from_backup.hpp	339
fix_csv/include/write_file.hpp	340
fix_csv/src/adjust_hardware_timestamp.cpp	341
fix_csv/src/convert_to_unix_line_endings.cpp	342
fix_csv/src/create_backup_file.cpp	343
fix_csv/src/delete_non_bosch_sensors.cpp	343
fix_csv/src/delete_out_of_bounds_values.cpp	344
fix_csv/src/main.cpp	201
fix_csv/src/remove_zeros_from_field.cpp	345
fix_csv/src/restore_from_backup.cpp	345
fix_csv/src/write_file.cpp	346
fix_csv/test/adjust_hardware_timestamp_test.cpp	347
fix_csv/test/main.cpp	202
fix_csv/test/remove_zeros_from_field_test.cpp	349

Chapter 5

Namespace Documentation

5.1 cl Namespace Reference

Namespaces

- [fs](#)

Classes

- struct [col_traits](#)
- struct [data_set_accessor](#)
- class [DataPoint](#)
- class [DataSet](#)
- class [Error](#)
- class [Exception](#)
- class [Process](#)

Typedefs

- template<Column Col>
using [column_type](#) = typename [col_traits](#)< Col >::type
- template<typename Ty >
using [Expected](#) = tl::expected< Ty, [Error](#) >

Enumerations

- enum [Channel](#) : std::uint64_t { [Channel::CL_CHANNEL_X](#), [Channel::CL_CHANNEL_Y](#) }
- enum [Column](#) : std::size_t {
[Column::Time](#), [Column::HardwareTimestamp](#), [Column::ExtractId](#), [Column::Trigger](#),
[Column::AccelerometerX](#), [Column::AccelerometerY](#), [Column::AccelerometerZ](#), [Column::GyroscopeX](#),
[Column::GyroscopeY](#), [Column::GyroscopeZ](#), [Column::SamplingRate](#) }
- enum [CsvFileKind](#) { [CsvFileKind::Raw](#), [CsvFileKind::Fixed](#) }
- enum [Sensor](#) : std::uint64_t { [Sensor::CL_SENSOR_X](#), [Sensor::CL_SENSOR_Y](#) }

Functions

- `DataSet::ChannelAccessor dataSetAccessor (Channel channel)`
- `std::ostream & operator<< (std::ostream &os, Channel channel)`
- `bool isAccelerometer (Channel channel)`
- `bool isGyroscope (Channel channel)`
- `long double threshold (Channel channel)`
- `CL_SPECIALIZE_COL_TRAITS (Column::Time, long double)`
- `CL_SPECIALIZE_COL_TRAITS (Column::HardwareTimestamp, std::uint64_t)`
- `CL_SPECIALIZE_COL_TRAITS (Column::ExtractId, Sensor)`
- `CL_SPECIALIZE_COL_TRAITS (Column::Trigger, std::uint64_t)`
- `CL_SPECIALIZE_COL_TRAITS (Column::AccelerometerX, long double)`
- `CL_SPECIALIZE_COL_TRAITS (Column::AccelerometerY, long double)`
- `CL_SPECIALIZE_COL_TRAITS (Column::AccelerometerZ, long double)`
- `CL_SPECIALIZE_COL_TRAITS (Column::GyroscopeX, long double)`
- `CL_SPECIALIZE_COL_TRAITS (Column::GyroscopeY, long double)`
- `CL_SPECIALIZE_COL_TRAITS (Column::GyroscopeZ, long double)`
- `CL_SPECIALIZE_COL_TRAITS (Column::SamplingRate, std::uint64_t)`
- `std::vector< pl::byte > dos2unix (const void *p, std::size_t size)`

Converts DOS / Microsoft Windows line endings to UNIX line endings.
- `Expected< std::vector< std::vector< std::string > > > readCsvFile (pl::string_view csvFilePath, std::vector< std::string > *columnNames=nullptr, CsvFileKind csvFileKind=CsvFileKind::Fixed) noexcept`
- `template<typename Integer> Expected< Integer > s2n (const std::string &str, std::size_t *pos=nullptr, [[maybe_unused]] int base=10)`
- `std::ostream & operator<< (std::ostream &os, Sensor sensor)`
- `template<typename Ty> std::string to_string (const Ty &ty)`
- `void useUnbufferedIo ()`
- `std::ostream & operator<< (std::ostream &os, const DataPoint &dataPoint)`
- `std::ostream & operator<< (std::ostream &os, const Error &error)`

Variables

- `constexpr std::size_t channelCount`
- `constexpr std::array< Channel, channelCount > channels`
- `template<Channel Chan> constexpr CL_CHANNEL DataSet::ChannelAccessor data_set_accessor_v = data_set_accessor<Chan>::f`
- `constexpr long double accelerometerThreshold {1.99L}`
- `constexpr long double gyroscopeThreshold {1999.99L}`
- `template<Column Col> constexpr std::size_t column_index = col_traits<Col>::index`
- `constexpr std::array< Sensor, 4 > sensors`

5.1.1 Typedef Documentation

5.1.1.1 column_type

```
template<Column Col>
using cl::column_type = typedef typename col_traits<Col>::type
```

Definition at line 49 of file column.hpp.

5.1.1.2 Expected

```
template<typename Ty >
using cl::Expected = typedef tl::expected<Ty, Error>
```

Definition at line 64 of file error.hpp.

5.1.2 Enumeration Type Documentation

5.1.2.1 Channel

```
enum cl::Channel : std::uint64_t [strong]
```

Enumerator

CL_CHANNEL←_X	
CL_CHANNEL	

Definition at line 20 of file channel.hpp.

5.1.2.2 Column

```
enum cl::Column : std::size_t [strong]
```

Enumerator

Time	
HardwareTimestamp	
ExtractId	
Trigger	
AccelerometerX	
AccelerometerY	
AccelerometerZ	
GyroscopeX	
GyroscopeY	
GyroscopeZ	
SamplingRate	

Definition at line 9 of file column.hpp.

5.1.2.3 CsvFileKind

```
enum cl::CsvFileKind [strong]
```

Enumerator

Raw	
Fixed	

Definition at line 11 of file read_csv_file.hpp.

5.1.2.4 Sensor

```
enum cl::Sensor : std::uint64_t [strong]
```

Enumerator

CL_SENSOR_X	
CL_SENSOR	

Definition at line 15 of file sensor.hpp.

5.1.3 Function Documentation

5.1.3.1 CL_SPECIALIZE_COL_TRAITS() [1/11]

```
cl::CL_SPECIALIZE_COL_TRAITS (
    Column::AccelerometerX ,
    long double )
```

5.1.3.2 CL_SPECIALIZE_COL_TRAITS() [2/11]

```
cl::CL_SPECIALIZE_COL_TRAITS (
    Column::AccelerometerY ,
    long double )
```

5.1.3.3 CL_SPECIALIZE_COL_TRAITS() [3/11]

```
cl::CL_SPECIALIZE_COL_TRAITS (
    Column::AccelerometerZ ,
    long double   )
```

5.1.3.4 CL_SPECIALIZE_COL_TRAITS() [4/11]

```
cl::CL_SPECIALIZE_COL_TRAITS (
    Column::ExtractId ,
    Sensor   )
```

5.1.3.5 CL_SPECIALIZE_COL_TRAITS() [5/11]

```
cl::CL_SPECIALIZE_COL_TRAITS (
    Column::GyroscopeX ,
    long double   )
```

5.1.3.6 CL_SPECIALIZE_COL_TRAITS() [6/11]

```
cl::CL_SPECIALIZE_COL_TRAITS (
    Column::GyroscopeY ,
    long double   )
```

5.1.3.7 CL_SPECIALIZE_COL_TRAITS() [7/11]

```
cl::CL_SPECIALIZE_COL_TRAITS (
    Column::GyroscopeZ ,
    long double   )
```

5.1.3.8 CL_SPECIALIZE_COL_TRAITS() [8/11]

```
cl::CL_SPECIALIZE_COL_TRAITS (
    Column::HardwareTimestamp ,
    std::uint64_t   )
```

5.1.3.9 CL_SPECIALIZE_COL_TRAITS() [9/11]

```
cl::CL_SPECIALIZE_COL_TRAITS (
    Column::SamplingRate ,
    std::uint64_t )
```

5.1.3.10 CL_SPECIALIZE_COL_TRAITS() [10/11]

```
cl::CL_SPECIALIZE_COL_TRAITS (
    Column::Time ,
    long double )
```

5.1.3.11 CL_SPECIALIZE_COL_TRAITS() [11/11]

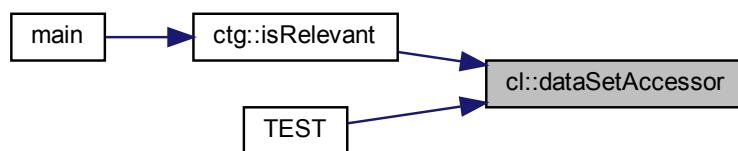
```
cl::CL_SPECIALIZE_COL_TRAITS (
    Column::Trigger ,
    std::uint64_t )
```

5.1.3.12 dataSetAccessor()

```
DataSet::ChannelAccessor cl::dataSetAccessor (
    Channel channel )
```

Definition at line 15 of file channel.cpp.

Here is the caller graph for this function:



5.1.3.13 dos2unix()

```
std::vector< pl::byte > cl::dos2unix (
    const void * p,
    std::size_t size )
```

Converts DOS / Microsoft Windows line endings to UNIX line endings.

Parameters

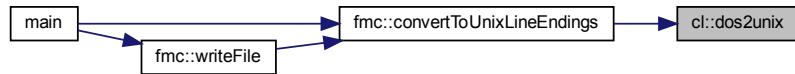
<i>p</i>	The beginning of the data to convert.
<i>size</i>	The size of the data to convert in bytes.

Returns

The resulting byte array.

Definition at line 4 of file dos2unix.cpp.

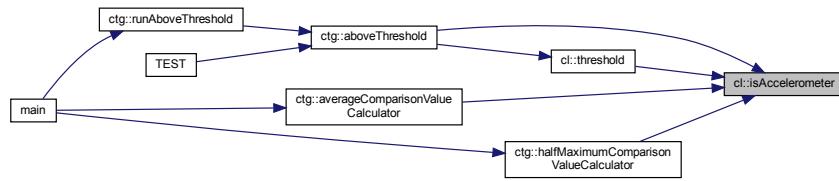
Here is the caller graph for this function:

**5.1.3.14 isAccelerometer()**

```
bool cl::isAccelerometer (
    Channel channel )
```

Definition at line 45 of file channel.cpp.

Here is the caller graph for this function:

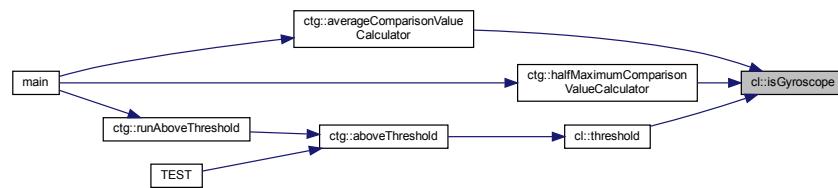


5.1.3.15 `isGyroscope()`

```
bool cl::isGyroscope (
    Channel channel )
```

Definition at line 50 of file channel.cpp.

Here is the caller graph for this function:



5.1.3.16 `operator<<()` [1/4]

```
std::ostream & cl::operator<< (
    std::ostream & os,
    Channel channel )
```

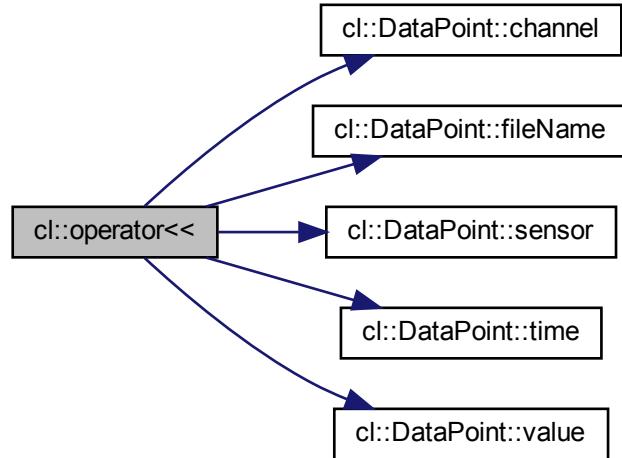
Definition at line 32 of file channel.cpp.

5.1.3.17 `operator<<()` [2/4]

```
std::ostream& cl::operator<< (
    std::ostream & os,
    const DataPoint & dataPoint )
```

Definition at line 10 of file data_point.cpp.

Here is the call graph for this function:

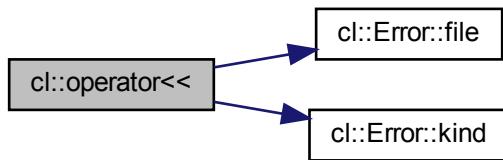


5.1.3.18 `operator<<()` [3/4]

```
std::ostream& cl::operator<< (
    std::ostream & os,
    const Error & error )
```

Definition at line 30 of file error.cpp.

Here is the call graph for this function:

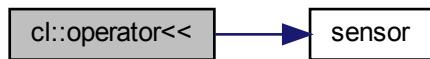


5.1.3.19 operator<<() [4/4]

```
std::ostream & cl::operator<< (
    std::ostream & os,
    Sensor sensor )
```

Definition at line 8 of file sensor.cpp.

Here is the call graph for this function:

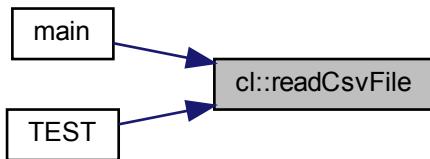


5.1.3.20 readCsvFile()

```
Expected< std::vector< std::vector< std::string > > > cl::readCsvFile (
    pl::string_view csvFilePath,
    std::vector< std::string > * columnNames = nullptr,
    CsvFileKind csvFileKind = CsvFileKind::Fixed ) [noexcept]
```

Definition at line 50 of file read_csv_file.cpp.

Here is the caller graph for this function:



5.1.3.21 s2n()

```
template<typename Integer >
Expected<Integer> cl::s2n (
    const std::string & str,
    std::size_t * pos = nullptr,
    [[maybe_unused]] int base = 10 ) [inline]
```

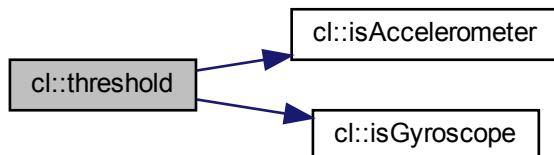
Definition at line 16 of file s2n.hpp.

5.1.3.22 threshold()

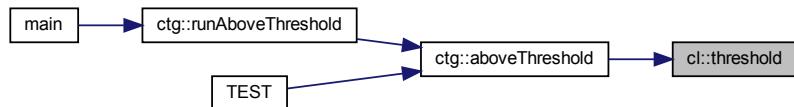
```
long double cl::threshold (
    Channel channel )
```

Definition at line 55 of file channel.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

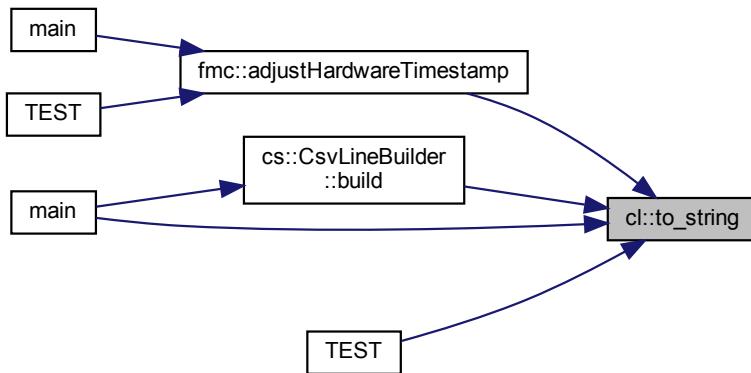


5.1.3.23 `to_string()`

```
template<typename Ty >
std::string cl::to_string (
    const Ty & ty ) [inline]
```

Definition at line 16 of file `to_string.hpp`.

Here is the caller graph for this function:



5.1.3.24 `useUnbufferedIo()`

```
void cl::useUnbufferedIo ( )
```

Definition at line 9 of file `use_unbuffered_io.cpp`.

Here is the caller graph for this function:



5.1.4 Variable Documentation

5.1.4.1 accelerometerThreshold

```
constexpr long double cl::accelerometerThreshold {1.99L} [inline], [constexpr]
```

Definition at line 61 of file channel.hpp.

5.1.4.2 channelCount

```
constexpr std::size_t cl::channelCount [inline], [constexpr]
```

Initial value:

```
{0  
#define CL_CHANNEL_X(enumerator, value, dataSetAccessor)  
    CL_CHANNEL  
}
```

Definition at line 26 of file channel.hpp.

5.1.4.3 channels

```
constexpr std::array<Channel, channelCount> cl::channels [inline], [constexpr]
```

Initial value:

```
{  
#define CL_CHANNEL_X(enm, v, a)  
    CL_CHANNEL  
} }
```

Definition at line 32 of file channel.hpp.

5.1.4.4 column_index

```
template<Column Col>  
constexpr std::size_t cl::column_index = col_traits<Col>::index [inline], [constexpr]
```

Definition at line 46 of file column.hpp.

5.1.4.5 data_set_accessor_v

```
template<Channel Chan>  
constexpr CL_CHANNEL DataSet::ChannelAccessor cl::data_set_accessor_v = data_set_accessor<Chan>↔  
::f [inline], [constexpr]
```

Definition at line 51 of file channel.hpp.

5.1.4.6 gyroscopeThreshold

```
constexpr long double cl::gyroscopeThreshold {1999.99L} [inline], [constexpr]
```

Definition at line 62 of file channel.hpp.

5.1.4.7 sensors

```
constexpr std::array<Sensor, 4> cl::sensors [inline], [constexpr]
```

Initial value:

```
{}  
#define CL_SENSOR_X(enm, v)  
    CL_SENSOR  
){}
```

Definition at line 21 of file sensor.hpp.

5.2 cl::fs Namespace Reference

Classes

- class [File](#)
Represents a file.
- class [FileStream](#)
A binary file stream.
- class [Path](#)
A filesystem path.

Enumerations

- enum [DirectoryListingOption](#) { [DirectoryListingOption::None](#), [DirectoryListingOption::ExcludeDotAndDotDot](#) }
Options for directoryListing.

Functions

- [Expected< std::vector< Path > > directoryListing](#) (const [Path](#) &directoryPath, [DirectoryListingOption](#) directoryListingOption=[DirectoryListingOption::ExcludeDotAndDotDot](#))
Creates a listing of the contents of a directory.
- [std::wstring utf8ToUtf16](#) ([pl::string_view](#) utf8)
Converts a UTF-8 encoded string to a UTF-16 encoded wstring.
- [std::string utf16ToUtf8](#) ([pl::wstring_view](#) utf16)
Converts a UTF-16 encoded wide character string to UTF-8 string.
- [std::wstring formatError](#) (DWORD errorCode)
Formats a WINAPI error code to a UTF-16 encoded wide character string.
- [std::ostream & operator<<](#) ([std::ostream](#) &os, const [Path](#) &path)
- [bool operator<](#) (const [Path](#) &lhs, const [Path](#) &rhs) noexcept
- [bool operator==](#) (const [Path](#) &lhs, const [Path](#) &rhs) noexcept

5.2.1 Enumeration Type Documentation

5.2.1.1 DirectoryListingOption

```
enum cl::fs::DirectoryListingOption [strong]
```

Options for directoryListing.

Enumerator

None	No option
ExcludeDotAndDotDot	Exclude the . and .. directories

Definition at line 13 of file directory_listing.hpp.

5.2.2 Function Documentation

5.2.2.1 directoryListing()

```
Expected< std::vector< Path > > cl::fs::directoryListing (
    const Path & directoryPath,
    DirectoryListingOption directoryListingOption = DirectoryListingOption::ExcludeDotAndDotDot
)
```

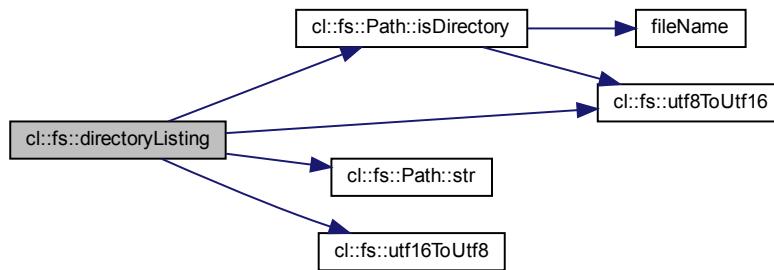
Creates a listing of the contents of a directory.

Parameters

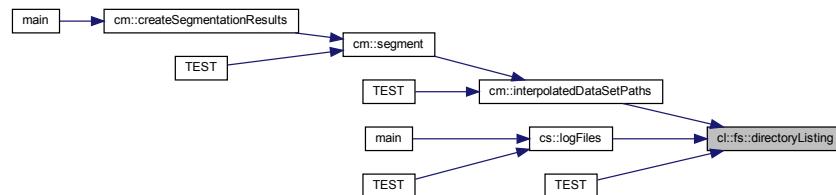
<i>directoryPath</i>	The directory to list.
<i>directoryListingOption</i>	The option to use.

Definition at line 24 of file directory_listing.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.2.2.2 `formatError()`

```
std::wstring cl::fs::formatError (
    DWORD errorCode )
```

Formats a WINAPI error code to a UTF-16 encoded wide character string.

Parameters

<code>errorCode</code>	The WINAPI error code.
------------------------	------------------------

Returns

The resulting UTF-16 encoded wide character string.

Note

Most WINAPIs expect UTF-16 encoded wide character strings, but we don't want to pollute the code base with UTF-16 strings.

Warning

Wide characters are only 16 bit wide on Microsoft Windows, they're 32 bit on GNU / Linux.

Definition at line 89 of file windows.cpp.

5.2.2.3 operator<()

```
bool cl::fs::operator< (
    const Path & lhs,
    const Path & rhs ) [noexcept]
```

Parameters

<i>lhs</i>	The left hand side operand.
<i>rhs</i>	The right hand side operand.

Returns

true if lhs < rhs; otherwise false.

Definition at line 27 of file path.cpp.

5.2.2.4 operator<<()

```
std::ostream& cl::fs::operator<< (
    std::ostream & os,
    const Path & path )
```

Parameters

<i>os</i>	the ostream to print to.
<i>path</i>	The path to print.

Returns

os

Definition at line 22 of file path.cpp.

5.2.2.5 operator==()

```
bool cl::fs::operator== (
    const Path & lhs,
    const Path & rhs ) [noexcept]
```

Parameters

<i>lhs</i>	The left hand side operand.
<i>rhs</i>	The right hand side operand.

Returns

true if *lhs* and *rhs* are equal.

Definition at line 32 of file path.cpp.

5.2.2.6 utf16ToUtf8()

```
std::string cl::fs::utf16ToUtf8 (
    pl::wstring_view utf16 )
```

Converts a UTF-16 encoded wide character string to UTF-8 string.

Parameters

<i>utf16</i>	The UTF-16 encoded wide character string to convert.
--------------	--

Returns

The resulting UTF-8 string.

Note

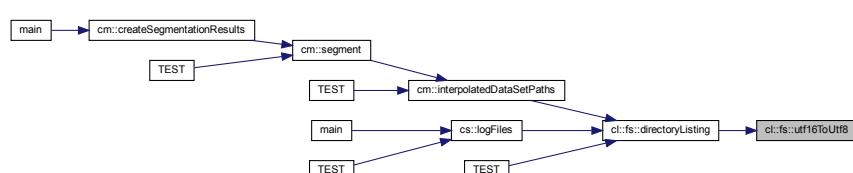
Most WINAPIs expect UTF-16 encoded wide character strings, but we don't want to pollute the code base with UTF-16 strings.

Warning

Wide characters are only 16 bit wide on Microsoft Windows, they're 32 bit on GNU / Linux.

Definition at line 61 of file windows.cpp.

Here is the caller graph for this function:



5.2.2.7 utf8ToUtf16()

```
std::wstring cl::fs::utf8ToUtf16 (
    pl::string_view utf8 )
```

Converts a UTF-8 encoded string to a UTF-16 encoded wstring.

Parameters

<i>utf8</i>	The UTF-8 encoded string to convert.
-------------	--------------------------------------

Returns

The resulting UTF-16 string.

Note

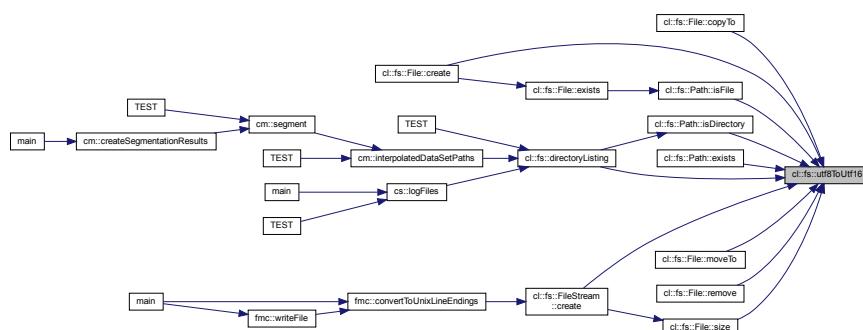
Most WINAPIs expect UTF-16 encoded wide character strings, but we don't want to pollute the code base with UTF-16 strings.

Warning

Wide characters are only 16 bit wide on Microsoft Windows, they're 32 bit on GNU / Linux.

Definition at line 35 of file windows.cpp.

Here is the caller graph for this function:



5.3 cm Namespace Reference

Classes

- class [Configuration](#)

Represents a possible configuration for the Python segmentor.
- struct [ConfigWithDistanceScore](#)
- struct [ConfigWithTotalConfusionMatrix](#)
- class [ConfusionMatrix](#)
- class [CsvFileInfo](#)

Type to hold the hardware timestamps of a CSV file.
- class [ManualSegmentationPoint](#)

Type used to represent a manual segmentation point.

Enumerations

- enum `DataSetIdentifier` { `DataSetIdentifier::CM_DATA_SET_IDENTIFIER_X`, `DataSetIdentifier::CM_DATA_SET_IDENTIFIER_Y` }
- enum `Imu` { `Imu::CM_IMU_X`, `Imu::CM_IMU` }

Scoped enum type for the IMUs.

Functions

- std::uint64_t `closestOne` (std::uint64_t algorithmicallyDeterminedSegmentationPoint, const std::vector< std::uint64_t > &manualSegmentationPoints)

Finds the segmentation point in manualSegmentationPoints that is the closest to algorithmicallyDeterminedSegmentationPoint.
- `CM_SORTER` (truePositives, >)
- `CM_SORTER` (trueNegatives, >)
- `CM_SORTER` (falsePositives, <)
- `CM_SORTER` (falseNegatives, <)
- bool `operator<` (const `ConfigWithTotalConfusionMatrix` &lhs, const `ConfigWithTotalConfusionMatrix` &rhs) noexcept
- std::ostream & `operator<<` (std::ostream &os, const `ConfigWithTotalConfusionMatrix` &obj)
- std::vector< `ConfigWithTotalConfusionMatrix` > `confusionMatrixBestConfigs` (const std::unordered_map< `DataSetIdentifier`, std::vector< std::uint64_t > > &manualSegmentationPoints, const std::unordered_map< `Configuration`, std::unordered_map< `cl::fs::Path`, std::vector< std::uint64_t > >> &algorithmicallyDeterminedSegmentationPoints)
- std::unordered_map< `cm::Configuration`, std::unordered_map< `cl::fs::Path`, std::vector< std::uint64_t > > > `createSegmentationResults` ()

Invokes Python to generate the segmentation points algorithmically.
- std::ostream & `operator<<` (std::ostream &os, `DataSetIdentifier` dsi)

Prints a DataSetIdentifier to an ostream.
- `DataSetIdentifier toDataSetIdentifier` (const `cl::fs::Path` &path)

Converts a path to a CSV file to the corresponding DataSetIdentifier.
- std::uint64_t `distance` (std::uint64_t a, std::uint64_t b)

Calculates the distance between a and b.
- std::uint64_t `distanceScore` (const std::unordered_map< `cl::fs::Path`, std::vector< std::uint64_t > > &segmentationPointsForConfig, const std::unordered_map< `DataSetIdentifier`, std::vector< std::uint64_t > >> &manualSegmentationPoints)
- template<typename Map , typename Key >
 auto `fetch` (const Map &map, const Key &key)

Fetches a value from a map for a given key.
- std::ostream & `operator<<` (std::ostream &os, `Imu` imu)

Prints imu to os.
- std::vector< `cl::fs::Path` > `interpolatedDataSetPaths` ()

Returns the paths to the interpolated data sets.
- bool `operator<` (const `ConfigWithDistanceScore` &lhs, const `ConfigWithDistanceScore` &rhs) noexcept
- std::ostream & `operator<<` (std::ostream &os, const `ConfigWithDistanceScore` &configWithDistScore)
- std::vector< `ConfigWithDistanceScore` > `orderConfigurationsByQuality` (const std::unordered_map< `DataSetIdentifier`, std::vector< std::uint64_t > > &manualSegmentationPoints, const std::unordered_map< `Configuration`, std::unordered_map< `cl::fs::Path`, std::vector< std::uint64_t > >> &algorithmicallyDeterminedSegmentationPoints)
- std::string `pythonOutput` (const `cl::fs::Path` &csvFilePath, const `Configuration` &segmentorConfiguration)

Runs the Python segmentor on path.
- std::unordered_map< `cl::fs::Path`, std::vector< std::uint64_t > > > `segment` (const `Configuration` &segmentorConfiguration)

- std::vector< std::string > [splitString](#) (std::string string, pl::string_view splitBy)
Splits string by splitBy.
- bool [operator==](#) (const Configuration &lhs, const Configuration &rhs) noexcept
- bool [operator!=](#) (const Configuration &lhs, const Configuration &rhs) noexcept
- std::ostream & [operator<<](#) (std::ostream &os, const Configuration &config)
- bool [operator==](#) (const ManualSegmentationPoint &lhs, const ManualSegmentationPoint &rhs) noexcept
- bool [operator!=](#) (const ManualSegmentationPoint &lhs, const ManualSegmentationPoint &rhs) noexcept
- std::ostream & [operator<<](#) (std::ostream &os, const ManualSegmentationPoint &manualSegmentationPoint)

Variables

- constexpr std::size_t [imuCount](#)
The amount of IMUs.
- constexpr std::array< [Imu](#), [imuCount](#) > [imus](#)
An array of the IMU enumerators.

5.3.1 Enumeration Type Documentation

5.3.1.1 [DataSetIdentifier](#)

```
enum cm::DataSetIdentifier [strong]
```

Enumerator

CM_DATA_SET_IDENTIFIER	← _X	
CM_DATA_SET_IDENTIFIER		

Definition at line 30 of file `data_set_identifier.hpp`.

5.3.1.2 [Imu](#)

```
enum cm::Imu [strong]
```

Scoped enum type for the IMUs.

Enumerator

CM_IMU	← _X	
CM_IMU		

Definition at line 17 of file `imu.hpp`.

5.3.2 Function Documentation

5.3.2.1 closestOne()

```
std::uint64_t cm::closestOne (
    std::uint64_t algorithmicallyDeterminedSegmentationPoint,
    const std::vector< std::uint64_t > & manualSegmentationPoints )
```

Finds the segmentation point in `manualSegmentationPoints` that is the closest to `algorithmicallyDeterminedSegmentationPoint`.

Parameters

<code>algorithmicallyDeterminedSegmentationPoint</code>	The segmentation point to find the closest one to.
<code>manualSegmentationPoints</code>	The manual segmentation points.

Returns

The manual segmentation point that is the closest to `algorithmicallyDeterminedSegmentationPoint`.

Exceptions

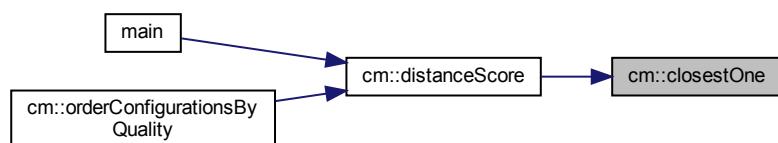
<code>cl::Exception</code>	if no segmentation point was found.
----------------------------	-------------------------------------

Definition at line 11 of file `closest_one.cpp`.

Here is the call graph for this function:



Here is the caller graph for this function:



5.3.2.2 CM_SORTER() [1/4]

```
cm::CM_SORTER (
    falseNegatives )
```

5.3.2.3 CM_SORTER() [2/4]

```
cm::CM_SORTER (
    falsePositives )
```

5.3.2.4 CM_SORTER() [3/4]

```
cm::CM_SORTER (
    trueNegatives )
```

5.3.2.5 CM_SORTER() [4/4]

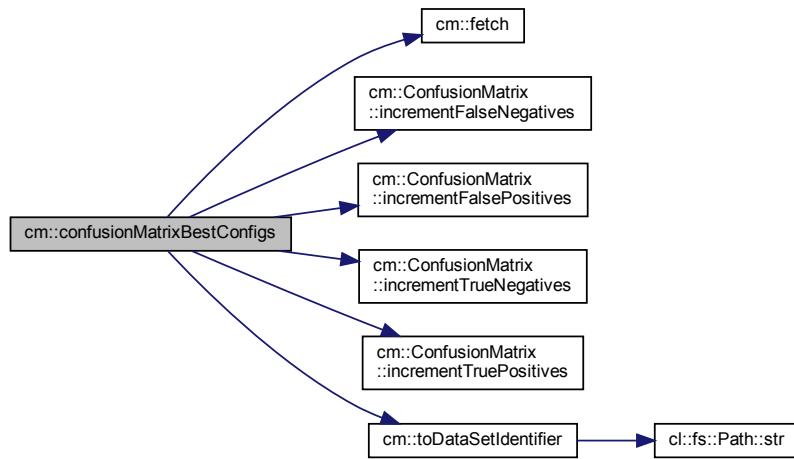
```
cm::CM_SORTER (
    truePositives )
```

5.3.2.6 confusionMatrixBestConfigs()

```
std::vector< ConfigWithTotalConfusionMatrix > cm::confusionMatrixBestConfigs (
    const std::unordered_map< DataSetIdentifier, std::vector< std::uint64_t >> &
manualSegmentationPoints,
    const std::unordered_map< Configuration, std::unordered_map< cl::fs::Path, std::vector< std::uint64_t >>> &
algorithmicallyDeterminedSegmentationPoints )
```

Definition at line 86 of file confusion_matrix_best_configs.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.3.2.7 createSegmentationResults()

```
std::unordered_map< Configuration, std::unordered_map< cl::fs::Path, std::vector< std::uint64_t > > > cm::createSegmentationResults( )
```

Invokes Python to generate the segmentation points algorithmically.

Returns

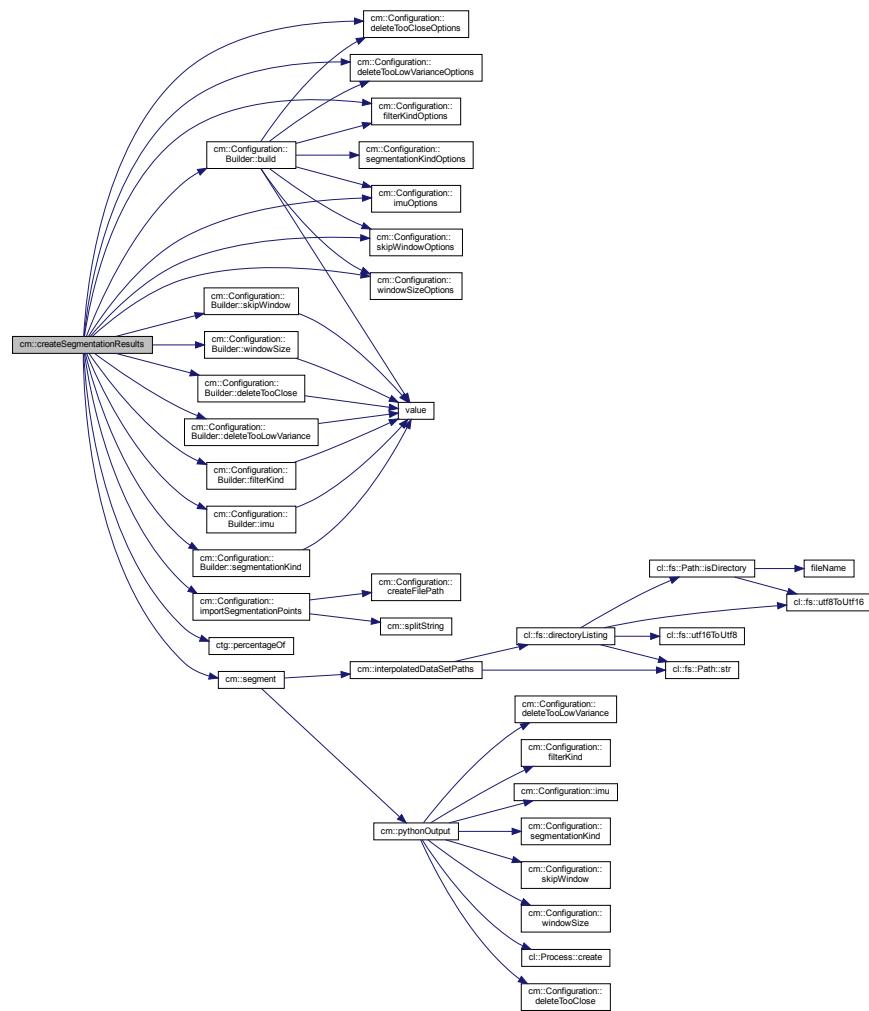
A map that maps the configurations to maps that map CSV file paths to segmentation points.

Exceptions

<i>cl::Exception</i>	on error.
----------------------	-----------

Definition at line 42 of file `create_segmentation_results.cpp`.

Here is the call graph for this function:



Here is the caller graph for this function:



5.3.2.8 distance()

```
std::uint64_t cm::distance (
    std::uint64_t a,
    std::uint64_t b )
```

Calculates the distance between a and b.

Parameters

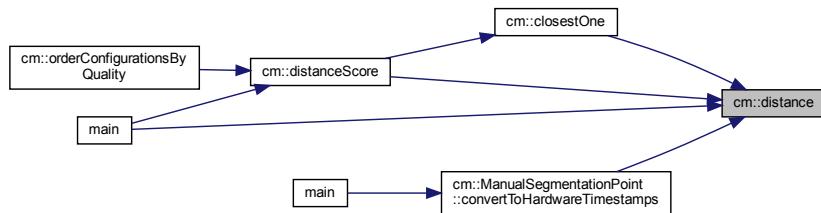
<i>a</i>	The first parameter.
<i>b</i>	The second parameter.

Returns

The difference between a and b.

Definition at line 6 of file `distance.cpp`.

Here is the caller graph for this function:

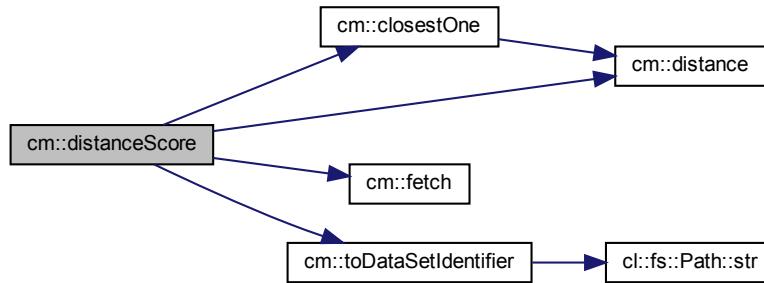


5.3.2.9 distanceScore()

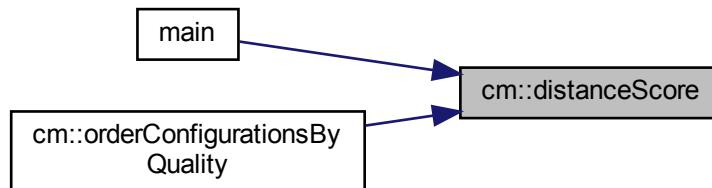
```
std::uint64_t cm::distanceScore (
    const std::unordered_map< cl::fs::Path, std::vector< std::uint64_t >> & segmentationPointsForConfig,
    const std::unordered_map< DataSetIdentifier, std::vector< std::uint64_t >> & manualSegmentationPoints )
```

Definition at line 7 of file `distance_score.cpp`.

Here is the call graph for this function:



Here is the caller graph for this function:



5.3.2.10 `fetch()`

```

template<typename Map , typename Key >
auto cm::fetch (
    const Map & map,
    const Key & key )
  
```

Fetches a value from a map for a given key.

Template Parameters

<code>Map</code>	The type of the map.
<code>Key</code>	The type of the Key.

Parameters

<code>map</code>	The map to fetch from.
------------------	------------------------

Parameters

<code>key</code>	The key to find the value for in map.
------------------	---------------------------------------

Returns

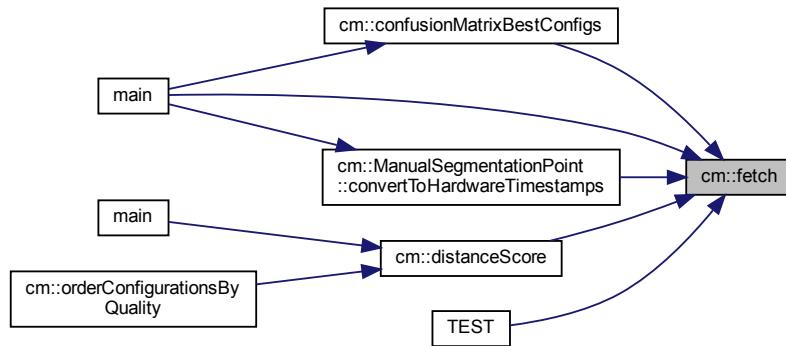
The value for `key` in map.

Exceptions

<code>cl::Exception</code>	if key is not found in map.
----------------------------	-----------------------------

Definition at line 16 of file fetch.hpp.

Here is the caller graph for this function:

**5.3.2.11 interpolatedDataSetPaths()**

```
std::vector< cl::fs::Path > cm::interpolatedDataSetPaths( )
```

Returns the paths to the interpolated data sets.

Returns

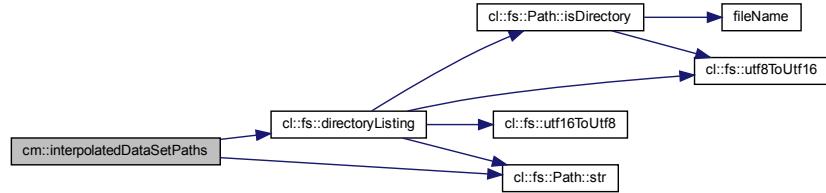
The interpolated data set paths.

Exceptions

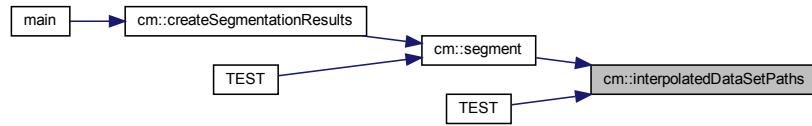
<code>cl::Exception</code>	on error.
----------------------------	-----------

Definition at line 61 of file interpolated_data_set_paths.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.3.2.12 operator"!=() [1/2]

```
bool cm::operator!= (
    const Configuration & lhs,
    const Configuration & rhs ) [noexcept]
```

Parameters

<i>lhs</i>	The first operand.
<i>rhs</i>	The second operand.

Returns

true if *lhs* and *rhs* are considered not to be equal.

Definition at line 230 of file configuration.cpp.

5.3.2.13 operator"!=() [2/2]

```
bool cm::operator!= (
    const ManualSegmentationPoint & lhs,
    const ManualSegmentationPoint & rhs ) [noexcept]
```

Parameters

<i>lhs</i>	The first operand.
<i>rhs</i>	The second operand.

Returns

true if *lhs* is considered not equal to *rhs*; false otherwise.

Definition at line 187 of file manual_segmentation_point.cpp.

5.3.2.14 operator<() [1/2]

```
bool cm::operator< (
    const ConfigWithDistanceScore & lhs,
    const ConfigWithDistanceScore & rhs ) [noexcept]
```

Definition at line 20 of file order_configurations_by_quality.cpp.

5.3.2.15 operator<() [2/2]

```
bool cm::operator< (
    const ConfigWithTotalConfusionMatrix & lhs,
    const ConfigWithTotalConfusionMatrix & rhs ) [noexcept]
```

Definition at line 48 of file confusion_matrix_best_configs.cpp.

5.3.2.16 operator<<() [1/6]

```
std::ostream& cm::operator<< (
    std::ostream & os,
    const Configuration & config )
```

Parameters

<i>os</i>	The ostream to print to.
<i>config</i>	The Configuration to print.

Returns

os

Definition at line 235 of file configuration.cpp.

5.3.2.17 operator<<() [2/6]

```
std::ostream & cm::operator<< (
    std::ostream & os,
    const ConfigWithDistanceScore & configWithDistScore )
```

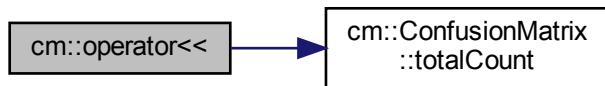
Definition at line 27 of file order_configurations_by_quality.cpp.

5.3.2.18 operator<<() [3/6]

```
std::ostream & cm::operator<< (
    std::ostream & os,
    const ConfigWithTotalConfusionMatrix & obj )
```

Definition at line 62 of file confusion_matrix_best_configs.cpp.

Here is the call graph for this function:



5.3.2.19 operator<<() [4/6]

```
std::ostream& cm::operator<< (
    std::ostream & os,
    const ManualSegmentationPoint & manualSegmentationPoint )
```

Parameters

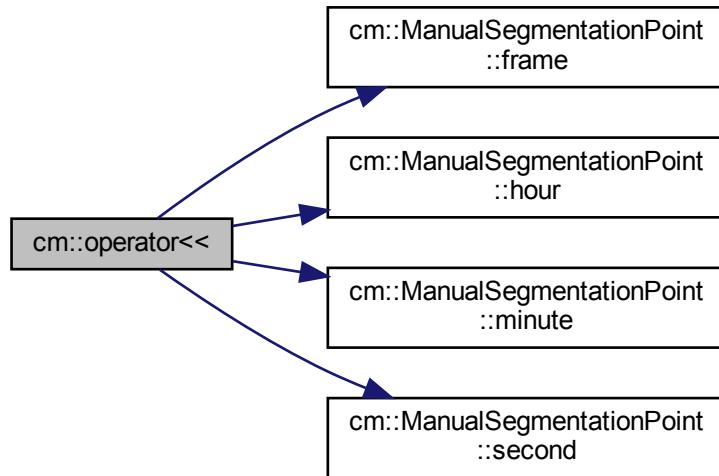
<i>os</i>	The ostream to print to
<i>manualSegmentationPoint</i>	The <code>ManualSegmentationPoint</code> to print.

Returns

os

Definition at line 194 of file manual_segmentation_point.cpp.

Here is the call graph for this function:



5.3.2.20 operator<<() [5/6]

```
std::ostream & cm::operator<< (
    std::ostream & os,
    DataSetIdentifier dsi )
```

Prints a DataSetIdentifier to an ostream.

Parameters

<i>os</i>	The ostream to print to.
<i>dsi</i>	The DataSetIdentifier to print.

Returns

os

Definition at line 33 of file data_set_identifier.cpp.

5.3.2.21 operator<<() [6/6]

```
std::ostream & cm::operator<< (
    std::ostream & os,
    Imu imu )
```

Prints `imu` to `os`.

Parameters

<code>os</code>	The ostream to print to
<code>imu</code>	The IMU enumerator to print.

Returns

`os`

Definition at line 35 of file `imu.cpp`.

5.3.2.22 operator==(()) [1/2]

```
bool cm::operator== (
    const Configuration & lhs,
    const Configuration & rhs ) [noexcept]
```

Parameters

<code>lhs</code>	The first operand.
<code>rhs</code>	The second operand.

Returns

true if `lhs` and `rhs` are considered to be equal.

Definition at line 210 of file `configuration.cpp`.

5.3.2.23 operator==(()) [2/2]

```
bool cm::operator== (
    const ManualSegmentationPoint & lhs,
    const ManualSegmentationPoint & rhs ) [noexcept]
```

Parameters

<code>lhs</code>	The first operand.
<code>rhs</code>	The second operand.

Returns

true if lhs is considered equal to rhs; false otherwise.

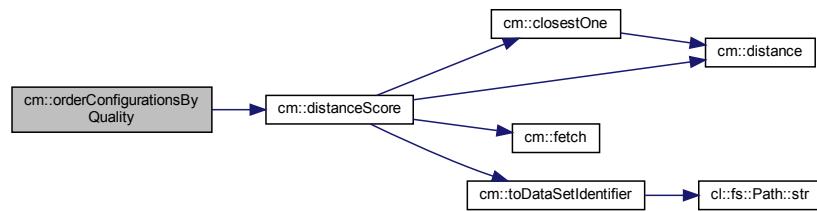
Definition at line 179 of file manual_segmentation_point.cpp.

5.3.2.24 orderConfigurationsByQuality()

```
std::vector< ConfigWithDistanceScore > cm::orderConfigurationsByQuality (
    const std::unordered_map< DataSetIdentifier, std::vector< uint64_t >> &
manualSegmentationPoints,
    const std::unordered_map< Configuration, std::unordered_map< cl::fs::Path, std::vector< uint64_t >>> & algorithmicallyDeterminedSegmentationPoints )
```

Definition at line 38 of file order_configurations_by_quality.cpp.

Here is the call graph for this function:

**5.3.2.25 pythonOutput()**

```
std::string cm::pythonOutput (
    const cl::fs::Path & csvFilePath,
    const Configuration & segmentorConfiguration )
```

Runs the Python segmentor on path.

Parameters

<code>path</code>	The path to the CSV file to segment.
<code>segmentorConfiguration</code>	The configuration to use.

Returns

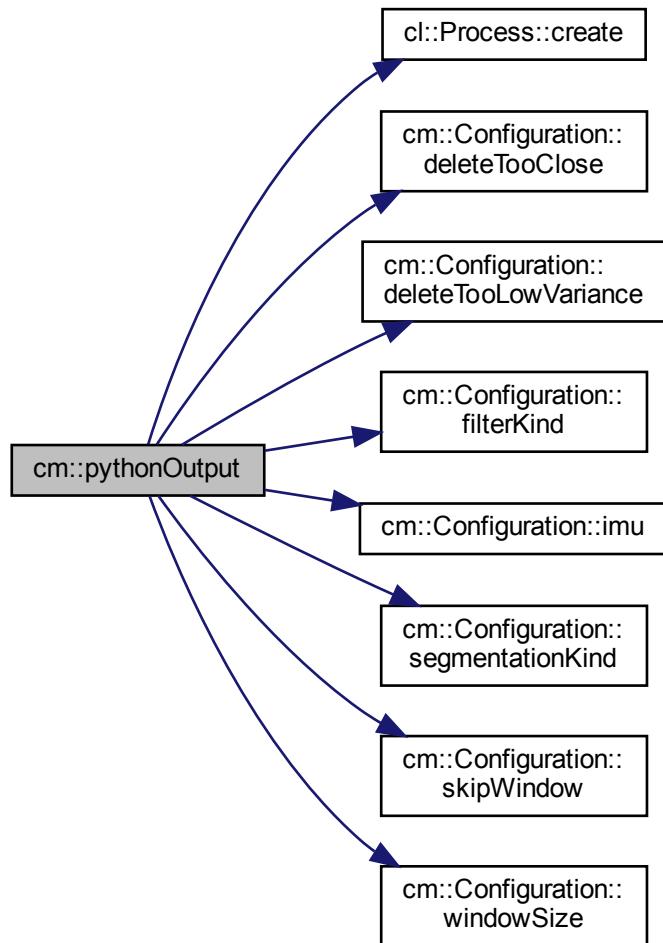
The output of the Python application.

Exceptions

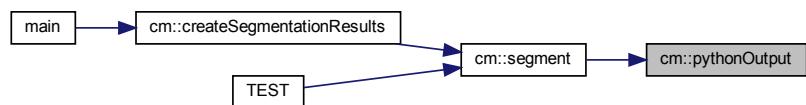
<i>cl::Exception</i>	if creating the process failed.
----------------------	---------------------------------

Definition at line 32 of file python_output.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.3.2.26 segment()

```
std::unordered_map< cl::fs::Path, std::vector< std::uint64_t > > cm::segment (
    const Configuration & segmentorConfiguration )
```

Invokes Python to segment the interpolated data sets.

Parameters

<i>segmentorConfiguration</i>	The Configuration to use for the Python segmentor.
-------------------------------	--

Returns

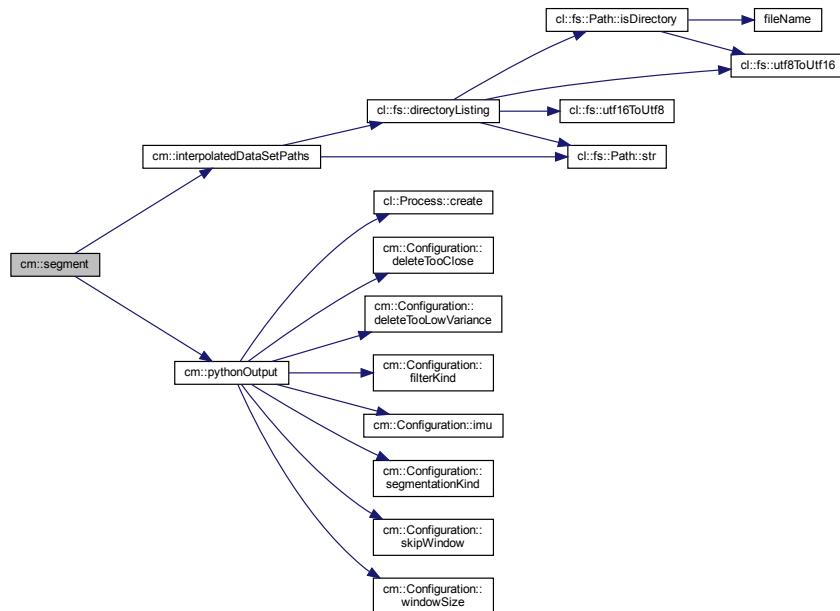
A map that maps the paths to the interpolated data sets to vectors of the hardware timestamps (in milliseconds) that are segmentation points.

Exceptions

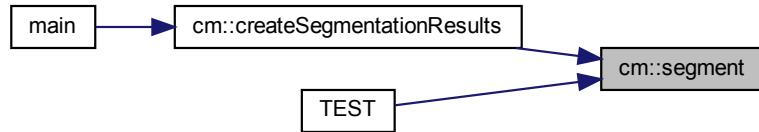
<i>cl::Exception</i>	if an error occurs.
----------------------	---------------------

Definition at line 64 of file segment.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.3.2.27 splitString()

```
std::vector< std::string > cm::splitString (
    std::string string,
    pl::string_view splitBy )
```

Splits `string` by `splitBy`.

Parameters

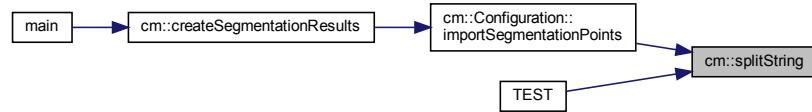
<code>string</code>	The string to split.
<code>splitBy</code>	What to split <code>string</code> by.

Returns

The resulting strings.

Definition at line 8 of file `split_string.cpp`.

Here is the caller graph for this function:



5.3.2.28 toDataSetIdentifier()

```
DataSetIdentifier cm::toDataSetIdentifier (
    const cl::fs::Path & path )
```

Converts a path to a CSV file to the corresponding `DataSetIdentifier`.

Parameters

<i>path</i>	The path.
-------------	-----------

Returns

The resulting DataSetIdentifier.

Exceptions

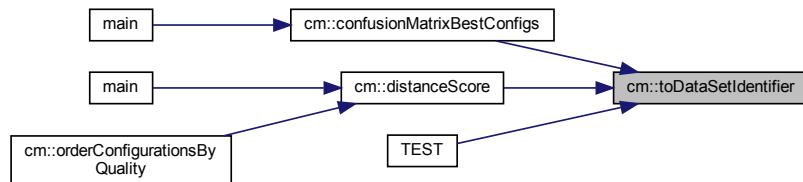
<i>cl::Exception</i>	if path is unrecognized.
----------------------	--------------------------

Definition at line 38 of file data_set_identifier.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.3.3 Variable Documentation

5.3.3.1 imuCount

```
constexpr std::size_t cm::imuCount [inline], [constexpr]
```

Initial value:

```
{
#define CM_IMU_X(enm)
                           CM_IMU
}
```

The amount of IMUs.

Definition at line 26 of file imu.hpp.

5.3.3.2 imus

```
constexpr std::array<Imu, imuCount> cm::imus [inline], [constexpr]
```

Initial value:

```
{  
#define CM_IMU_X(enm)  
    CM_IMU  
}  
}
```

An array of the IMU enumerators.

Definition at line 35 of file imu.hpp.

5.4 cs Namespace Reference

Classes

- class [CsvLineBuilder](#)
Builder for a CSV line.
- struct [data_set_info](#)
Meta function for data set tags.
- class [LogInfo](#)
Information about a log file.
- class [LogLine](#)
A line out of a log file.

Enumerations

- enum [FilterKind](#) { [FilterKind::Butterworth](#), [FilterKind::MovingAverage](#) }
Type for the different kinds of filters.
- enum [SegmentationKind](#) : pl::byte { [SegmentationKind::Minima](#) = 0b0000'0001, [SegmentationKind::Maxima](#) = 0b0000'0010, [SegmentationKind::Both](#) = Minima | Maxima }
The segmentation kind.

Functions

- [PL_DEFINE_EXCEPTION_TYPE](#) ([NoSuchDataSetException](#), [std::logic_error](#))
- [CS_SPECIALIZE_DATA_SET_INFO](#) ([Felix1](#), "11.17.39", 24)
- [CS_SPECIALIZE_DATA_SET_INFO](#) ([Felix2](#), "12.50.00", 20)
- [CS_SPECIALIZE_DATA_SET_INFO](#) ([Felix3](#), "13.00.09", 15)
- [CS_SPECIALIZE_DATA_SET_INFO](#) ([Marcelle1](#), "14.59.59", 10)
- [CS_SPECIALIZE_DATA_SET_INFO](#) ([Marcelle2](#), "15.13.22", 16)
- [CS_SPECIALIZE_DATA_SET_INFO](#) ([Marcelle3](#), "15.31.36", 18)
- [CS_SPECIALIZE_DATA_SET_INFO](#) ([Mike1](#), "14.07.33", 26)
- [CS_SPECIALIZE_DATA_SET_INFO](#) ([Mike2](#), "14.14.32", 22)
- [CS_SPECIALIZE_DATA_SET_INFO](#) ([Mike3](#), "14.20.28", 18)
- [CS_SPECIALIZE_DATA_SET_INFO](#) ([Andre1](#), "Andre_liegestuetzen1", 27)
- [CS_SPECIALIZE_DATA_SET_INFO](#) ([Andre2](#), "Andre_liegestuetzen2", 20)
- [CS_SPECIALIZE_DATA_SET_INFO](#) ([Andre3](#), "Andre_liegestuetzen3", 17)
- [CS_SPECIALIZE_DATA_SET_INFO](#) ([AndreSquats1](#), "Andre_Squats", 30)

- `CS_SPECIALIZE_DATA_SET_INFO` (AndreSquats2, "Andre_Squats2", 49)
- `CS_SPECIALIZE_DATA_SET_INFO` (Jan1, "Jan_liegestuetzen1", 25)
- `CS_SPECIALIZE_DATA_SET_INFO` (Jan2, "Jan_liegestuetzen2", 19)
- `CS_SPECIALIZE_DATA_SET_INFO` (Jan3, "Jan_liegestuetzen3", 13)
- `CS_SPECIALIZE_DATA_SET_INFO` (Lucas1, "Lucas_liegestuetzen1", 24)
- `CS_SPECIALIZE_DATA_SET_INFO` (Lucas2, "Lucas_liegestuetzen2", 19)
- `CS_SPECIALIZE_DATA_SET_INFO` (Lucas3, "Lucas_liegestuetzen3", 11)
- `std::uint64_t repetitionCount (pl::string_view dataSet)`
Fetches the repetition count for a given data set identified by its string.
- `std::ostream & operator<< (std::ostream &os, FilterKind filterKind)`
Prints a FilterKind to an ostream.
- `cl::Expected< std::vector< cl::fs::Path > > logFiles (pl::string_view directoryPath)`
Fetches the paths to the log files in the given directory.
- `std::ostream & operator<< (std::ostream &os, SegmentationKind segmentationKind)`
Prints a SegmentationKind to an ostream.
- `bool operator== (const LogInfo &lhs, const LogInfo &rhs) noexcept`
- `bool operator!= (const LogInfo &lhs, const LogInfo &rhs) noexcept`
- `std::ostream & operator<< (std::ostream &os, const LogInfo &logInfo)`

Variables

- `constexpr pl::string_view logPath {"segmentation_comparison/logs"}`
Relative path to the directory containing the preprocessed log files.
- `constexpr pl::string_view oldLogPath {"segmentation_comparison/logs/old"}`
Relative path to the directory containing the old log files.

5.4.1 Enumeration Type Documentation

5.4.1.1 FilterKind

enum `cs::FilterKind` [strong]

Type for the different kinds of filters.

Enumerator

Butterworth	
MovingAverage	

Definition at line 9 of file filter_kind.hpp.

5.4.1.2 SegmentationKind

enum `cs::SegmentationKind` : `pl::byte` [strong]

The segmentation kind.

Enumerator

Minima	Segmentation by local minima
Maxima	Segmentation by local maxima
Both	Segmentation by both local extrema

Definition at line 12 of file segmentation_kind.hpp.

5.4.2 Function Documentation

5.4.2.1 CS_SPECIALIZE_DATA_SET_INFO() [1/20]

```
cs::CS_SPECIALIZE_DATA_SET_INFO (
    Andre1 ,
    "Andre_liegestuetzen1" ,
    27 )
```

5.4.2.2 CS_SPECIALIZE_DATA_SET_INFO() [2/20]

```
cs::CS_SPECIALIZE_DATA_SET_INFO (
    Andre2 ,
    "Andre_liegestuetzen2" ,
    20 )
```

5.4.2.3 CS_SPECIALIZE_DATA_SET_INFO() [3/20]

```
cs::CS_SPECIALIZE_DATA_SET_INFO (
    Andre3 ,
    "Andre_liegestuetzen3" ,
    17 )
```

5.4.2.4 CS_SPECIALIZE_DATA_SET_INFO() [4/20]

```
cs::CS_SPECIALIZE_DATA_SET_INFO (
    AndreSquats1 ,
    "Andre_Squats" ,
    30 )
```

5.4.2.5 CS_SPECIALIZE_DATA_SET_INFO() [5/20]

```
cs::CS_SPECIALIZE_DATA_SET_INFO (
    AndreSquats2 ,
    "Andre_Squats2" ,
    49  )
```

5.4.2.6 CS_SPECIALIZE_DATA_SET_INFO() [6/20]

```
cs::CS_SPECIALIZE_DATA_SET_INFO (
    Felix1 ,
    "11.17.39" ,
    24  )
```

5.4.2.7 CS_SPECIALIZE_DATA_SET_INFO() [7/20]

```
cs::CS_SPECIALIZE_DATA_SET_INFO (
    Felix2 ,
    "12.50.00" ,
    20  )
```

5.4.2.8 CS_SPECIALIZE_DATA_SET_INFO() [8/20]

```
cs::CS_SPECIALIZE_DATA_SET_INFO (
    Felix3 ,
    "13.00.09" ,
    15  )
```

5.4.2.9 CS_SPECIALIZE_DATA_SET_INFO() [9/20]

```
cs::CS_SPECIALIZE_DATA_SET_INFO (
    Jan1 ,
    "Jan_liegestuetzen1" ,
    25  )
```

5.4.2.10 CS_SPECIALIZE_DATA_SET_INFO() [10/20]

```
cs::CS_SPECIALIZE_DATA_SET_INFO (
    Jan2 ,
    "Jan_liegestuetzen2" ,
    19  )
```

5.4.2.11 CS_SPECIALIZE_DATA_SET_INFO() [11/20]

```
cs::CS_SPECIALIZE_DATA_SET_INFO (
    Jan3 ,
    "Jan_liegestuetzen3" ,
    13  )
```

5.4.2.12 CS_SPECIALIZE_DATA_SET_INFO() [12/20]

```
cs::CS_SPECIALIZE_DATA_SET_INFO (
    Lucas1 ,
    "Lukas_liegestuetzen1" ,
    24  )
```

5.4.2.13 CS_SPECIALIZE_DATA_SET_INFO() [13/20]

```
cs::CS_SPECIALIZE_DATA_SET_INFO (
    Lucas2 ,
    "Lukas_liegestuetzen2" ,
    19  )
```

5.4.2.14 CS_SPECIALIZE_DATA_SET_INFO() [14/20]

```
cs::CS_SPECIALIZE_DATA_SET_INFO (
    Lucas3 ,
    "Lukas_liegestuetzen3" ,
    11  )
```

5.4.2.15 CS_SPECIALIZE_DATA_SET_INFO() [15/20]

```
cs::CS_SPECIALIZE_DATA_SET_INFO (
    Marcellle1 ,
    "14.59.59" ,
    10  )
```

5.4.2.16 CS_SPECIALIZE_DATA_SET_INFO() [16/20]

```
cs::CS_SPECIALIZE_DATA_SET_INFO (
    Marcellle2 ,
    "15.13.22" ,
    16  )
```

5.4.2.17 CS_SPECIALIZE_DATA_SET_INFO() [17/20]

```
cs::CS_SPECIALIZE_DATA_SET_INFO (
    Marcelle3 ,
    "15.31.36" ,
    18  )
```

5.4.2.18 CS_SPECIALIZE_DATA_SET_INFO() [18/20]

```
cs::CS_SPECIALIZE_DATA_SET_INFO (
    Mikel ,
    "14.07.33" ,
    26  )
```

5.4.2.19 CS_SPECIALIZE_DATA_SET_INFO() [19/20]

```
cs::CS_SPECIALIZE_DATA_SET_INFO (
    Mike2 ,
    "14.14.32" ,
    22  )
```

5.4.2.20 CS_SPECIALIZE_DATA_SET_INFO() [20/20]

```
cs::CS_SPECIALIZE_DATA_SET_INFO (
    Mike3 ,
    "14.20.28" ,
    18  )
```

5.4.2.21 logFiles()

```
cl::Expected< std::vector< cl::fs::Path > > cs::logFiles (
    pl::string_view directoryPath )
```

Fetches the paths to the log files in the given directory.

Parameters

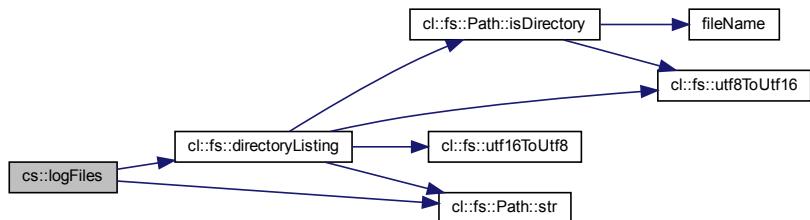
<i>directoryPath</i>	The path to a directory to search for log files.
----------------------	--

Returns

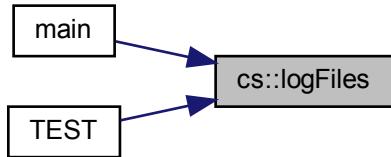
The log files found or an error.

Definition at line 9 of file log_files.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

**5.4.2.22 operator"!=()**

```

bool cs::operator!= (
    const LogInfo & lhs,
    const LogInfo & rhs ) [noexcept]
  
```

Parameters

<i>lhs</i>	The left hand side operand.
<i>rhs</i>	The right hand side operand.

Returns

true if *lhs* and *rhs* are considered not equal; otherwise false.

Definition at line 287 of file log_info.cpp.

5.4.2.23 operator<<() [1/3]

```
std::ostream& cs::operator<< (
    std::ostream & os,
    const LogInfo & logInfo )
```

Parameters

<i>os</i>	The ostream to print to.
<i>logInfo</i>	The LogInfo to print.

Returns

os

Definition at line 292 of file log_info.cpp.

5.4.2.24 operator<<() [2/3]

```
std::ostream & cs::operator<< (
    std::ostream & os,
    FilterKind filterKind )
```

Prints a FilterKind to an ostream.

Parameters

<i>os</i>	The ostream to print to.
<i>filterKind</i>	The FilterKind to print.

Returns

os

Definition at line 6 of file filter_kind.cpp.

5.4.2.25 operator<<() [3/3]

```
std::ostream & cs::operator<< (
    std::ostream & os,
    SegmentationKind segmentationKind )
```

Prints a SegmentationKind to an ostream.

Parameters

<i>os</i>	The ostream to print to.
<i>segmentationKind</i>	The SegmentationKind to print.

Returns

```
os
```

Definition at line 6 of file segmentation_kind.cpp.

5.4.2.26 operator==()

```
bool cs::operator== (
    const LogInfo & lhs,
    const LogInfo & rhs ) [noexcept]
```

Parameters

<i>lhs</i>	The left hand side operand.
<i>rhs</i>	The right hand side operand.

Returns

true if *lhs* and *rhs* are considered equal; otherwise false.

Definition at line 264 of file log_info.cpp.

5.4.2.27 PL_DEFINE_EXCEPTION_TYPE()

```
cs::PL_DEFINE_EXCEPTION_TYPE (
    NoSuchDataSetException ,
    std::logic_error )
```

5.4.2.28 repetitionCount()

```
std::uint64_t cs::repetitionCount (
    pl::string_view dataSet )
```

Fetches the repetition count for a given data set identified by its string.

Parameters

<code>dataSet</code>	The data set to fetch the repetition count of.
----------------------	--

Returns

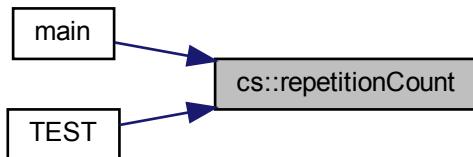
The repetition count of `dataSet`.

Warning

`dataSet` may not be invalid!

Definition at line 10 of file `data_set_info.cpp`.

Here is the caller graph for this function:



5.4.3 Variable Documentation

5.4.3.1 logPath

```
constexpr pl::string_view cs::logPath {"segmentation_comparison/logs"} [inline], [constexpr]
```

Relative path to the directory containing the preprocessed log files.

Definition at line 9 of file `paths.hpp`.

5.4.3.2 oldLogPath

```
constexpr pl::string_view cs::oldLogPath {"segmentation_comparison/logs/old"} [inline], [constexpr]
```

Relative path to the directory containing the old log files.

Definition at line 14 of file `paths.hpp`.

5.5 ctg Namespace Reference

Functions

- `std::vector< cl::DataPoint > aboveThreshold (const cl::DataSet &dataSet, long double accelerometerThreshold, long double gyroscopeThreshold)`
- `long double averageComparisonValueCalculator (cl::Sensor sensor, cl::Channel channel, const cl::DataSet &dataSet)`
- `long double halfMaximumComparisonValueCalculator (cl::Sensor sensor, cl::Channel channel, const cl::DataSet &dataSet)`
- `template<typename ComparisonValueCalculator> bool isRelevant (cl::Sensor sensor, cl::Channel channel, const cl::DataSet &dataSet, ComparisonValueCalculator comparisonValueCalculator)`
- `constexpr long double percentageOf (std::size_t amount, std::size_t totalCount) noexcept`
- `void runAboveThreshold (std::ostream &aboveThresholdLogFileStream, const cl::DataSet &dataSet)`

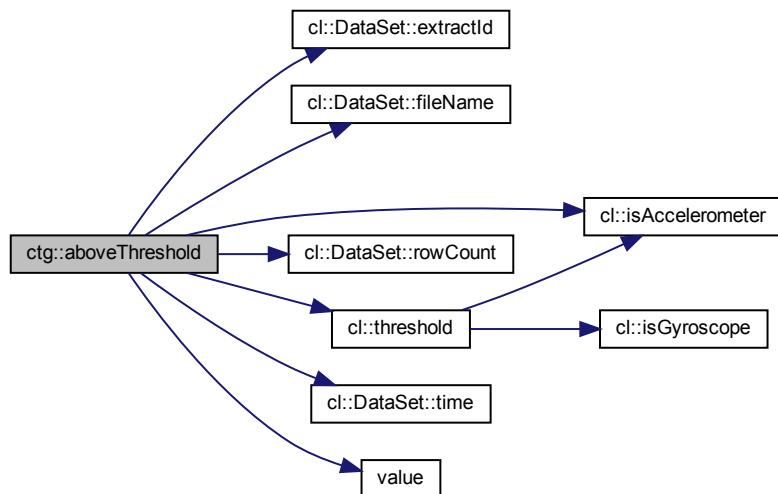
5.5.1 Function Documentation

5.5.1.1 aboveThreshold()

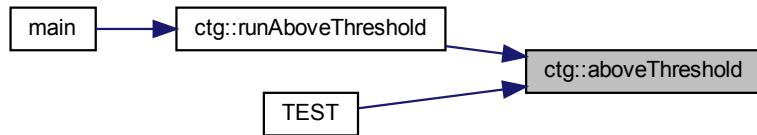
```
std::vector< cl::DataPoint > ctg::aboveThreshold (
    const cl::DataSet & dataSet,
    long double accelerometerThreshold,
    long double gyroscopeThreshold )
```

Definition at line 28 of file above_threshold.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



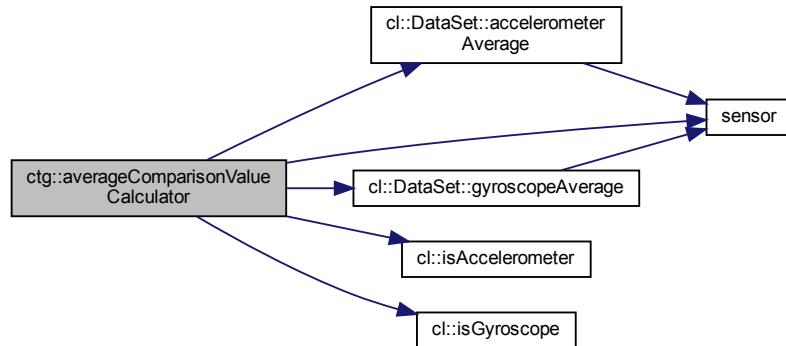
5.5.1.2 averageComparisonValueCalculator()

```

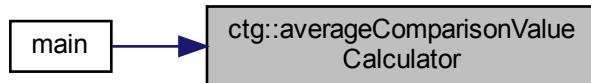
long double ctg::averageComparisonValueCalculator (
    cl::Sensor sensor,
    cl::Channel channel,
    const cl::DataSet & dataSet )
  
```

Definition at line 10 of file average_comparison_value_calculator.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

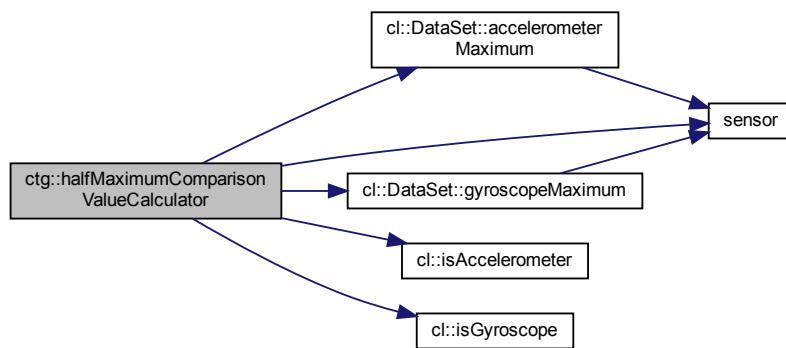


5.5.1.3 halfMaximumComparisonValueCalculator()

```
long double ctg::halfMaximumComparisonValueCalculator (
    cl::Sensor sensor,
    cl::Channel channel,
    const cl::DataSet & dataSet )
```

Definition at line 10 of file half_maximum_comparison_value_calculator.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

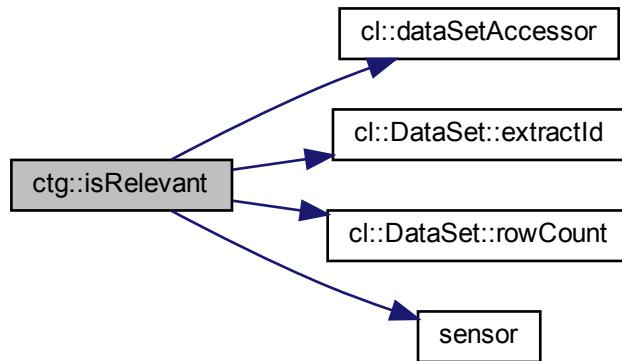


5.5.1.4 isRelevant()

```
template<typename ComparisonValueCalculator >
bool ctg::isRelevant (
    cl::Sensor sensor,
    cl::Channel channel,
    const cl::DataSet & dataSet,
    ComparisonValueCalculator comparisonValueCalculator )
```

Definition at line 11 of file is_relevant.hpp.

Here is the call graph for this function:



Here is the caller graph for this function:

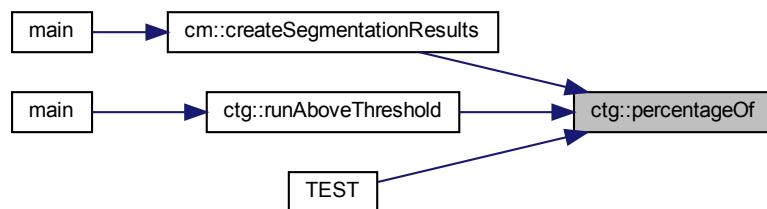


5.5.1.5 percentageOf()

```
constexpr long double ctg::percentageOf (
    std::size_t amount,
    std::size_t totalCount ) [constexpr], [noexcept]
```

Definition at line 6 of file percentage_of.hpp.

Here is the caller graph for this function:

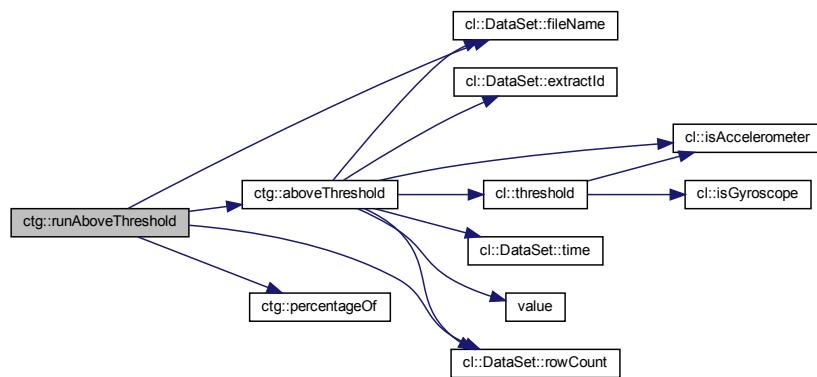


5.5.1.6 runAboveThreshold()

```
void ctg::runAboveThreshold (
    std::ostream & aboveThresholdLogFileStream,
    const cl::DataSet & dataSet )
```

Definition at line 14 of file run_above_threshold.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.6 fmc Namespace Reference

Functions

- void [adjustHardwareTimestamp](#) (std::string *cellContent, const std::string &nextRowHardwareTimestamp, std::uint64_t *overflowCount)
- bool [convertToUnixLineEndings](#) (const std::string &csvPath)
- bool [createBackupFile](#) (const std::string &csvFilePath, const std::string &backupFilePath)
- void [deleteNonBoschSensors](#) (std::vector< std::vector< std::string >> *data)
- [cl::Expected< void >](#) [deleteOutOfBoundsValues](#) (std::vector< std::vector< std::string >> *data)
- void [removeZerosFromField](#) (std::string *field)
- bool [restoreFromBackup](#) (const std::string &csvFilePath, const std::string &backupFilePath)
- bool [writeFile](#) (pl::string_view csvPath, pl::string_view csvFileExtension, const std::vector< std::string > &columnNames, const std::vector< std::vector< std::string >> &data)

5.6.1 Function Documentation

5.6.1.1 `adjustHardwareTimestamp()`

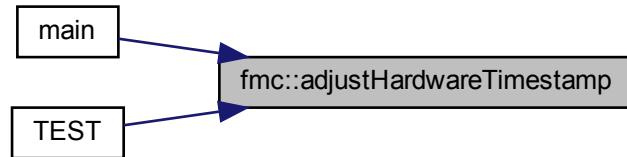
```
void fmc::adjustHardwareTimestamp (
    std::string * cellContent,
    const std::string & nextRowHardwareTimestamp,
    std::uint64_t * overflowCount )
```

Definition at line 16 of file `adjust_hardware_timestamp.cpp`.

Here is the call graph for this function:



Here is the caller graph for this function:

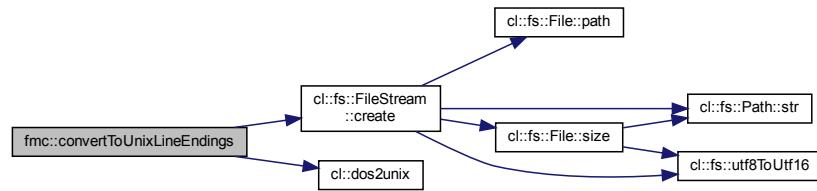


5.6.1.2 `convertToUnixLineEndings()`

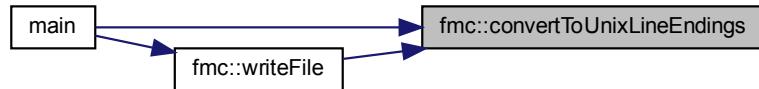
```
bool fmc::convertToUnixLineEndings (
    const std::string & csvPath )
```

Definition at line 18 of file `convert_to_unix_line_endings.cpp`.

Here is the call graph for this function:



Here is the caller graph for this function:



5.6.1.3 `createBackupFile()`

```

bool fmc::createBackupFile (
    const std::string & csvFilePath,
    const std::string & backupFilePath )
  
```

Definition at line 6 of file `create_backup_file.cpp`.

Here is the caller graph for this function:



5.6.1.4 deleteNonBoschSensors()

```
void fmc::deleteNonBoschSensors (
    std::vector< std::vector< std::string >> * data )
```

Definition at line 30 of file `delete_non_bosch_sensors.cpp`.

Here is the caller graph for this function:



5.6.1.5 deleteOutOfBoundsValues()

```
cl::Expected< void > fmc::deleteOutOfBoundsValues (
    std::vector< std::vector< std::string >> * data )
```

Definition at line 29 of file `delete_out_of_bounds_values.cpp`.

Here is the caller graph for this function:

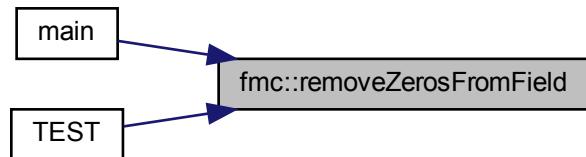


5.6.1.6 removeZerosFromField()

```
void fmc::removeZerosFromField (
    std::string * field )
```

Definition at line 6 of file `remove_zeros_from_field.cpp`.

Here is the caller graph for this function:



5.6.1.7 restoreFromBackup()

```
bool fmc::restoreFromBackup (
    const std::string & csvFilePath,
    const std::string & backupFilePath )
```

Definition at line 11 of file restore_from_backup.cpp.

Here is the caller graph for this function:

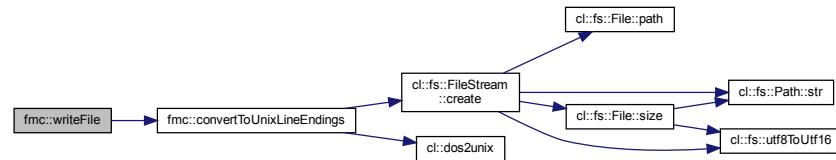


5.6.1.8 writeFile()

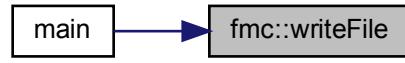
```
bool fmc::writeFile (
    pl::string_view csvPath,
    pl::string_view csvFileExtension,
    const std::vector< std::string > & columnNames,
    const std::vector< std::vector< std::string >> & data )
```

Definition at line 12 of file write_file.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



Chapter 6

Class Documentation

6.1 cm::Configuration::Builder Class Reference

[Builder](#) type for [Configuration](#).

```
#include <configuration.hpp>
```

Public Member Functions

- [Builder \(\) noexcept](#)
Creates an empty [Builder](#).
- [Builder & skipWindow \(bool value\)](#)
Sets the [skipWindow](#) property.
- [Builder & deleteTooClose \(bool value\)](#)
Sets the [deleteTooClose](#) property.
- [Builder & deleteTooLowVariance \(bool value\)](#)
Sets the [deleteTooLowVariance](#) property.
- [Builder & imu \(Imu value\)](#)
Sets the [imu](#) property.
- [Builder & segmentationKind \(std::string value\)](#)
Sets the [segmentationKind](#) property.
- [Builder & windowSize \(std::size_t value\)](#)
Sets the [windowSize](#) property.
- [Builder & filterKind \(std::string value\)](#)
Sets the [filterKind](#) property.
- [Configuration build \(\) const](#)
Builds a [Configuration](#).

6.1.1 Detailed Description

[Builder](#) type for [Configuration](#).

Definition at line 40 of file configuration.hpp.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 Builder()

```
cm::Configuration::Builder::Builder () [noexcept]
```

Creates an empty [Builder](#).

Definition at line 39 of file configuration.cpp.

6.1.3 Member Function Documentation

6.1.3.1 build()

```
Configuration cm::Configuration::Builder::build () const
```

Builds a [Configuration](#).

Returns

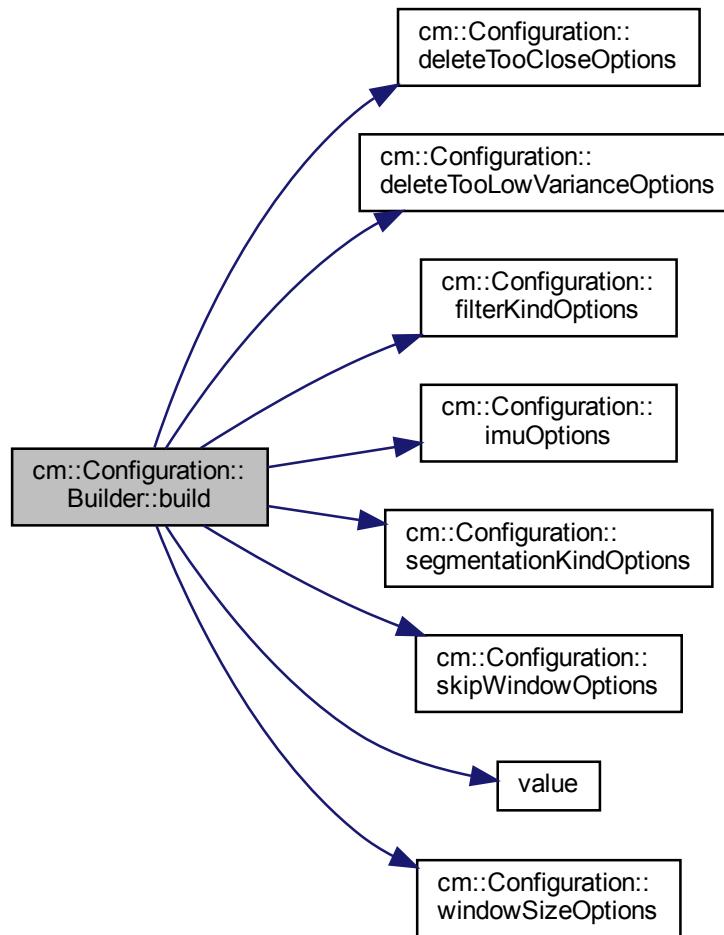
The [Configuration](#) built.

Exceptions

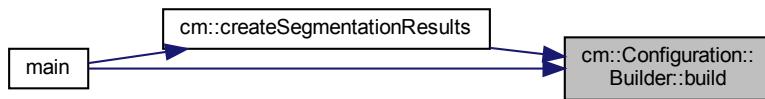
cl::Exception	if one of the properties has not been set or is invalid.
-------------------------------	--

Definition at line 93 of file configuration.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



6.1.3.2 `deleteTooClose()`

```
Configuration::Builder & cm::Configuration::Builder::deleteTooClose (
    bool value )
```

Sets the deleteTooClose property.

Parameters

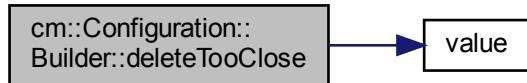
<code>value</code>	The value to use.
--------------------	-------------------

Returns

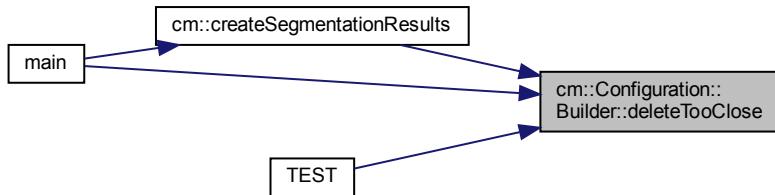
`*this`

Definition at line 56 of file configuration.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



6.1.3.3 `deleteTooLowVariance()`

```
Configuration::Builder & cm::Configuration::Builder::deleteTooLowVariance (
    bool value )
```

Sets the deleteTooLowVariance property.

Parameters

<code>value</code>	The value to use.
--------------------	-------------------

Returns

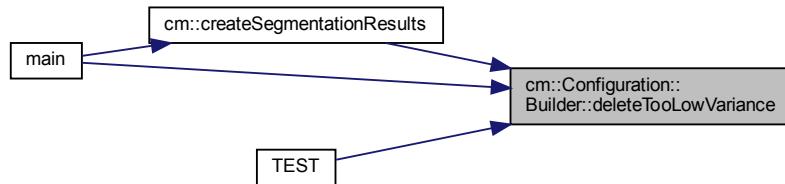
*this

Definition at line 62 of file configuration.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



6.1.3.4 filterKind()

```
Configuration::Builder & cm::Configuration::Builder::filterKind (
    std::string value )
```

Sets the filterKind property.

Parameters

value	The value to use.
-------	-------------------

Returns

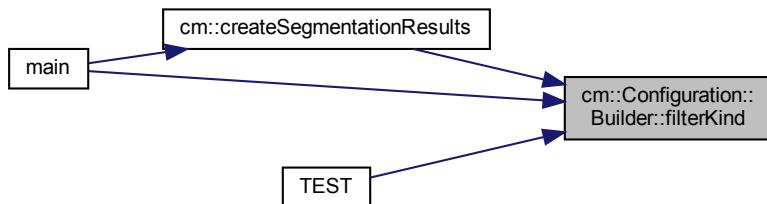
*this

Definition at line 87 of file configuration.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



6.1.3.5 imu()

```
Configuration::Builder & cm::Configuration::Builder::imu ( Imu value )
```

Sets the imu property.

Parameters

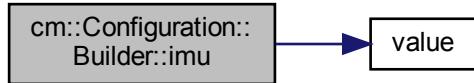
<code>value</code>	The value to use.
--------------------	-------------------

Returns

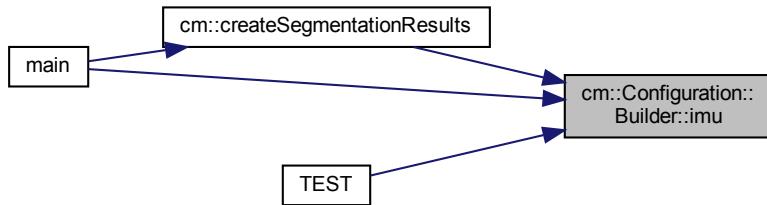
`*this`

Definition at line 68 of file configuration.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



6.1.3.6 segmentationKind()

```
Configuration::Builder & cm::Configuration::Builder::segmentationKind ( std::string value )
```

Sets the segmentationKind property.

Parameters

<code>value</code>	The value to use.
--------------------	-------------------

Returns

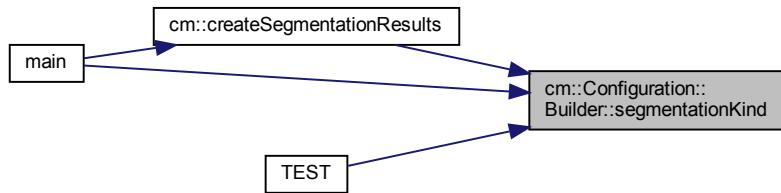
`*this`

Definition at line 74 of file configuration.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



6.1.3.7 skipWindow()

```
Configuration::Builder & cm::Configuration::Builder::skipWindow ( bool value )
```

Sets the skipWindow property.

Parameters

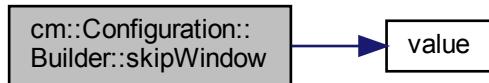
value	The value to use.
-------	-------------------

Returns

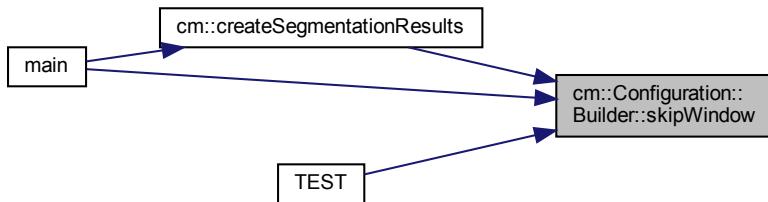
*this

Definition at line 50 of file configuration.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



6.1.3.8 windowSize()

```
Configuration::Builder & cm::Configuration::Builder::windowSize (
    std::size_t value )
```

Sets the windowSize property.

Parameters

value	The value to use.
-------	-------------------

Returns

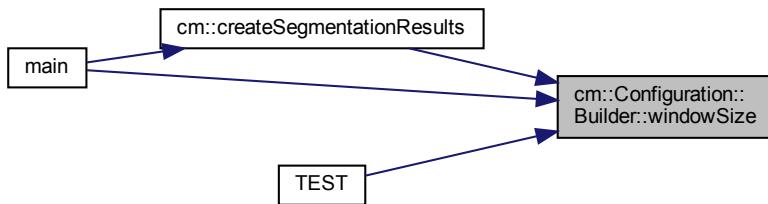
*this

Definition at line 81 of file configuration.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



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

- confusion_matrix/include/[configuration.hpp](#)
- confusion_matrix/src/[configuration.cpp](#)

6.2 cl::col_traits< Col > Struct Template Reference

```
#include <column.hpp>
```

6.2.1 Detailed Description

```
template<Column Col>
struct cl::col_traits< Col >
```

Definition at line 24 of file column.hpp.

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

- csv_lib/include/cl/column.hpp

6.3 cm::Configuration Class Reference

Represents a possible configuration for the Python segmentor.

```
#include <configuration.hpp>
```

Classes

- class [Builder](#)
Builder type for [Configuration](#).

Public Member Functions

- [Configuration \(\)](#)
Default constructor.
- bool [skipWindow \(\) const noexcept](#)
Read accessor for the skipWindow property.
- bool [deleteTooClose \(\) const noexcept](#)
Read accessor for the deleteTooClose property.
- bool [deleteTooLowVariance \(\) const noexcept](#)
Read accessor for the deleteTooLowVariance property.
- [Imu imu \(\) const noexcept](#)
Read accessor for the imu property.
- const std::string & [segmentationKind \(\) const noexcept](#)
Read accessor for the segmentationKind property.
- std::size_t [windowSize \(\) const noexcept](#)
Read accessor for the windowSize property.
- const std::string & [filterKind \(\) const noexcept](#)
Read accessor for the filterKind property.
- bool [isInitialized \(\) const noexcept](#)
Checks if this object is initialized and thus valid.
- [cl::fs::Path createFilePath \(\) const](#)
Create a file path for this kind of Configuration.
- bool [serializeSegmentationPoints \(const std::unordered_map< cl::fs::Path, std::vector< std::uint64_t >> &segmentationPointsMap\) const](#)
Serializes a map of segmentation points to the file path for this Configuration.
- std::unordered_map< [cl::fs::Path](#), std::vector< std::uint64_t > > [importSegmentationPoints \(\) const](#)
Imports segmentation points from the file path for this Configuration.

Static Public Member Functions

- static const std::deque< bool > & [skipWindowOptions \(\) noexcept](#)
Returns the possible skipWindow options.
- static const std::deque< bool > & [deleteTooCloseOptions \(\) noexcept](#)
Returns the possible deleteTooClose options.
- static const std::deque< bool > & [deleteTooLowVarianceOptions \(\) noexcept](#)
Returns the possible deleteTooLowVariance options.
- static const std::vector< [Imu](#) > & [imuOptions \(\) noexcept](#)
Returns the possible imu options.
- static const std::vector< std::string > & [segmentationKindOptions \(\) noexcept](#)
Returns the possible segmentationKind options.
- static const std::vector< std::size_t > & [windowSizeOptions \(\) noexcept](#)
Returns the possible windowSize options.
- static const std::vector< std::string > & [filterKindOptions \(\) noexcept](#)
Returns the possible filterKind options.

Friends

- class `Builder`
- struct `std::hash< Configuration >`
- bool `operator==(const Configuration &lhs, const Configuration &rhs) noexcept`
Compares two Configurations for equality.
- bool `operator!=(const Configuration &lhs, const Configuration &rhs) noexcept`
Compares two Configurations for inequality.
- `std::ostream & operator<< (std::ostream &os, const Configuration &config)`
Prints config to os.

6.3.1 Detailed Description

Represents a possible configuration for the Python segmentor.

Definition at line 32 of file configuration.hpp.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 Configuration()

```
cm::Configuration::Configuration ( )
```

Default constructor.

Warning

This constructor is only there to work around Microsoft buggedness, don't use.

Note

Creates an uninitialized object!

Definition at line 256 of file configuration.cpp.

6.3.3 Member Function Documentation

6.3.3.1 createFilePath()

```
cl::fs::Path cm::Configuration::createFilePath ( ) const
```

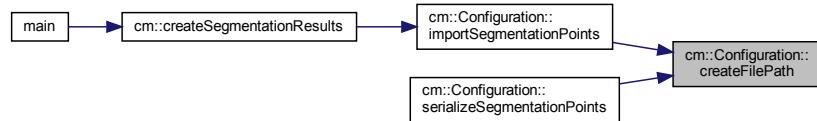
Create a file path for this kind of Configuration.

Returns

The file path for this kind of Configuration.

Definition at line 293 of file configuration.cpp.

Here is the caller graph for this function:



6.3.3.2 deleteTooClose()

```
bool cm::Configuration::deleteTooClose ( ) const [noexcept]
```

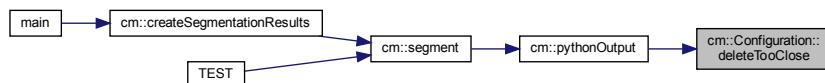
Read accessor for the `deleteTooClose` property.

Returns

The `deleteTooClose` option.

Definition at line 270 of file configuration.cpp.

Here is the caller graph for this function:



6.3.3.3 deleteTooCloseOptions()

```
const std::deque< bool > & cm::Configuration::deleteTooCloseOptions ( ) [static], [noexcept]
```

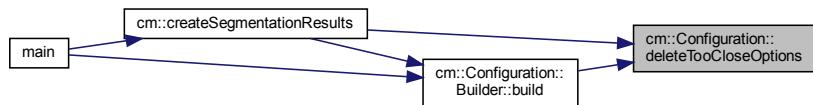
Returns the possible deleteTooClose options.

Returns

The deleteTooClose options.

Definition at line 154 of file configuration.cpp.

Here is the caller graph for this function:



6.3.3.4 deleteTooLowVariance()

```
bool cm::Configuration::deleteTooLowVariance ( ) const [noexcept]
```

Read accessor for the deleteTooLowVariance property.

Returns

The deleteTooLowVariance option.

Definition at line 272 of file configuration.cpp.

Here is the caller graph for this function:



6.3.3.5 deleteTooLowVarianceOptions()

```
const std::deque< bool > & cm::Configuration::deleteTooLowVarianceOptions ( ) [static], [noexcept]
```

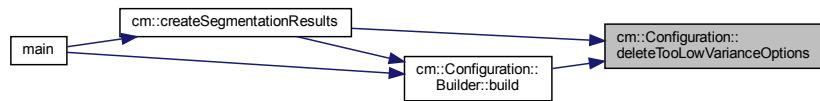
Returns the possible deleteTooLowVariance options.

Returns

The deleteTooLowVariance options.

Definition at line 160 of file configuration.cpp.

Here is the caller graph for this function:



6.3.3.6 filterKind()

```
const std::string & cm::Configuration::filterKind ( ) const [noexcept]
```

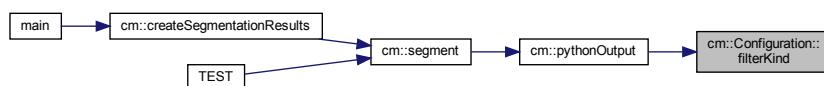
Read accessor for the filterKind property.

Returns

The filterKind option.

Definition at line 286 of file configuration.cpp.

Here is the caller graph for this function:



6.3.3.7 filterKindOptions()

```
const std::vector< std::string > & cm::Configuration::filterKindOptions ( ) [static], [noexcept]
```

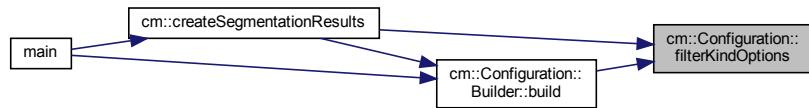
Returns the possible filterKind options.

Returns

The filterKind options.

Definition at line 204 of file configuration.cpp.

Here is the caller graph for this function:



6.3.3.8 importSegmentationPoints()

```
std::unordered_map< cl::fs::Path, std::vector< std::uint64_t > > cm::Configuration::importSegmentationPoints ( ) const
```

Imports segmentation points from the file path for this [Configuration](#).

Returns

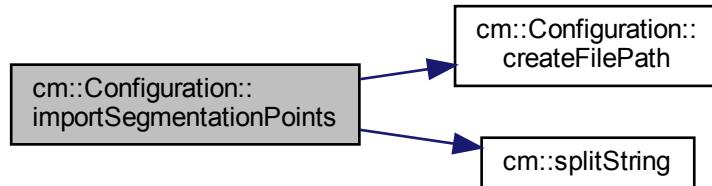
The imported segmentation points.

Exceptions

cl::Exception	if the file path for this Configuration does not exist or an error occurs while reading / parsing.
-------------------------------	--

Definition at line 332 of file configuration.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



6.3.3.9 imu()

`Imu cm::Configuration::imu () const [noexcept]`

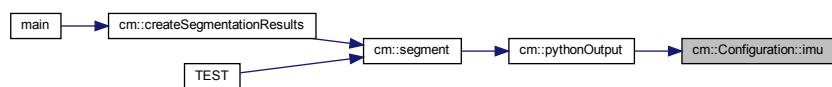
Read accessor for the imu property.

Returns

The imu option.

Definition at line 277 of file configuration.cpp.

Here is the caller graph for this function:



6.3.3.10 imuOptions()

```
const std::vector< Imu > & cm::Configuration::imuOptions ( ) [static], [noexcept]
```

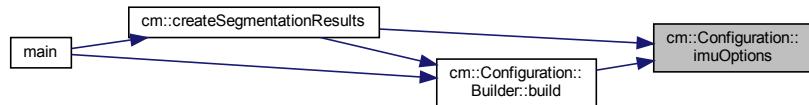
Returns the possible imu options.

Returns

The imu options.

Definition at line 166 of file configuration.cpp.

Here is the caller graph for this function:



6.3.3.11 isInitialized()

```
bool cm::Configuration::isInitialized ( ) const [noexcept]
```

Checks if this object is initialized and thus valid.

Returns

true if this object is initialized; false otherwise.

Warning

If you use the [Builder](#) to construct (as you should) this member function will always return true. false will only be returned if you use the (invalid) default constructor that servers as a workaround for MSVC bugs.

Definition at line 291 of file configuration.cpp.

6.3.3.12 segmentationKind()

```
const std::string & cm::Configuration::segmentationKind() const [noexcept]
```

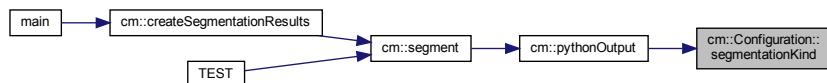
Read accessor for the segmentationKind property.

Returns

The segmentationKind option.

Definition at line 279 of file configuration.cpp.

Here is the caller graph for this function:



6.3.3.13 segmentationKindOptions()

```
const std::vector< std::string > & cm::Configuration::segmentationKindOptions() [static], [noexcept]
```

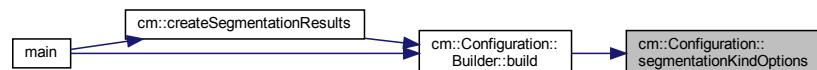
Returns the possible segmentationKind options.

Returns

The segmentationKind options.

Definition at line 173 of file configuration.cpp.

Here is the caller graph for this function:



6.3.3.14 serializeSegmentationPoints()

```
bool cm::Configuration::serializeSegmentationPoints(
    const std::unordered_map< cl::fs::Path, std::vector< std::uint64_t >> & segmentationPointsMap ) const
```

Serializes a map of segmentation points to the file path for this [Configuration](#).

Parameters

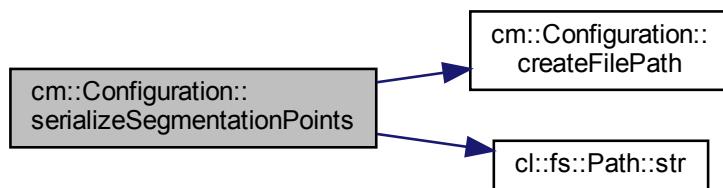
<i>segmentationPointsMap</i>	The map to serialize.
------------------------------	-----------------------

Returns

true on success; false otherwise.

Definition at line 311 of file configuration.cpp.

Here is the call graph for this function:

**6.3.3.15 skipWindow()**

```
bool cm::Configuration::skipWindow() const [noexcept]
```

Read accessor for the skipWindow property.

Returns

The skipWindow option.

Definition at line 268 of file configuration.cpp.

Here is the caller graph for this function:



6.3.3.16 skipWindowOptions()

```
const std::deque< bool > & cm::Configuration::skipWindowOptions ( ) [static], [noexcept]
```

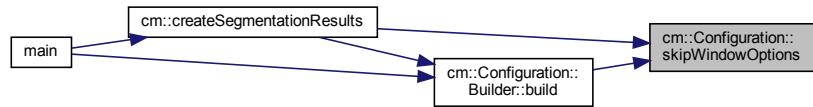
Returns the possible skipWindow options.

Returns

The skipWindow options.

Definition at line 148 of file configuration.cpp.

Here is the caller graph for this function:



6.3.3.17 windowHeight()

```
std::size_t cm::Configuration::windowSize ( ) const [noexcept]
```

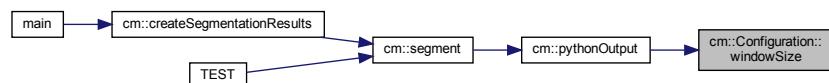
Read accessor for the windowHeight property.

Returns

The windowHeight option.

Definition at line 284 of file configuration.cpp.

Here is the caller graph for this function:



6.3.3.18 `windowSizeOptions()`

```
const std::vector< std::size_t > & cm::Configuration::windowSizeOptions ( ) [static], [noexcept]
```

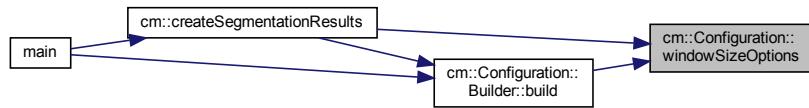
Returns the possible `windowSize` options.

Returns

The `windowSize` options.

Definition at line 179 of file `configuration.cpp`.

Here is the caller graph for this function:



6.3.4 Friends And Related Function Documentation

6.3.4.1 `Builder`

```
friend class Builder [friend]
```

Definition at line 34 of file `configuration.hpp`.

6.3.4.2 `operator"!=`

```
bool operator!= (
    const Configuration & lhs,
    const Configuration & rhs ) [friend]
```

Compares two Configurations for inequality.

Parameters

<code>lhs</code>	The first operand.
<code>rhs</code>	The second operand.

Returns

true if lhs and rhs are considered not to be equal.

Definition at line 230 of file configuration.cpp.

6.3.4.3 operator<<

```
std::ostream& operator<< (
    std::ostream & os,
    const Configuration & config ) [friend]
```

Prints config to os.

Parameters

<i>os</i>	The ostream to print to.
<i>config</i>	The Configuration to print.

Returns

os

Definition at line 235 of file configuration.cpp.

6.3.4.4 operator==

```
bool operator== (
    const Configuration & lhs,
    const Configuration & rhs ) [friend]
```

Compares two Configurations for equality.

Parameters

<i>lhs</i>	The first operand.
<i>rhs</i>	The second operand.

Returns

true if lhs and rhs are considered to be equal.

Definition at line 210 of file configuration.cpp.

6.3.4.5 std::hash< Configuration >

```
friend struct std::hash< Configuration > [friend]
```

Definition at line 35 of file configuration.hpp.

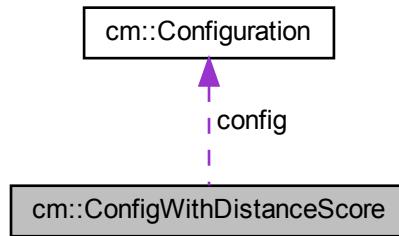
The documentation for this class was generated from the following files:

- confusion_matrix/include/configuration.hpp
- confusion_matrix/src/configuration.cpp

6.4 cm::ConfigWithDistanceScore Struct Reference

```
#include <order_configurations_by_quality.hpp>
```

Collaboration diagram for cm::ConfigWithDistanceScore:



Public Member Functions

- [ConfigWithDistanceScore \(Configuration p_config, std::uint64_t p_distScore\)](#)

Public Attributes

- [Configuration config](#)
- [std::uint64_t distScore](#)

6.4.1 Detailed Description

Definition at line 15 of file order_configurations_by_quality.hpp.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 ConfigWithDistanceScore()

```
cm::ConfigWithDistanceScore::ConfigWithDistanceScore (
    Configuration p_config,
    std::uint64_t p_distScore )
```

Definition at line 13 of file order_configurations_by_quality.cpp.

6.4.3 Member Data Documentation

6.4.3.1 config

```
Configuration cm::ConfigWithDistanceScore::config
```

Definition at line 18 of file order_configurations_by_quality.hpp.

6.4.3.2 distScore

```
std::uint64_t cm::ConfigWithDistanceScore::distScore
```

Definition at line 19 of file order_configurations_by_quality.hpp.

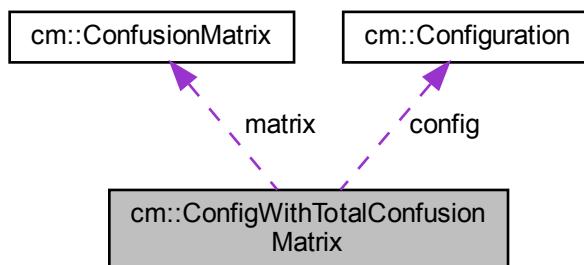
The documentation for this struct was generated from the following files:

- confusion_matrix/include/[order_configurations_by_quality.hpp](#)
- confusion_matrix/src/[order_configurations_by_quality.cpp](#)

6.5 cm::ConfigWithTotalConfusionMatrix Struct Reference

```
#include <confusion_matrix_best_configs.hpp>
```

Collaboration diagram for cm::ConfigWithTotalConfusionMatrix:



Public Member Functions

- `ConfigWithTotalConfusionMatrix ()=default`
- `ConfigWithTotalConfusionMatrix (Configuration p_config, ConfusionMatrix p_matrix)`

Public Attributes

- `Configuration config`
- `ConfusionMatrix matrix`

6.5.1 Detailed Description

Definition at line 12 of file confusion_matrix_best_configs.hpp.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 ConfigWithTotalConfusionMatrix() [1/2]

```
cm::ConfigWithTotalConfusionMatrix::ConfigWithTotalConfusionMatrix ( ) [default]
```

6.5.2.2 ConfigWithTotalConfusionMatrix() [2/2]

```
cm::ConfigWithTotalConfusionMatrix::ConfigWithTotalConfusionMatrix ( Configuration p_config, ConfusionMatrix p_matrix )
```

Definition at line 41 of file confusion_matrix_best_configs.cpp.

6.5.3 Member Data Documentation

6.5.3.1 config

```
Configuration cm::ConfigWithTotalConfusionMatrix::config
```

Definition at line 19 of file confusion_matrix_best_configs.hpp.

6.5.3.2 matrix

```
ConfusionMatrix cm::ConfigWithTotalConfusionMatrix::matrix
```

Definition at line 20 of file confusion_matrix_best_configs.hpp.

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

- confusion_matrix/include/confusion_matrix_best_configs.hpp
- confusion_matrix/src/confusion_matrix_best_configs.cpp

6.6 cm::ConfusionMatrix Class Reference

```
#include <confusion_matrix.hpp>
```

Public Types

- using `this_type` = ConfusionMatrix

Public Member Functions

- `ConfusionMatrix()`
- `std::uint64_t truePositives() const noexcept`
- `std::uint64_t trueNegatives() const noexcept`
- `std::uint64_t falsePositives() const noexcept`
- `std::uint64_t falseNegatives() const noexcept`
- `std::uint64_t totalCount() const noexcept`
- `this_type & incrementTruePositives() noexcept`
- `this_type & incrementTrueNegatives() noexcept`
- `this_type & incrementFalsePositives() noexcept`
- `this_type & incrementFalseNegatives() noexcept`
- `this_type & operator+=(const ConfusionMatrix &other) noexcept`

6.6.1 Detailed Description

Definition at line 6 of file confusion_matrix.hpp.

6.6.2 Member Typedef Documentation

6.6.2.1 this_type

```
using cm::ConfusionMatrix::this_type = ConfusionMatrix
```

Definition at line 8 of file confusion_matrix.hpp.

6.6.3 Constructor & Destructor Documentation

6.6.3.1 ConfusionMatrix()

```
cm::ConfusionMatrix::ConfusionMatrix ( )
```

Definition at line 4 of file confusion_matrix.cpp.

6.6.4 Member Function Documentation

6.6.4.1 falseNegatives()

```
std::uint64_t cm::ConfusionMatrix::falseNegatives ( ) const [noexcept]
```

Definition at line 28 of file confusion_matrix.cpp.

6.6.4.2 falsePositives()

```
std::uint64_t cm::ConfusionMatrix::falsePositives ( ) const [noexcept]
```

Definition at line 23 of file confusion_matrix.cpp.

6.6.4.3 incrementFalseNegatives()

```
ConfusionMatrix & cm::ConfusionMatrix::incrementFalseNegatives ( ) [noexcept]
```

Definition at line 59 of file confusion_matrix.cpp.

Here is the caller graph for this function:



6.6.4.4 incrementFalsePositives()

```
ConfusionMatrix & cm::ConfusionMatrix::incrementFalsePositives () [noexcept]
```

Definition at line 52 of file confusion_matrix.cpp.

Here is the caller graph for this function:



6.6.4.5 incrementTrueNegatives()

```
ConfusionMatrix & cm::ConfusionMatrix::incrementTrueNegatives () [noexcept]
```

Definition at line 45 of file confusion_matrix.cpp.

Here is the caller graph for this function:



6.6.4.6 incrementTruePositives()

```
ConfusionMatrix & cm::ConfusionMatrix::incrementTruePositives () [noexcept]
```

Definition at line 38 of file confusion_matrix.cpp.

Here is the caller graph for this function:



6.6.4.7 operator+=()

```
ConfusionMatrix & cm::ConfusionMatrix::operator+= ( const ConfusionMatrix & other ) [noexcept]
```

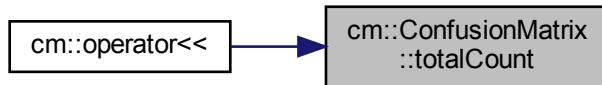
Definition at line 66 of file confusion_matrix.cpp.

6.6.4.8 totalCount()

```
std::uint64_t cm::ConfusionMatrix::totalCount ( ) const [noexcept]
```

Definition at line 33 of file confusion_matrix.cpp.

Here is the caller graph for this function:



6.6.4.9 trueNegatives()

```
std::uint64_t cm::ConfusionMatrix::trueNegatives ( ) const [noexcept]
```

Definition at line 18 of file confusion_matrix.cpp.

6.6.4.10 truePositives()

```
std::uint64_t cm::ConfusionMatrix::truePositives ( ) const [noexcept]
```

Definition at line 13 of file confusion_matrix.cpp.

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

- confusion_matrix/include/[confusion_matrix.hpp](#)
- confusion_matrix/src/[confusion_matrix.cpp](#)

6.7 cm::CsvFileInfo Class Reference

Type to hold the hardware timestamps of a CSV file.

```
#include <csv_file_info.hpp>
```

Public Member Functions

- `CsvFileInfo` (const `cl::fs::Path` &`csvFilePath`)
Reads the hardware timestamps from csvFilePath.
- const `std::vector< std::uint64_t >` & `hardwareTimestamps` () const noexcept
Read accessor for the hardware timestamps.

6.7.1 Detailed Description

Type to hold the hardware timestamps of a CSV file.

Definition at line 13 of file `csv_file_info.hpp`.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 CsvFileInfo()

```
cm::CsvFileInfo::CsvFileInfo (
    const cl::fs::Path & csvFilePath ) [explicit]
```

Reads the hardware timestamps from `csvFilePath`.

Parameters

<code>csvFilePath</code>	The CSV to read the hardware timestamps from.
--------------------------	---

Exceptions

<code>cl::Exception</code>	on error.
----------------------------	-----------

Definition at line 10 of file `csv_file_info.cpp`.

6.7.3 Member Function Documentation

6.7.3.1 hardwareTimestamps()

```
const std::vector< std::uint64_t > & cm::CsvFileInfo::hardwareTimestamps() const [noexcept]
```

Read accessor for the hardware timestamps.

Returns

The hardware timestamps.

Definition at line 56 of file csv_file_info.cpp.

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

- confusion_matrix/include/csv_file_info.hpp
- confusion_matrix/src/csv_file_info.cpp

6.8 cs::CsvLineBuilder Class Reference

Builder for a CSV line.

```
#include <csv_line.hpp>
```

Public Types

- using `this_type` = CsvLineBuilder

Public Member Functions

- `CsvLineBuilder()`
Creates an empty, invalid CsvLineBuilder.
- `this_type & skipWindow(bool value)`
Write accessor for the skip window property.
- `this_type & deleteTooClose(bool value)`
Write accessor for the delete too close property.
- `this_type & deleteLowVariance(bool value)`
Write accessor for the delete low variance property.
- `this_type & kind(SegmentationKind value)`
Write accessor for the kind property.
- `this_type & windowSize(std::uint64_t value)`
Write accessor for the window size property.
- `this_type & filter(FilterKind value)`
Write accessor for the filter property.
- `this_type & dataSet(std::string value)`
Write accessor for the data set property.
- `this_type & sensor(std::uint64_t value)`
Write accessor for the sensor property.
- `this_type & repetitions(std::uint64_t value)`
Write accessor for the repetitions property.
- `this_type & segmentationPoints(std::uint64_t value)`
Write accessor for the segmentation points property.
- `this_type & isOld(bool value)`
Write accessor for the is old property.
- `std::vector< std::string > build() const`
Builds the CSV line as a vector containing the cells of the CSV line.

6.8.1 Detailed Description

Builder for a CSV line.

Builder type for a CSV line. All write accessors have to be called before the build member function is called!

Definition at line 21 of file csv_line.hpp.

6.8.2 Member Typedef Documentation

6.8.2.1 this_type

```
using cs::CsvLineBuilder::this_type = CsvLineBuilder
```

Definition at line 23 of file csv_line.hpp.

6.8.3 Constructor & Destructor Documentation

6.8.3.1 CsvLineBuilder()

```
cs::CsvLineBuilder::CsvLineBuilder( )
```

Creates an empty, invalid [CsvLineBuilder](#).

Definition at line 44 of file csv_line.cpp.

6.8.4 Member Function Documentation

6.8.4.1 build()

```
std::vector< std::string > cs::CsvLineBuilder::build () const
```

Builds the CSV line as a vector containing the cells of the CSV line.

Returns

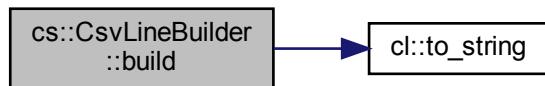
The resulting vector of strings.

Warning

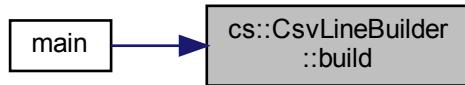
May only be called after all the write accessors have been called.

Definition at line 124 of file csv_line.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



6.8.4.2 dataSet()

```
CsvLineBuilder & cs::CsvLineBuilder::dataSet (
    std::string value )
```

Write accessor for the data set property.

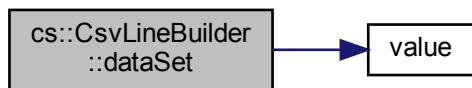
Parameters

<code>value</code>	The value to use.
--------------------	-------------------

Returns`*this`

Definition at line 94 of file csv_line.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



6.8.4.3 `deleteLowVariance()`

```
CsvLineBuilder & cs::CsvLineBuilder::deleteLowVariance ( bool value )
```

Write accessor for the delete low variance property.

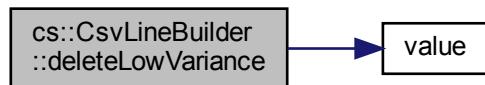
Parameters

<code>value</code>	The value to use.
--------------------	-------------------

Returns`*this`

Definition at line 70 of file csv_line.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



6.8.4.4 deleteTooClose()

```
CsvLineBuilder & cs::CsvLineBuilder::deleteTooClose (
    bool value )
```

Write accessor for the delete too close property.

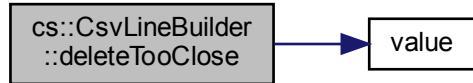
Parameters

<code>value</code>	The value to use.
--------------------	-------------------

Returns`*this`

Definition at line 64 of file csv_line.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



6.8.4.5 filter()

```
CsvLineBuilder & cs::CsvLineBuilder::filter (  
    FilterKind value )
```

Write accessor for the filter property.

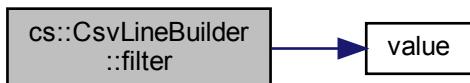
Parameters

<code>value</code>	The value to use.
--------------------	-------------------

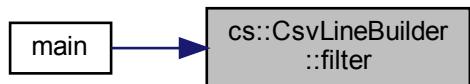
Returns`*this`

Definition at line 88 of file csv_line.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



6.8.4.6 `isOld()`

```
CsvLineBuilder & cs::CsvLineBuilder::isOld (
    bool value )
```

Write accessor for the is old property.

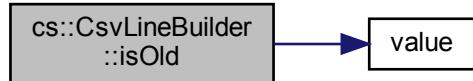
Parameters

<code>value</code>	The value to use.
--------------------	-------------------

Returns`*this`

Definition at line 118 of file csv_line.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



6.8.4.7 kind()

```
CsvLineBuilder & cs::CsvLineBuilder::kind ( SegmentationKind value )
```

Write accessor for the kind property.

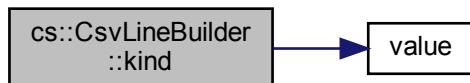
Parameters

<code>value</code>	The value to use.
--------------------	-------------------

Returns`*this`

Definition at line 76 of file csv_line.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



6.8.4.8 repetitions()

```
CsvLineBuilder & cs::CsvLineBuilder::repetitions (
    std::uint64_t value )
```

Write accessor for the repetitions property.

Parameters

<code>value</code>	The value to use.
--------------------	-------------------

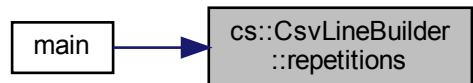
Returns`*this`

Definition at line 106 of file csv_line.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



6.8.4.9 segmentationPoints()

```
CsvLineBuilder & cs::CsvLineBuilder::segmentationPoints ( std::uint64_t value )
```

Write accessor for the segmentation points property.

Parameters

<code>value</code>	The value to use.
--------------------	-------------------

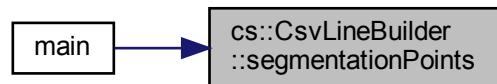
Returns`*this`

Definition at line 112 of file csv_line.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



6.8.4.10 `sensor()`

```
CsvLineBuilder & cs::CsvLineBuilder::sensor (  
    std::uint64_t value )
```

Write accessor for the sensor property.

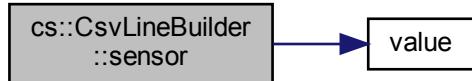
Parameters

<code>value</code>	The value to use.
--------------------	-------------------

Returns`*this`

Definition at line 100 of file csv_line.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



6.8.4.11 skipWindow()

```
CsvLineBuilder & cs::CsvLineBuilder::skipWindow ( bool value )
```

Write accessor for the skip window property.

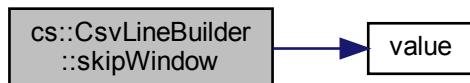
Parameters

<code>value</code>	The value to use.
--------------------	-------------------

Returns`*this`

Definition at line 58 of file csv_line.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



6.8.4.12 `windowSize()`

```
CsvLineBuilder & cs::CsvLineBuilder::windowSize( std::uint64_t value )
```

Write accessor for the window size property.

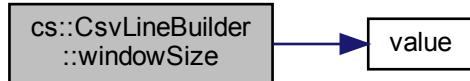
Parameters

<code>value</code>	The value to use.
--------------------	-------------------

Returns`*this`

Definition at line 82 of file csv_line.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



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

- compare_segmentation/include/[csv_line.hpp](#)
- compare_segmentation/src/[csv_line.cpp](#)

6.9 cl::data_set_accessor< Chan > Struct Template Reference

```
#include <channel.hpp>
```

6.9.1 Detailed Description

```
template<Channel Chan>
struct cl::data_set_accessor< Chan >
```

Definition at line 39 of file channel.hpp.

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

- [csv_lib/include/cl/channel.hpp](#)

6.10 cs::data_set_info< Tag > Struct Template Reference

Meta function for data set tags.

```
#include <data_set_info.hpp>
```

6.10.1 Detailed Description

```
template<typename Tag>
struct cs::data_set_info< Tag >
```

Meta function for data set tags.

Template Parameters

<i>Tag</i>	The data set tag to use.
------------	--------------------------

Meta function for data set tags. Contains a text for the data set tag and its repetition count.

Definition at line 21 of file data_set_info.hpp.

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

- compare_segmentation/include/[data_set_info.hpp](#)

6.11 cl::DataPoint Class Reference

```
#include <data_point.hpp>
```

Public Member Functions

- [DataPoint](#) (std::string *fileName*, long double *time*, Sensor *sensor*, Channel *channel*, long double *value*) noexcept
- const std::string & *fileName* () const noexcept
- long double *time* () const noexcept
- Sensor *sensor* () const noexcept
- Channel *channel* () const noexcept
- long double *value* () const noexcept

Friends

- std::ostream & [operator<<](#) (std::ostream &*os*, const [DataPoint](#) &*dataPoint*)

6.11.1 Detailed Description

Definition at line 10 of file data_point.hpp.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 DataPoint()

```
DataPoint::DataPoint (
    std::string fileName,
    long double time,
    Sensor sensor,
    Channel channel,
    long double value ) [noexcept]
```

Definition at line 21 of file data_point.cpp.

Here is the call graph for this function:



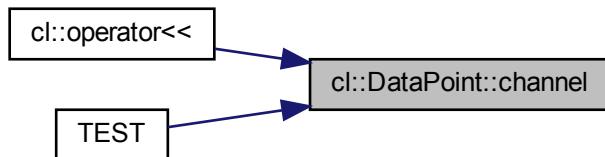
6.11.3 Member Function Documentation

6.11.3.1 channel()

```
Channel DataPoint::channel () const [noexcept]
```

Definition at line 41 of file data_point.cpp.

Here is the caller graph for this function:

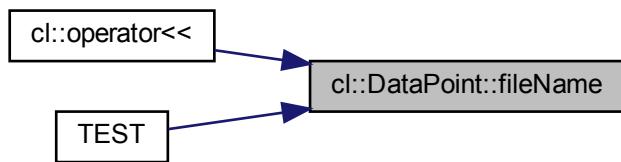


6.11.3.2 fileName()

```
const std::string & DataPoint::fileName ( ) const [noexcept]
```

Definition at line 35 of file data_point.cpp.

Here is the caller graph for this function:

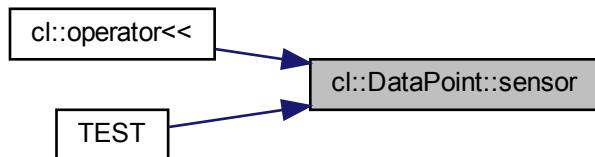


6.11.3.3 sensor()

```
Sensor DataPoint::sensor ( ) const [noexcept]
```

Definition at line 39 of file data_point.cpp.

Here is the caller graph for this function:

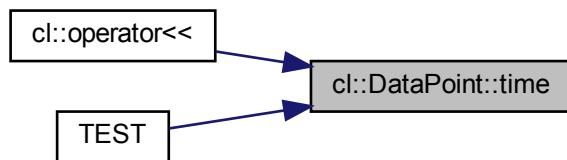


6.11.3.4 time()

```
long double DataPoint::time ( ) const [noexcept]
```

Definition at line 37 of file data_point.cpp.

Here is the caller graph for this function:

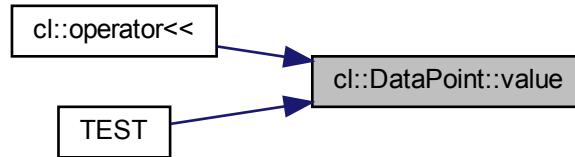


6.11.3.5 value()

```
long double DataPoint::value ( ) const [noexcept]
```

Definition at line 43 of file data_point.cpp.

Here is the caller graph for this function:



6.11.4 Friends And Related Function Documentation

6.11.4.1 operator<<

```
std::ostream& operator<< (
    std::ostream & os,
    const DataPoint & dataPoint ) [friend]
```

Definition at line 10 of file `data_point.cpp`.

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

- [csv_lib/include/cl/data_point.hpp](#)
- [csv_lib/src/cl/data_point.cpp](#)

6.12 cl::DataSet Class Reference

```
#include <data_set.hpp>
```

Public Types

- using `size_type` = `std::size_t`
- using `ChannelAccessor` = `long double(DataSet::*)(size_type) const`

Public Member Functions

- `size_type rowCount () const noexcept`
- `const std::string & fileName () const noexcept`
- `column_type< Column::Time > time (size_type index) const`
- `column_type< Column::HardwareTimestamp > hardwareTimestamp (size_type index) const`
- `column_type< Column::ExtractId > extractId (size_type index) const`
- `column_type< Column::Trigger > trigger (size_type index) const`
- `column_type< Column::AccelerometerX > accelerometerX (size_type index) const`
- `column_type< Column::AccelerometerY > accelerometerY (size_type index) const`
- `column_type< Column::AccelerometerZ > accelerometerZ (size_type index) const`
- `column_type< Column::GyroscopeX > gyroscopeX (size_type index) const`
- `column_type< Column::GyroscopeY > gyroscopeY (size_type index) const`
- `column_type< Column::GyroscopeZ > gyroscopeZ (size_type index) const`
- `long double accelerometerAverage (Sensor sensor) const`
- `long double gyroscopeAverage (Sensor sensor) const`
- `long double accelerometerMaximum (Sensor sensor) const`
- `long double gyroscopeMaximum (Sensor sensor) const`

Static Public Member Functions

- static `Expected< DataSet > create (std::string fileName, const std::vector< std::vector< std::string >> &matrix)`

6.12.1 Detailed Description

Definition at line 14 of file data_set.hpp.

6.12.2 Member Typedef Documentation

6.12.2.1 ChannelAccessor

```
using cl::DataSet::ChannelAccessor = long double (DataSet::*)(size_type) const
```

Definition at line 17 of file data_set.hpp.

6.12.2.2 size_type

```
using cl::DataSet::size_type = std::size_t
```

Definition at line 16 of file data_set.hpp.

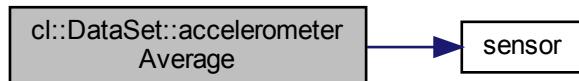
6.12.3 Member Function Documentation

6.12.3.1 accelerometerAverage()

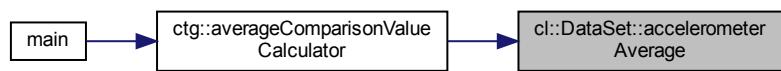
```
long double cl::DataSet::accelerometerAverage (
    Sensor sensor ) const
```

Definition at line 255 of file data_set.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

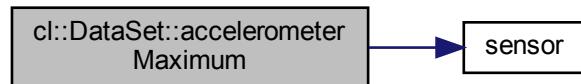


6.12.3.2 accelerometerMaximum()

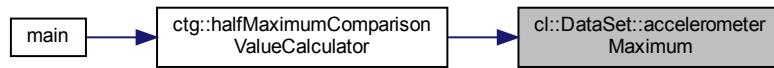
```
long double cl::DataSet::accelerometerMaximum (
    Sensor sensor ) const
```

Definition at line 265 of file data_set.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

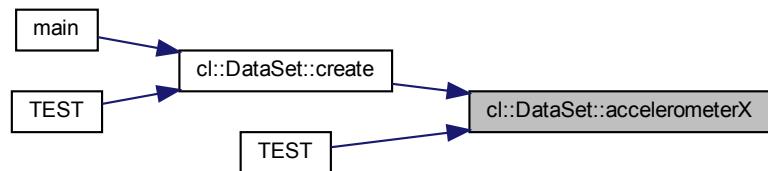


6.12.3.3 accelerometerX()

```
column_type< Column::AccelerometerX > cl::DataSet::accelerometerX (
    size_type index ) const
```

Definition at line 200 of file data_set.cpp.

Here is the caller graph for this function:

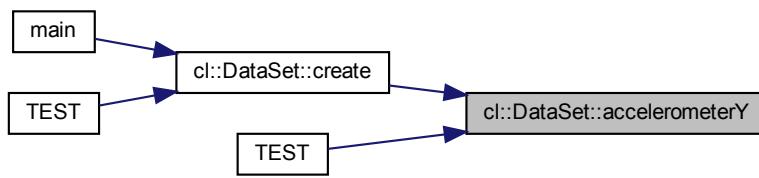


6.12.3.4 accelerometerY()

```
column_type< Column::AccelerometerY > cl::DataSet::accelerometerY ( size_type index ) const
```

Definition at line 208 of file data_set.cpp.

Here is the caller graph for this function:

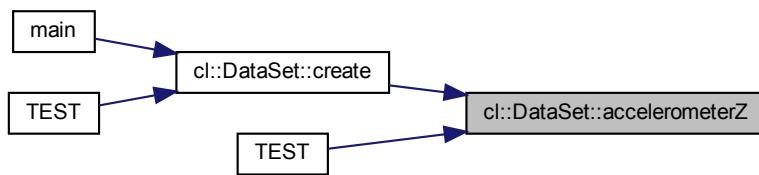


6.12.3.5 accelerometerZ()

```
column_type< Column::AccelerometerZ > cl::DataSet::accelerometerZ ( size_type index ) const
```

Definition at line 216 of file data_set.cpp.

Here is the caller graph for this function:

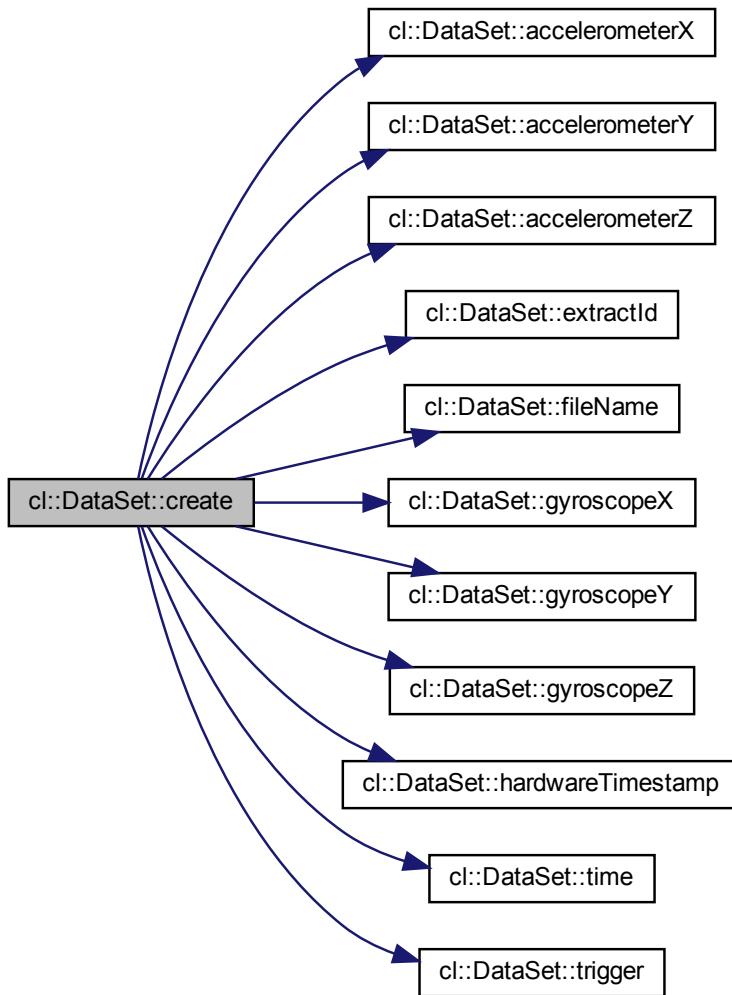


6.12.3.6 create()

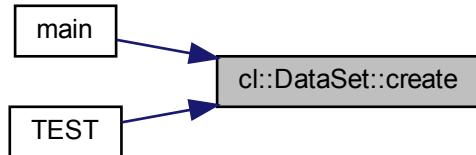
```
Expected< DataSet > cl::DataSet::create (
    std::string fileName,
    const std::vector< std::vector< std::string >> & matrix ) [static]
```

Definition at line 42 of file data_set.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

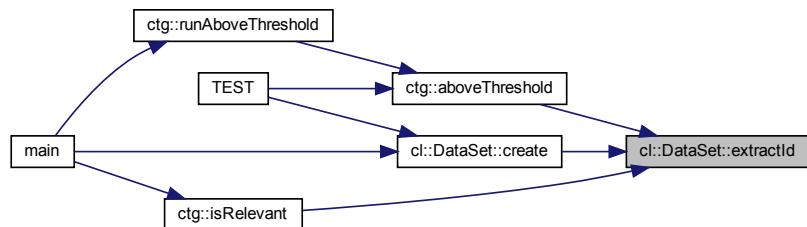


6.12.3.7 extractId()

```
column_type< Column::ExtractId > cl::DataSet::extractId ( size_type index ) const
```

Definition at line 186 of file data_set.cpp.

Here is the caller graph for this function:

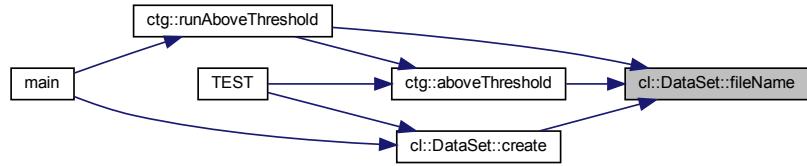


6.12.3.8 fileName()

```
const std::string & cl::DataSet::fileName ( ) const [noexcept]
```

Definition at line 169 of file data_set.cpp.

Here is the caller graph for this function:



6.12.3.9 gyroscopeAverage()

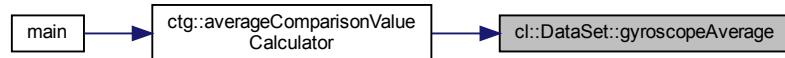
```
long double cl::DataSet::gyroscopeAverage (
    Sensor sensor ) const
```

Definition at line 260 of file `data_set.cpp`.

Here is the call graph for this function:



Here is the caller graph for this function:



6.12.3.10 gyroscopeMaximum()

```
long double cl::DataSet::gyroscopeMaximum (
    Sensor sensor ) const
```

Definition at line 270 of file data_set.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

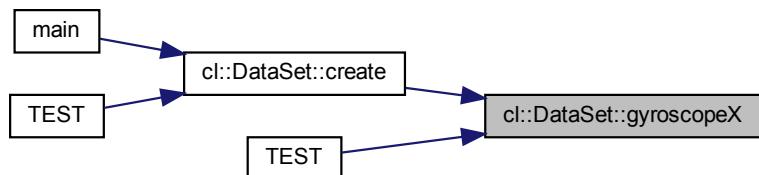


6.12.3.11 gyroscopeX()

```
column_type< Column::GyroscopeX > cl::DataSet::gyroscopeX (
    size_type index ) const
```

Definition at line 224 of file data_set.cpp.

Here is the caller graph for this function:

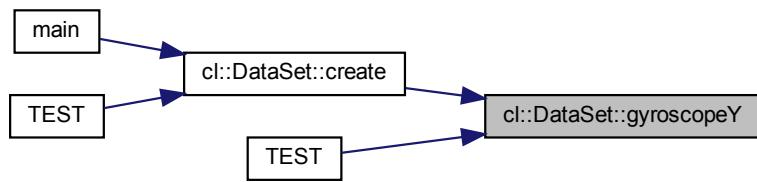


6.12.3.12 gyroscopeY()

```
column_type< Column::GyroscopeY > cl::DataSet::gyroscopeY ( size_type index ) const
```

Definition at line 231 of file data_set.cpp.

Here is the caller graph for this function:

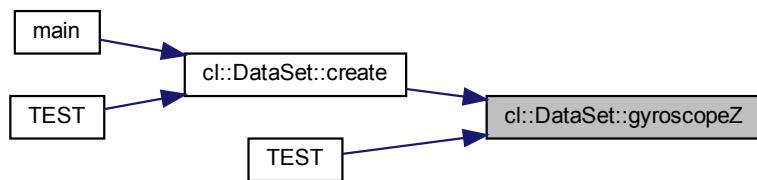


6.12.3.13 gyroscopeZ()

```
column_type< Column::GyroscopeZ > cl::DataSet::gyroscopeZ ( size_type index ) const
```

Definition at line 238 of file data_set.cpp.

Here is the caller graph for this function:

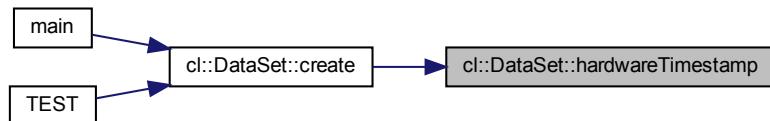


6.12.3.14 hardwareTimestamp()

```
column_type< Column::HardwareTimestamp > cl::DataSet::hardwareTimestamp ( size_type index ) const
```

Definition at line 178 of file data_set.cpp.

Here is the caller graph for this function:

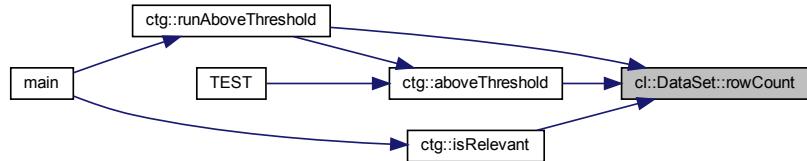


6.12.3.15 rowCount()

```
DataSet::size_type cl::DataSet::rowCount ( ) const [noexcept]
```

Definition at line 152 of file data_set.cpp.

Here is the caller graph for this function:

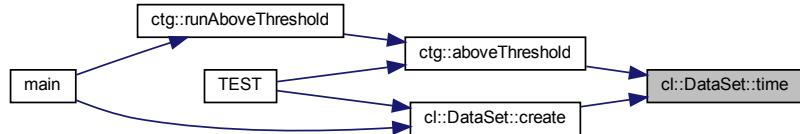


6.12.3.16 time()

```
column_type< Column::Time > cl::DataSet::time ( size_type index ) const
```

Definition at line 171 of file data_set.cpp.

Here is the caller graph for this function:

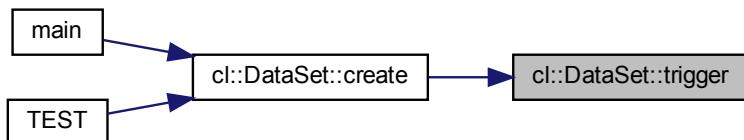


6.12.3.17 trigger()

```
column_type< Column::Trigger > cl::DataSet::trigger (
    size_type index ) const
```

Definition at line 193 of file `data_set.cpp`.

Here is the caller graph for this function:



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

- `csv_lib/include/cl/data_set.hpp`
- `csv_lib/src/cl/data_set.cpp`

6.13 cl::Error Class Reference

```
#include <error.hpp>
```

Public Types

- enum `Kind { CL_ERROR_KIND }`

Public Member Functions

- `Error (Kind kind, std::string file, std::string function, std::size_t line, std::string message)`
- `Kind kind () const noexcept`
- `const std::string & file () const noexcept`
- `const std::string & function () const noexcept`
- `std::size_t line () const noexcept`
- `const std::string & message () const noexcept`
- `void raise () const`
- `std::string to_string () const`

Friends

- `std::ostream & operator<< (std::ostream &os, const Error &error)`

6.13.1 Detailed Description

Definition at line 23 of file error.hpp.

6.13.2 Member Enumeration Documentation

6.13.2.1 Kind

`enum cl::Error::Kind`

Enumerator

<code>CL_ERROR_KIND</code>	<input type="button" value=" "/>
----------------------------	----------------------------------

Definition at line 26 of file error.hpp.

6.13.3 Constructor & Destructor Documentation

6.13.3.1 Error()

```
cl::Error::Error (
    Kind kind,
    std::string file,
    std::string function,
    std::size_t line,
    std::string message )
```

Definition at line 41 of file error.cpp.

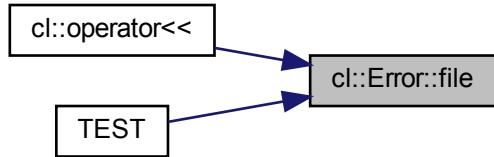
6.13.4 Member Function Documentation

6.13.4.1 file()

```
const std::string & cl::Error::file() const [noexcept]
```

Definition at line 57 of file error.cpp.

Here is the caller graph for this function:

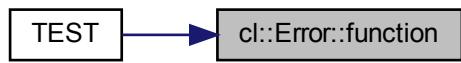


6.13.4.2 function()

```
const std::string & cl::Error::function() const [noexcept]
```

Definition at line 59 of file error.cpp.

Here is the caller graph for this function:

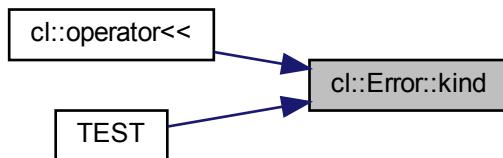


6.13.4.3 kind()

```
Error::Kind cl::Error::kind () const [noexcept]
```

Definition at line 55 of file error.cpp.

Here is the caller graph for this function:



6.13.4.4 line()

```
std::size_t cl::Error::line () const [noexcept]
```

Definition at line 61 of file error.cpp.

6.13.4.5 message()

```
const std::string & cl::Error::message () const [noexcept]
```

Definition at line 63 of file error.cpp.

Here is the caller graph for this function:



6.13.4.6 `raise()`

```
void cl::Error::raise ( ) const
```

Definition at line 65 of file error.cpp.

6.13.4.7 `to_string()`

```
std::string cl::Error::to_string ( ) const
```

Definition at line 74 of file error.cpp.

6.13.5 Friends And Related Function Documentation

6.13.5.1 `operator<<`

```
std::ostream& operator<< (
    std::ostream & os,
    const Error & error ) [friend]
```

Definition at line 30 of file error.cpp.

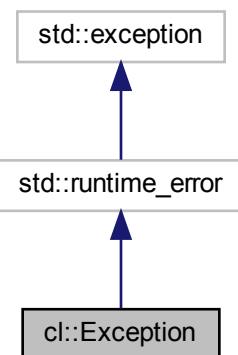
The documentation for this class was generated from the following files:

- csv_lib/include/cl/error.hpp
- csv_lib/src/cl/error.cpp

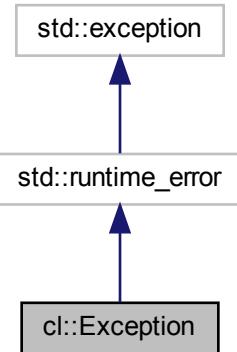
6.14 cl::Exception Class Reference

```
#include <exception.hpp>
```

Inheritance diagram for cl::Exception:



Collaboration diagram for cl::Exception:



Public Types

- using `base_type` = `std::runtime_error`

Public Member Functions

- `Exception (std::string file, std::string function, std::size_t line, const std::string &what_arg)`
- `Exception (std::string file, std::string function, std::size_t line, const char *what_arg)`
- `const std::string & file () const noexcept`
- `const std::string & function () const noexcept`
- `std::size_t line () const noexcept`

6.14.1 Detailed Description

Definition at line 14 of file exception.hpp.

6.14.2 Member Typedef Documentation

6.14.2.1 `base_type`

```
using cl::Exception::base_type = std::runtime_error
```

Definition at line 16 of file exception.hpp.

6.14.3 Constructor & Destructor Documentation

6.14.3.1 Exception() [1/2]

```
cl::Exception::Exception (
    std::string file,
    std::string function,
    std::size_t line,
    const std::string & what_arg )
```

Definition at line 6 of file exception.cpp.

6.14.3.2 Exception() [2/2]

```
cl::Exception::Exception (
    std::string file,
    std::string function,
    std::size_t line,
    const char * what_arg )
```

Definition at line 18 of file exception.cpp.

6.14.4 Member Function Documentation

6.14.4.1 file()

```
const std::string & cl::Exception::file ( ) const [noexcept]
```

Definition at line 30 of file exception.cpp.

Here is the caller graph for this function:

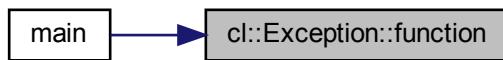


6.14.4.2 function()

```
const std::string & cl::Exception::function() const [noexcept]
```

Definition at line 32 of file exception.cpp.

Here is the caller graph for this function:

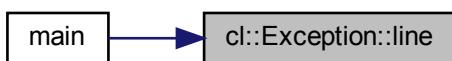


6.14.4.3 line()

```
std::size_t cl::Exception::line() const [noexcept]
```

Definition at line 34 of file exception.cpp.

Here is the caller graph for this function:



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

- csv_lib/include/cl/exception.hpp
- csv_lib/src/cl/exception.cpp

6.15 cl::fs::File Class Reference

Represents a file.

```
#include <file.hpp>
```

Public Member Functions

- **File (Path path)**
Creates a [File](#) from the given `path`.
- **bool exists () const noexcept**
Determines if this file exists.
- **bool create () const noexcept**
Creates this file.
- **bool copyTo (const Path ©ToPath) const noexcept**
Copies this file in the filesystem.
- **bool moveTo (const Path &newPath)**
Moves this file in the filesystem.
- **bool remove () noexcept**
Deletes this file.
- **std::int64_t size () const noexcept**
Determines the size of this file in bytes.
- **const Path & path () const noexcept**
Read accessor for the path of this file.

6.15.1 Detailed Description

Represents a file.

Definition at line 11 of file file.hpp.

6.15.2 Constructor & Destructor Documentation

6.15.2.1 File()

```
cl::fs::File::File (
    Path path ) [explicit]
```

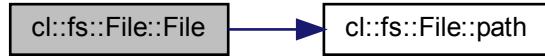
Creates a [File](#) from the given `path`.

Parameters

<code>path</code>	The path to use.
-------------------	------------------

Definition at line 21 of file file.cpp.

Here is the call graph for this function:



6.15.3 Member Function Documentation

6.15.3.1 copyTo()

```
bool cl::fs::File::copyTo (
    const Path & copyToPath ) const [noexcept]
```

Copies this file in the filesystem.

Parameters

<i>copyToPath</i>	The path to copy to.
-------------------	----------------------

Returns

true if the file was successfully copied to *copyToPath*; otherwise false.

Warning

There should be no file that already exists at *copyToPath*.

Definition at line 56 of file file.cpp.

Here is the call graph for this function:



6.15.3.2 `create()`

```
bool cl::fs::File::create() const [noexcept]
```

Creates this file.

Returns

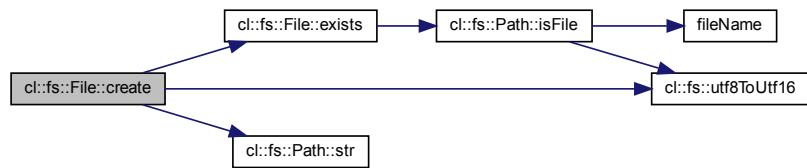
true if the file was successfully created; otherwise false.

Note

Will fail if the file already exists.

Definition at line 25 of file file.cpp.

Here is the call graph for this function:



6.15.3.3 `exists()`

```
bool cl::fs::File::exists() const [noexcept]
```

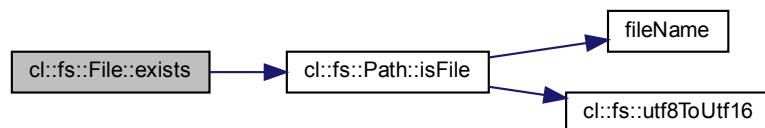
Determines if this file exists.

Returns

true if the file exists; otherwise false.

Definition at line 23 of file file.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



6.15.3.4 moveTo()

```
bool cl::fs::File::moveTo (
    const Path & newPath )
```

Moves this file in the filesystem.

Parameters

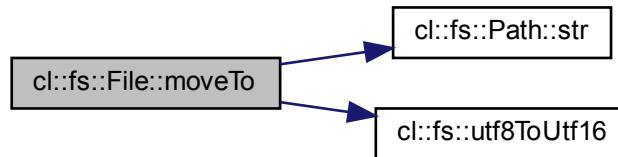
<i>newPath</i>	The path to move this file to.
----------------	--------------------------------

Returns

true if the file was successfully moved to newPath; otherwise false.

Definition at line 100 of file file.cpp.

Here is the call graph for this function:



6.15.3.5 path()

```
const Path & cl::fs::File::path() const [noexcept]
```

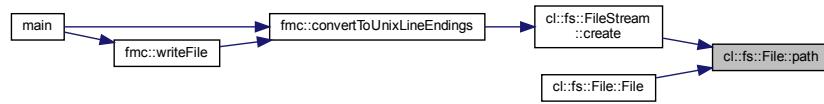
Read accessor for the path of this file.

Returns

The path of this file.

Definition at line 169 of file file.cpp.

Here is the caller graph for this function:



6.15.3.6 remove()

```
bool cl::fs::File::remove() [noexcept]
```

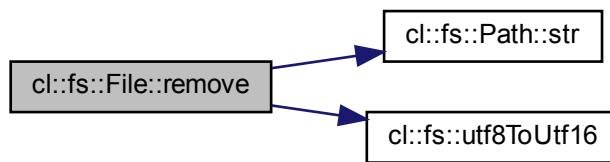
Deletes this file.

Returns

true if deleting succeeded; otherwise false.

Definition at line 117 of file file.cpp.

Here is the call graph for this function:



6.15.3.7 size()

```
std::int64_t cl::fs::File::size() const [noexcept]
```

Determines the size of this file in bytes.

Returns

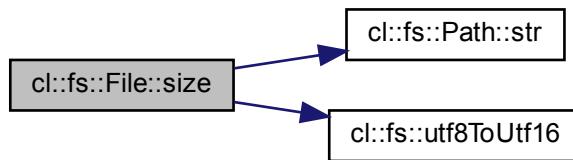
The size of this file in bytes or -1 on error.

Warning

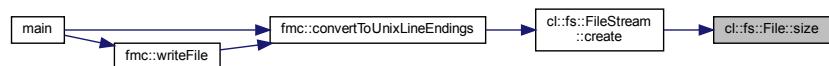
Returns -1 on error.

Definition at line 128 of file file.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



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

- csv_lib/include/cl/fs/file.hpp
- csv_lib/src/cl/fs/file.cpp

6.16 cl::fs::FileStream Class Reference

A binary file stream.

```
#include <file_stream.hpp>
```

Public Types

- enum `OpenMode` : `std::uint8_t` { `Read` = 0b0000'0001, `Write` = 0b0000'0010, `ReadWrite` = `Read` | `Write` }
The file open mode.
- using `this_type` = `FileStream`

Public Member Functions

- `PL_NONCOPYABLE (FileStream)`
- `FileStream (this_type &&other) noexcept`
Move constructs from other.
- `this_type & operator= (this_type &&other) noexcept`
Move assigns other to this file stream.
- `~FileStream ()`
Closes this file stream.
- bool `write (const void *data, std::size_t byteCount)`
Writes data to the file.
- `std::vector< pl::byte > readAll () const`
Reads the entire file into RAM.

Static Public Member Functions

- static `Expected< FileStream > create (const File &file, OpenMode openMode)`
Creates a file stream.

6.16.1 Detailed Description

A binary file stream.

Definition at line 19 of file `file_stream.hpp`.

6.16.2 Member Typedef Documentation

6.16.2.1 `this_type`

```
using cl::fs::FileStream::this_type = FileStream
```

Definition at line 30 of file `file_stream.hpp`.

6.16.3 Member Enumeration Documentation

6.16.3.1 `OpenMode`

```
enum cl::fs::FileStream::OpenMode : std::uint8_t
```

The file open mode.

Enumerator

Read	Read only access
Write	Write only access
ReadWrite	Read and write access

Definition at line 24 of file file_stream.hpp.

6.16.4 Constructor & Destructor Documentation

6.16.4.1 FileStream()

```
cl::fs::FileStream::FileStream (
    this_type && other ) [noexcept]
```

Move constructs from *other*.

Parameters

<i>other</i>	The file stream to move construct from.
--------------	---

Definition at line 70 of file file_stream.cpp.

6.16.4.2 ~FileStream()

```
cl::fs::FileStream::~FileStream ( )
```

Closes this file stream.

Definition at line 84 of file file_stream.cpp.

6.16.5 Member Function Documentation

6.16.5.1 create()

```
Expected< FileStream > cl::fs::FileStream::create (
    const File & file,
    OpenMode openMode ) [static]
```

Creates a file stream.

Parameters

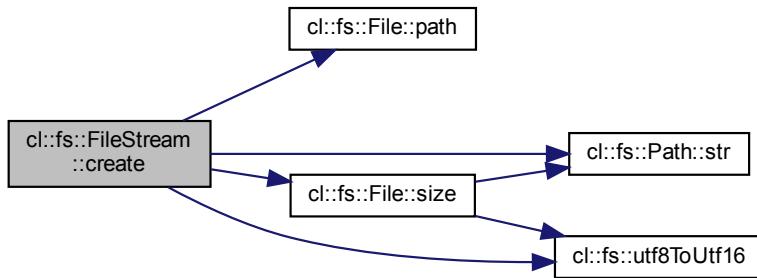
<i>file</i>	The file to open.
<i>openMode</i>	The open mode to use.

Returns

The file stream or an error.

Definition at line 36 of file `file_stream.cpp`.

Here is the call graph for this function:



Here is the caller graph for this function:

**6.16.5.2 operator=()**

```
FileStream & cl::fs::FileStream::operator= (
    this_type && other ) [noexcept]
```

Move assigns `other` to this file stream.

Parameters

<i>other</i>	The file stream to move assign to this file stream.
--------------	---

Returns`*this`

Definition at line 77 of file `file_stream.cpp`.

6.16.5.3 PL_NOCOPYABLE()

```
cl::fs::FileStream::PL_NOCOPYABLE (
    FileStream )
```

6.16.5.4 readAll()

```
std::vector< pl::byte > cl::fs::FileStream::readAll ( ) const
```

Reads the entire file into RAM.

Returns

The bytes read.

Definition at line 103 of file `file_stream.cpp`.

6.16.5.5 write()

```
bool cl::fs::FileStream::write (
    const void * data,
    std::size_t byteCount )
```

Writes data to the file.

Parameters

<code>data</code>	Pointer to the beginning of the memory region to write.
<code>byteCount</code>	The amount of bytes to write, starting from <code>data</code> .

Returns

true on success; otherwise false.

Definition at line 96 of file `file_stream.cpp`.

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

- `csv_lib/include/cl/fs/file_stream.hpp`
- `csv_lib/src/cl/fs/file_stream.cpp`

6.17 std::hash<::cl::fs::Path> Struct Reference

```
#include <path.hpp>
```

Public Member Functions

- size_t [operator\(\)](#) (const ::cl::fs::Path &path) const

6.17.1 Detailed Description

Definition at line 90 of file path.hpp.

6.17.2 Member Function Documentation

6.17.2.1 operator()()

```
size_t std::hash<::cl::fs::Path>::operator() (
    const ::cl::fs::Path & path ) const [inline]
```

Definition at line 91 of file path.hpp.

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

- csv_lib/include/cl/fs/[path.hpp](#)

6.18 std::hash<::cm::Configuration> Struct Reference

```
#include <configuration.hpp>
```

Public Member Functions

- size_t [operator\(\)](#) (const ::cm::Configuration &configuration) const

6.18.1 Detailed Description

Definition at line 296 of file configuration.hpp.

6.18.2 Member Function Documentation

6.18.2.1 operator()

```
size_t std::hash<::cm::Configuration>::operator() (
    const ::cm::Configuration & configuration ) const [inline]
```

Definition at line 297 of file configuration.hpp.

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

- confusion_matrix/include/configuration.hpp

6.19 cs::LogInfo Class Reference

Information about a log file.

```
#include <log_info.hpp>
```

Public Member Functions

- [LogInfo \(\)](#)
Creates an uninitialized LogInfo.
- [const cl::fs::Path & logFilePath \(\) const noexcept](#)
Read accessor for the log file path.
- [bool skipWindow \(\) const noexcept](#)
Read accessor for the skip window option.
- [bool deleteTooClose \(\) const noexcept](#)
Read accessor for the delete too close option.
- [bool deleteLowVariance \(\) const noexcept](#)
Read accessor for the delete low variance option.
- [SegmentationKind segmentationKind \(\) const noexcept](#)
Read accessor for the segmentation kind.
- [std::uint64_t windowSize \(\) const noexcept](#)
Read accessor for the window size.
- [FilterKind filterKind \(\) const noexcept](#)
Read accessor for the filter kind.
- [std::uint64_t sensor \(\) const noexcept](#)
Read accessor for the sensor.
- [bool isInitialized \(\) const noexcept](#)
Checks whether this LogInfo is initialized.

Static Public Member Functions

- [static cl::Expected< LogInfo > create \(cl::fs::Path logFilePath\) noexcept](#)
Creates a LogInfo from the given log file path.

Static Public Attributes

- [static const std::uint64_t invalidSensor = UINT64_C\(0xFFFFFFFFFFFFFF\)](#)
Represents an invalid sensor.

Friends

- bool `operator==` (const `LogInfo` &lhs, const `LogInfo` &rhs) noexcept
Compares two LogInfos for equality.
- bool `operator!=` (const `LogInfo` &lhs, const `LogInfo` &rhs) noexcept
Compares two LogInfos for inequality.
- std::ostream & `operator<<` (std::ostream &os, const `LogInfo` &logInfo)
Prints a LogInfo to an ostream.

6.19.1 Detailed Description

Information about a log file.

Information about a log file that is extracted from the log file name.

Definition at line 20 of file `log_info.hpp`.

6.19.2 Constructor & Destructor Documentation

6.19.2.1 `LogInfo()`

```
cs::LogInfo::LogInfo ( )
```

Creates an uninitialized `LogInfo`.

Warning

Should only be used in order to be assigned with an initialized `LogInfo`; otherwise use the `create` static member function.

Definition at line 304 of file `log_info.cpp`.

6.19.3 Member Function Documentation

6.19.3.1 `create()`

```
cl::Expected< LogInfo > cs::LogInfo::create (
    cl::fs::Path filePath ) [static], [noexcept]
```

Creates a `LogInfo` from the given log file path.

Parameters

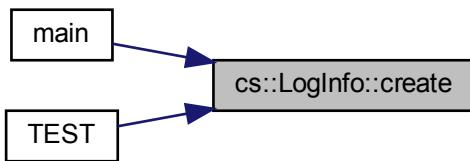
<i>logFilePath</i>	The log file path to create a LogInfo from.
--------------------	---

Returns

The [LogInfo](#) created or an error.

Definition at line 90 of file `log_info.cpp`.

Here is the caller graph for this function:



6.19.3.2 deleteLowVariance()

```
bool cs::LogInfo::deleteLowVariance( ) const [noexcept]
```

Read accessor for the delete low variance option.

Returns

true if delete low variance is active; false otherwise.

Definition at line 326 of file `log_info.cpp`.

6.19.3.3 deleteTooClose()

```
bool cs::LogInfo::deleteTooClose( ) const [noexcept]
```

Read accessor for the delete too close option.

Returns

true if delete too close is active; false otherwise.

Definition at line 324 of file `log_info.cpp`.

6.19.3.4 filterKind()

```
FilterKind cs::LogInfo::filterKind () const [noexcept]
```

Read accessor for the filter kind.

Returns

The filter kind.

Definition at line 335 of file log_info.cpp.

6.19.3.5 isInitialized()

```
bool cs::LogInfo::isInitialized () const [noexcept]
```

Checks whether this [LogInfo](#) is initialized.

Returns

true if this [LogInfo](#) is initialized; false otherwise.

Note

Will return true if this [LogInfo](#) was created with the create static member function.

Definition at line 339 of file log_info.cpp.

6.19.3.6 logFilePath()

```
const cl::fs::Path & cs::LogInfo::logFilePath () const [noexcept]
```

Read accessor for the log file path.

Returns

The log file path.

Definition at line 317 of file log_info.cpp.

Here is the caller graph for this function:



6.19.3.7 segmentationKind()

```
SegmentationKind cs::LogInfo::segmentationKind () const [noexcept]
```

Read accessor for the segmentation kind.

Returns

The segmentation kind.

Definition at line 328 of file log_info.cpp.

6.19.3.8 sensor()

```
std::uint64_t cs::LogInfo::sensor () const [noexcept]
```

Read accessor for the sensor.

Returns

The sensor.

Note

Will be the invalid sensor unless the log file is old.

Definition at line 337 of file log_info.cpp.

6.19.3.9 skipWindow()

```
bool cs::LogInfo::skipWindow () const [noexcept]
```

Read accessor for the skip window option.

Returns

true if skip window is active; false otherwise.

Definition at line 322 of file log_info.cpp.

6.19.3.10 `windowSize()`

```
std::uint64_t cs::LogInfo::windowSize ( ) const [noexcept]
```

Read accessor for the window size.

Returns

The window size.

Definition at line 333 of file log_info.cpp.

6.19.4 Friends And Related Function Documentation

6.19.4.1 `operator"!=`

```
bool operator!= (
    const LogInfo & lhs,
    const LogInfo & rhs ) [friend]
```

Compares two LogInfos for inequality.

Parameters

<i>lhs</i>	The left hand side operand.
<i>rhs</i>	The right hand side operand.

Returns

true if *lhs* and *rhs* are considered not equal; otherwise false.

Definition at line 287 of file log_info.cpp.

6.19.4.2 `operator<<`

```
std::ostream& operator<< (
    std::ostream & os,
    const LogInfo & logInfo ) [friend]
```

Prints a [LogInfo](#) to an ostream.

Parameters

<i>os</i>	The ostream to print to.
<i>logInfo</i>	The LogInfo to print.

Returns

```
os
```

Definition at line 292 of file log_info.cpp.

6.19.4.3 operator==

```
bool operator== (
    const LogInfo & lhs,
    const LogInfo & rhs ) [friend]
```

Compares two LogInfos for equality.

Parameters

<i>lhs</i>	The left hand side operand.
<i>rhs</i>	The right hand side operand.

Returns

true if *lhs* and *rhs* are considered equal; otherwise false.

Definition at line 264 of file log_info.cpp.

6.19.5 Member Data Documentation**6.19.5.1 invalidSensor**

```
const std::uint64_t cs::LogInfo::invalidSensor = UINT64_C(0xFFFFFFFFFFFFFF) [static]
```

Represents an invalid sensor.

Definition at line 25 of file log_info.hpp.

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

- compare_segmentation/include/[log_info.hpp](#)
- compare_segmentation/src/[log_info.cpp](#)

6.20 cs::LogLine Class Reference

A line out of a log file.

```
#include <log_line.hpp>
```

Public Member Functions

- std::uint64_t [segmentationPointCount\(\)](#) const noexcept
Read accessor for the segmentation point count.
- const [cl::fs::Path & filePath\(\)](#) const noexcept
Read accessor for the file path.
- [cl::Expected< std::string > fileName\(\)](#) const
Creates the short file name for the file in the log line.
- std::uint64_t [sensor\(\)](#) const noexcept
Read accessor for the sensor.

Static Public Member Functions

- static [cl::Expected< LogLine > parse\(const std::string &line\)](#)
Parses a [LogLine](#) out of a line of text read from a log file.

Static Public Attributes

- static const std::uint64_t [invalidSensor = UINT64_C\(0xFFFFFFFFFFFFFF\)](#)
Indicates an invalid sensor.

6.20.1 Detailed Description

A line out of a log file.

Definition at line 14 of file `log_line.hpp`.

6.20.2 Member Function Documentation

6.20.2.1 `fileName()`

`cl::Expected< std::string > cs::LogLine::fileName() const`

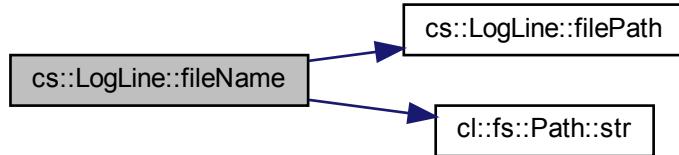
Creates the short file name for the file in the log line.

Returns

The resulting short file name or an error.

Definition at line 126 of file log_line.cpp.

Here is the call graph for this function:



6.20.2.2 filePath()

```
const cl::fs::Path & cs::LogLine::filePath ( ) const [noexcept]
```

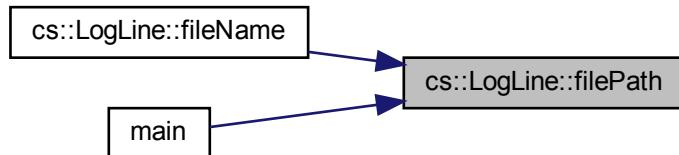
Read accessor for the file path.

Returns

The file path of the file in the log line.

Definition at line 124 of file log_line.cpp.

Here is the caller graph for this function:



6.20.2.3 parse()

```
cl::Expected< LogLine > cs::LogLine::parse (const std::string & line) [static]
```

Parses a [LogLine](#) out of a line of text read from a log file.

Parameters

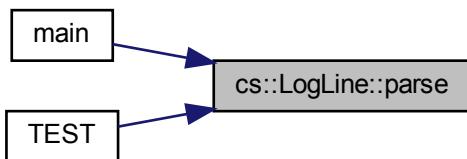
<i>line</i>	The line read.
-------------	----------------

Returns

The resulting [LogLine](#) or an error.

Definition at line 31 of file log_line.cpp.

Here is the caller graph for this function:

**6.20.2.4 segmentationPointCount()**

```
std::uint64_t cs::LogLine::segmentationPointCount ( ) const [noexcept]
```

Read accessor for the segmentation point count.

Returns

The segmentation point count.

Definition at line 119 of file log_line.cpp.

6.20.2.5 sensor()

```
std::uint64_t cs::LogLine::sensor ( ) const [noexcept]
```

Read acccessor for the sensor.

Returns

The sensor.

Note

Will only return a valid sensor if the [LogLine](#) is for a preprocessed file.

Definition at line 164 of file log_line.cpp.

6.20.3 Member Data Documentation

6.20.3.1 invalidSensor

```
const std::uint64_t cs::LogLine::invalidSensor = UINT64_C(0xFFFFFFFFFFFFFF) [static]
```

Indicates an invalid sensor.

Definition at line 19 of file log_line.hpp.

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

- compare_segmentation/include/[log_line.hpp](#)
- compare_segmentation/src/[log_line.cpp](#)

6.21 cm::ManualSegmentationPoint Class Reference

Type used to represent a manual segmentation point.

```
#include <manual_segmentation_point.hpp>
```

Public Member Functions

- [ManualSegmentationPoint](#) (std::uint32_t **hour**, std::uint32_t **minute**, std::uint32_t **second**, std::uint32_t **frame**)
Creates a ManualSegmentationPoint.
- std::uint32_t **hour** () const noexcept
Read accessor for the hour property.
- std::uint32_t **minute** () const noexcept
Read accessor for the minute property.
- std::uint32_t **second** () const noexcept
Read accessor for the second property.
- std::uint32_t **frame** () const noexcept
Read accessor for the frame property.
- std::uint64_t **asMilliseconds** () const noexcept
Converts this manual segmentation point into a millisecond representation.

Static Public Member Functions

- static std::unordered_map< [DataSetIdentifier](#), std::vector< [ManualSegmentationPoint](#) > > [readCsvFile](#) ()
Reads the CSV file of the manual segmentation points.
- static std::unordered_map< [DataSetIdentifier](#), std::vector< std::uint64_t > > [convertToHardwareTimestamps](#) (const std::unordered_map< [DataSetIdentifier](#), std::vector< [ManualSegmentationPoint](#) >> &[manualSegmentationPoints](#), const std::unordered_map< [cl::fs::Path](#), std::vector< std::uint64_t >> &[pythonResult](#))
Converts manualSegmentationPoints imported from the CSV file to hardware timestamps.

Friends

- bool `operator==` (const `ManualSegmentationPoint` &lhs, const `ManualSegmentationPoint` &rhs) noexcept
Compares two manual segmentation points for equality.
- bool `operator!=` (const `ManualSegmentationPoint` &lhs, const `ManualSegmentationPoint` &rhs) noexcept
Compares two manual segmentation points for inequality.
- std::ostream & `operator<<` (std::ostream &os, const `ManualSegmentationPoint` &manualSegmentationPoint)
Prints manualSegmentationPoint to os.

6.21.1 Detailed Description

Type used to represent a manual segmentation point.

Definition at line 17 of file `manual_segmentation_point.hpp`.

6.21.2 Constructor & Destructor Documentation

6.21.2.1 `ManualSegmentationPoint()`

```
cm::ManualSegmentationPoint::ManualSegmentationPoint (
    std::uint32_t hour,
    std::uint32_t minute,
    std::uint32_t second,
    std::uint32_t frame )
```

Creates a `ManualSegmentationPoint`.

Parameters

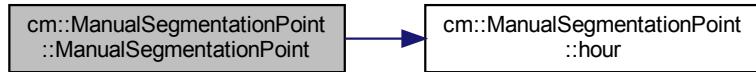
<code>hour</code>	The hour to use. Must be within [0,59].
<code>minute</code>	The minute to use. Must be within [0,59].
<code>second</code>	The second to use. Must be within [0,59].
<code>frame</code>	The frame to use. Must be within [0,29].

Exceptions

<code>cl::Exception</code>	if one of the arguments is out of bounds.
----------------------------	---

Definition at line 415 of file `manual_segmentation_point.cpp`.

Here is the call graph for this function:



6.21.3 Member Function Documentation

6.21.3.1 asMilliseconds()

```
std::uint64_t cm::ManualSegmentationPoint::asMilliseconds( ) const [noexcept]
```

Converts this manual segmentation point into a millisecond representation.

Returns

This manual segmentation point converted to milliseconds.

Definition at line 473 of file manual_segmentation_point.cpp.

6.21.3.2 convertToHardwareTimestamps()

```
std::unordered_map< DataSetIdentifier, std::vector< std::uint64_t > > cm::ManualSegmentationPoint::convertToHardwareTimestamps(
    const std::unordered_map< DataSetIdentifier, std::vector< ManualSegmentationPoint >> & manualSegmentationPoints,
    const std::unordered_map< cl::fs::Path, std::vector< std::uint64_t >> & pythonResult ) [static]
```

Converts `manualSegmentationPoints` imported from the CSV file to hardware timestamps.

Parameters

<code>manualSegmentationPoints</code>	The manual segmentation points that were read from the CSV file.
<code>pythonResult</code>	The result from Python of a (good) Configuration to use for the first segmentation point in order to convert the manual segmentation points to ones that are based on hardware timestamps.

Returns

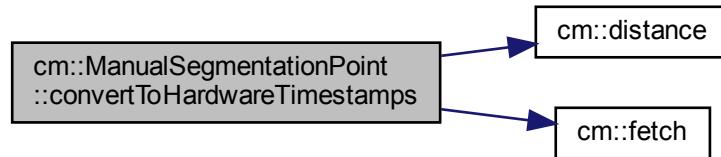
The resulting hardware timestamp based manual segmentation points.

Exceptions

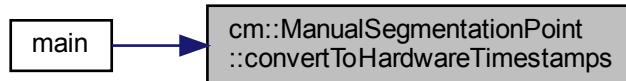
cl::Exception on error.

Definition at line 359 of file manual_segmentation_point.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



6.21.3.3 frame()

```
std::uint32_t cm::ManualSegmentationPoint::frame( ) const [noexcept]
```

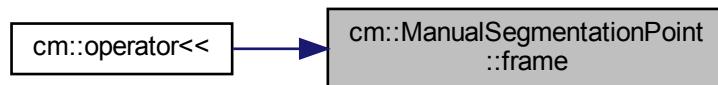
Read accessor for the frame property.

Returns

The frame within the second of this manual segmentation point.

Definition at line 468 of file manual_segmentation_point.cpp.

Here is the caller graph for this function:



6.21.3.4 hour()

```
std::uint32_t cm::ManualSegmentationPoint::hour( ) const [noexcept]
```

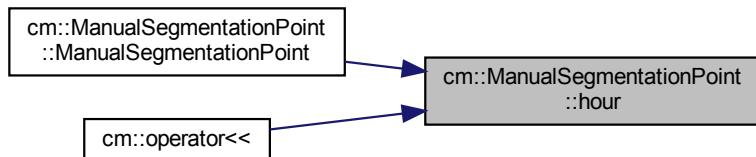
Read accessor for the hour property.

Returns

The hour.

Definition at line 456 of file manual_segmentation_point.cpp.

Here is the caller graph for this function:



6.21.3.5 minute()

```
std::uint32_t cm::ManualSegmentationPoint::minute ( ) const [noexcept]
```

Read accessor for the minute property.

Returns

The minute.

Definition at line 458 of file manual_segmentation_point.cpp.

Here is the caller graph for this function:



6.21.3.6 readCsvFile()

```
std::unordered_map< DataSetIdentifier, std::vector< ManualSegmentationPoint > > cm::ManualSegmentationPoint::readCsvFile ( ) [static]
```

Reads the CSV file of the manual segmentation points.

Returns

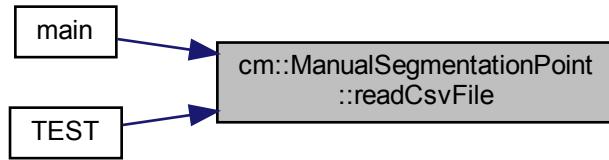
A map that maps the `DataSetIdentifier` enumerators to vectors of the corresponding manual segmentation points extracted from the CSV file.

Exceptions

<code>cl::Exception</code>	if parsing fails, CSV processing fails or the CSV file is missing.
----------------------------	--

Definition at line 209 of file manual_segmentation_point.cpp.

Here is the caller graph for this function:



6.21.3.7 second()

```
std::uint32_t cm::ManualSegmentationPoint::second() const [noexcept]
```

Read accessor for the second property.

Returns

The second.

Definition at line 463 of file manual_segmentation_point.cpp.

Here is the caller graph for this function:



6.21.4 Friends And Related Function Documentation

6.21.4.1 operator"!=

```
bool operator!=(
    const ManualSegmentationPoint & lhs,
    const ManualSegmentationPoint & rhs) [friend]
```

Compares two manual segmentation points for inequality.

Parameters

<i>lhs</i>	The first operand.
<i>rhs</i>	The second operand.

Returns

true if *lhs* is considered not equal to *rhs*; false otherwise.

Definition at line 187 of file manual_segmentation_point.cpp.

6.21.4.2 operator<<

```
std::ostream& operator<< (
    std::ostream & os,
    const ManualSegmentationPoint & manualSegmentationPoint ) [friend]
```

Prints *manualSegmentationPoint* to *os*.

Parameters

<i>os</i>	The ostream to print to
<i>manualSegmentationPoint</i>	The <i>ManualSegmentationPoint</i> to print.

Returns

os

Definition at line 194 of file manual_segmentation_point.cpp.

6.21.4.3 operator==

```
bool operator== (
    const ManualSegmentationPoint & lhs,
    const ManualSegmentationPoint & rhs ) [friend]
```

Compares two manual segmentation points for equality.

Parameters

<i>lhs</i>	The first operand.
<i>rhs</i>	The second operand.

Returns

true if `lhs` is considered equal to `rhs`; false otherwise.

Definition at line 179 of file manual_segmentation_point.cpp.

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

- confusion_matrix/include/manual_segmentation_point.hpp
- confusion_matrix/src/manual_segmentation_point.cpp

6.22 cl::fs::Path Class Reference

A filesystem path.

```
#include <path.hpp>
```

Public Member Functions

- [Path \(\)](#)
Default constructs an (empty) path.
- [PL_IMPLICIT Path \(std::string path\)](#)
Creates a path.
- [PL_IMPLICIT Path \(const char *path\)](#)
Creates a path.
- [bool exists \(\) const noexcept](#)
Checks if the path exists.
- [bool isFile \(\) const noexcept](#)
Checks if the path is a file.
- [bool isDirectory \(\) const noexcept](#)
Checks if the path is a directory.
- [const std::string & str \(\) const noexcept](#)
Read accessor for the underlying string.

Friends

- [std::ostream & operator<< \(std::ostream &os, const Path &path\)](#)
Prints a `Path` to an ostream.
- [bool operator< \(const Path &lhs, const Path &rhs\) noexcept](#)
Checks if `lhs` is less than `rhs`.
- [bool operator== \(const Path &lhs, const Path &rhs\) noexcept](#)
Equality compares `lhs` and `rhs`.

6.22.1 Detailed Description

A filesystem path.

Definition at line 14 of file path.hpp.

6.22.2 Constructor & Destructor Documentation

6.22.2.1 Path() [1/3]

```
cl::fs::Path::Path ( )
```

Default constructs an (empty) path.

Definition at line 37 of file path.cpp.

6.22.2.2 Path() [2/3]

```
cl::fs::Path::Path ( std::string path )
```

Creates a path.

Parameters

<i>path</i>	The string to construct from.
-------------	-------------------------------

Definition at line 39 of file path.cpp.

6.22.2.3 Path() [3/3]

```
cl::fs::Path::Path ( const char * path )
```

Creates a path.

Parameters

<i>path</i>	The string to construct from.
-------------	-------------------------------

Definition at line 46 of file path.cpp.

6.22.3 Member Function Documentation

6.22.3.1 exists()

```
bool cl::fs::Path::exists () const [noexcept]
```

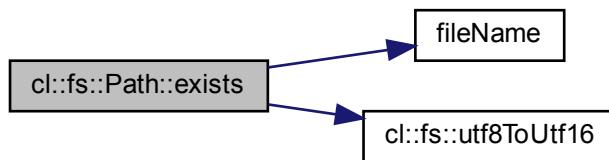
Checks if the path exists.

Returns

true if the path exists; otherwise false.

Definition at line 48 of file path.cpp.

Here is the call graph for this function:



6.22.3.2 isDirectory()

```
bool cl::fs::Path::isDirectory () const [noexcept]
```

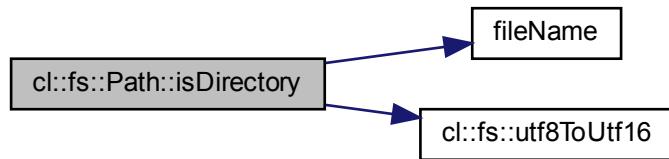
Checks if the path is a directory.

Returns

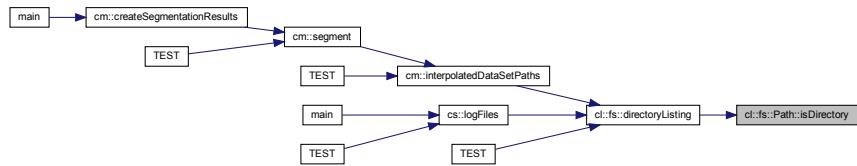
true if the path is a directory; otherwise false.

Definition at line 104 of file path.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



6.22.3.3 `isFile()`

```
bool cl::fs::Path::isFile ( ) const [noexcept]
```

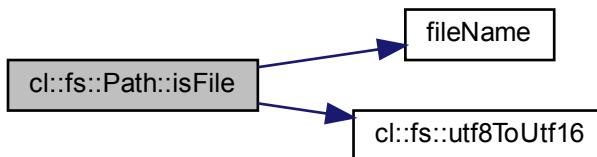
Checks if the path is a file.

Returns

true if the path is a file; otherwise false.

Definition at line 77 of file path.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



6.22.3.4 str()

```
const std::string & cl::fs::Path::str() const [noexcept]
```

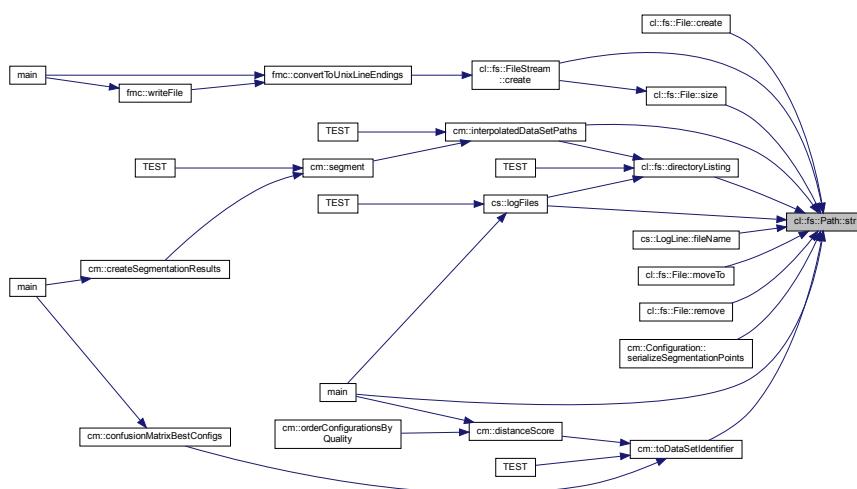
Read accessor for the underlying string.

Returns

The underlying string.

Definition at line 127 of file path.cpp.

Here is the caller graph for this function:



6.22.4 Friends And Related Function Documentation

6.22.4.1 operator<

```
bool operator< (
    const Path & lhs,
    const Path & rhs ) [friend]
```

Checks if `lhs` is less than `rhs`.

Parameters

<code>lhs</code>	The left hand side operand.
<code>rhs</code>	The right hand side operand.

Returns

true if lhs < rhs; otherwise false.

Definition at line 27 of file path.cpp.

6.22.4.2 operator<<

```
std::ostream& operator<< (
    std::ostream & os,
    const Path & path ) [friend]
```

Prints a [Path](#) to an ostream.

Parameters

<i>os</i>	the ostream to print to.
<i>path</i>	The path to print.

Returns

os

Definition at line 22 of file path.cpp.

6.22.4.3 operator==

```
bool operator== (
    const Path & lhs,
    const Path & rhs ) [friend]
```

Equality compares lhs and rhs.

Parameters

<i>lhs</i>	The left hand side operand.
<i>rhs</i>	The right hand side operand.

Returns

true if lhs and rhs are equal.

Definition at line 32 of file path.cpp.

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

- [csv_lib/include/cl/fs/path.hpp](#)
- [csv_lib/src/cl/fs/path.cpp](#)

6.23 cl::Process Class Reference

```
#include <process.hpp>
```

Public Types

- using `this_type = Process`

Public Member Functions

- `PL_NONCOPYABLE (Process)`
- `Process (this_type &&other) noexcept`
- `this_type & operator= (this_type &&other) noexcept`
- `~Process ()`
- `std::FILE * file () noexcept`
- `const std::FILE * file () const noexcept`

Static Public Member Functions

- static `Expected< Process > create (pl::string_view command, pl::string_view mode)`

6.23.1 Detailed Description

Definition at line 11 of file process.hpp.

6.23.2 Member Typedef Documentation

6.23.2.1 this_type

```
using cl::Process::this_type = Process
```

Definition at line 15 of file process.hpp.

6.23.3 Constructor & Destructor Documentation

6.23.3.1 Process()

```
cl::Process::Process (
    this_type && other ) [noexcept]
```

Definition at line 56 of file process.cpp.

6.23.3.2 ~Process()

```
cl::Process::~Process ( )
```

Definition at line 69 of file process.cpp.

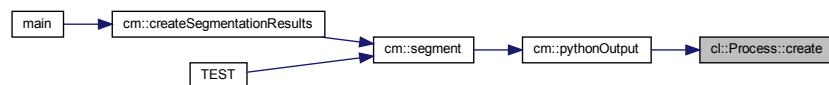
6.23.4 Member Function Documentation

6.23.4.1 create()

```
Expected< Process > cl::Process::create (
    pl::string_view command,
    pl::string_view mode ) [static]
```

Definition at line 36 of file process.cpp.

Here is the caller graph for this function:



6.23.4.2 file() [1/2]

```
const std::FILE* cl::Process::file ( ) const [noexcept]
```

6.23.4.3 file() [2/2]

```
const std::FILE * cl::Process::file ( ) [noexcept]
```

Definition at line 79 of file process.cpp.

6.23.4.4 operator=()

```
Process & cl::Process::operator= (
    this_type && other ) [noexcept]
```

Definition at line 61 of file process.cpp.

6.23.4.5 PL_NONCOPYABLE()

```
cl::Process::PL_NONCOPYABLE (
    Process )
```

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

- csv_lib/include/cl/process.hpp
- csv_lib/src/cl/process.cpp

Chapter 7

File Documentation

7.1 compare_segmentation/CMakeLists.txt File Reference

Functions

- `set (LIB_NAME compare_segmentation_lib) set(LIB_HEADERS include/csv_line.hpp include/data_set_info.hpp include/filter_kind.hpp include/log_files.hpp include/log_info.hpp include/log_line.hpp include/paths.hpp include/segmentation_kind.hpp) set(LIB_SOURCES src/csv_line.cpp src/data_set_info.cpp src/filter_kind.cpp src/log_files.cpp src/log_info.cpp src/log_line.cpp src/segmentation_kind.cpp) add_library($`

7.1.1 Function Documentation

7.1.1.1 set()

```
set (
    LIB_NAME compare_segmentation_lib )
```

Definition at line 2 of file CMakeLists.txt.

7.2 compare_segmentation/test/CMakeLists.txt File Reference

Functions

- `include (GoogleTest) set(TEST_NAME compare_segmentation_test) set(TEST_SOURCES csv_line_test.cpp data_set_info_test.cpp log_files_test.cpp log_info_test.cpp log_line_test.cpp main.cpp) add_executable($`

7.2.1 Function Documentation

7.2.1.1 include()

```
include (
    GoogleTest    )
```

Definition at line 1 of file CMakeLists.txt.

7.3 counting/CMakeLists.txt File Reference

Functions

- `set (LIB_NAME counting_lib) set(LIB_HEADERS include/above_threshold.hpp include/average_← comparison_value_calculator.hpp include/half_maximum_comparison_value_calculator.hpp include/is_← relevant.hpp include/percentage_of.hpp include/run_above_threshold.hpp) set(LIB_SOURCES src/above← _threshold.cpp src/average_comparison_value_calculator.cpp src/half_maximum_comparison_value← calculator.cpp src/run_above_threshold.cpp) add_library($`

7.3.1 Function Documentation

7.3.1.1 set()

```
set (
    LIB_NAME counting_lib )
```

Definition at line 2 of file CMakeLists.txt.

7.4 counting/test/CMakeLists.txt File Reference

Functions

- `include (GoogleTest) set(TEST_NAME counting_test) set(TEST_SOURCES above_threshold_test.cpp main.cpp percentage_of_test.cpp) add_executable($`

7.4.1 Function Documentation

7.4.1.1 include()

```
include (
    GoogleTest    )
```

Definition at line 1 of file CMakeLists.txt.

7.5 csv_lib/CMakeLists.txt File Reference

Functions

- `set (LIB_NAME csv_lib) set(LIB_HEADERS include/cl/fs/directory_listing.hpp include/cl/fs/file.hpp include/cl/fs/file_stream.hpp include/cl/fs/path.hpp include/cl/fs/sePARATOR.hpp include/cl/fs/windows.hpp include/cl/channel.hpp include/cl/column.hpp include/cl/data_point.hpp include/cl/data_set.hpp include/cl/dos2unix.hpp include/cl/error.hpp include/cl/exception.hpp include/cl/process.hpp include/cl/read_csv_file.hpp include/cl/s2n.hpp include/cl/sensor.hpp include/cl/to_string.hpp include/cl/use_unbuffered_io.hpp) set(LIB_SOURCES src/cl/fs/directory_listing.cpp src/cl/fs/file.cpp src/cl/fs/file_stream.cpp src/cl/fs/path.cpp src/cl/fs/windows.cpp src/cl/channel.cpp src/cl/data_point.cpp src/cl/data_set.cpp src/cl/dos2unix.cpp src/cl/error.cpp src/cl/exception.cpp src/cl/process.cpp src/cl/read_csv_file.cpp src/cl/sensor.cpp src/cl/use_unbuffered_io.cpp) add_library($`

7.5.1 Function Documentation

7.5.1.1 set()

```
set (  
    LIB_NAME csv_lib )
```

Definition at line 2 of file CMakeLists.txt.

7.6 csv_lib/test/CMakeLists.txt File Reference

Functions

- `include (GoogleTest) set(TEST_NAME csv_lib_test) set(TEST_SOURCES channel_test.cpp column_test.cpp data_point_test.cpp directory_listing_test.cpp error_test.cpp exception_test.cpp main.cpp sensor_test.cpp to_string_test.cpp read_csv_file_test.cpp data_set_test.cpp s2n_test.cpp) add_executable($`

7.6.1 Function Documentation

7.6.1.1 include()

```
include (  
    GoogleTest )
```

Definition at line 1 of file CMakeLists.txt.

7.7 fix_csv/CMakeLists.txt File Reference

Functions

- `set (LIB_NAME fix_mogasens_csv_lib) set(LIB_HEADERS include/adjust_hardware_timestamp.hpp include/convert_to_unix_line_endings.hpp include/create_backup_file.hpp include/delete_non_bosch_sensors.hpp include/delete_out_of_bounds_values.hpp include/remove_zeros_from_field.hpp include/restore_from_backup.hpp include/write_file.hpp) set(LIB_SOURCES src/adjust_hardware_timestamp.cpp src/convert_to_unix_line_endings.cpp src/create_backup_file.cpp src/delete_non_bosch_sensors.cpp src/delete_out_of_bounds_values.cpp src/remove_zeros_from_field.cpp src/restore_from_backup.cpp src/write_file.cpp) add_library($`

7.7.1 Function Documentation

7.7.1.1 `set()`

```
set (
    LIB_NAME fix_mogasens_csv_lib )
```

Definition at line 2 of file CMakeLists.txt.

7.8 fix_csv/test/CMakeLists.txt File Reference

Functions

- `include (GoogleTest) set(TEST_NAME fmc_test) set(TEST_SOURCES main.cpp remove_zeros_from_field_test.cpp adjust_hardware_timestamp_test.cpp) add_executable($`

7.8.1 Function Documentation

7.8.1.1 `include()`

```
include (
    GoogleTest )
```

Definition at line 1 of file CMakeLists.txt.

7.9 confusion_matrix/CMakeLists.txt File Reference

Functions

- `set (LIB_NAME confusion_matrix_lib) set(LIB_HEADERS include/closest_one.hpp include/configuration.hpp include/confusion_matrix.hpp include/confusion_matrix_best_configs.hpp include/create_segmentation_results.hpp include/csv_file_info.hpp include/data_set_identifier.hpp include/distance.hpp include/distance_score.hpp include/fetch.hpp include imu.hpp include/interpolated_data_set_paths.hpp include/manual_segmentation_point.hpp include/order_configurations_by_quality.hpp include/python_output.hpp include/segment.hpp include/split_string.hpp) set(LIB_SOURCES src/closest_one.cpp src/configuration.cpp src/confusion_matrix.cpp src/confusion_matrix_best_configs.cpp src/create_segmentation_results.cpp src/csv_file_info.cpp src/data_set_identifier.cpp src/distance.cpp src/distance_score.cpp src/imu.cpp src/interpolated_data_set_paths.cpp src/manual_segmentation_point.cpp src/order_configurations_by_quality.cpp src/python_output.cpp src/segment.cpp src/split_string.cpp) add_library($`

7.9.1 Function Documentation

7.9.1.1 set()

```
set (  
    LIB_NAME confusion_matrix_lib )
```

Definition at line 2 of file CMakeLists.txt.

7.10 confusion_matrix/test/CMakeLists.txt File Reference

Functions

- `include (GoogleTest) set(TEST_NAME confusion_matrix_test) set(TEST_SOURCES data_set_identifier_test.cpp interpolated_data_set_paths_test.cpp main.cpp manual_segmentation_point_test.cpp segment_test.cpp split_string_test.cpp) add_executable($`

7.10.1 Function Documentation

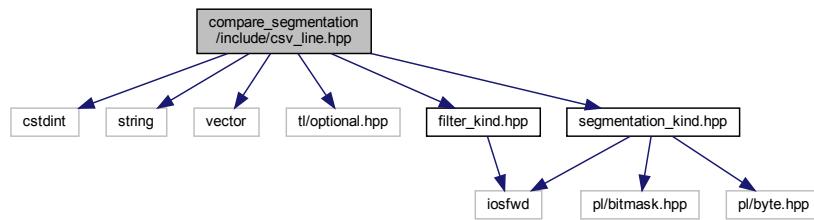
7.10.1.1 include()

```
include (  
    GoogleTest )
```

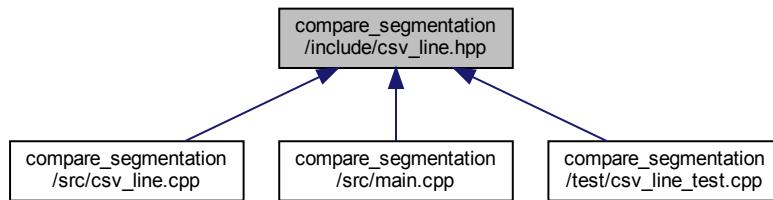
Definition at line 1 of file CMakeLists.txt.

7.11 compare_segmentation/include/csv_line.hpp File Reference

```
#include <cstdint>
#include <string>
#include <vector>
#include <tl/optional.hpp>
#include "filter_kind.hpp"
#include "segmentation_kind.hpp"
Include dependency graph for csv_line.hpp:
```



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



Classes

- class [cs::CsvLineBuilder](#)

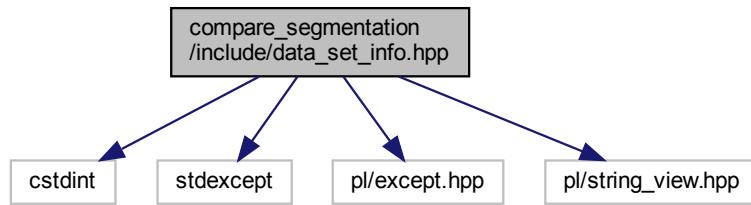
Builder for a CSV line.

Namespaces

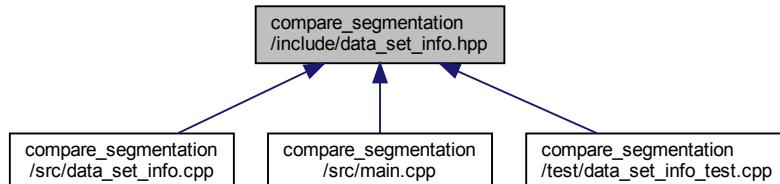
- [CS](#)

7.12 compare_segmentation/include/data_set_info.hpp File Reference

```
#include <cstdint>
#include <stdexcept>
#include <pl/except.hpp>
#include <pl/string_view.hpp>
Include dependency graph for data_set_info.hpp:
```



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



Classes

- struct `cs::data_set_info< Tag >`

Meta function for data set tags.

Namespaces

- `cs`

Macros

- `#define CS_SPECIALIZE_DATA_SET_INFO(tag, string, repetitionCount)`

Functions

- `cs::PL_DEFINE_EXCEPTION_TYPE` (NoSuchDataSetException, std::logic_error)
- `cs::CS_SPECIALIZE_DATA_SET_INFO` (Felix1, "11.17.39", 24)
- `cs::CS_SPECIALIZE_DATA_SET_INFO` (Felix2, "12.50.00", 20)
- `cs::CS_SPECIALIZE_DATA_SET_INFO` (Felix3, "13.00.09", 15)
- `cs::CS_SPECIALIZE_DATA_SET_INFO` (Marcelle1, "14.59.59", 10)
- `cs::CS_SPECIALIZE_DATA_SET_INFO` (Marcelle2, "15.13.22", 16)
- `cs::CS_SPECIALIZE_DATA_SET_INFO` (Marcelle3, "15.31.36", 18)
- `cs::CS_SPECIALIZE_DATA_SET_INFO` (Mike1, "14.07.33", 26)
- `cs::CS_SPECIALIZE_DATA_SET_INFO` (Mike2, "14.14.32", 22)
- `cs::CS_SPECIALIZE_DATA_SET_INFO` (Mike3, "14.20.28", 18)
- `cs::CS_SPECIALIZE_DATA_SET_INFO` (Andre1, "Andre_liegestuetzen1", 27)
- `cs::CS_SPECIALIZE_DATA_SET_INFO` (Andre2, "Andre_liegestuetzen2", 20)
- `cs::CS_SPECIALIZE_DATA_SET_INFO` (Andre3, "Andre_liegestuetzen3", 17)
- `cs::CS_SPECIALIZE_DATA_SET_INFO` (AndreSquats1, "Andre_Squats", 30)
- `cs::CS_SPECIALIZE_DATA_SET_INFO` (AndreSquats2, "Andre_Squats2", 49)
- `cs::CS_SPECIALIZE_DATA_SET_INFO` (Jan1, "Jan_liegestuetzen1", 25)
- `cs::CS_SPECIALIZE_DATA_SET_INFO` (Jan2, "Jan_liegestuetzen2", 19)
- `cs::CS_SPECIALIZE_DATA_SET_INFO` (Jan3, "Jan_liegestuetzen3", 13)
- `cs::CS_SPECIALIZE_DATA_SET_INFO` (Lucas1, "Lukas_liegestuetzen1", 24)
- `cs::CS_SPECIALIZE_DATA_SET_INFO` (Lucas2, "Lukas_liegestuetzen2", 19)
- `cs::CS_SPECIALIZE_DATA_SET_INFO` (Lucas3, "Lukas_liegestuetzen3", 11)
- `std::uint64_t cs::repetitionCount` (pl::string_view dataSet)

Fetches the repetition count for a given data set identified by its string.

7.12.1 Macro Definition Documentation

7.12.1.1 CS_SPECIALIZE_DATA_SET_INFO

```
#define CS_SPECIALIZE_DATA_SET_INFO( \
    tag, \
    string, \
    repetitionCount )
```

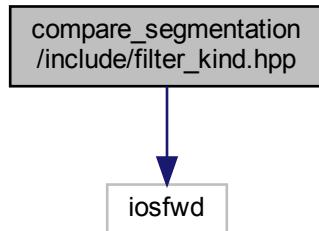
Value:

```
struct tag { \
}; \
constexpr bool contains##tag(pl::string_view other) \
{ \
    return other.contains(string); \
} \
template<> \
struct data_set_info<tag> { \
    static constexpr pl::string_view text      = string; \
    static constexpr std::uint64_t   repetitions = UINT64_C(repetitionCount); \
}
```

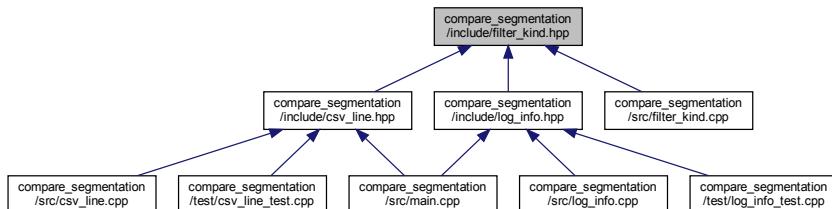
Definition at line 23 of file data_set_info.hpp.

7.13 compare_segmentation/include/filter_kind.hpp File Reference

```
#include <iostream>
Include dependency graph for filter_kind.hpp:
```



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



Namespaces

- `cs`

Enumerations

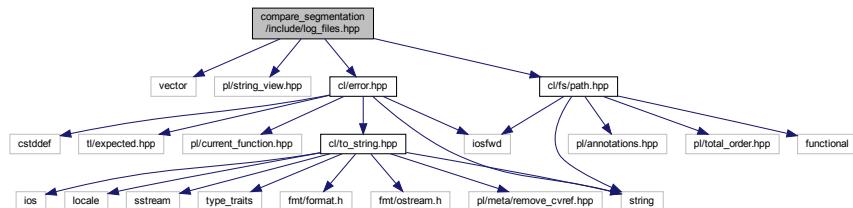
- enum `cs::FilterKind` { `cs::FilterKind::Butterworth`, `cs::FilterKind::MovingAverage` }
- Type for the different kinds of filters.*

Functions

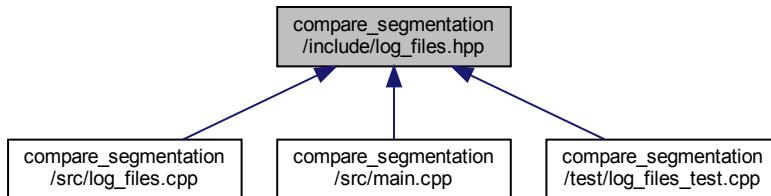
- `std::ostream & cs::operator<< (std::ostream &os, FilterKind filterKind)`
- Prints a FilterKind to an ostream.*

7.14 compare_segmentation/include/log_files.hpp File Reference

```
#include <vector>
#include <pl/string_view.hpp>
#include <cl/error.hpp>
#include <cl/fs/path.hpp>
Include dependency graph for log_files.hpp:
```



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



Namespaces

- `cs`

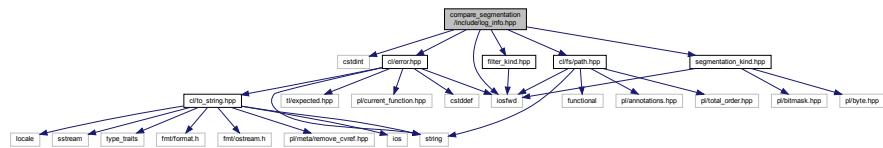
Functions

- `cl::Expected< std::vector< cl::fs::Path > > cs::logFiles (pl::string_view directoryPath)`
Fetches the paths to the log files in the given directory.

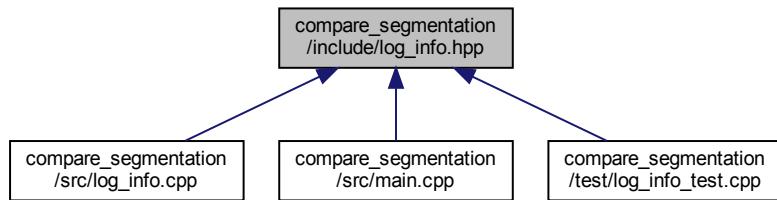
7.15 compare_segmentation/include/log_info.hpp File Reference

```
#include <cstdint>
#include <iostfwd>
#include <cl/error.hpp>
#include <cl/fs/path.hpp>
#include "filter_kind.hpp"
```

```
#include "segmentation_kind.hpp"
Include dependency graph for log_info.hpp:
```



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



Classes

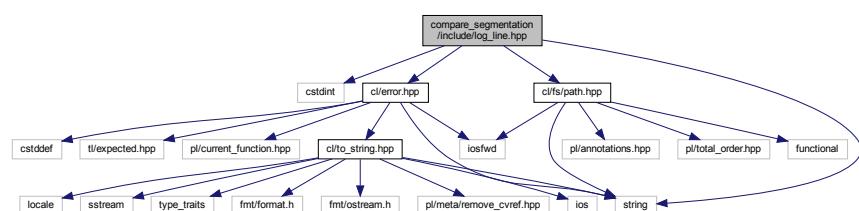
- class [cs::LogInfo](#)
Information about a log file.

Namespaces

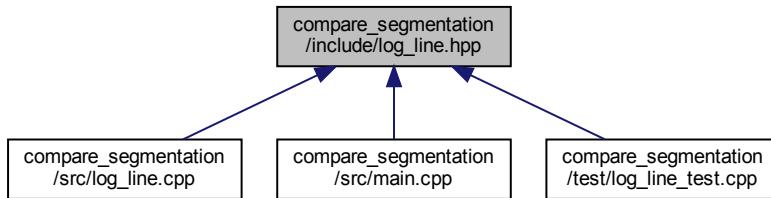
- [cs](#)

7.16 compare_segmentation/include/log_line.hpp File Reference

```
#include <cstdint>
#include <string>
#include "cl/error.hpp"
#include "cl/fs/path.hpp"
Include dependency graph for log_line.hpp:
```



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



Classes

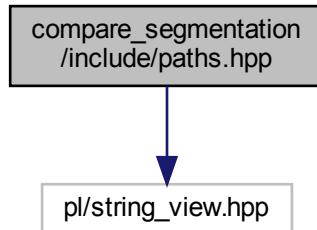
- class [cs::LogLine](#)
A line out of a log file.

Namespaces

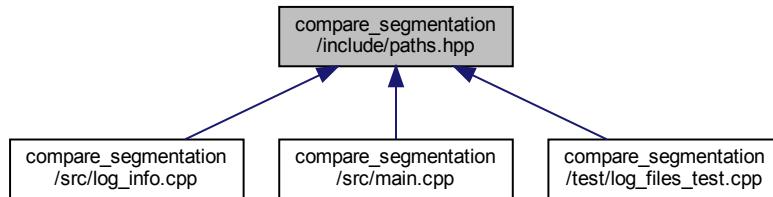
- [cs](#)

7.17 compare_segmentation/include/paths.hpp File Reference

```
#include <pl/string_view.hpp>
Include dependency graph for paths.hpp:
```



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



Namespaces

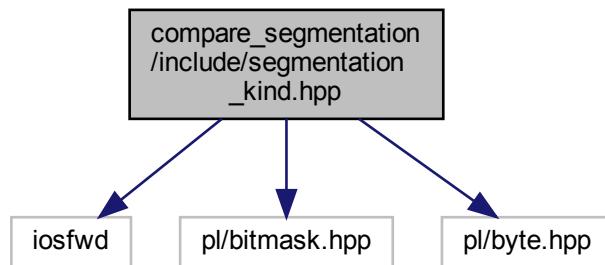
- `cs`

Variables

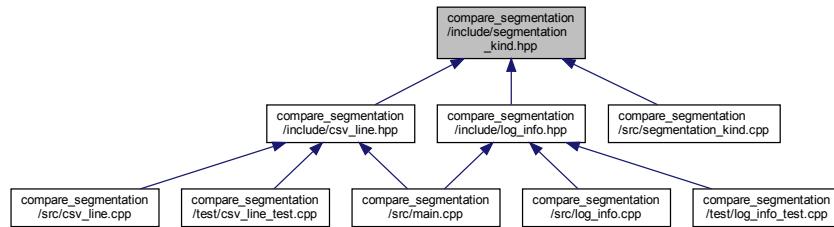
- `constexpr pl::string_view cs::logPath {"segmentation_comparison/logs"}`
Relative path to the directory containing the preprocessed log files.
- `constexpr pl::string_view cs::oldLogPath {"segmentation_comparison/logs/old"}`
Relative path to the directory containing the old log files.

7.18 compare_segmentation/include/segmentation_kind.hpp File Reference

```
#include <iostream>
#include <pl/bitmask.hpp>
#include <pl/byte.hpp>
Include dependency graph for segmentation_kind.hpp:
```



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



Namespaces

- [cs](#)

Enumerations

- enum [cs::SegmentationKind](#) : pl::byte { [cs::SegmentationKind::Minima](#) = 0b0000'0001, [cs::SegmentationKind::Maxima](#) = 0b0000'0010, [cs::SegmentationKind::Both](#) = Minima | Maxima }

The segmentation kind.

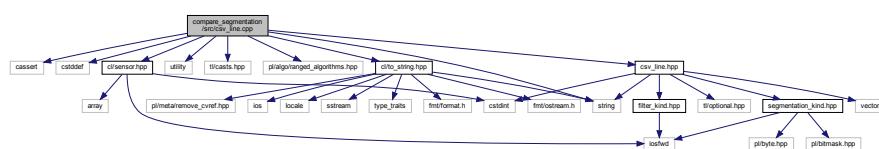
Functions

- std::ostream & [cs::operator<<](#) (std::ostream &os, SegmentationKind segmentationKind)

Prints a SegmentationKind to an ostream.

7.19 compare_segmentation/src/csv_line.cpp File Reference

```
#include <cassert>
#include <cstddef>
#include <string>
#include <utility>
#include <tl/casts.hpp>
#include <pl/algo/ranged_algorithms.hpp>
#include "cl/sensor.hpp"
#include "cl/to_string.hpp"
#include "csv_line.hpp"
Include dependency graph for csv_line.cpp:
```

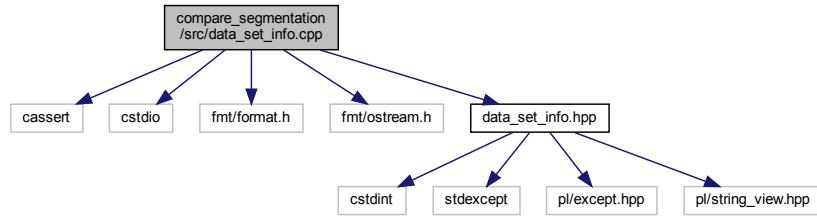


Namespaces

- [cs](#)

7.20 compare_segmentation/src/data_set_info.cpp File Reference

```
#include <cassert>
#include <cstdio>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include "data_set_info.hpp"
Include dependency graph for data_set_info.cpp:
```



Namespaces

- [cs](#)

Functions

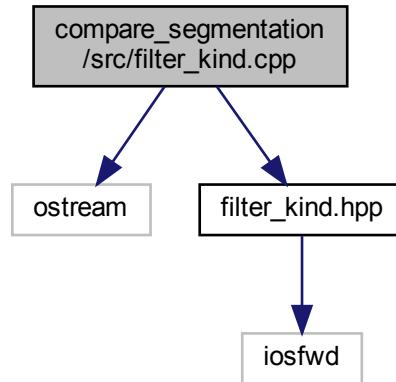
- `std::uint64_t cs::repetitionCount (pl::string_view dataSet)`

Fetches the repetition count for a given data set identified by its string.

7.21 compare_segmentation/src/filter_kind.cpp File Reference

```
#include <iostream>
#include "filter_kind.hpp"
```

Include dependency graph for filter_kind.cpp:



Namespaces

- [cs](#)

Functions

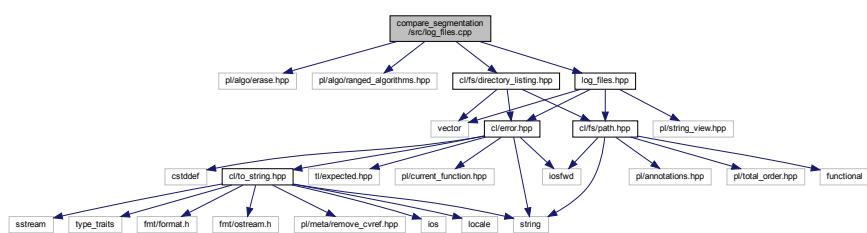
- `std::ostream & cs::operator<< (std::ostream &os, FilterKind filterKind)`

Prints a FilterKind to an ostream.

7.22 compare_segmentation/src/log_files.cpp File Reference

```
#include <pl/algo/erase.hpp>
#include <pl/algo/ranged_algorithms.hpp>
#include <cl/fs/directory_listing.hpp>
#include "log_files.hpp"
```

Include dependency graph for log_files.cpp:



Namespaces

- CS

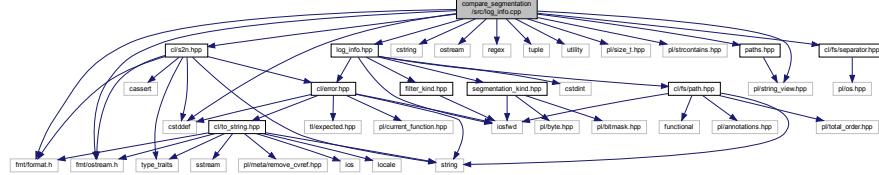
Functions

- `cl::Expected< std::vector< cl::fs::Path > > cs::logFiles (pl::string_view directoryPath)`
Fetches the paths to the log files in the given directory.

7.23 compare_segmentation/src/log_info.cpp File Reference

```
#include <cstddef>
#include <cstring>
#include <iostream>
#include <regex>
#include <tuple>
#include <utility>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <pl/size_t.hpp>
#include <pl/strcontains.hpp>
#include <pl/string_view.hpp>
#include "cl/fs/sePARATOR.hpp"
#include "cl/s2n.hpp"
#include "log_info.hpp"
#include "paths.hpp"
Include dependency graph for log_info.cpp:
```

Include dependency graph for log_info.cpp:



Namespaces

- cs

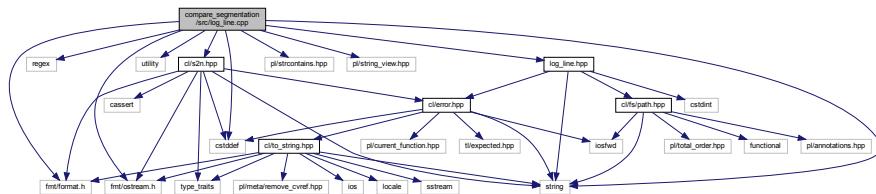
Functions

- bool `cs::operator==` (const LogInfo &lhs, const LogInfo &rhs) noexcept
 - bool `cs::operator!=` (const LogInfo &lhs, const LogInfo &rhs) noexcept
 - std::ostream & `cs::operator<<` (std::ostream &os, const LogInfo &logInfo)

7.24 compare_segmentation/src/log_line.cpp File Reference

```
#include <cstdint>
#include <regex>
#include <string>
#include <utility>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <pl/strcontains.hpp>
#include <pl/string_view.hpp>
#include "cl/s2n.hpp"
#include "log_line.hpp"
Include dependency graph for log_line.cpp:
```

Include dependency graph for log_line.cpp:



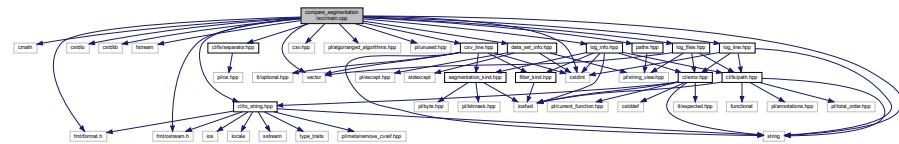
Namespaces

- CS

7.25 compare_segmentation/src/main.cpp File Reference

```
#include <cmath>
#include <cstdint>
#include <cstdio>
#include <cstdlib>
#include <fstream>
#include <string>
#include <vector>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <csv.hpp>
#include <pl/algo/ranged_algorithms.hpp>
#include <pl/unused.hpp>
#include "cl/fs/sePARATOR.hpp"
#include "cl/to_string.hpp"
#include "csv_line.hpp"
#include "data_set_info.hpp"
#include "log_files.hpp"
#include "log_info.hpp"
#include "log_line.hpp"
```

```
#include "paths.hpp"
Include dependency graph for main.cpp:
```



Functions

- int main (int argc, char *argv[])

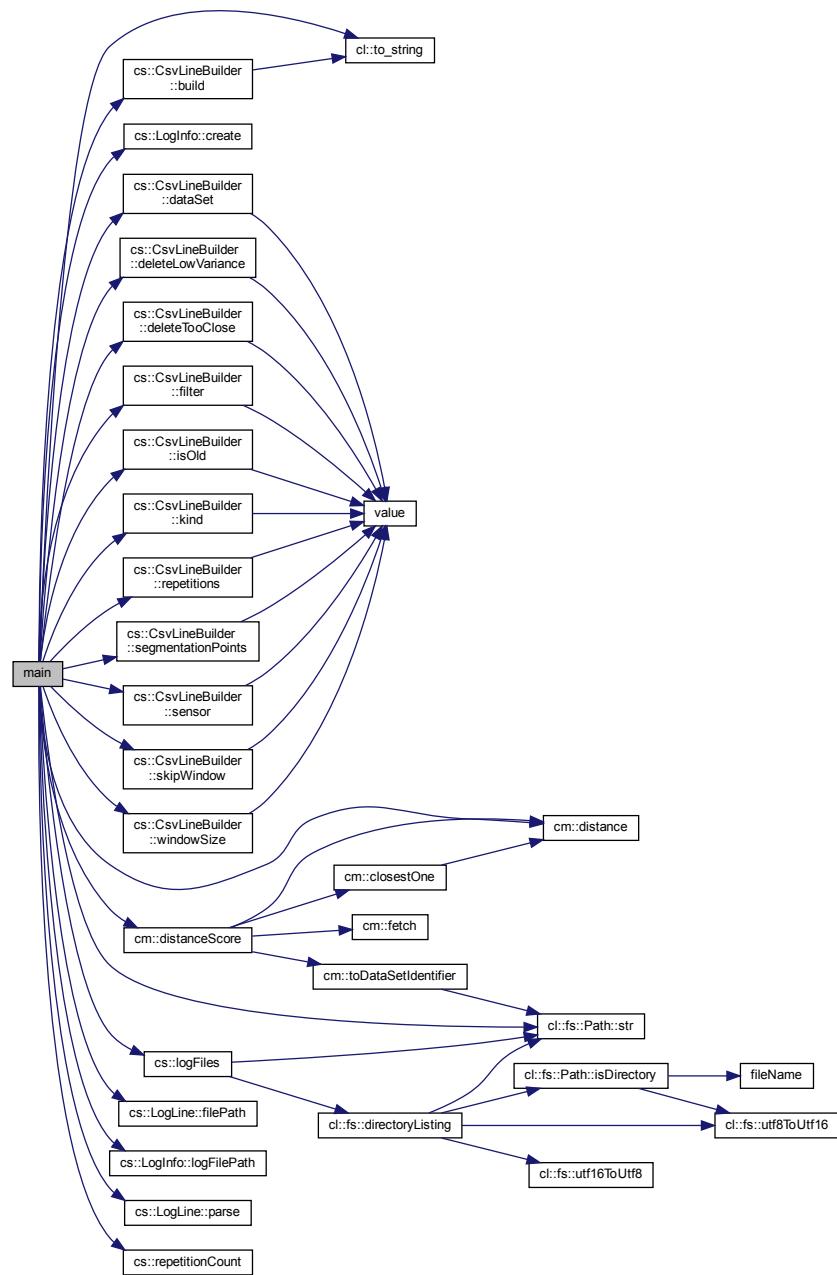
7.25.1 Function Documentation

7.25.1.1 main()

```
int main ( int argc,  
           char * argv[ ] )
```

Definition at line 28 of file main.cpp.

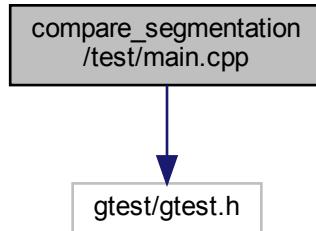
Here is the call graph for this function:



7.26 compare_segmentation/test/main.cpp File Reference

```
#include "gtest/gtest.h"
```

Include dependency graph for main.cpp:



Functions

- int `main` (int argc, char *argv[])

7.26.1 Function Documentation

7.26.1.1 `main()`

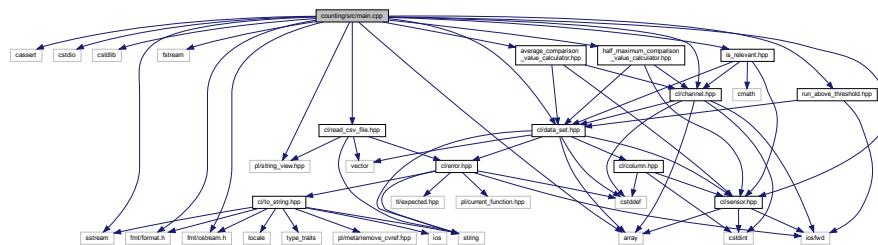
```
int main (
    int argc,
    char * argv[] )
```

Definition at line 3 of file main.cpp.

7.27 counting/src/main.cpp File Reference

```
#include <cassert>
#include <cstdio>
#include <cstdlib>
#include <array>
#include <fstream>
#include <sstream>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <pl/string_view.hpp>
#include "cl/channel.hpp"
#include "cl/data_set.hpp"
#include "cl/read_csv_file.hpp"
#include "cl/sensor.hpp"
#include "average_comparison_value_calculator.hpp"
#include "half_maximum_comparison_value_calculator.hpp"
```

```
#include "is_relevant.hpp"
#include "run_above_threshold.hpp"
Include dependency graph for main.cpp:
```



Functions

- int `main` (int argc, char *argv[])

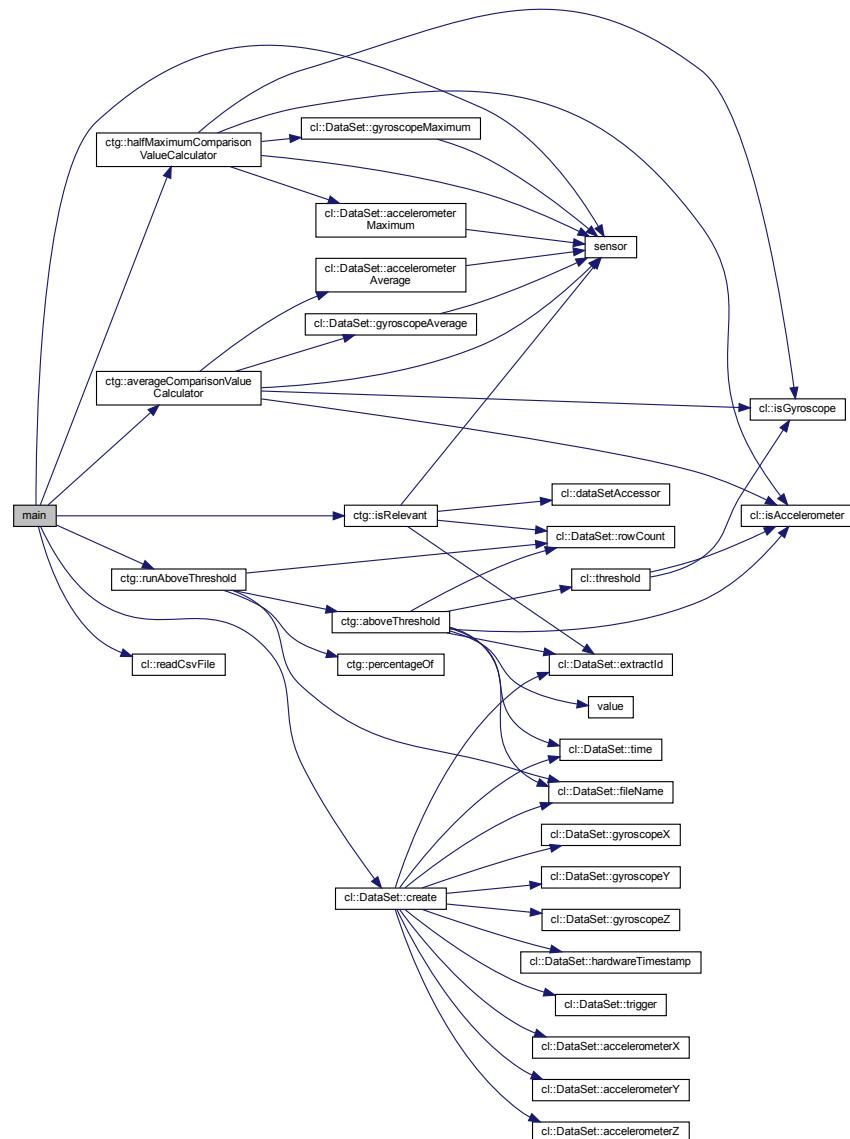
7.27.1 Function Documentation

7.27.1.1 `main()`

```
int main (
    int argc,
    char * argv[ ] )
```

Definition at line 24 of file `main.cpp`.

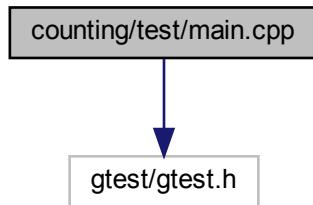
Here is the call graph for this function:



7.28 counting/test/main.cpp File Reference

```
#include "gtest/gtest.h"
```

Include dependency graph for main.cpp:



Functions

- int `main` (int argc, char *argv[])

7.28.1 Function Documentation

7.28.1.1 `main()`

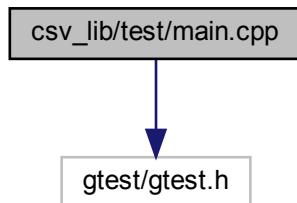
```
int main (
    int argc,
    char * argv[] )
```

Definition at line 3 of file main.cpp.

7.29 csv_lib/test/main.cpp File Reference

```
#include "gtest/gtest.h"
```

Include dependency graph for main.cpp:



Functions

- int main (int argc, char *argv[])

7.29.1 Function Documentation

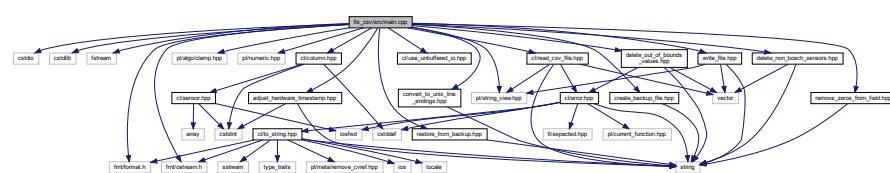
7.29.1.1 main()

```
int main ( int argc,  
           char * argv[ ] )
```

Definition at line 3 of file main.cpp.

7.30 fix_csv/src/main.cpp File Reference

```
#include <cstdio>
#include <cstdlib>
#include <fstream>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <pl/algo/clamp.hpp>
#include <pl/numeric.hpp>
#include <pl/string_view.hpp>
#include "cl/column.hpp"
#include "cl/read_csv_file.hpp"
#include "cl/use_unbuffered_io.hpp"
#include "adjust_hardware_timestamp.hpp"
#include "convert_to_unix_line_endings.hpp"
#include "create_backup_file.hpp"
#include "delete_non_bosch_sensors.hpp"
#include "delete_out_of_bounds_values.hpp"
#include "remove_zeros_from_field.hpp"
#include "restore_from_backup.hpp"
#include "write_file.hpp"
Include dependency graph for main.cpp:
```



Functions

- int main (int argc, char *argv[])

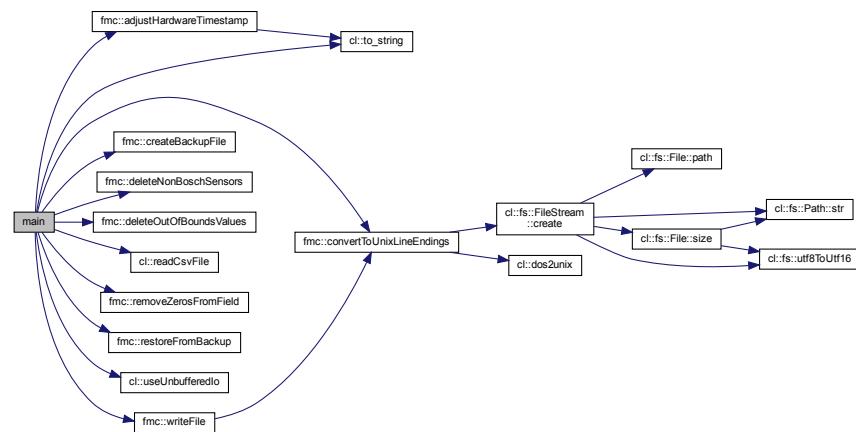
7.30.1 Function Documentation

7.30.1.1 main()

```
int main (
    int argc,
    char * argv[] )
```

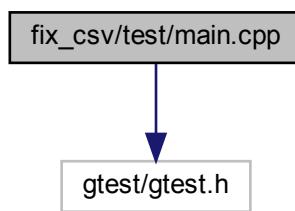
Definition at line 26 of file main.cpp.

Here is the call graph for this function:



7.31 fix_csv/test/main.cpp File Reference

```
#include "gtest/gtest.h"
Include dependency graph for main.cpp:
```



Functions

- int main (int argc, char *argv[])

7.31.1 Function Documentation

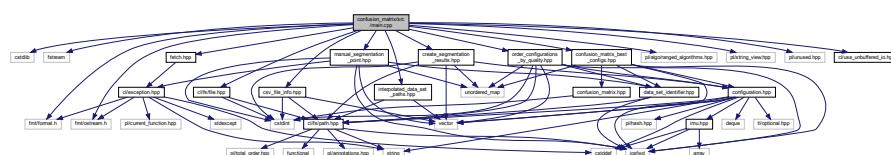
7.31.1.1 main()

```
int main ( int argc, char * argv[ ] )
```

Definition at line 3 of file main.cpp.

7.32 confusion_matrix/src/main.cpp File Reference

```
#include <cstdlib>
#include <fstream>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <pl/algo/ranged_algorithms.hpp>
#include <pl/string_view.hpp>
#include <pl/unused.hpp>
#include <cl/fs/file.hpp>
#include <cl/use_unbuffered_io.hpp>
#include "confusion_matrix_best_configs.hpp"
#include "create_segmentation_results.hpp"
#include "csv_file_info.hpp"
#include "fetch.hpp"
#include "interpolated_data_set_paths.hpp"
#include "manual_segmentation_point.hpp"
#include "order_configurations_by_quality.hpp"
Include dependency graph for main.cpp:
```



Macros

- #define SORT PRINT(kind)

Functions

- int main (int argc, char *argv[1])

7.32.1 Macro Definition Documentation

7.32.1.1 SORT_PRINT

```
#define SORT_PRINT( kind )
```

Value:

```
pl::algo::sort(bestConfigs, cm::kind##Sorter);
print("{}\n", #kind);
for (const cm::ConfigWithTotalConfusionMatrix& cur : bestConfigs) {
    print("{}\n", cur);
}
print("\nBest configuration (" #kind "): {}\n", bestConfigs.front())
```

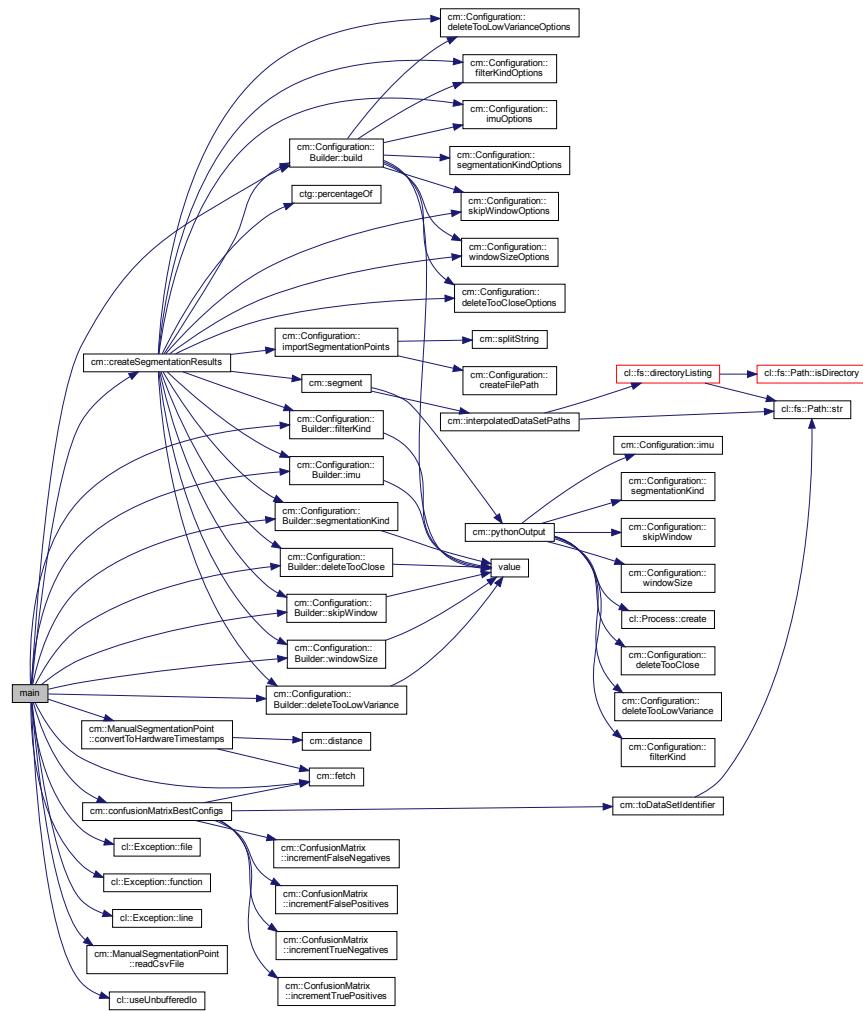
7.32.2 Function Documentation

7.32.2.1 main()

```
int main (
    int argc,
    char * argv[] )
```

Definition at line 36 of file main.cpp.

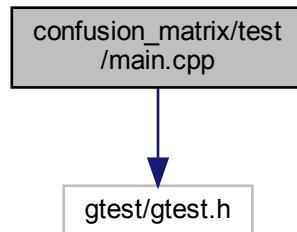
Here is the call graph for this function:



7.33 confusion_matrix/test/main.cpp File Reference

```
#include "gtest/gtest.h"
```

Include dependency graph for main.cpp:



Functions

- int [main](#) (int argc, char *argv[])

7.33.1 Function Documentation

7.33.1.1 [main\(\)](#)

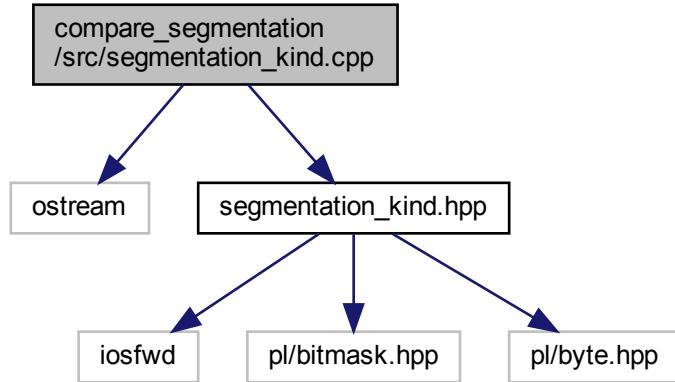
```
int main (
    int argc,
    char * argv[] )
```

Definition at line 3 of file main.cpp.

7.34 compare_segmentation/src/segmentation_kind.cpp File Reference

```
#include <iostream>
#include "segmentation_kind.hpp"
```

Include dependency graph for segmentation_kind.cpp:



Namespaces

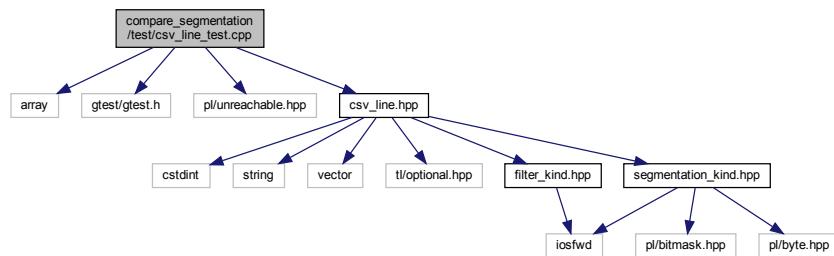
- `cs`

Functions

- `std::ostream & cs::operator<< (std::ostream &os, SegmentationKind segmentationKind)`
Prints a SegmentationKind to an ostream.

7.35 compare_segmentation/test/csv_line_test.cpp File Reference

```
#include <array>
#include "gtest/gtest.h"
#include <pl/unreachable.hpp>
#include "csv_line.hpp"
Include dependency graph for csv_line_test.cpp:
```



Functions

- [TEST](#) (CsvLine, shouldWork)

7.35.1 Function Documentation

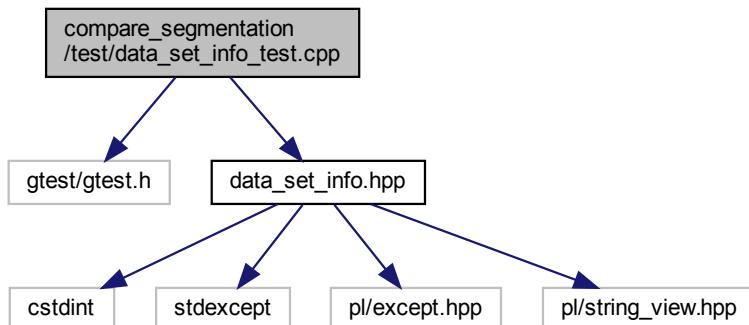
7.35.1.1 TEST()

```
TEST (
    CsvLine ,
    shouldWork )
```

Definition at line 30 of file csv_line_test.cpp.

7.36 compare_segmentation/test/data_set_info_test.cpp File Reference

```
#include "gtest/gtest.h"
#include "data_set_info.hpp"
Include dependency graph for data_set_info_test.cpp:
```



Functions

- [TEST](#) (dataSetInfo, repetitionCount)

7.36.1 Function Documentation

7.36.1.1 TEST()

```
TEST (
    dataSetInfo ,
    repetitionCount )
```

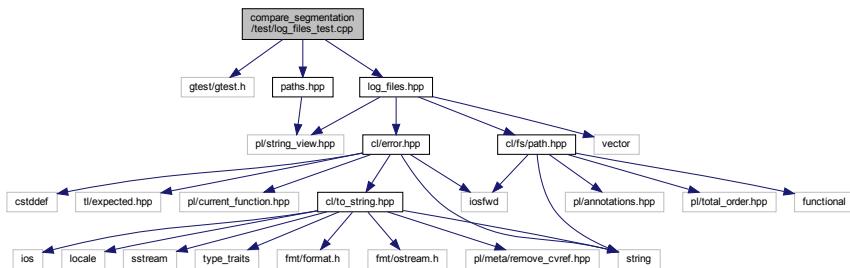
Definition at line 5 of file `data_set_info_test.cpp`.

Here is the call graph for this function:



7.37 compare_segmentation/test/log_files_test.cpp File Reference

```
#include "gtest/gtest.h"
#include <log_files.hpp>
#include <paths.hpp>
Include dependency graph for log_files_test.cpp:
```



Functions

- [TEST](#) (`logFiles`, `shouldFindLogFiles`)
- [TEST](#) (`logFiles`, `shouldFindOldLogFiles`)
- [TEST](#) (`logFiles`, `shouldNotFindGarbage`)

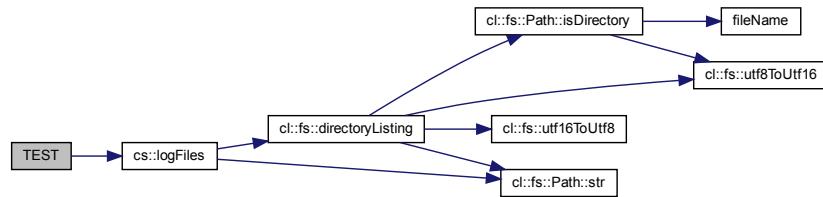
7.37.1 Function Documentation

7.37.1.1 TEST() [1/3]

```
TEST (
    logFiles ,
    shouldFindLogFiles )
```

Definition at line 6 of file log_files_test.cpp.

Here is the call graph for this function:

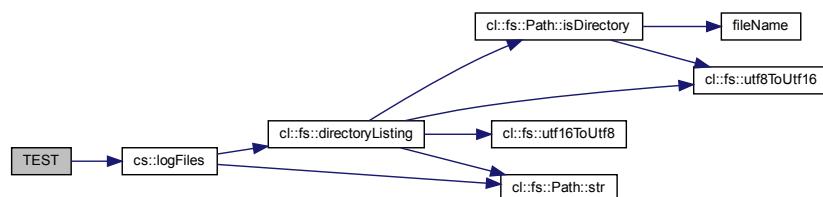


7.37.1.2 TEST() [2/3]

```
TEST (
    logFiles ,
    shouldFindOldLogFiles )
```

Definition at line 23 of file log_files_test.cpp.

Here is the call graph for this function:

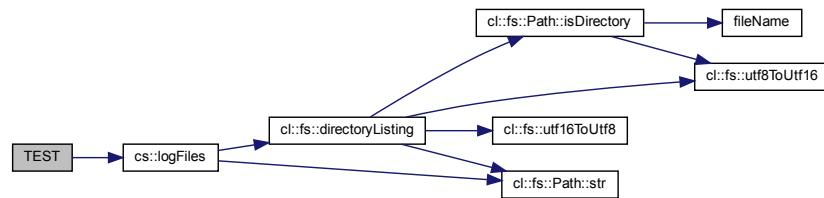


7.37.1.3 TEST() [3/3]

```
TEST (
    logFiles ,
    shouldNotFindGarbage
)
```

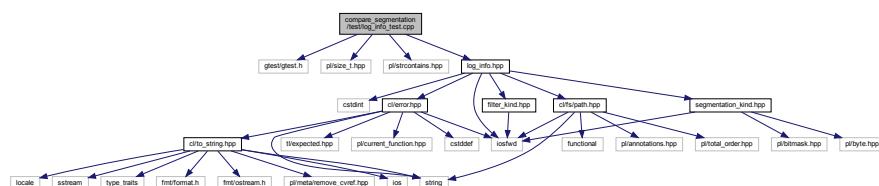
Definition at line 40 of file log_files_test.cpp.

Here is the call graph for this function:



7.38 compare_segmentation/test/log_info_test.cpp File Reference

```
#include "gtest/gtest.h"
#include <pl/size_t.hpp>
#include <pl/strcontains.hpp>
#include "log_info.hpp"
Include dependency graph for log_info_test.cpp:
```



Functions

- [TEST \(LogInfo, shouldWork\)](#)
- [TEST \(LogInfo, shouldWork2\)](#)
- [TEST \(LogInfo, shouldWork3\)](#)
- [TEST \(LogInfo, shouldWork4\)](#)
- [TEST \(LogInfo, shouldWork5\)](#)
- [TEST \(LogInfo, shouldWork6\)](#)
- [TEST \(LogInfo, shouldWork7\)](#)
- [TEST \(LogInfo, shouldWork8\)](#)
- [TEST \(LogInfo, shouldWork9\)](#)
- [TEST \(LogInfo, shouldWorkWithOldPath\)](#)
- [TEST \(LogInfo, shouldWorkWithOldPath2\)](#)
- [TEST \(LogInfo, shouldResultInErrorIfLogFilePathIsTooShort\)](#)

- [TEST](#) (LogInfo, shouldFailIfSkipWindowIsInvalid)
- [TEST](#) (LogInfo, shouldFailIfDeleteTooCloseIsInvalid)
- [TEST](#) (LogInfo, shouldFailIfDeleteTooLowVarianceIsInvalid)
- [TEST](#) (LogInfo, shouldFailIfSegmentationKindIsInvalid)
- [TEST](#) (LogInfo, shouldFailIfWindowSizeIsInvalid)
- [TEST](#) (LogInfo, shouldFailIfFilterIsInvalid)
- [TEST](#) (LogInfo, shouldCreateUninitializedObjectWhenDefaultConstructorIsCalled)

7.38.1 Function Documentation

7.38.1.1 TEST() [1/19]

```
TEST (
    LogInfo ,
    shouldCreateUninitializedObjectWhenDefaultConstructorIsCalled )
```

Definition at line 388 of file log_info_test.cpp.

7.38.1.2 TEST() [2/19]

```
TEST (
    LogInfo ,
    shouldFailIfDeleteTooCloseIsInvalid )
```

Definition at line 341 of file log_info_test.cpp.

Here is the call graph for this function:



7.38.1.3 TEST() [3/19]

```
TEST (
    LogInfo ,
    shouldFailIfDeleteTooLowVarianceIsInvalid )
```

Definition at line 350 of file log_info_test.cpp.

Here is the call graph for this function:

**7.38.1.4 TEST() [4/19]**

```
TEST (
    LogInfo ,
    shouldFailIfFilterIsInvalid )
```

Definition at line 379 of file log_info_test.cpp.

Here is the call graph for this function:



7.38.1.5 TEST() [5/19]

```
TEST (
    LogInfo ,
    shouldFailIfSegmentationKindIsInvalid )
```

Definition at line 359 of file log_info_test.cpp.

Here is the call graph for this function:



7.38.1.6 TEST() [6/19]

```
TEST (
    LogInfo ,
    shouldFailIfSkipWindowIsInvalid )
```

Definition at line 332 of file log_info_test.cpp.

Here is the call graph for this function:



7.38.1.7 TEST() [7/19]

```
TEST (
    LogInfo ,
    shouldFailIfWindowSizeIsValid )
```

Definition at line 368 of file log_info_test.cpp.

Here is the call graph for this function:

**7.38.1.8 TEST() [8/19]**

```
TEST (
    LogInfo ,
    shouldResultInErrorIfFilePathIsTooShort )
```

Definition at line 325 of file log_info_test.cpp.

Here is the call graph for this function:



7.38.1.9 TEST() [9/19]

```
TEST (
    LogInfo ,
    shouldWork )
```

Definition at line 8 of file log_info_test.cpp.

Here is the call graph for this function:

**7.38.1.10 TEST() [10/19]**

```
TEST (
    LogInfo ,
    shouldWork2 )
```

Definition at line 37 of file log_info_test.cpp.

Here is the call graph for this function:

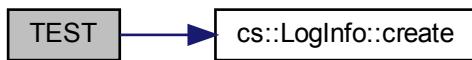


7.38.1.11 TEST() [11/19]

```
TEST (
    LogInfo ,
    shouldWork3 )
```

Definition at line 66 of file log_info_test.cpp.

Here is the call graph for this function:

**7.38.1.12 TEST() [12/19]**

```
TEST (
    LogInfo ,
    shouldWork4 )
```

Definition at line 95 of file log_info_test.cpp.

Here is the call graph for this function:



7.38.1.13 TEST() [13/19]

```
TEST (
    LogInfo ,
    shouldWork5 )
```

Definition at line 124 of file log_info_test.cpp.

Here is the call graph for this function:



7.38.1.14 TEST() [14/19]

```
TEST (
    LogInfo ,
    shouldWork6 )
```

Definition at line 153 of file log_info_test.cpp.

Here is the call graph for this function:



7.38.1.15 TEST() [15/19]

```
TEST (
    LogInfo ,
    shouldWork7 )
```

Definition at line 182 of file log_info_test.cpp.

Here is the call graph for this function:

**7.38.1.16 TEST() [16/19]**

```
TEST (
    LogInfo ,
    shouldWork8 )
```

Definition at line 211 of file log_info_test.cpp.

Here is the call graph for this function:



7.38.1.17 TEST() [17/19]

```
TEST (
    LogInfo ,
    shouldWork9 )
```

Definition at line 240 of file log_info_test.cpp.

Here is the call graph for this function:

**7.38.1.18 TEST() [18/19]**

```
TEST (
    LogInfo ,
    shouldWorkWithPath )
```

Definition at line 269 of file log_info_test.cpp.

Here is the call graph for this function:



7.38.1.19 TEST() [19/19]

```
TEST (
    LogInfo ,
    shouldWorkWithOldPath2 )
```

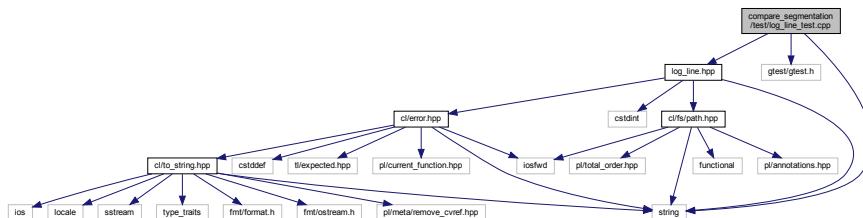
Definition at line 297 of file log_info_test.cpp.

Here is the call graph for this function:



7.39 compare_segmentation/test/log_line_test.cpp File Reference

```
#include <string>
#include "gtest/gtest.h"
#include "log_line.hpp"
Include dependency graph for log_line_test.cpp:
```



Functions

- [TEST](#) (LogLine, shouldWorkWithPreprocessedLine)
- [TEST](#) (LogLine, shouldWorkWithOldLine)
- [TEST](#) (LogLine, shouldNotMatchGarbage)
- [TEST](#) (LogLine, shouldNotParseGarbageSensor)

7.39.1 Function Documentation

7.39.1.1 TEST() [1/4]

```
TEST (
    LogLine ,
    shouldNotMatchGarbage   )
```

Definition at line 41 of file log_line_test.cpp.

Here is the call graph for this function:



7.39.1.2 TEST() [2/4]

```
TEST (
    LogLine ,
    shouldNotParseGarbageSensor   )
```

Definition at line 48 of file log_line_test.cpp.

Here is the call graph for this function:



7.39.1.3 TEST() [3/4]

```
TEST (
    LogLine ,
    shouldWorkWithOldLine )
```

Definition at line 25 of file log_line_test.cpp.

Here is the call graph for this function:

**7.39.1.4 TEST() [4/4]**

```
TEST (
    LogLine ,
    shouldWorkWithPreprocessedLine )
```

Definition at line 9 of file log_line_test.cpp.

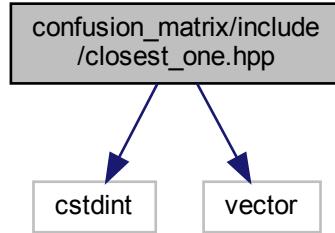
Here is the call graph for this function:



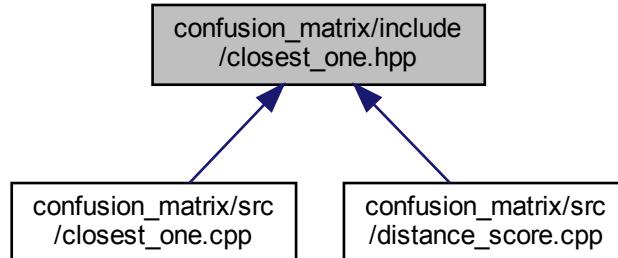
7.40 confusion_matrix/include/closest_one.hpp File Reference

```
#include <cstdint>
#include <vector>
```

Include dependency graph for closest_one.hpp:



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



Namespaces

- [cm](#)

Functions

- std::uint64_t [cm::closestOne](#) (std::uint64_t algorithmicallyDeterminedSegmentationPoint, const std::vector<std::uint64_t > &manualSegmentationPoints)

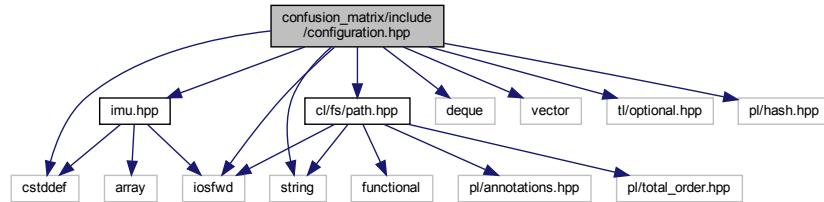
Finds the segmentation point in manualSegmentationPoints that is the closest to algorithmicallyDeterminedSegmentationPoint.

7.41 confusion_matrix/include/configuration.hpp File Reference

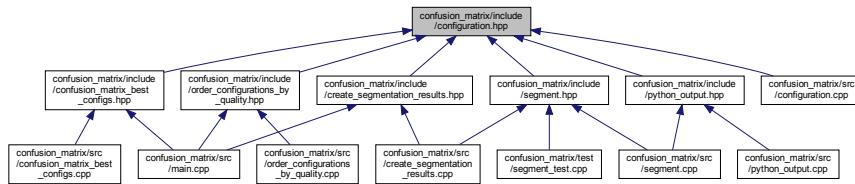
```
#include <cstdint>
#include <deque>
```

```
#include <iostream>
#include <string>
#include <vector>
#include <t1/optional.hpp>
#include <pl/hash.hpp>
#include <cl/fs/path.hpp>
#include "imu.hpp"

Include dependency graph for configuration.hpp:
```



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



Classes

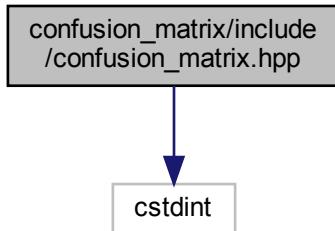
- class `cm::Configuration`
Represents a possible configuration for the Python segmentor.
- class `cm::Configuration::Builder`
Builder type for Configuration.
- struct `std::hash<::cm::Configuration >`

Namespaces

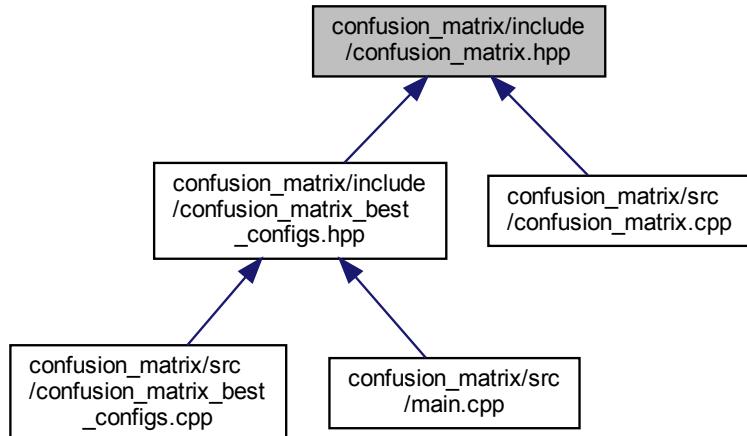
- `cm`

7.42 confusion_matrix/include/confusion_matrix.hpp File Reference

```
#include <cstdint>
Include dependency graph for confusion_matrix.hpp:
```



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



Classes

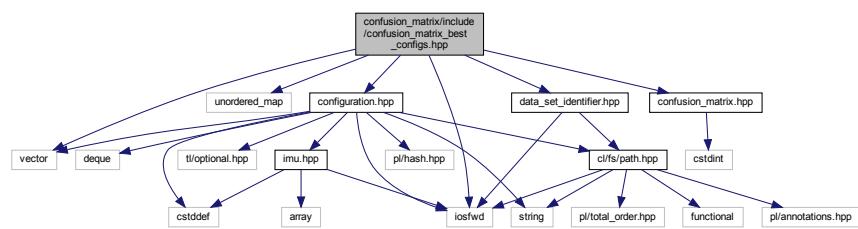
- class [cm::ConfusionMatrix](#)

Namespaces

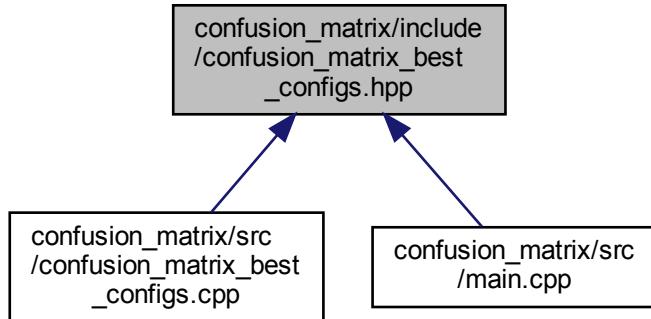
- [cm](#)

7.43 confusion_matrix/include/confusion_matrix_best_configs.hpp File Reference

```
#include <iostream>
#include <unordered_map>
#include <vector>
#include "configuration.hpp"
#include "confusion_matrix.hpp"
#include "data_set_identifier.hpp"
Include dependency graph for confusion_matrix_best_configs.hpp:
```



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



Classes

- struct `cm::ConfigWithTotalConfusionMatrix`

Namespaces

- `cm`

Macros

- `#define CM_SORTER(criterion, op)`

Functions

- `cm::CM_SORTER (truePositives, >)`
- `cm::CM_SORTER (trueNegatives, >)`
- `cm::CM_SORTER (falsePositives,<)`
- `cm::CM_SORTER (falseNegatives,<)`
- `bool cm::operator< (const ConfigWithTotalConfusionMatrix &lhs, const ConfigWithTotalConfusionMatrix &rhs) noexcept`
- `std::ostream & cm::operator<< (std::ostream &os, const ConfigWithTotalConfusionMatrix &obj)`
- `std::vector< ConfigWithTotalConfusionMatrix > cm::confusionMatrixBestConfigs (const std::unordered_map< DataSetIdentifier, std::vector< std::uint64_t > > &manualSegmentationPoints, const std::unordered_map< Configuration, std::unordered_map< cl::fs::Path, std::vector< std::uint64_t > > > &algorithmicallyDeterminedSegmentationPoints)`

7.43.1 Macro Definition Documentation

7.43.1.1 CM_SORTER

```
#define CM_SORTER(
    criterion,
    op )
```

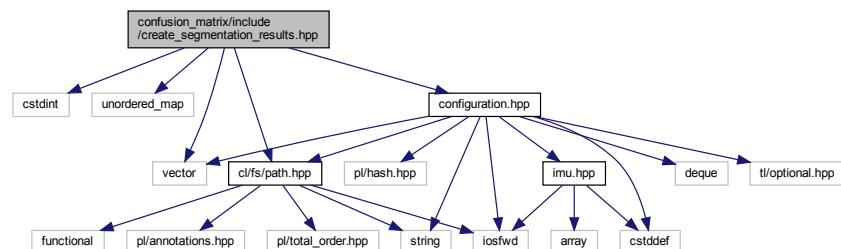
Value:

```
inline constexpr struct {
    bool operator()(
        const ConfigWithTotalConfusionMatrix& lhs,
        const ConfigWithTotalConfusionMatrix& rhs) const noexcept
    {
        return lhs.matrix.criterion() op rhs.matrix.criterion();
    }
} criterion##Sorter
```

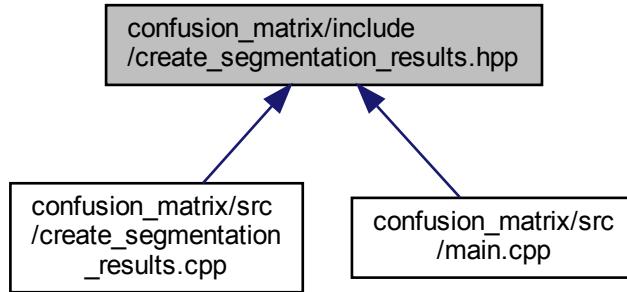
Definition at line 23 of file confusion_matrix_best_configs.hpp.

7.44 confusion_matrix/include/create_segmentation_results.hpp File Reference

```
#include <cstdint>
#include <unordered_map>
#include <vector>
#include <cl/fs/path.hpp>
#include "configuration.hpp"
Include dependency graph for create_segmentation_results.hpp:
```



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



Namespaces

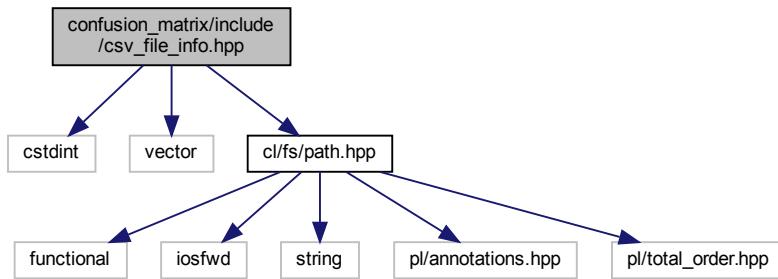
- cm

Functions

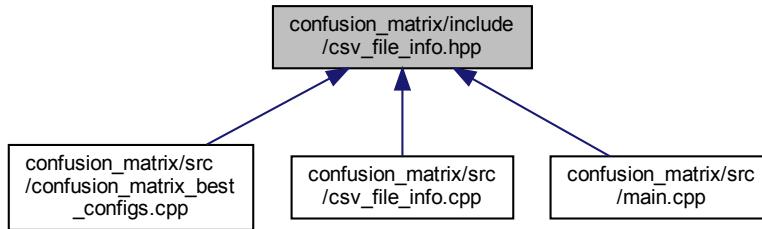
- std::unordered_map< cm::Configuration, std::unordered_map< cl::fs::Path, std::vector< std::uint64_t > > >
> cm::createSegmentationResults ()
Invokes Python to generate the segmentation points algorithmically.

7.45 confusion_matrix/include/csv_file_info.hpp File Reference

```
#include <cstdint>
#include <vector>
#include <cl/fs/path.hpp>
Include dependency graph for csv_file_info.hpp:
```



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



Classes

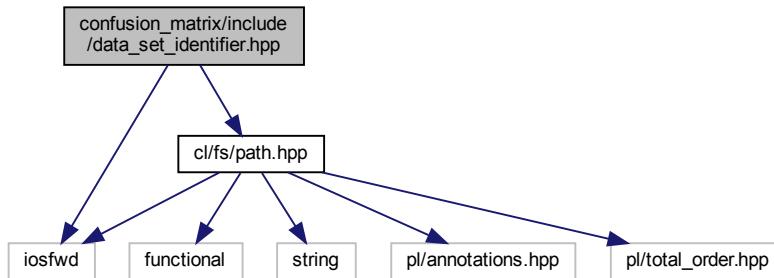
- class [cm::CsvFileInfo](#)
Type to hold the hardware timestamps of a CSV file.

Namespaces

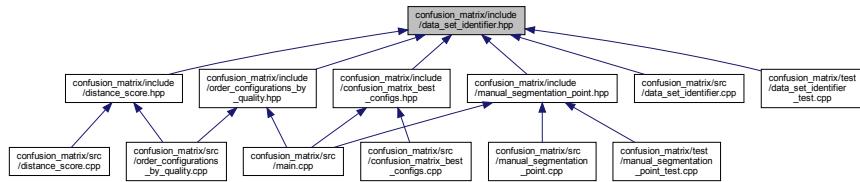
- [cm](#)

7.46 confusion_matrix/include/data_set_identifier.hpp File Reference

```
#include <iostream>
#include <c1/fs/path.hpp>
Include dependency graph for data_set_identifier.hpp:
```



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



Namespaces

- `cm`

Macros

- `#define CM_DATA_SET_IDENTIFIER`
- `#define CM_DATA_SET_IDENTIFIER_X(enm) enm,`

Enumerations

- enum `cm::DataSetIdentifier { cm::DataSetIdentifier::CM_DATA_SET_IDENTIFIER_X, cm::DataSetIdentifier::CM_DATA_SET_IDENTIFIER_Y }`

Functions

- `std::ostream & cm::operator<< (std::ostream &os, DataSetIdentifier dsi)`
Prints a DataSetIdentifier to an ostream.
- `DataSetIdentifier cm::toDataSetIdentifier (const cl::fs::Path &path)`
Converts a path to a CSV file to the corresponding DataSetIdentifier.

7.46.1 Macro Definition Documentation

7.46.1.1 CM_DATA_SET_IDENTIFIER

```
#define CM_DATA_SET_IDENTIFIER
```

Value:

```
CM_DATA_SET_IDENTIFIER_X(Felix_11_17_39) \
CM_DATA_SET_IDENTIFIER_X(Felix_12_50_00) \
CM_DATA_SET_IDENTIFIER_X(Felix_13_00_09) \
CM_DATA_SET_IDENTIFIER_X(Mike_14_07_33) \
CM_DATA_SET_IDENTIFIER_X(Mike_14_14_32) \
CM_DATA_SET_IDENTIFIER_X(Mike_14_20_28) \
CM_DATA_SET_IDENTIFIER_X(Marsi_14_59_59) \
CM_DATA_SET_IDENTIFIER_X(Marsi_15_13_22) \
CM_DATA_SET_IDENTIFIER_X(Marsi_15_31_36) \
CM_DATA_SET_IDENTIFIER_X(Jan_1) \
CM_DATA_SET_IDENTIFIER_X(Jan_2) \
CM_DATA_SET_IDENTIFIER_X(Jan_3) \
CM_DATA_SET_IDENTIFIER_X(Andre_1) \
CM_DATA_SET_IDENTIFIER_X(Andre_2) \
CM_DATA_SET_IDENTIFIER_X(Andre_3) \
CM_DATA_SET_IDENTIFIER_X(Andre_Squats_1) \
CM_DATA_SET_IDENTIFIER_X(Andre_Squats_2) \
CM_DATA_SET_IDENTIFIER_X(Lucas_1) \
CM_DATA_SET_IDENTIFIER_X(Lucas_2) \
CM_DATA_SET_IDENTIFIER_X(Lucas_3)
```

Definition at line 8 of file data_set_identifier.hpp.

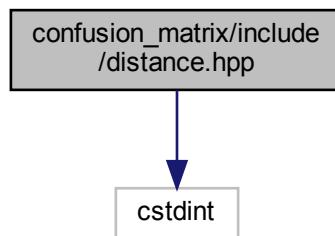
7.46.1.2 CM_DATA_SET_IDENTIFIER_X

```
#define CM_DATA_SET_IDENTIFIER_X(
    enm ) enm,
```

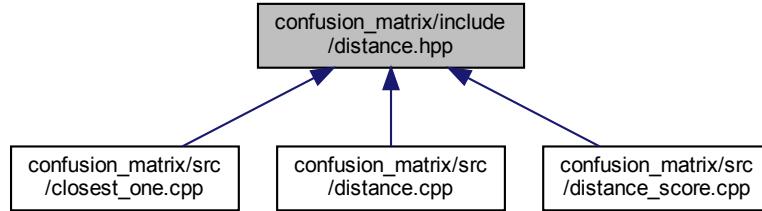
Definition at line 31 of file data_set_identifier.hpp.

7.47 confusion_matrix/include/distance.hpp File Reference

```
#include <cstdint>
Include dependency graph for distance.hpp:
```



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



Namespaces

- `cm`

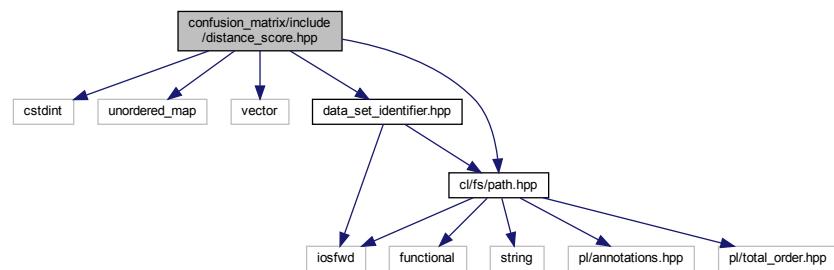
Functions

- `std::uint64_t cm::distance (std::uint64_t a, std::uint64_t b)`

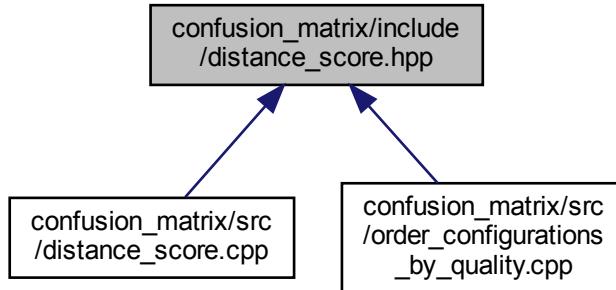
Calculates the distance between a and b.

7.48 confusion_matrix/include/distance_score.hpp File Reference

```
#include <cstdint>
#include <unordered_map>
#include <vector>
#include <cl/fs/path.hpp>
#include "data_set_identifier.hpp"
Include dependency graph for distance_score.hpp:
```



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



Namespaces

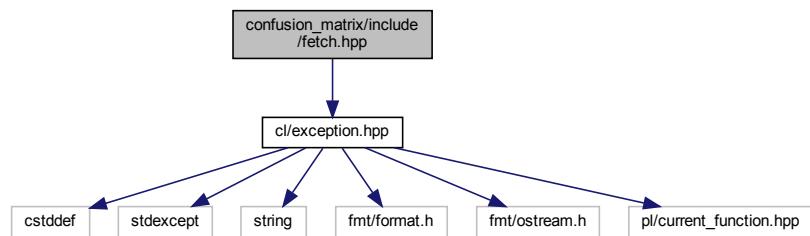
- `cm`

Functions

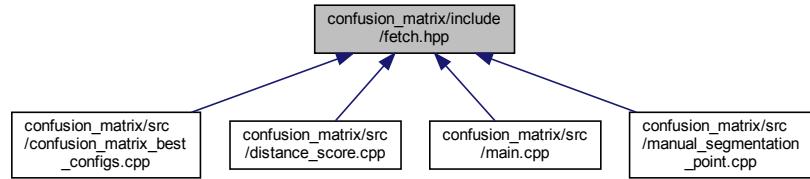
- `std::uint64_t cm::distanceScore (const std::unordered_map< cl::fs::Path, std::vector< std::uint64_t >> &segmentationPointsForConfig, const std::unordered_map< DataSetIdentifier, std::vector< std::uint64_t >> &manualSegmentationPoints)`

7.49 confusion_matrix/include/fetch.hpp File Reference

```
#include <cl/exception.hpp>
Include dependency graph for fetch.hpp:
```



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



Namespaces

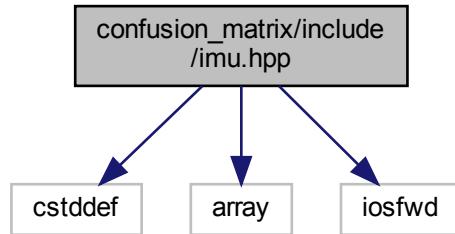
- `cm`

Functions

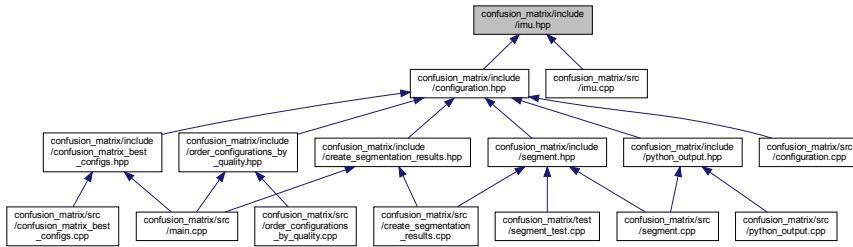
- template<typename Map , typename Key >
auto `cm::fetch` (const Map &map, const Key &key)
- Fetches a value from a map for a given key.*

7.50 confusion_matrix/include imu.hpp File Reference

```
#include <cstddef>
#include <array>
#include <iostream>
Include dependency graph for imu.hpp:
```



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



Namespaces

- `cm`

Macros

- `#define CM_IMU`
- `#define CM_IMU_X(enm) enm,`
- `#define CM_IMU_X(enm) +1`
- `#define CM_IMU_X(enm) ::cm::imu::enm,`

Enumerations

- enum `cm::imu { cm::imu::CM_IMU_X, cm::imu::CM_IMU }`
- Scoped enum type for the IMUs.*

Functions

- `std::ostream & cm::operator<< (std::ostream &os, Imu imu)`
- Prints `imu` to `os`.*

Variables

- `constexpr std::size_t cm::imuCount`

The amount of IMUs.

- `constexpr std::array<Imu, imuCount> cm::imus`

An array of the IMU enumerators.

7.50.1 Macro Definition Documentation

7.50.1.1 CM_IMU

```
#define CM_IMU
```

Value:

```
CM_IMU_X (Accelerometer) \
CM_IMU_X (Gyroscope)
```

Definition at line 10 of file imu.hpp.

7.50.1.2 CM_IMU_X [1/3]

```
#define CM_IMU_X(
    enm ) enm,
```

Definition at line 18 of file imu.hpp.

7.50.1.3 CM_IMU_X [2/3]

```
#define CM_IMU_X(
    enm ) +1
```

Definition at line 18 of file imu.hpp.

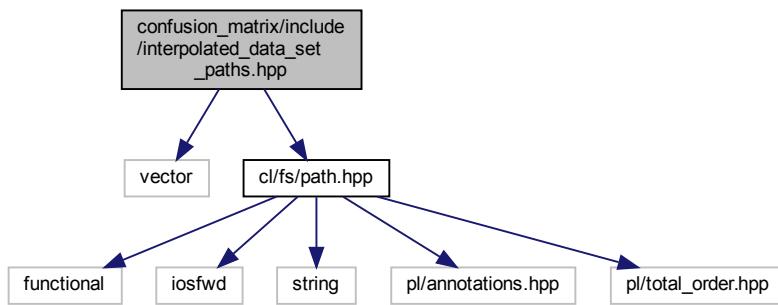
7.50.1.4 CM_IMU_X [3/3]

```
#define CM_IMU_X(
    enm ) ::cm::Imu::enm,
```

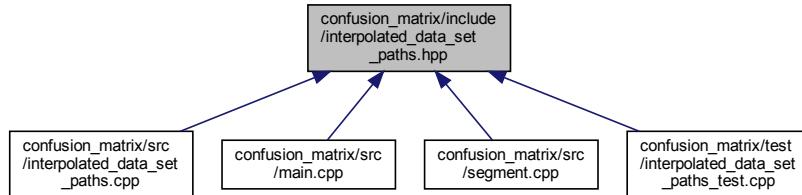
Definition at line 18 of file imu.hpp.

7.51 confusion_matrix/include/interpolated_data_set_paths.hpp File Reference

```
#include <vector>
#include <cl/fs/path.hpp>
Include dependency graph for interpolated_data_set_paths.hpp:
```



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



Namespaces

- `cm`

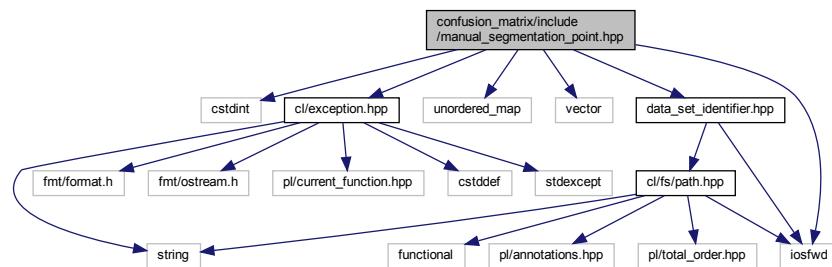
Functions

- `std::vector< cl::fs::Path > cm::interpolatedDataSetPaths ()`

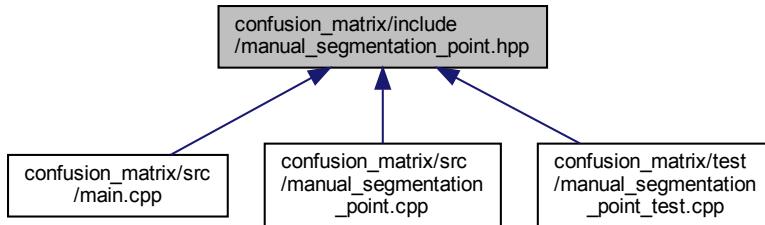
Returns the paths to the interpolated data sets.

7.52 confusion_matrix/include/manual_segmentation_point.hpp File Reference

```
#include <cstdint>
#include <iostream>
#include <unordered_map>
#include <vector>
#include <cl/exception.hpp>
#include "data_set_identifier.hpp"
Include dependency graph for manual_segmentation_point.hpp:
```



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



Classes

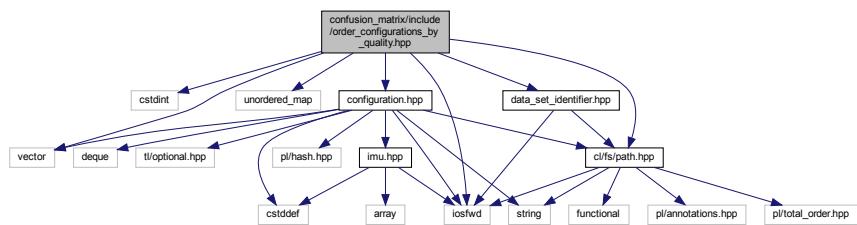
- class [cm::ManualSegmentationPoint](#)
Type used to represent a manual segmentation point.

Namespaces

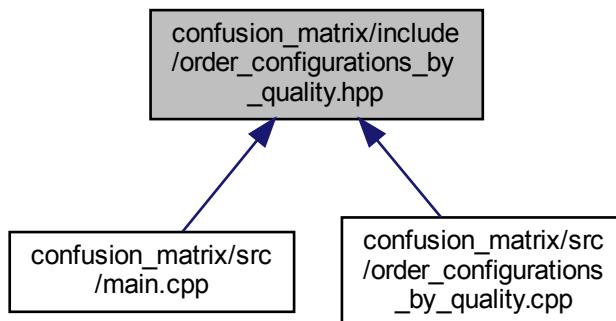
- [cm](#)

7.53 confusion_matrix/include/order_configurations_by_quality.hpp File Reference

```
#include <cstdint>
#include <iostream>
#include <unordered_map>
#include <vector>
#include <cl/fs/path.hpp>
#include "configuration.hpp"
#include "data_set_identifier.hpp"
Include dependency graph for order_configurations_by_quality.hpp:
```



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



Classes

- struct [cm::ConfigWithDistanceScore](#)

Namespaces

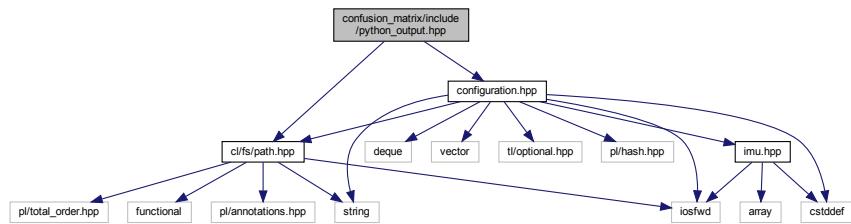
- [cm](#)

Functions

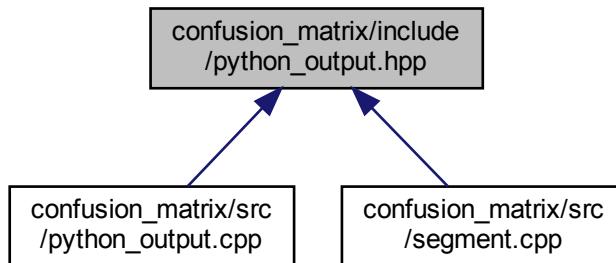
- bool `cm::operator<` (const ConfigWithDistanceScore &lhs, const ConfigWithDistanceScore &rhs) noexcept
- std::ostream & `cm::operator<<` (std::ostream &os, const ConfigWithDistanceScore &configWithDistScore)
- std::vector< ConfigWithDistanceScore > `cm::orderConfigurationsByQuality` (const std::unordered_map< DataSetIdentifier, std::vector< std::uint64_t > >> &manualSegmentationPoints, const std::unordered_map< Configuration, std::unordered_map< cl::fs::Path, std::vector< std::uint64_t > >>> &algorithmicallyDeterminedSegmentationPoints)

7.54 confusion_matrix/include/python_output.hpp File Reference

```
#include <cl/fs/path.hpp>
#include "configuration.hpp"
Include dependency graph for python_output.hpp:
```



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



Namespaces

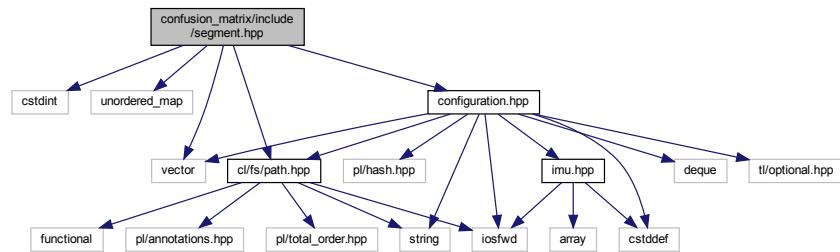
- `cm`

Functions

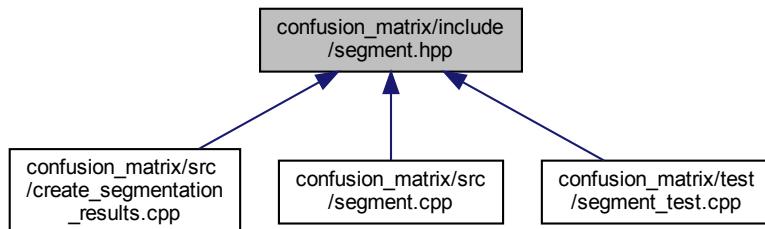
- std::string `cm::pythonOutput` (const `cl::fs::Path` &csvFilePath, const Configuration &segmentorConfiguration)
Runs the Python segmentor on path.

7.55 confusion_matrix/include/segment.hpp File Reference

```
#include <cstdint>
#include <unordered_map>
#include <vector>
#include <cl/fs/path.hpp>
#include "configuration.hpp"
Include dependency graph for segment.hpp:
```



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



Namespaces

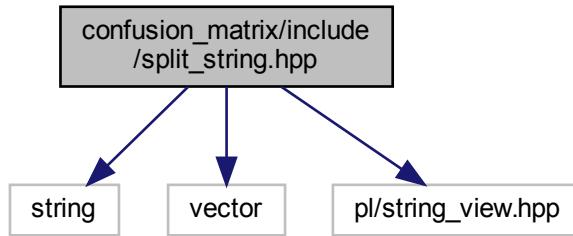
- `cm`

Functions

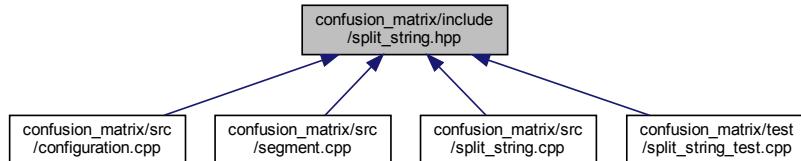
- `std::unordered_map< cl::fs::Path, std::vector< std::uint64_t > > cm::segment (const Configuration &segmentorConfiguration)`

7.56 confusion_matrix/include/split_string.hpp File Reference

```
#include <string>
#include <vector>
#include <pl/string_view.hpp>
Include dependency graph for split_string.hpp:
```



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



Namespaces

- [cm](#)

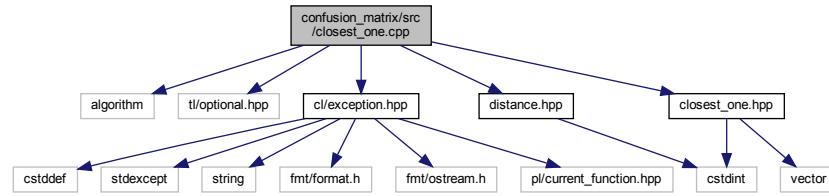
Functions

- `std::vector< std::string > cm::splitString (std::string string, pl::string_view splitBy)`
Splits string by splitBy.

7.57 confusion_matrix/src/closest_one.cpp File Reference

```
#include <algorithm>
#include <t1/optional.hpp>
#include <cl/exception.hpp>
#include "closest_one.hpp"
```

```
#include "distance.hpp"
Include dependency graph for closest_one.cpp:
```



Namespaces

- cm

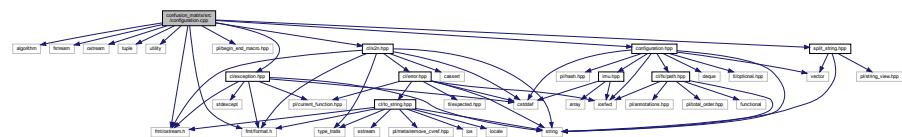
Functions

- std::uint64_t **cm::closestOne** (std::uint64_t algorithmicallyDeterminedSegmentationPoint, const std::vector<std::uint64_t> &manualSegmentationPoints)

Finds the segmentation point in manualSegmentationPoints that is the closest to algorithmicallyDeterminedSegmentationPoint.

7.58 confusion_matrix/src/configuration.cpp File Reference

```
#include <algorithm>
#include <fstream>
#include <ostream>
#include <tuple>
#include <utility>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <pl/begin_end_macro.hpp>
#include <cl/exception.hpp>
#include <cl/s2n.hpp>
#include "configuration.hpp"
#include "split_string.hpp"
Include dependency graph for configuration.cpp:
```



Namespaces

- cm

Macros

- `#define CM_ENSURE_HAS_VALUE(dataMember)`
- `#define CM_ENSURE_CONTAINS(container, dataMember)`

Functions

- `bool cm::operator==(const Configuration &lhs, const Configuration &rhs) noexcept`
- `bool cm::operator!=(const Configuration &lhs, const Configuration &rhs) noexcept`
- `std::ostream & cm::operator<< (std::ostream &os, const Configuration &config)`

7.58.1 Macro Definition Documentation

7.58.1.1 CM_ENSURE_CONTAINS

```
#define CM_ENSURE_CONTAINS (
    container,
    dataMember )
```

Value:

```
PL_BEGIN_MACRO
if (!contains(container, dataMember)) {
    CL_THROW_FMT(
        "\\"{}\\" is not a valid option for \"{}\"", *dataMember, #dataMember); \
}
PL_END_MACRO
```

7.58.1.2 CM_ENSURE_HAS_VALUE

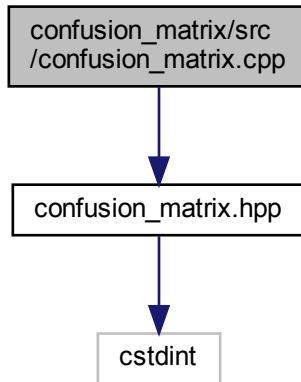
```
#define CM_ENSURE_HAS_VALUE (
    dataMember )
```

Value:

```
PL_BEGIN_MACRO
if (!dataMember.has_value()) {
    CL_THROW_FMT("\"{}\" was nullopt!", #dataMember); \
}
PL_END_MACRO
```

7.59 confusion_matrix/src/confusion_matrix.cpp File Reference

```
#include "confusion_matrix.hpp"
Include dependency graph for confusion_matrix.cpp:
```

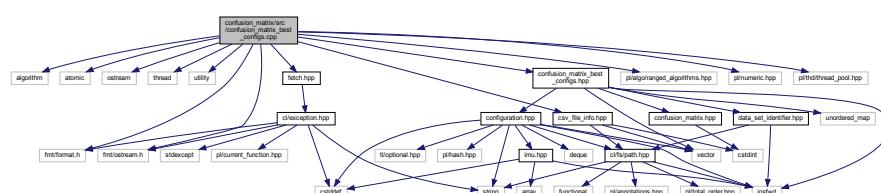


Namespaces

- cm

7.60 confusion_matrix/src/confusion_matrix_best_configs.cpp File Reference

```
#include <algorithm>
#include <atomic>
#include <iostream>
#include <thread>
#include <utility>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <pl/algo/ranged_algorithms.hpp>
#include <pl/numeric.hpp>
#include <pl/thread/thread_pool.hpp>
#include "confusion_matrix_best_configs.hpp"
#include "csv_file_info.hpp"
#include "fetch.hpp"
Include dependency graph for confusion_matrix_best_configs.cpp:
```



Namespaces

- [cm](#)

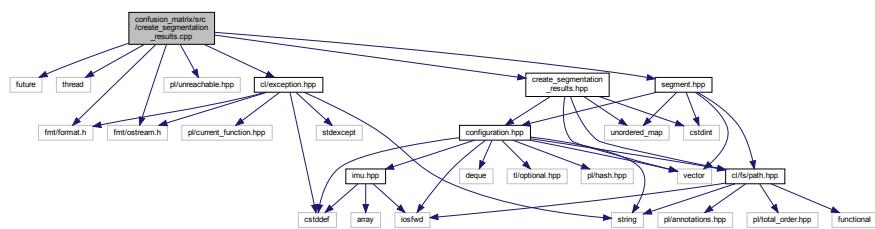
Functions

- `bool cm::operator< (const ConfigWithTotalConfusionMatrix &lhs, const ConfigWithTotalConfusionMatrix &rhs) noexcept`
- `std::ostream & cm::operator<< (std::ostream &os, const ConfigWithTotalConfusionMatrix &obj)`
- `std::vector< ConfigWithTotalConfusionMatrix > cm::confusionMatrixBestConfigs (const std::unordered_map< DataSetIdentifier, std::vector< std::uint64_t > > &manualSegmentationPoints, const std::unordered_map< Configuration, std::unordered_map< cl::fs::Path, std::vector< std::uint64_t > > > &algorithmicallyDeterminedSegmentationPoints)`

7.61 confusion_matrix/src/create_segmentation_results.cpp File Reference

```
#include <future>
#include <thread>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <pl/unreachable.hpp>
#include <cl/exception.hpp>
#include "create_segmentation_results.hpp"
#include "segment.hpp"

Include dependency graph for create_segmentation_results.cpp:
```



Namespaces

- [cm](#)

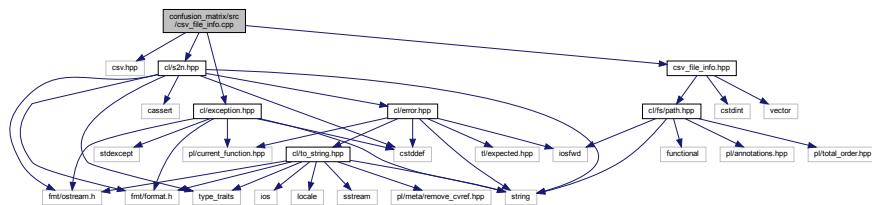
Functions

- `std::unordered_map< cm::Configuration, std::unordered_map< cl::fs::Path, std::vector< std::uint64_t > > > cm::createSegmentationResults ()`

Invokes Python to generate the segmentation points algorithmically.

7.62 confusion_matrix/src/csv_file_info.cpp File Reference

```
#include <csv.hpp>
#include <cl/exception.hpp>
#include <cl/s2n.hpp>
#include "csv_file_info.hpp"
Include dependency graph for csv_file_info.cpp:
```

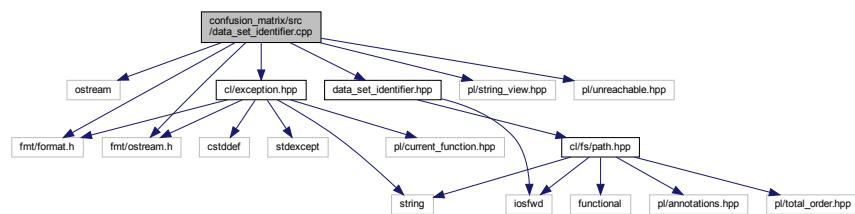


Namespaces

- `cm`

7.63 confusion_matrix/src/data_set_identifier.cpp File Reference

```
#include <iostream>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <pl/string_view.hpp>
#include <pl/unreachable.hpp>
#include <cl/exception.hpp>
#include "data_set_identifier.hpp"
Include dependency graph for data_set_identifier.cpp:
```



Namespaces

- `cm`

Macros

- `#define CM_DATA_SET_IDENTIFIER_X(enm) case DataSetIdentifier::enm: return #enm; /* stringify */`
- `#define DSI DataSetIdentifier`

Functions

- std::ostream & `cm::operator<<` (std::ostream &os, DataSetIdentifier dsi)
Prints a DataSetIdentifier to an ostream.
- DataSetIdentifier `cm::toDataSetIdentifier` (const cl::fs::Path &path)
Converts a path to a CSV file to the corresponding DataSetIdentifier.

7.63.1 Macro Definition Documentation

7.63.1.1 CM_DATA_SET_IDENTIFIER_X

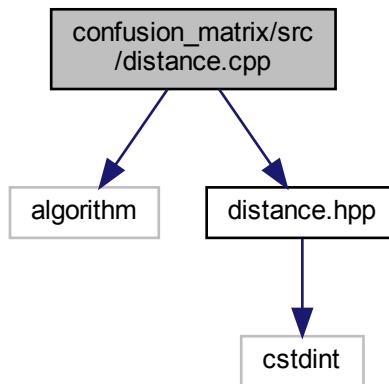
```
#define CM_DATA_SET_IDENTIFIER_X(  
    enm ) case DataSetIdentifier::enm:    return #enm; /* stringify */
```

7.63.1.2 DSI

```
#define DSI DataSetIdentifier
```

7.64 confusion_matrix/src/distance.cpp File Reference

```
#include <algorithm>  
#include "distance.hpp"  
Include dependency graph for distance.cpp:
```



Namespaces

- cm

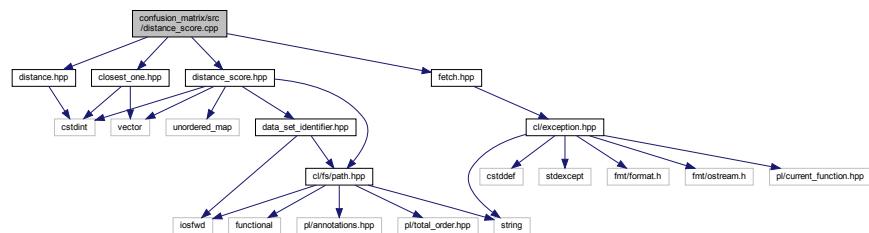
Functions

- std::uint64_t cm::distance (std::uint64_t a, std::uint64_t b)

Calculates the distance between a and b.

7.65 confusion_matrix/src/distance_score.cpp File Reference

```
#include "distance_score.hpp"
#include "closest_one.hpp"
#include "distance.hpp"
#include "fetch.hpp"
Include dependency graph for distance_score.cpp:
```



Namespaces

- cm

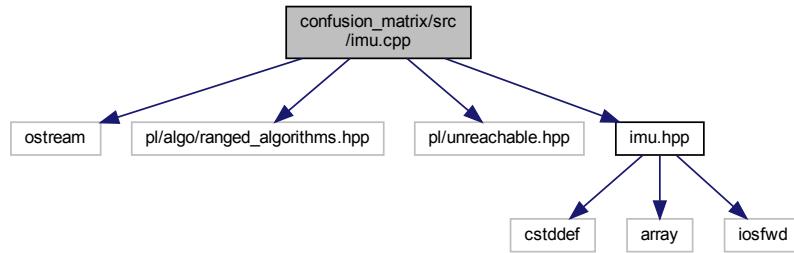
Functions

- std::uint64_t **cm::distanceScore** (const std::unordered_map< cl::fs::Path, std::vector< std::uint64_t > >& segmentationPointsForConfig, const std::unordered_map< DataSetIdentifier, std::vector< std::uint64_t >> & manualSegmentationPoints)

7.66 confusion_matrix/src/imu.cpp File Reference

```
#include <iostream>
#include <pl/algo/ranged_algorithms.hpp>
#include <pl/unreachable.hpp>
```

```
#include "imu.hpp"
Include dependency graph for imu.cpp:
```



Namespaces

- `cm`

Macros

- `#define CM_IMU_X(enm) case Imu::enm: return os << toLower(#enm);`

Functions

- `std::ostream & cm::operator<< (std::ostream &os, Imu imu)`

Prints imu to os.

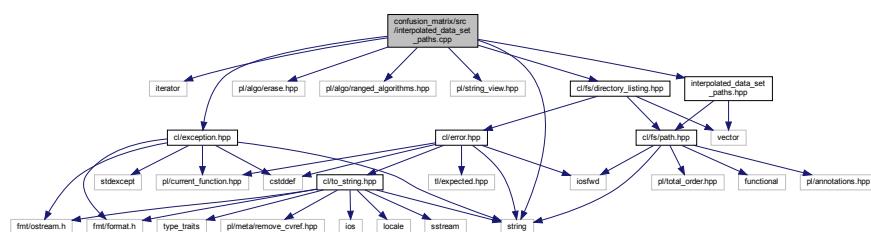
7.66.1 Macro Definition Documentation

7.66.1.1 CM_IMU_X

```
#define CM_IMU_X(
    enm ) case Imu::enm:    return os << toLower(#enm);
```

7.67 confusion_matrix/src/interpolated_data_set_paths.cpp File Reference

```
#include <iterator>
#include <string>
#include <pl/algo/erase.hpp>
#include <pl/algo/ranged_algorithms.hpp>
#include <pl/string_view.hpp>
#include <cl/exception.hpp>
#include <cl/fs/directory_listing.hpp>
#include "interpolated_data_set_paths.hpp"
Include dependency graph for interpolated_data_set_paths.cpp:
```



Namespaces

- `cm`

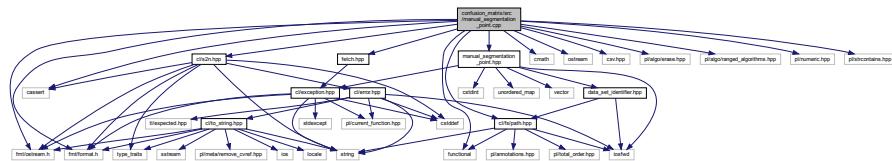
Functions

- `std::vector< cl::fs::Path > cm::interpolatedDataSetPaths ()`
Returns the paths to the interpolated data sets.

7.68 confusion_matrix/src/manual_segmentation_point.cpp File Reference

```
#include <cassert>
#include <cmath>
#include <functional>
#include <iostream>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <csv.hpp>
#include <pl/algo/erase.hpp>
#include <pl/algo/ranged_algorithms.hpp>
#include <pl/numeric.hpp>
#include <pl/strcontains.hpp>
#include <cl/fs/path.hpp>
#include <cl/s2n.hpp>
#include "fetch.hpp"
```

```
#include "manual_segmentation_point.hpp"
Include dependency graph for manual_segmentation_point.cpp:
```



Namespaces

- `cm`

Macros

- `#define DSI DataSetIdentifier`

Functions

- `bool cm::operator==(const ManualSegmentationPoint &lhs, const ManualSegmentationPoint &rhs) noexcept`
- `bool cm::operator!=(const ManualSegmentationPoint &lhs, const ManualSegmentationPoint &rhs) noexcept`
- `std::ostream & cm::operator<< (std::ostream &os, const ManualSegmentationPoint &manualSegmentationPoint)`

7.68.1 Macro Definition Documentation

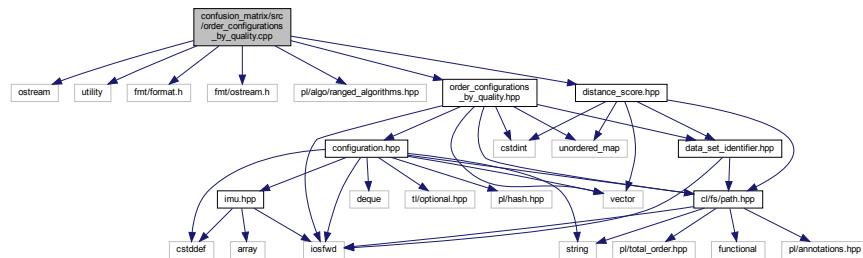
7.68.1.1 DSI

```
#define DSI DataSetIdentifier
```

7.69 confusion_matrix/src/order_configurations_by_quality.cpp File Reference

```
#include <iostream>
#include <utility>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <ppl/algo/ranged_algorithms.hpp>
#include "distance_score.hpp"
```

```
#include "order_configurations_by_quality.hpp"
Include dependency graph for order_configurations_by_quality.cpp:
```



Namespaces

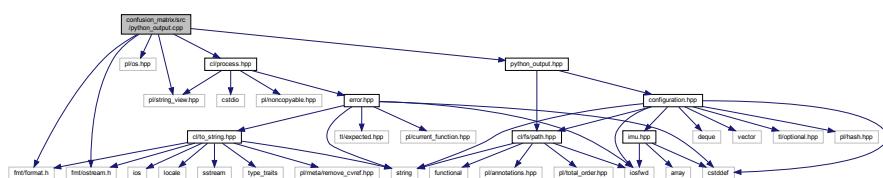
- `cm`

Functions

- `bool cm::operator< (const ConfigWithDistanceScore &lhs, const ConfigWithDistanceScore &rhs) noexcept`
- `std::ostream & cm::operator<< (std::ostream &os, const ConfigWithDistanceScore &configWithDistScore)`
- `std::vector< ConfigWithDistanceScore > cm::orderConfigurationsByQuality (const std::unordered_map< DataSetIdentifier, std::vector< std::uint64_t > >> &manualSegmentationPoints, const std::unordered_map< Configuration, std::unordered_map< cl::fs::Path, std::vector< std::uint64_t > >>> &algorithmicallyDeterminedSegmentationPoints)`

7.70 confusion_matrix/src/python_output.cpp File Reference

```
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <pl/os.hpp>
#include <pl/string_view.hpp>
#include <c1/process.hpp>
#include "python_output.hpp"
Include dependency graph for python_output.cpp:
```



Namespaces

- `cm`

Macros

- `#define CM_SEGMENTOR "./preprocessed_segment.sh"`
Object like macro for the segmentor script.
- `#define CM_DEV_NULL "/dev/null"`
Object like macro for /dev/null.

Functions

- `std::string cm::pythonOutput (const cl::fs::Path &csvFilePath, const Configuration &segmentorConfiguration)`
Runs the Python segmentor on path.

7.70.1 Macro Definition Documentation

7.70.1.1 CM_DEV_NULL

```
#define CM_DEV_NULL "/dev/null"
```

Object like macro for /dev/null.

Definition at line 24 of file python_output.cpp.

7.70.1.2 CM_SEGMENTOR

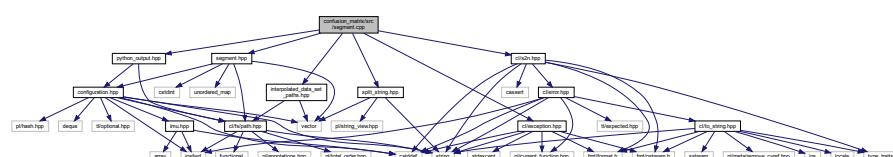
```
#define CM_SEGMENTOR "./preprocessed_segment.sh"
```

Object like macro for the segmentor script.

Definition at line 23 of file python_output.cpp.

7.71 confusion_matrix/src/segment.cpp File Reference

```
#include <cl/exception.hpp>
#include <cl/s2n.hpp>
#include "interpolated_data_set_paths.hpp"
#include "python_output.hpp"
#include "segment.hpp"
#include "split_string.hpp"
Include dependency graph for segment.cpp:
```



Namespaces

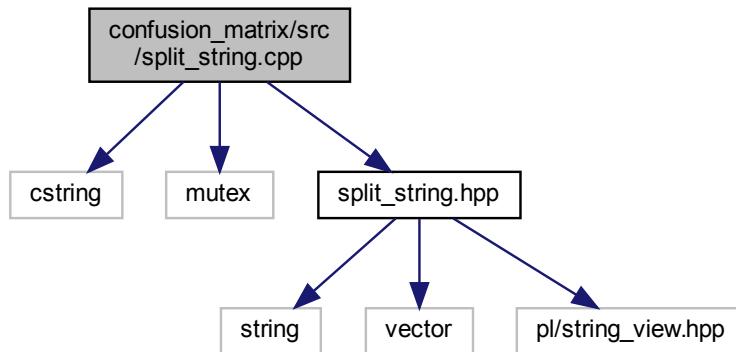
- [cm](#)

Functions

- `std::unordered_map< cl::fs::Path, std::vector< std::uint64_t > > cm::segment (const Configuration &segmentorConfiguration)`

7.72 confusion_matrix/src/split_string.cpp File Reference

```
#include <cstring>
#include <mutex>
#include "split_string.hpp"
Include dependency graph for split_string.cpp:
```



Namespaces

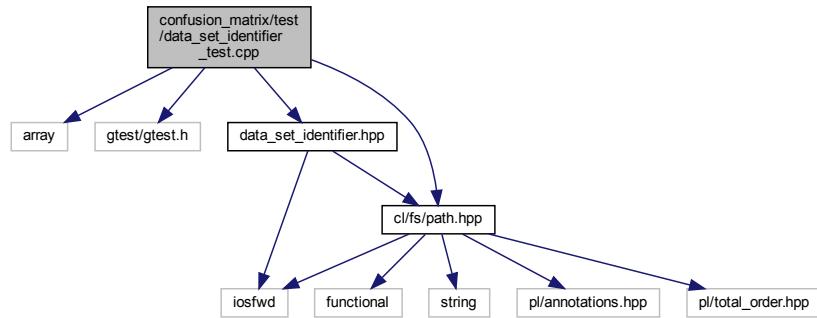
- [cm](#)

Functions

- `std::vector< std::string > cm::splitString (std::string string, pl::string_view splitBy)`
Splits string by splitBy.

7.73 confusion_matrix/test/data_set_identifier_test.cpp File Reference

```
#include <array>
#include "gtest/gtest.h"
#include <c1/fs/path.hpp>
#include "data_set_identifier.hpp"
Include dependency graph for data_set_identifier.cpp:
```



Macros

- `#define DSI ::cm::DataSetIdentifier`

Functions

- `TEST (DataSetIdentifier, shouldConvertPaths)`

7.73.1 Macro Definition Documentation

7.73.1.1 DSI

```
#define DSI ::cm::DataSetIdentifier
```

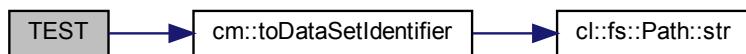
7.73.2 Function Documentation

7.73.2.1 TEST()

```
TEST (
    DataSetIdentifier ,
    shouldConvertPaths )
```

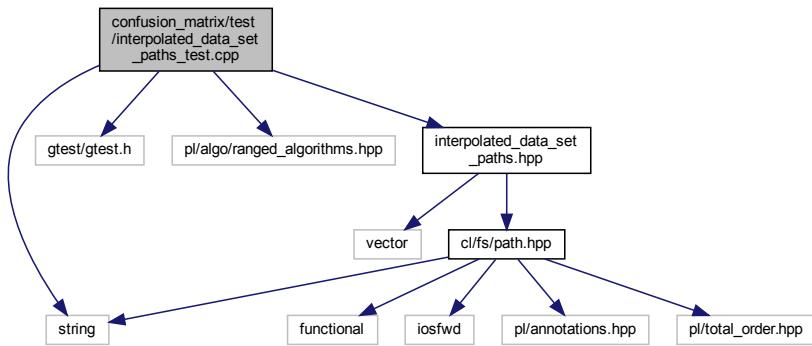
Definition at line 9 of file `data_set_identifier_test.cpp`.

Here is the call graph for this function:



7.74 confusion_matrix/test/interpolated_data_set_paths_test.cpp File Reference

```
#include <string>
#include "gtest/gtest.h"
#include <pl/algo/ranged_algorithms.hpp>
#include "interpolated_data_set_paths.hpp"
Include dependency graph for interpolated_data_set_paths_test.cpp:
```



Functions

- [TEST](#) (`interpolatedDataSetPaths`, `shouldFetchPaths`)

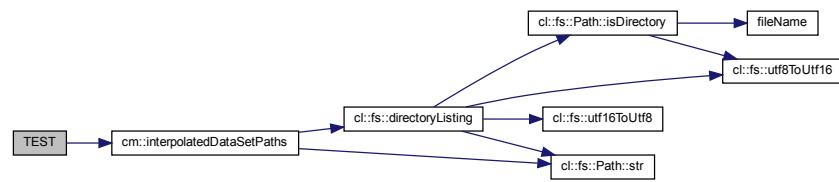
7.74.1 Function Documentation

7.74.1.1 TEST()

```
TEST (
    interpolatedDataSetPaths ,
    shouldFetchPaths )
```

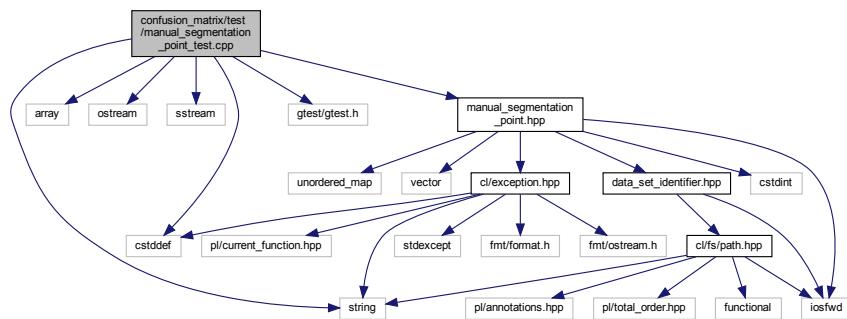
Definition at line 9 of file interpolated_data_set_paths_test.cpp.

Here is the call graph for this function:



7.75 confusion_matrix/test/manual_segmentation_point_test.cpp File Reference

```
#include <cstddef>
#include <array>
#include <iostream>
#include <sstream>
#include <sstream>
#include <string>
#include "gtest/gtest.h"
#include "manual_segmentation_point.hpp"
Include dependency graph for manual_segmentation_point_test.cpp:
```



Macros

- #define DSI ::cm::DataSetIdentifier

Functions

- `TEST (ManualSegmentationPoint, shouldConstruct)`
- `TEST (ManualSegmentationPoint, shouldThrowWhenConstructingWithInvalidMinute)`
- `TEST (ManualSegmentationPoint, shouldThrowWhenConstructingWithInvalidSecond)`
- `TEST (ManualSegmentationPoint, shouldThrowWhenConstructingWithInvalidFrame)`
- `TEST (ManualSegmentationPoint, shouldConvertToMilliseconds)`
- `TEST (ManualSegmentationPoint, shouldConvertHourToMilliseconds)`
- `TEST (ManualSegmentationPoint, shouldConvertMinuteToMilliseconds)`
- `TEST (ManualSegmentationPoint, shouldConvertSecondToMilliseconds)`
- `TEST (ManualSegmentationPoint, shouldConvertFramesToMilliseconds)`
- `TEST (ManualSegmentationPoint, shouldBeAbleToImportCsvFile)`
- `TEST (ManualSegmentationPoint, shouldPrint)`

7.75.1 Macro Definition Documentation

7.75.1.1 DSI

```
#define DSI ::cm::DataSetIdentifier
```

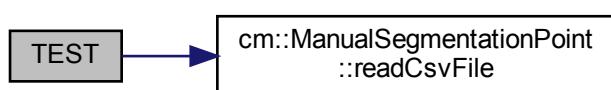
7.75.2 Function Documentation

7.75.2.1 TEST() [1/11]

```
TEST (
    ManualSegmentationPoint ,
    shouldBeAbleToImportCsvFile )
```

Definition at line 98 of file manual_segmentation_point_test.cpp.

Here is the call graph for this function:



7.75.2.2 TEST() [2/11]

```
TEST (
    ManualSegmentationPoint ,
    shouldConstruct )
```

Definition at line 12 of file manual_segmentation_point_test.cpp.

7.75.2.3 TEST() [3/11]

```
TEST (
    ManualSegmentationPoint ,
    shouldConvertFramesToMilliseconds )
```

Definition at line 82 of file manual_segmentation_point_test.cpp.

7.75.2.4 TEST() [4/11]

```
TEST (
    ManualSegmentationPoint ,
    shouldConvertHourToMilliseconds )
```

Definition at line 64 of file manual_segmentation_point_test.cpp.

7.75.2.5 TEST() [5/11]

```
TEST (
    ManualSegmentationPoint ,
    shouldConvertMinuteToMilliseconds )
```

Definition at line 70 of file manual_segmentation_point_test.cpp.

7.75.2.6 TEST() [6/11]

```
TEST (
    ManualSegmentationPoint ,
    shouldConvertSecondToMilliseconds )
```

Definition at line 76 of file manual_segmentation_point_test.cpp.

7.75.2.7 TEST() [7/11]

```
TEST (
    ManualSegmentationPoint ,
    shouldConvertToMilliseconds )
```

Definition at line 58 of file manual_segmentation_point_test.cpp.

7.75.2.8 TEST() [8/11]

```
TEST (
    ManualSegmentationPoint ,
    shouldPrint )
```

Definition at line 370 of file manual_segmentation_point_test.cpp.

7.75.2.9 TEST() [9/11]

```
TEST (
    ManualSegmentationPoint ,
    shouldThrowWhenConstructingWithInvalidFrame )
```

Definition at line 46 of file manual_segmentation_point_test.cpp.

7.75.2.10 TEST() [10/11]

```
TEST (
    ManualSegmentationPoint ,
    shouldThrowWhenConstructingWithInvalidMinute )
```

Definition at line 22 of file manual_segmentation_point_test.cpp.

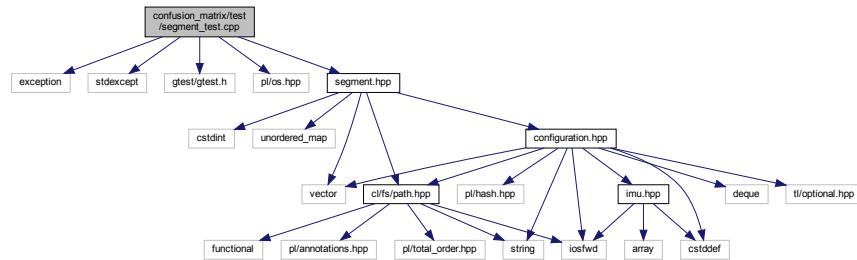
7.75.2.11 TEST() [11/11]

```
TEST (
    ManualSegmentationPoint ,
    shouldThrowWhenConstructingWithInvalidSecond )
```

Definition at line 34 of file manual_segmentation_point_test.cpp.

7.76 confusion_matrix/test/segment_test.cpp File Reference

```
#include <exception>
#include <stdexcept>
#include "gtest/gtest.h"
#include <pl/os.hpp>
#include "segment.hpp"
Include dependency graph for segment_test.cpp:
```



Macros

- `#define EXPECT_SEGMENTATION_POINTS(path, ...)` `EXPECT_EQ((std::vector<std::uint64_t>{__VA_ARGS__}), fetch(path))`

Functions

- `TEST` (`segment`, `shouldGetExpectedSegmentationPointsFromPython`)

7.76.1 Macro Definition Documentation

7.76.1.1 EXPECT_SEGMENTATION_POINTS

```
#define EXPECT_SEGMENTATION_POINTS(
    path,
    ...
) EXPECT_EQ((std::vector<std::uint64_t>{__VA_ARGS__}), fetch(path))
```

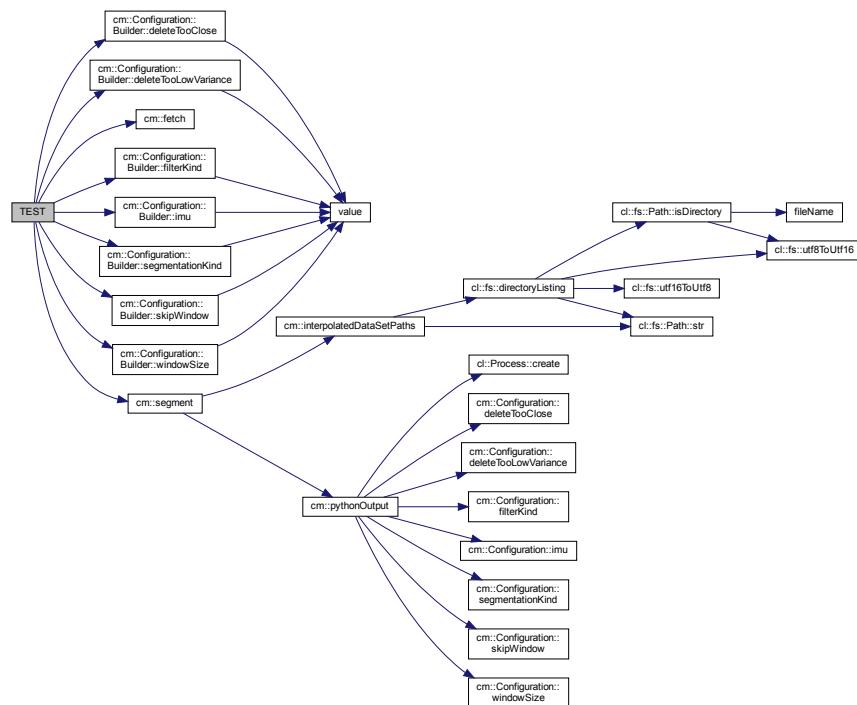
7.76.2 Function Documentation

7.76.2.1 TEST()

```
TEST (
    segment ,
    shouldGetExpectedSegmentationPointsFromPython )
```

Definition at line 11 of file segment_test.cpp.

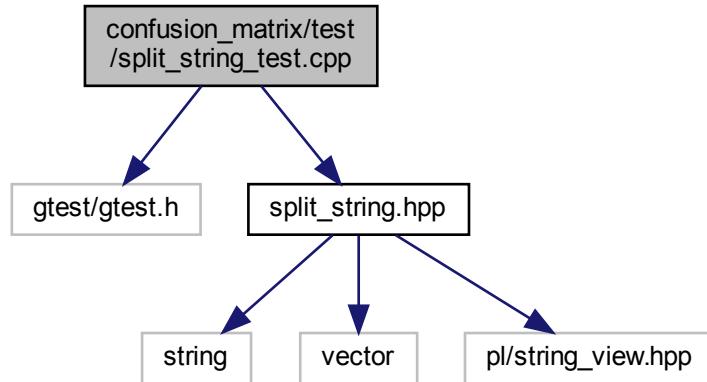
Here is the call graph for this function:



7.77 confusion_matrix/test/split_string_test.cpp File Reference

```
#include "gtest/gtest.h"
#include "split_string.hpp"
```

Include dependency graph for split_string_test.cpp:



Functions

- [TEST](#) (`splitString`, `shouldSplitString`)

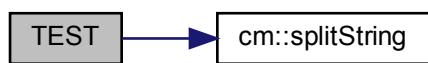
7.77.1 Function Documentation

7.77.1.1 TEST()

```
TEST (splitString , shouldSplitString )
```

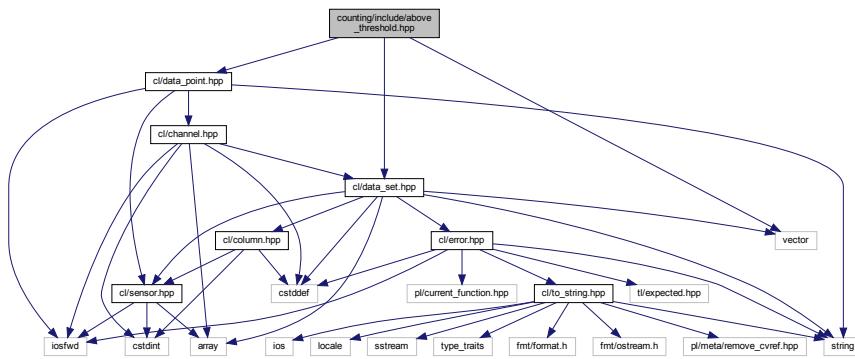
Definition at line 5 of file `split_string_test.cpp`.

Here is the call graph for this function:

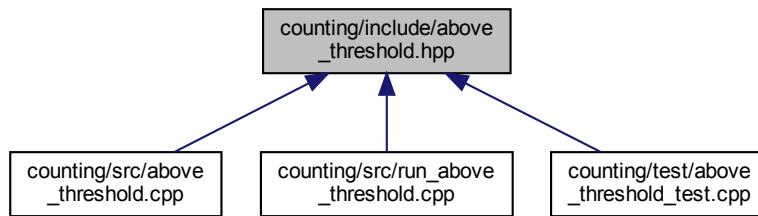


7.78 counting/include/above_threshold.hpp File Reference

```
#include <vector>
#include "cl/data_point.hpp"
#include "cl/data_set.hpp"
Include dependency graph for above_threshold.hpp:
```



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



Namespaces

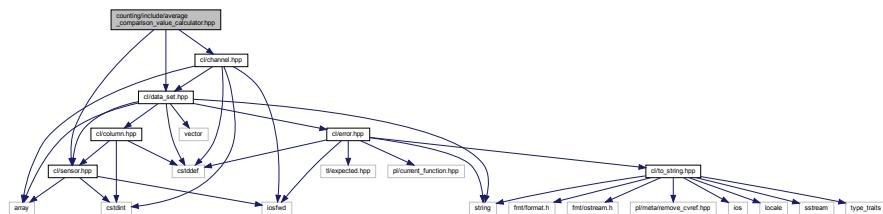
- `ctg`

Functions

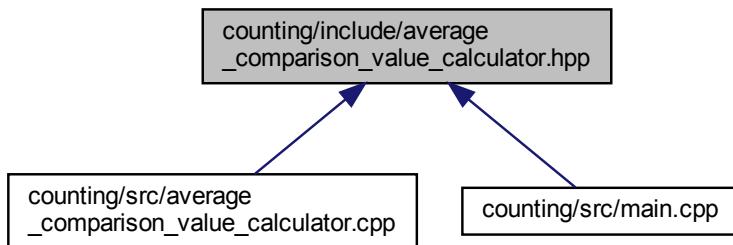
- `std::vector< cl::DataPoint > ctg::aboveThreshold (const cl::DataSet &dataSet, long double accelerometerThreshold, long double gyroscopeThreshold)`

7.79 counting/include/average_comparison_value_calculator.hpp File Reference

```
#include "cl/channel.hpp"
#include "cl/data_set.hpp"
#include "cl/sensor.hpp"
Include dependency graph for average_comparison_value_calculator.hpp:
```



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



Namespaces

- `ctg`

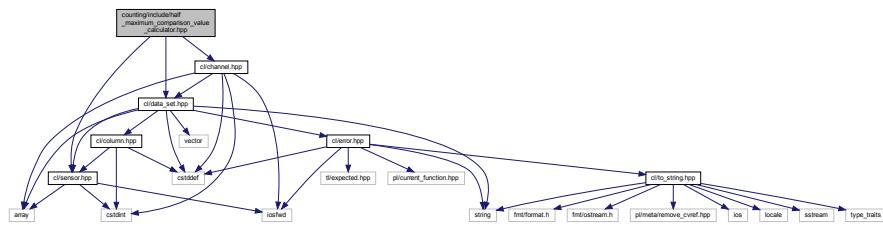
Functions

- `long double ctg::averageComparisonValueCalculator (cl::Sensor sensor, cl::Channel channel, const cl::DataSet &dataSet)`

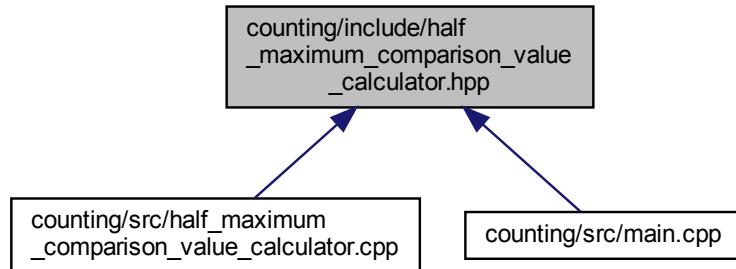
7.80 counting/include/half_maximum_comparison_value_calculator.hpp File Reference

```
#include "cl/channel.hpp"
#include "cl/data_set.hpp"
```

```
#include "cl/sensor.hpp"
Include dependency graph for half_maximum_comparison_value_calculator.hpp:
```



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



Namespaces

- ctg

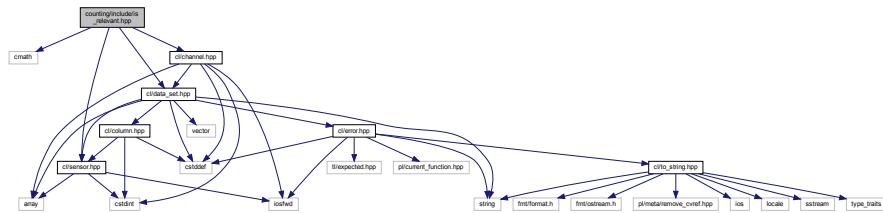
Functions

- long double `ctg::halfMaximumComparisonValueCalculator` (`cl::Sensor sensor, cl::Channel channel, const cl::DataSet &dataSet`)

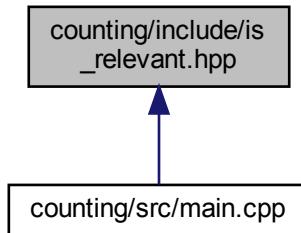
7.81 counting/include/is_relevant.hpp File Reference

```
#include <cmath>
#include "cl/channel.hpp"
#include "cl/data_set.hpp"
```

```
#include "cl/sensor.hpp"
Include dependency graph for is_relevant.hpp:
```



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



Namespaces

- `ctg`

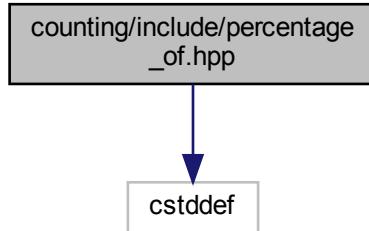
Functions

- template<typename ComparisonValueCalculator >
`bool ctg::isRelevant (cl::Sensor sensor, cl::Channel channel, const cl::DataSet &dataSet, ComparisonValueCalculator comparisonValueCalculator)`

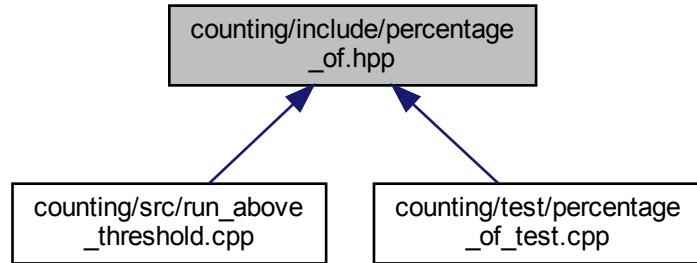
7.82 counting/include/percentage_of.hpp File Reference

```
#include <cstddef>
```

Include dependency graph for percentage_of.hpp:



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



Namespaces

- [ctg](#)

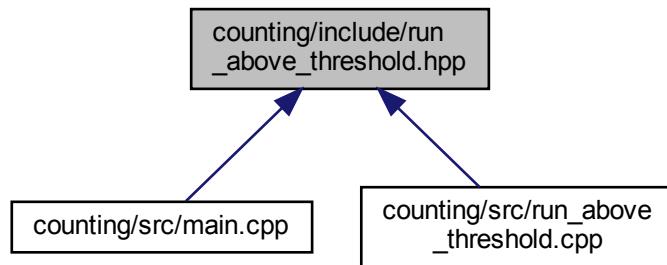
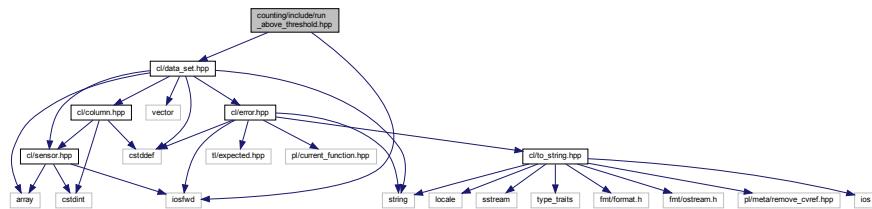
Functions

- `constexpr long double ctg::percentageOf (std::size_t amount, std::size_t totalCount) noexcept`

7.83 counting/include/run_above_threshold.hpp File Reference

```
#include <iostream>
#include "cl/data_set.hpp"
```

Include dependency graph for run_above_threshold.hpp:



Namespaces

- `ctg`

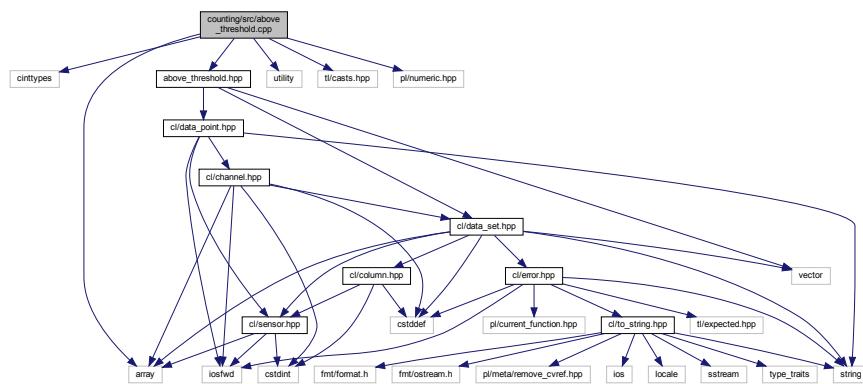
Functions

- void `ctg::runAboveThreshold` (`std::ostream &aboveThresholdLogFileStream, const cl::DataSet &dataSet`)

7.84 counting/src/above_threshold.cpp File Reference

```
#include <cinttypes>
#include <array>
#include <utility>
#include <tl/casts.hpp>
#include <pl/numeric.hpp>
```

```
#include "above_threshold.hpp"
Include dependency graph for above_threshold.cpp:
```



Namespaces

- `ctg`

Macros

- `#define CL_CHANNEL_X(enm, v, accessor) {accessor, cl::Channel::enm},`

Functions

- `std::vector< cl::DataPoint > ctg::aboveThreshold (const cl::DataSet &dataSet, long double accelerometerThreshold, long double gyroscopeThreshold)`

7.84.1 Macro Definition Documentation

7.84.1.1 CL_CHANNEL_X

```
#define CL_CHANNEL_X(
    enm,
    v,
    accessor ) {accessor, cl::Channel::enm},
```

7.84.2 Variable Documentation

7.84.2.1 channel

```
cl::Channel channel
```

Definition at line 18 of file above_threshold.cpp.

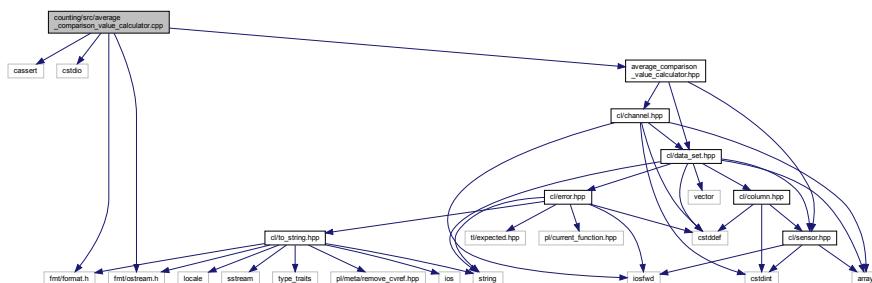
7.84.2.2 channelAccessor

```
cl::DataSet::ChannelAccessor channelAccessor
```

Definition at line 17 of file above_threshold.cpp.

7.85 counting/src/average_comparison_value_calculator.cpp File Reference

```
#include <cassert>
#include <cstdio>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include "average_comparison_value_calculator.hpp"
Include dependency graph for average_comparison_value_calculator.cpp:
```



Namespaces

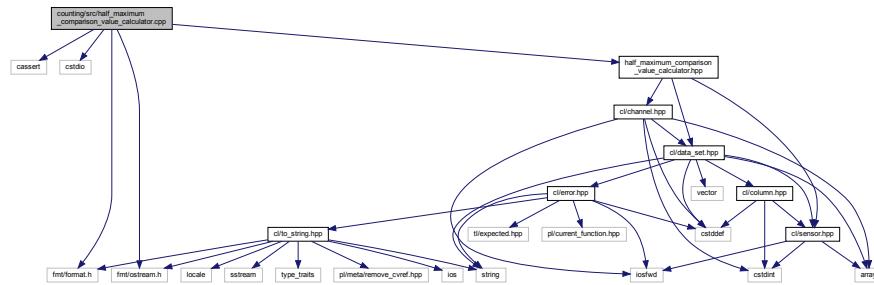
- `ctg`

Functions

- long double `ctg::averageComparisonValueCalculator (cl::Sensor sensor, cl::Channel channel, const cl::DataSet &dataSet)`

7.86 counting/src/half_maximum_comparison_value_calculator.cpp File Reference

```
#include <cassert>
#include <cstdio>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include "half_maximum_comparison_value_calculator.hpp"
Include dependency graph for half_maximum_comparison_value_calculator.cpp:
```



Namespaces

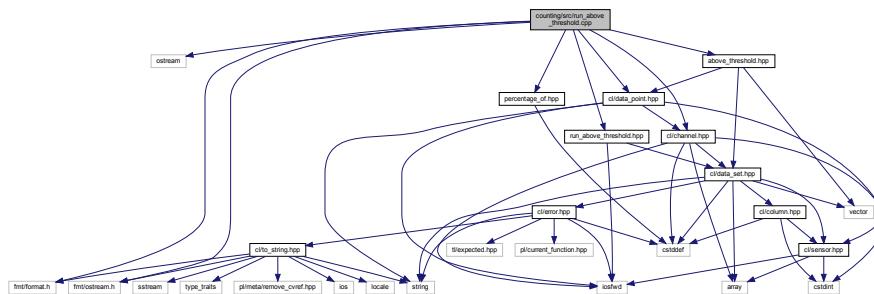
- `ctg`

Functions

- long double `ctg::halfMaximumComparisonValueCalculator (cl::Sensor sensor, cl::Channel channel, const cl::DataSet &dataSet)`

7.87 counting/src/run_above_threshold.cpp File Reference

```
#include <iostream>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include "cl/channel.hpp"
#include "cl/data_point.hpp"
#include "above_threshold.hpp"
#include "percentage_of.hpp"
#include "run_above_threshold.hpp"
Include dependency graph for run_above_threshold.cpp:
```



Namespaces

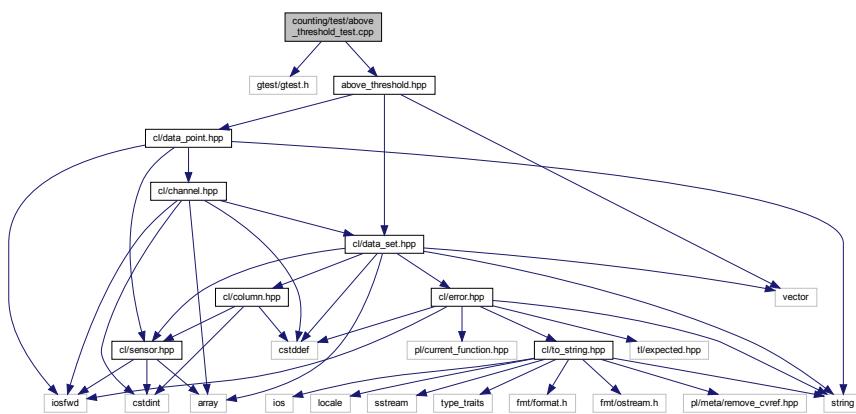
- `ctg`

Functions

- void `ctg::runAboveThreshold` (`std::ostream &aboveThresholdLogFileStream, const cl::DataSet &dataSet)`

7.88 counting/test/above_threshold_test.cpp File Reference

```
#include "gtest/gtest.h"
#include "above_threshold.hpp"
Include dependency graph for above_threshold_test.cpp:
```



Macros

- `#define EXPECT_LONG_DOUBLE_EQ(a, b) EXPECT_DOUBLE_EQ(static_cast<double>(a), static_cast<double>(b))`

Functions

- `TEST` (`aboveThreshold, shouldFindDataPointsIfThereAreAny`)

7.88.1 Macro Definition Documentation

7.88.1.1 EXPECT_LONG_DOUBLE_EQ

```
#define EXPECT_LONG_DOUBLE_EQ( a, b ) EXPECT_DOUBLE_EQ( static_cast<double>(a), static_cast<double>(b) )
```

Definition at line 6 of file above_threshold_test.cpp.

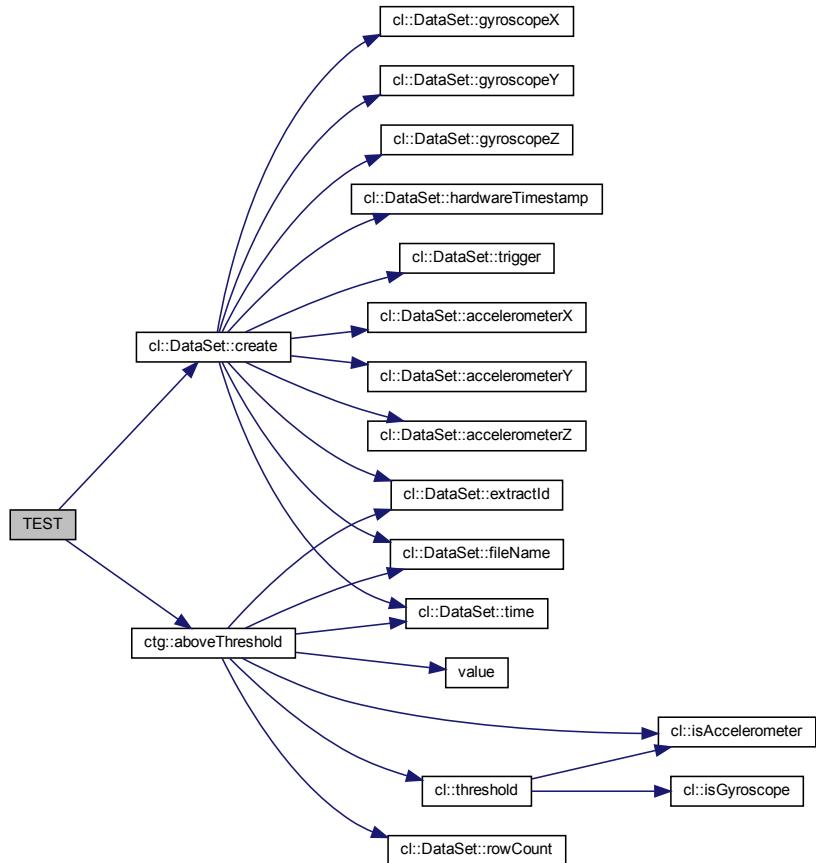
7.88.2 Function Documentation

7.88.2.1 TEST()

```
TEST( aboveThreshold , shouldFindDataPointsIfThereAreAny )
```

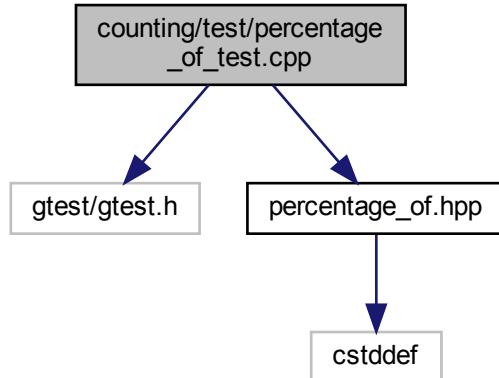
Definition at line 10 of file above_threshold_test.cpp.

Here is the call graph for this function:



7.89 counting/test/percentage_of_test.cpp File Reference

```
#include "gtest/gtest.h"
#include "percentage_of.hpp"
Include dependency graph for percentage_of_test.cpp:
```



Macros

- `#define EXPECT_LONG_DOUBLE_EQ(a, b) EXPECT_DOUBLE_EQ(static_cast<double>(a), static_cast<double>(b))`

Functions

- `TEST(percentageOf, shouldWork)`

7.89.1 Macro Definition Documentation

7.89.1.1 EXPECT_LONG_DOUBLE_EQ

```
#define EXPECT_LONG_DOUBLE_EQ(
    a,
    b ) EXPECT_DOUBLE_EQ(static_cast<double>(a), static_cast<double>(b))
```

Definition at line 6 of file percentage_of_test.cpp.

7.89.2 Function Documentation

7.89.2.1 TEST()

```
TEST ( percentageOf ,  
       shouldWork )
```

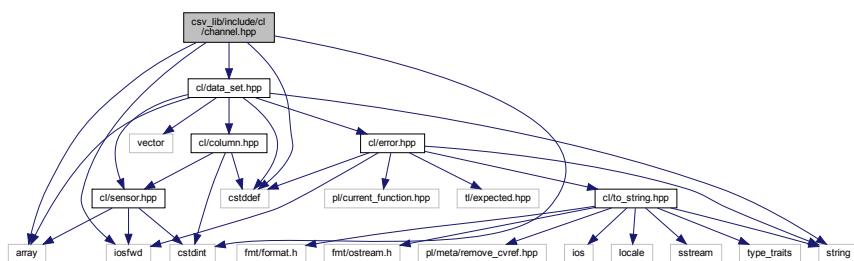
Definition at line 10 of file percentage_of_test.cpp.

Here is the call graph for this function:

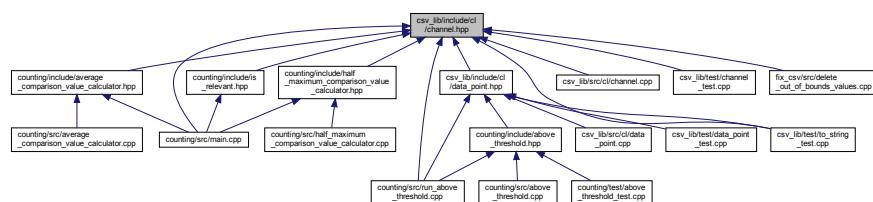


7.90 csv_lib/include/cl/channel.hpp File Reference

```
#include <cstddef>
#include <cstdint>
#include <array>
#include <iostream>
#include "cl/data_set.hpp"
Include dependency graph for channel.hpp:
```



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



Classes

- struct `cl::data_set_accessor< Chan >`

Namespaces

- `cl`

Macros

- `#define CL_CHANNEL`
- `#define CL_CHANNEL_X(enumerator, value, dataSetAccessor) enumerator = value,`
- `#define CL_CHANNEL_X(enumerator, value, dataSetAccessor) +1`
- `#define CL_CHANNEL_X(enm, v, a) ::cl::Channel::enm,`
- `#define CL_CHANNEL_X(enumerator, value, dataSetAccessor)`

Enumerations

- enum `cl::Channel : std::uint64_t { CL_CHANNEL, CL_CHANNEL }`

Functions

- `DataSet::ChannelAccessor cl::dataSetAccessor (Channel channel)`
- `std::ostream & cl::operator<< (std::ostream &os, Channel channel)`
- `bool cl::isAccelerometer (Channel channel)`
- `bool cl::isGyroscope (Channel channel)`
- `long double cl::threshold (Channel channel)`

Variables

- `constexpr std::size_t cl::channelCount`
- `constexpr std::array< Channel, channelCount > cl::channels`
- `template<Channel Chan>`
`constexpr CL_CHANNEL DataSet::ChannelAccessor cl::data_set_accessor_v = data_set_accessor<Chan>::f`
- `constexpr long double cl::accelerometerThreshold {1.99L}`
- `constexpr long double cl::gyroscopeThreshold {1999.99L}`

7.90.1 Macro Definition Documentation

7.90.1.1 CL_CHANNEL

```
#define CL_CHANNEL
```

Value:

```
CL_CHANNEL_X(AccelerometerX, 1, &::cl::DataSet::accelerometerX) \
CL_CHANNEL_X(AccelerometerY, 2, &::cl::DataSet::accelerometerY) \
CL_CHANNEL_X(AccelerometerZ, 3, &::cl::DataSet::accelerometerZ) \
CL_CHANNEL_X(GyroscopeX, 4, &::cl::DataSet::gyroscopeX) \
CL_CHANNEL_X(GyroscopeY, 5, &::cl::DataSet::gyroscopeY) \
CL_CHANNEL_X(GyroscopeZ, 6, &::cl::DataSet::gyroscopeZ)
```

Definition at line 11 of file channel.hpp.

7.90.1.2 CL_CHANNEL_X [1/4]

```
#define CL_CHANNEL_X(
    enm,
    v,
    a ) ::cl::Channel::enm,
```

Definition at line 41 of file channel.hpp.

7.90.1.3 CL_CHANNEL_X [2/4]

```
#define CL_CHANNEL_X(
    enumerator,
    value,
    dataSetAccessor ) enumerator = value,
```

Definition at line 41 of file channel.hpp.

7.90.1.4 CL_CHANNEL_X [3/4]

```
#define CL_CHANNEL_X(
    enumerator,
    value,
    dataSetAccessor ) +1
```

Definition at line 41 of file channel.hpp.

7.90.1.5 CL_CHANNEL_X [4/4]

```
#define CL_CHANNEL_X(enumerator, value, dataSetAccessor )
```

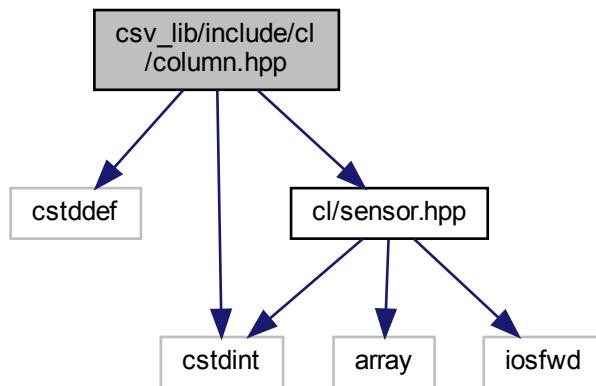
Value:

```
template<>
struct data_set_accessor<Channel::enumerator> {
    static constexpr ::cl::DataSet::ChannelAccessor f = dataSetAccessor; \
};
```

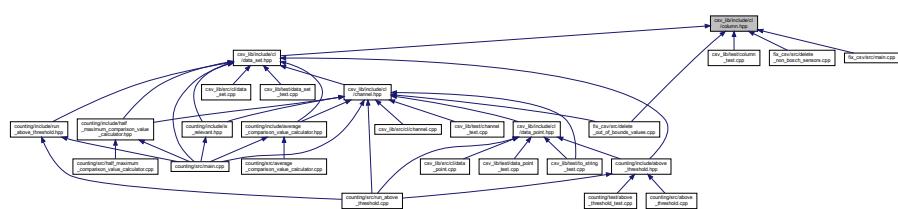
Definition at line 41 of file channel.hpp.

7.91 csv_lib/include/cl/column.hpp File Reference

```
#include <cstdint>
#include "cl/sensor.hpp"
Include dependency graph for column.hpp:
```



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



Classes

- struct `cl::col_traits< Col >`

Namespaces

- `cl`

Macros

- `#define CL_SPECIALIZE_COL_TRAITS(column, columnType)`

Typedefs

- template<Column Col>
using `cl::column_type` = typename `col_traits< Col >::type`

Enumerations

- enum `cl::Column` : `std::size_t` {
`cl::Column::Time, cl::Column::HardwareTimestamp, cl::Column::ExtractId, cl::Column::Trigger,`
`cl::Column::AccelerometerX, cl::Column::AccelerometerY, cl::Column::AccelerometerZ, cl::Column::GyroscopeX,`
`cl::Column::GyroscopeY, cl::Column::GyroscopeZ, cl::Column::SamplingRate }`

Functions

- `cl::CL_SPECIALIZE_COL_TRAITS (Column::Time, long double)`
- `cl::CL_SPECIALIZE_COL_TRAITS (Column::HardwareTimestamp, std::uint64_t)`
- `cl::CL_SPECIALIZE_COL_TRAITS (Column::ExtractId, Sensor)`
- `cl::CL_SPECIALIZE_COL_TRAITS (Column::Trigger, std::uint64_t)`
- `cl::CL_SPECIALIZE_COL_TRAITS (Column::AccelerometerX, long double)`
- `cl::CL_SPECIALIZE_COL_TRAITS (Column::AccelerometerY, long double)`
- `cl::CL_SPECIALIZE_COL_TRAITS (Column::AccelerometerZ, long double)`
- `cl::CL_SPECIALIZE_COL_TRAITS (Column::GyroscopeX, long double)`
- `cl::CL_SPECIALIZE_COL_TRAITS (Column::GyroscopeY, long double)`
- `cl::CL_SPECIALIZE_COL_TRAITS (Column::GyroscopeZ, long double)`
- `cl::CL_SPECIALIZE_COL_TRAITS (Column::SamplingRate, std::uint64_t)`

Variables

- template<Column Col>
constexpr `std::size_t cl::column_index` = `col_traits<Col>::index`

7.91.1 Macro Definition Documentation

7.91.1.1 CL_SPECIALIZE_COL_TRAITS

```
#define CL_SPECIALIZE_COL_TRAITS( \
    column, \
    columnType )
```

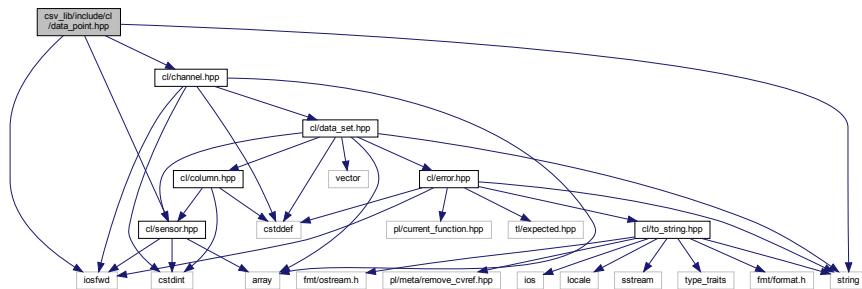
Value:

```
template<> \
struct col_traits<column> { \
    static constexpr std::size_t index = static_cast<std::size_t>(column); \
    using type = columnType; \
}
```

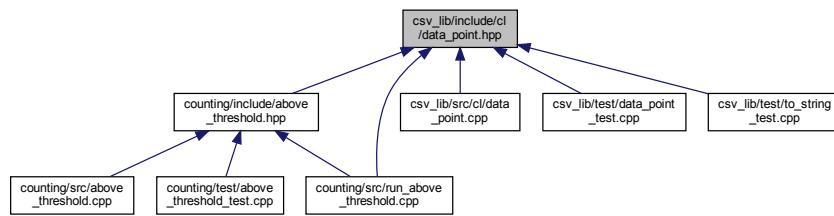
Definition at line 26 of file column.hpp.

7.92 csv_lib/include/cl/data_point.hpp File Reference

```
#include <iostream>
#include <string>
#include "cl/channel.hpp"
#include "cl/sensor.hpp"
Include dependency graph for data_point.hpp:
```



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



Classes

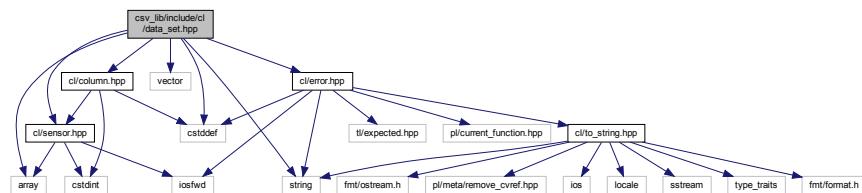
- class `cl::DataPoint`

Namespaces

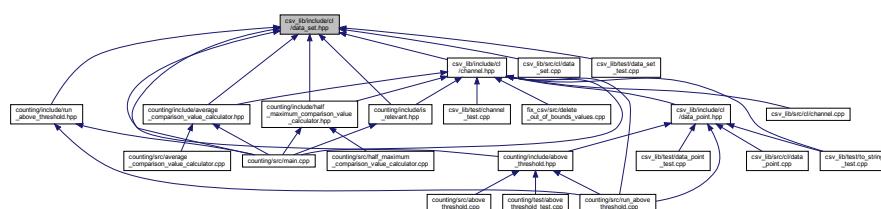
- cl

7.93 csv_lib/include/cl/data_set.hpp File Reference

```
#include <cstddef>
#include <array>
#include <string>
#include <vector>
#include "cl/column.hpp"
#include "cl/error.hpp"
#include "cl/sensor.hpp"
Include dependency graph for data_set.hpp
```



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



Classes

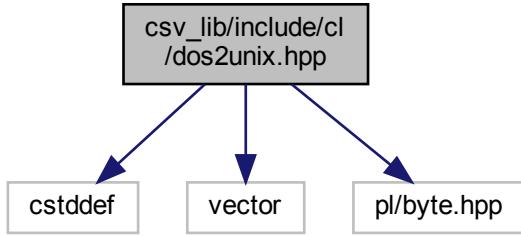
- class cl::DataSet

Namespaces

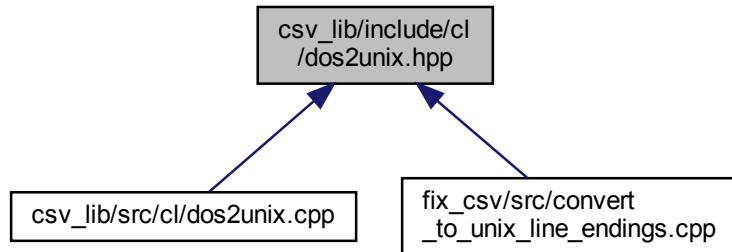
- C

7.94 csv_lib/include/cl/dos2unix.hpp File Reference

```
#include <cstddef>
#include <vector>
#include <pl/byte.hpp>
Include dependency graph for dos2unix.hpp:
```



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



Namespaces

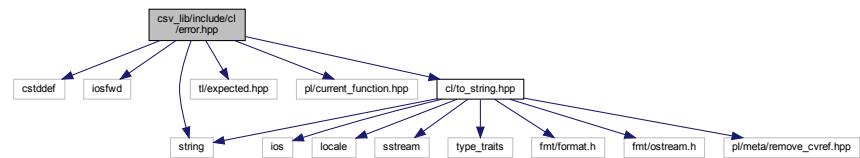
- `cl`

Functions

- `std::vector< pl::byte > cl::dos2unix (const void *p, std::size_t size)`
Converts DOS / Microsoft Windows line endings to UNIX line endings.

7.95 csv_lib/include/cl/error.hpp File Reference

```
#include <cstddef>
#include <iostream>
#include <string>
#include <tl/expected.hpp>
#include <pl/current_function.hpp>
#include "cl/to_string.hpp"
Include dependency graph for error.hpp:
```



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



Classes

- class [cl::Error](#)

Namespaces

- [cl](#)

Macros

- `#define CL_ERROR_KIND`
- `#define CL_ERROR_KIND_X(kind) kind,`
- `#define CL_UNEXPECTED(kind, message)`

Typedefs

- template<typename Ty >
using [cl::Expected](#) = tl::expected< Ty, Error >

7.95.1 Macro Definition Documentation

7.95.1.1 CL_ERROR_KIND

```
#define CL_ERROR_KIND
```

Value:

```
CL_ERROR_KIND_X(Filesystem) \
CL_ERROR_KIND_X(InvalidArgumentException) \
CL_ERROR_KIND_X(OutOfRange) \
CL_ERROR_KIND_X(Parsing) \
CL_ERROR_KIND_X(Logic) \
CL_ERROR_KIND_X(OperatingSystem)
```

Definition at line 14 of file error.hpp.

7.95.1.2 CL_ERROR_KIND_X

```
#define CL_ERROR_KIND_X(
    kind ) kind,
```

Definition at line 27 of file error.hpp.

7.95.1.3 CL_UNEXPECTED

```
#define CL_UNEXPECTED (
    kind,
    message )
```

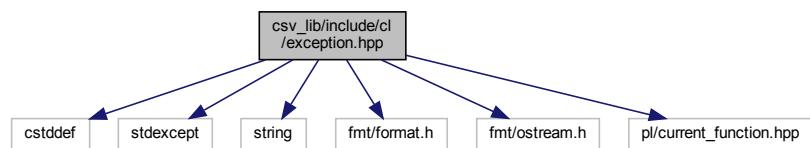
Value:

```
::tl::make_unexpected(
    ::cl::Error{kind, __FILE__, PL_CURRENT_FUNCTION, __LINE__, message})
```

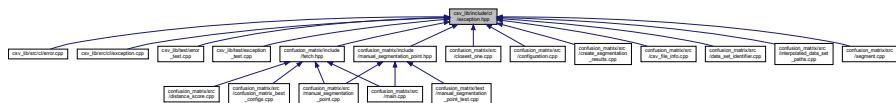
Definition at line 67 of file error.hpp.

7.96 csv_lib/include/cl/exception.hpp File Reference

```
#include <cstddef>
#include <stdexcept>
#include <string>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <pl/current_function.hpp>
Include dependency graph for exception.hpp:
```



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



Classes

- class cl::Exception

Namespaces

- cl

Macros

- `#define CL_THROW(what_arg) throw ::cl::Exception { __FILE__, PL_CURRENT_FUNCTION, __LINE__, what_arg }`
 - `#define CL_THROW_FMT(fmt_str, ...) CL_THROW(::fmt::format(fmt_str, __VA_ARGS__))`

7.96.1 Macro Definition Documentation

7.96.1.1 CL THROW

```
#define CL_THROW( what_arg ) throw ::cl::Exception { __FILE__, PL_CURRENT_FUNCTION, __LINE__←
, what_arg }
```

Definition at line 42 of file exception.hpp.

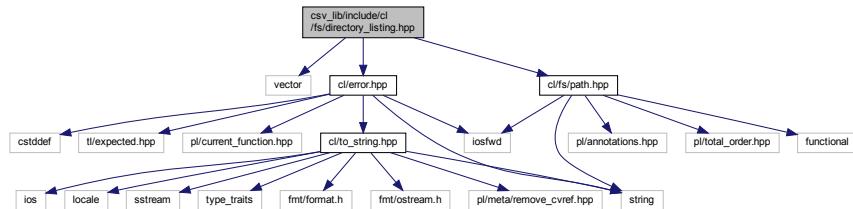
7.96.1.2 CL THROW FMT

```
#define CL_THROW_FMT(          fmt_str,          ... ) CL_THROW(::fmt::format(fmt_str, VA_ARGS ))
```

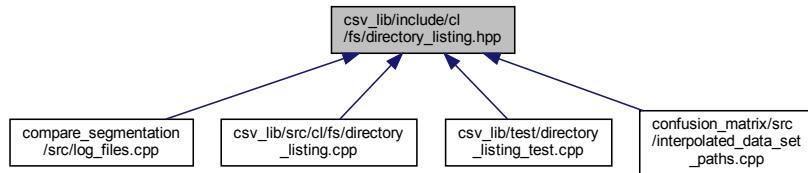
Definition at line 45 of file exception.hpp.

7.97 csv_lib/include/cl/fs/directory_listing.hpp File Reference

```
#include <vector>
#include <cl/error.hpp>
#include <cl/fs/path.hpp>
Include dependency graph for directory_listing.hpp:
```



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



Namespaces

- `cl`
- `cl::fs`

Enumerations

- enum `cl::fs::DirectoryListingOption` { `cl::fs::DirectoryListingOption::None`, `cl::fs::DirectoryListingOption::ExcludeDotAndDotDot` }

Options for directoryListing.

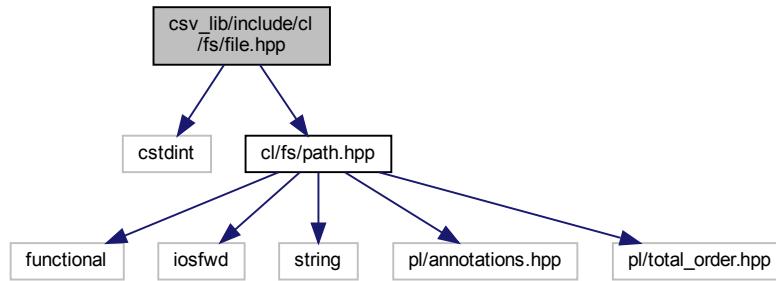
Functions

- `Expected< std::vector< Path > > cl::fs::directoryListing (const Path &directoryPath, DirectoryListingOption directoryListingOption=DirectoryListingOption::ExcludeDotAndDotDot)`

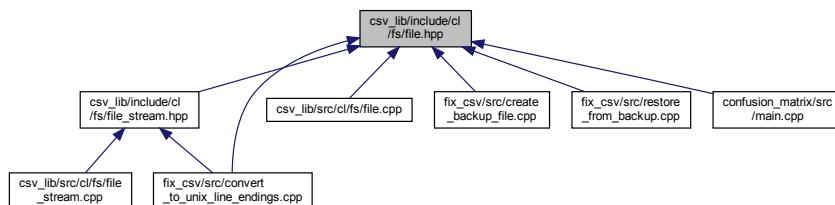
Creates a listing of the contents of a directory.

7.98 csv_lib/include/cl/fs/file.hpp File Reference

```
#include <cstdint>
#include "cl/fs/path.hpp"
Include dependency graph for file.hpp:
```



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



Classes

- class `cl::fs::File`
Represents a file.

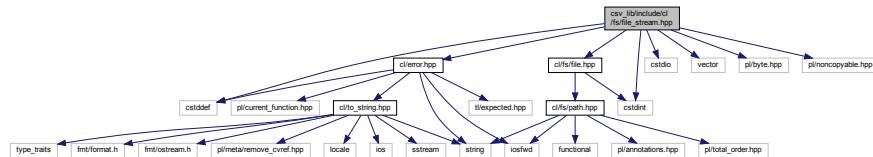
Namespaces

- `cl`
- `cl::fs`

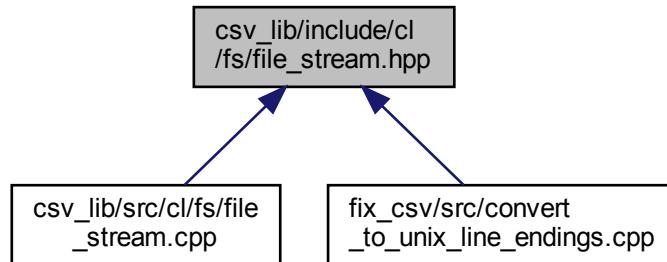
7.99 csv_lib/include/cl/fs/file_stream.hpp File Reference

```
#include <cstddef>
#include <cstdint>
#include <cstdio>
#include <vector>
```

```
#include <pl/byte.hpp>
#include <pl/noncopyable.hpp>
#include "cl/error.hpp"
#include "cl/fs/file.hpp"
Include dependency graph for file_stream.hpp:
```



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



Classes

- class [cl::fs::FileStream](#)

A binary file stream.

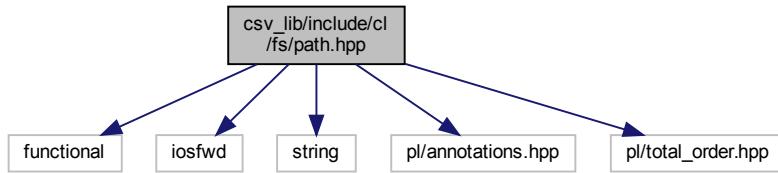
Namespaces

- [cl](#)
- [cl::fs](#)

7.100 csv_lib/include/cl/fs/path.hpp File Reference

```
#include <functional>
#include <iostream>
#include <string>
#include <pl/annotations.hpp>
```

```
#include <pl/total_order.hpp>
Include dependency graph for path.hpp:
```



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



Classes

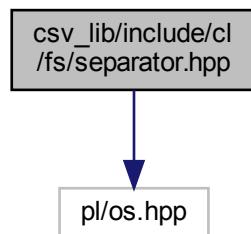
- class [cl::fs::Path](#)
A filesystem path.
- struct [std::hash<::cl::fs::Path >](#)

Namespaces

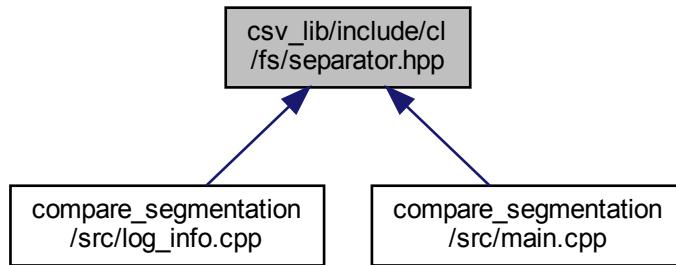
- [cl](#)
- [cl::fs](#)

7.101 csv_lib/include/cl/fs/sePARATOR.hpp File Reference

```
#include <pl/os.hpp>
Include dependency graph for separator.hpp:
```



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



Macros

- `#define CL_FS_SEPARATOR "\\\"`
The filesystem separator of the operating system.

7.101.1 Macro Definition Documentation

7.101.1.1 CL_FS_SEPARATOR

```
#define CL_FS_SEPARATOR "\\\"
```

The filesystem separator of the operating system.

Definition at line 11 of file separator.hpp.

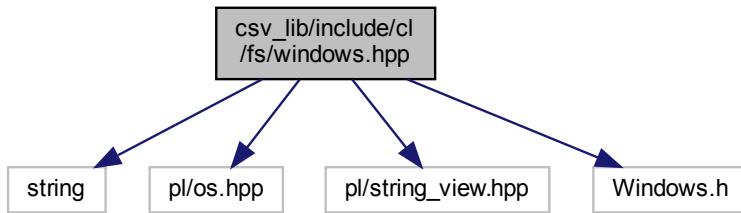
7.102 csv_lib/include/cl/fs/windows.hpp File Reference

Contains Microsoft Windows specific functions.

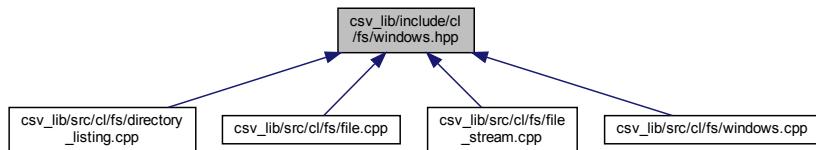
```
#include <string>
#include <pl/os.hpp>
#include <pl/string_view.hpp>
```

```
#include <Windows.h>
```

Include dependency graph for windows.hpp:



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



Namespaces

- [cl](#)
- [cl::fs](#)

Functions

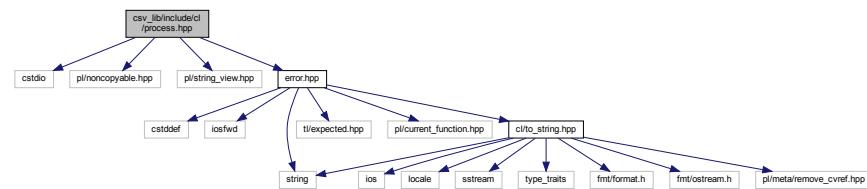
- std::wstring [cl::fs::utf8ToUtf16](#) (pl::string_view utf8)
Converts a UTF-8 encoded string to a UTF-16 encoded wstring.
- std::string [cl::fs::utf16ToUtf8](#) (pl::wstring_view utf16)
Converts a UTF-16 encoded wide character string to UTF-8 string.
- std::wstring [cl::fs::formatError](#) (DWORD errorCode)
Formats a WINAPI error code to a UTF-16 encoded wide character string.

7.102.1 Detailed Description

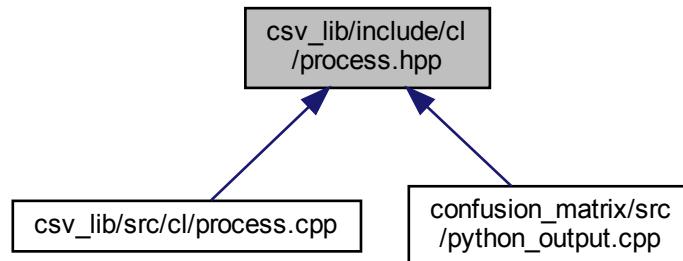
Contains Microsoft Windows specific functions.

7.103 csv_lib/include/cl/process.hpp File Reference

```
#include <cstdio>
#include <pl/noncopyable.hpp>
#include <pl/string_view.hpp>
#include "error.hpp"
Include dependency graph for process.hpp:
```



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



Classes

- class [cl::Process](#)

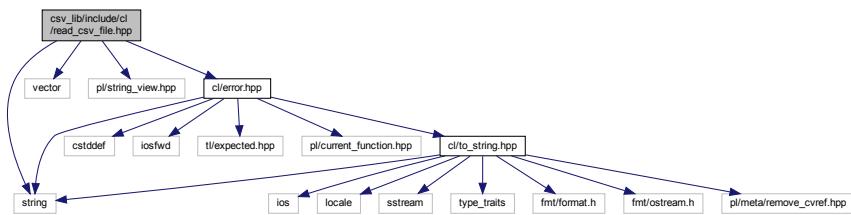
Namespaces

- [cl](#)

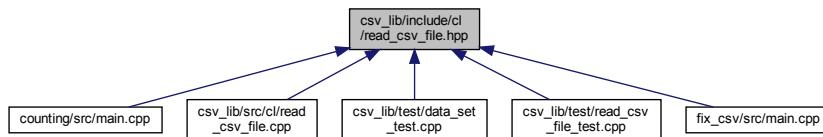
7.104 csv_lib/include/cl/read_csv_file.hpp File Reference

```
#include <string>
#include <vector>
#include <pl/string_view.hpp>
```

```
#include "cl/error.hpp"
Include dependency graph for read_csv_file.hpp:
```



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



Namespaces

- `cl`

Enumerations

- enum `cl::CsvFileKind` { `cl::CsvFileKind::Raw`, `cl::CsvFileKind::Fixed` }

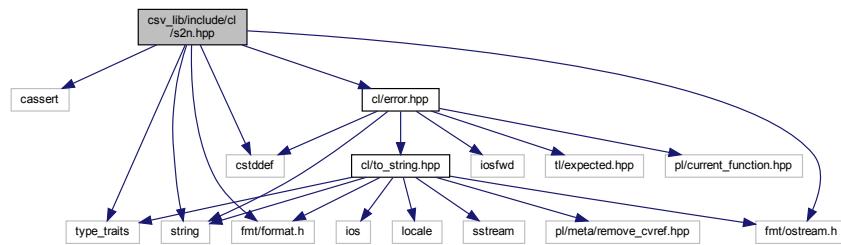
Functions

- `Expected< std::vector< std::vector< std::string > > > cl::readCsvFile(pl::string_view csvFilePath, std::vector< std::string > *columnNames=nullptr, CsvFileKind csvFileKind=CsvFileKind::Fixed) noexcept`

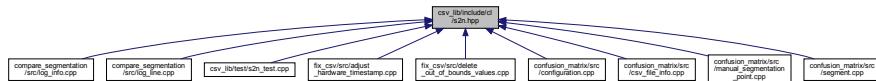
7.105 csv_lib/include/cl/s2n.hpp File Reference

```
#include <cassert>
#include <cstdint>
#include <string>
#include <type_traits>
#include <fmt/format.h>
#include <fmt/ostream.h>
```

```
#include "cl/error.hpp"
Include dependency graph for s2n.hpp:
```



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



Namespaces

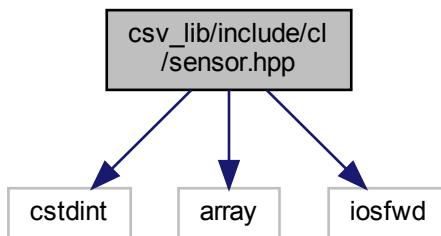
- `cl`

Functions

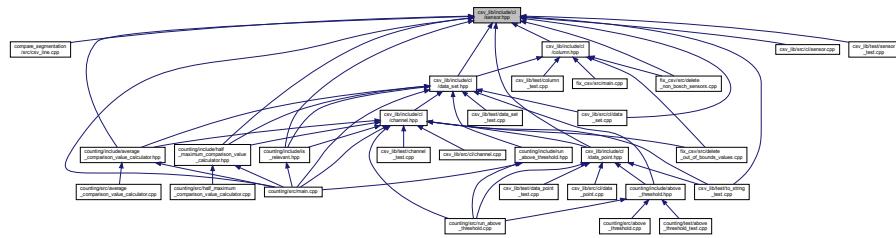
- template<typename Integer >
Expected< Integer > `cl::s2n` (const std::string &str, std::size_t *pos=nullptr, [[maybe_unused]] int base=10)

7.106 csv_lib/include/cl/sensor.hpp File Reference

```
#include <cstdint>
#include <array>
#include <iostfwd>
Include dependency graph for sensor.hpp:
```



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



Namespaces

- cl

Macros

- `#define CL_SENSOR`
 - `#define CL_SENSOR_X(enumerator, value) enumerator = value,`
 - `#define CL_SENSOR_X(enm, v) ::cl::Sensor::enm,`

Enumerations

- enum cl::Sensor : std::uint64_t { cl::Sensor::CL SENSOR_X, cl::Sensor::CL SENSOR }

Functions

- std::ostream & `cl::operator<<` (std::ostream &os, Sensor `sensor`)

Variables

- `constexpr std::array< Sensor, 4 > cl::sensors`

7.106.1 Macro Definition Documentation

7.106.1.1 CL SENSOR

```
#define CL_SENSOR
```

Value:

```
CL_SENSOR_X(LeftArm, 769) \
CL_SENSOR_X(Belly, 770) \
CL_SENSOR_X(RightArm, 771) \
CL_SENSOR_X(Chest, 772)
```

Definition at line 9 of file sensor.hpp.

7.106.1.2 CL_SENSOR_X [1/2]

```
#define CL_SENSOR_X(
    enm,
    v ) ::cl::Sensor::enm,
```

Definition at line 16 of file sensor.hpp.

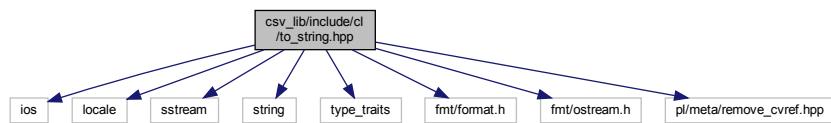
7.106.1.3 CL_SENSOR_X [2/2]

```
#define CL_SENSOR_X(
    enumerator,
    value ) enumerator = value,
```

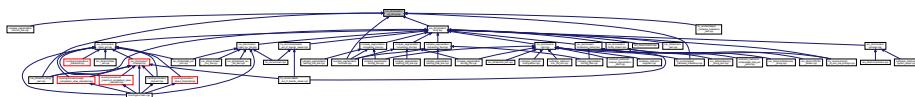
Definition at line 16 of file sensor.hpp.

7.107 csv_lib/include/cl/to_string.hpp File Reference

```
#include <iostream>
#include <locale>
#include <iostream>
#include <string>
#include <type_traits>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <pl/meta/remove_cvref.hpp>
Include dependency graph for to_string.hpp:
```



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



Namespaces

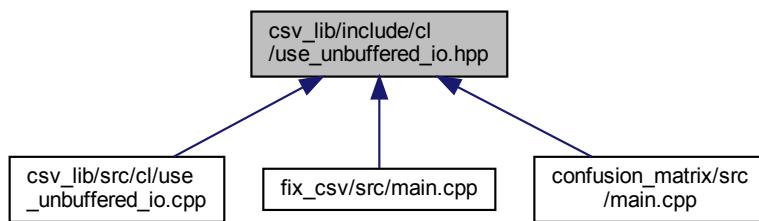
- cl

Functions

- template<typename Ty >
std::string [cl::to_string](#) (const Ty &ty)

7.108 csv_lib/include/cl/use_unbuffered_io.hpp File Reference

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



Namespaces

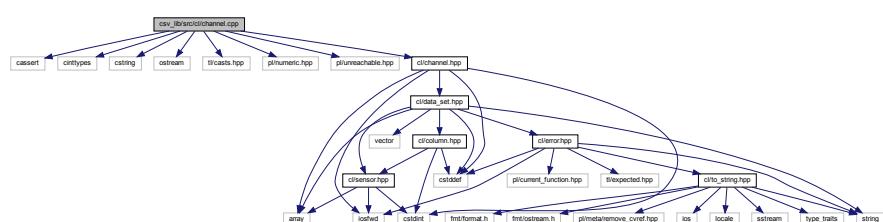
- [cl](#)

Functions

- void [cl::useUnbufferedIo\(\)](#)

7.109 csv_lib/src/cl/channel.cpp File Reference

```
#include <cassert>
#include <cinttypes>
#include <cstring>
#include <iostream>
#include <t1/casts.hpp>
#include <p1/numeric.hpp>
#include <p1/unreachable.hpp>
#include "cl/channel.hpp"
Include dependency graph for channel.cpp:
```



Namespaces

- `cl`

Macros

- `#define CL_CHANNEL_X(enm, v, acc) case Channel::enm: return data_set_accessor_v<Channel::enm>;`
- `#define CL_CHANNEL_X(enumerator, value, dataSetAccessor) case Channel::enumerator: return os << #enumerator;`

Functions

- `DataSet::ChannelAccessor cl::dataSetAccessor (Channel channel)`
- `std::ostream & cl::operator<< (std::ostream &os, Channel channel)`
- `bool cl::isAccelerometer (Channel channel)`
- `bool cl::isGyroscope (Channel channel)`
- `long double cl::threshold (Channel channel)`

7.109.1 Macro Definition Documentation

7.109.1.1 CL_CHANNEL_X [1/2]

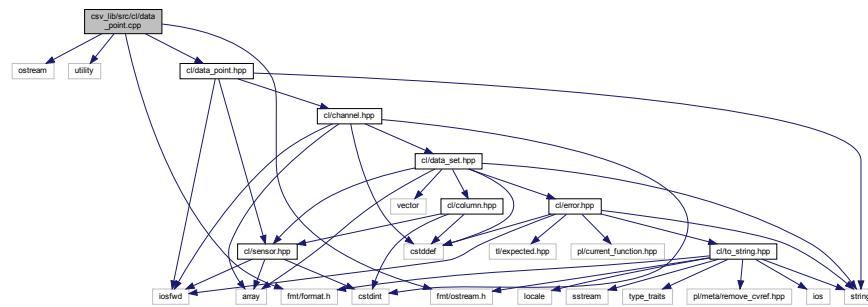
```
#define CL_CHANNEL_X(
    enm,
    v,
    acc ) case Channel::enm: return data_set_accessor_v<Channel::enm>;
```

7.109.1.2 CL_CHANNEL_X [2/2]

```
#define CL_CHANNEL_X(
    enumerator,
    value,
    dataSetAccessor ) case Channel::enumerator: return os << #enumerator;
```

7.110 csv_lib/src/cl/data_point.cpp File Reference

```
#include <iostream>
#include <utility>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include "cl/data_point.hpp"
Include dependency graph for data_point.cpp:
```



Namespaces

- `cl`

Functions

- `std::ostream & cl::operator<< (std::ostream &os, const DataPoint &dataPoint)`
- `dataPoint fileName ()`
- `dataPoint dataPoint time ()`
- `dataPoint dataPoint dataPoint sensor ()`
- `dataPoint dataPoint dataPoint dataPoint channel ()`
- `dataPoint dataPoint dataPoint dataPoint value ()`

7.110.1 Function Documentation

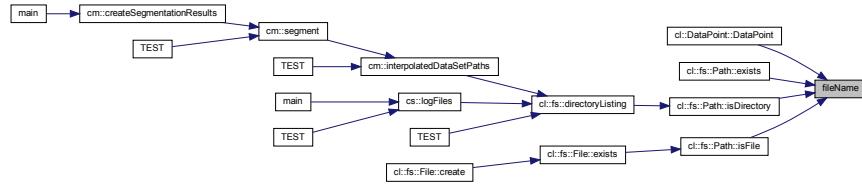
7.110.1.1 channel()

```
dataPoint dataPoint dataPoint dataPoint channel ( )
```

7.110.1.2 fileName()

```
dataPoint fileName ( )
```

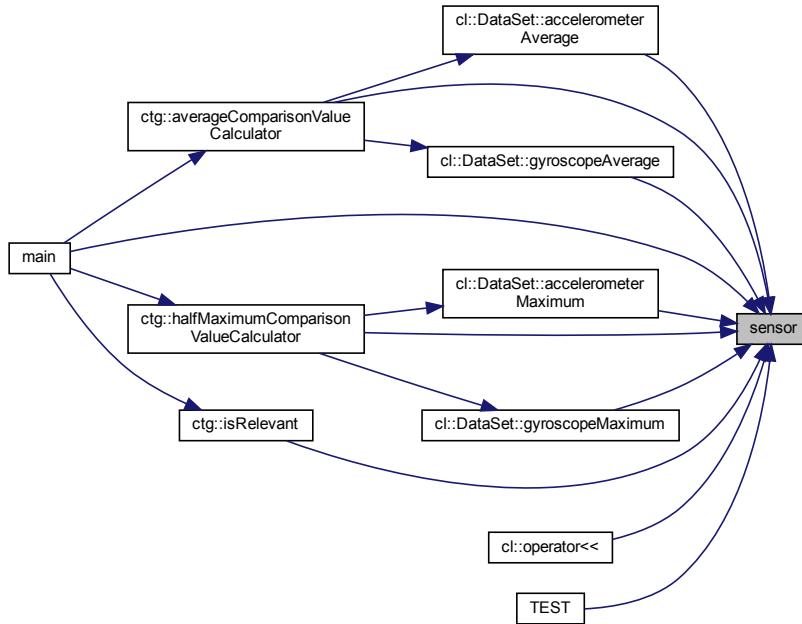
Here is the caller graph for this function:



7.110.1.3 sensor()

```
dataPoint dataPoint dataPoint sensor ( )
```

Here is the caller graph for this function:



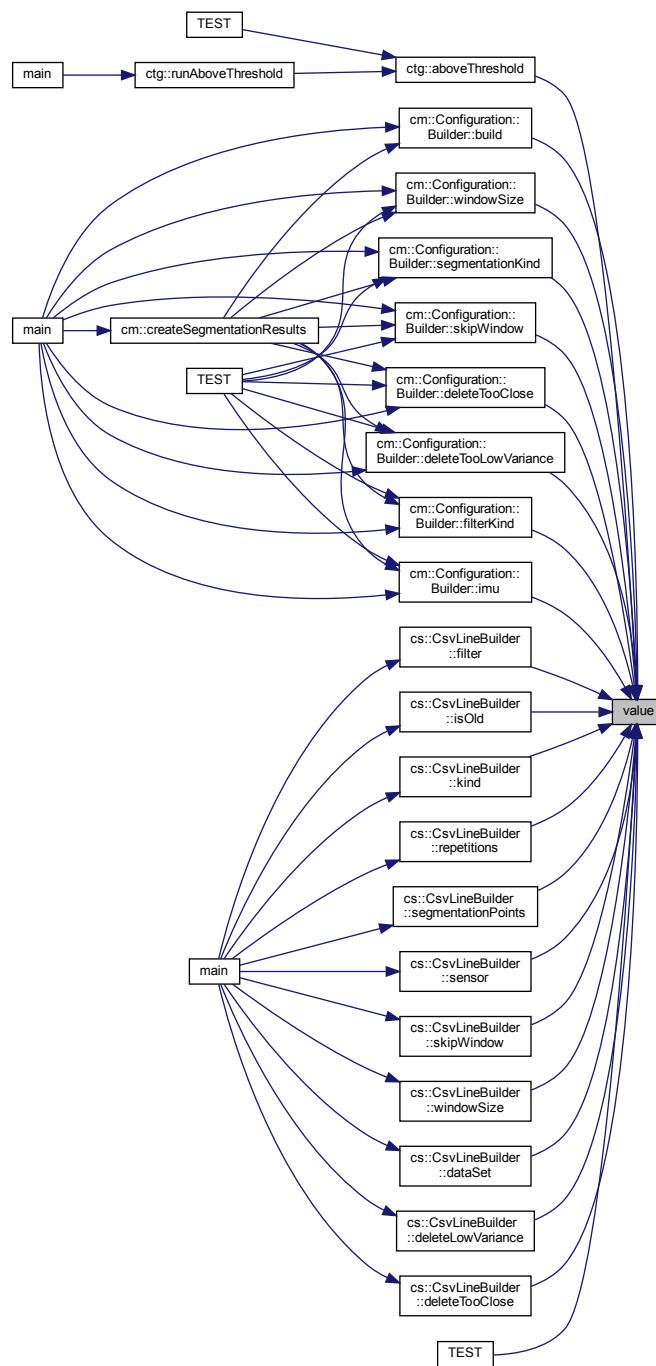
7.110.1.4 time()

```
dataPoint dataPoint time ( )
```

7.110.1.5 value()

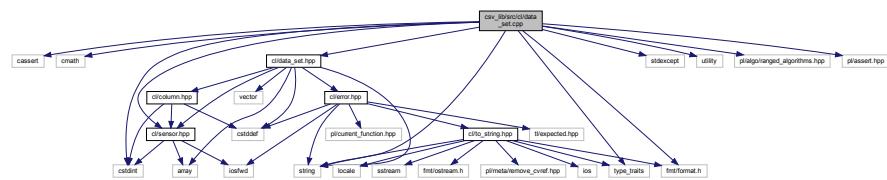
```
dataPoint dataPoint dataPoint dataPoint value ( )
```

Here is the caller graph for this function:



7.111 csv_lib/src/cl/data_set.cpp File Reference

```
#include <cassert>
#include <cmath>
#include <cstdint>
#include <stdexcept>
#include <string>
#include <type_traits>
#include <utility>
#include <fmt/format.h>
#include <pl/algo/ranged_algorithms.hpp>
#include <pl/assert.hpp>
#include "cl/data_set.hpp"
#include "cl/sensor.hpp"
Include dependency graph for data_set.cpp:
```

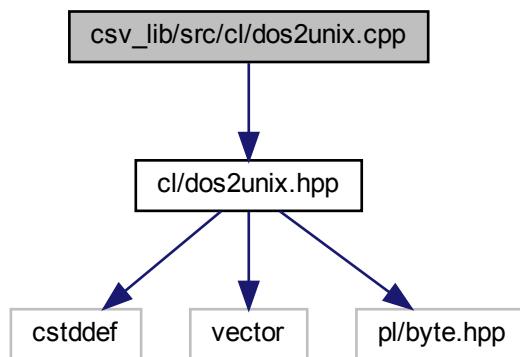


Namespaces

- c|

7.112 csv_lib/src/cl/dos2unix.cpp File Reference

```
#include "cl/dos2unix.hpp"
Include dependency graph for dos2unix.cpp:
```



Namespaces

- [cl](#)

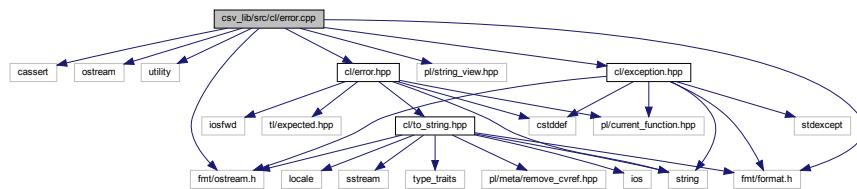
Functions

- `std::vector< pl::byte > cl::dos2unix (const void *p, std::size_t size)`

Converts DOS / Microsoft Windows line endings to UNIX line endings.

7.113 csv_lib/src/cl/error.cpp File Reference

```
#include <cassert>
#include <iostream>
#include <utility>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <pl/string_view.hpp>
#include "cl/error.hpp"
#include "cl/exception.hpp"
Include dependency graph for error.cpp:
```



Namespaces

- [cl](#)

Macros

- `#define CL_ERROR_KIND_X(kind) case Error::kind: return #kind;`

Functions

- `std::ostream & cl::operator<< (std::ostream &os, const Error &error)`

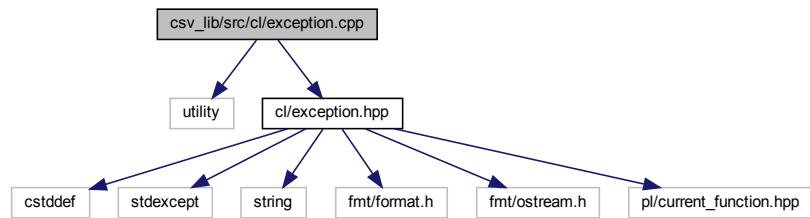
7.113.1 Macro Definition Documentation

7.113.1.1 CL_ERROR_KIND_X

```
#define CL_ERROR_KIND_X( kind ) case Error::kind: return #kind;
```

7.114 csv_lib/src/cl/exception.cpp File Reference

```
#include <utility>
#include "cl/exception.hpp"
Include dependency graph for exception.cpp:
```

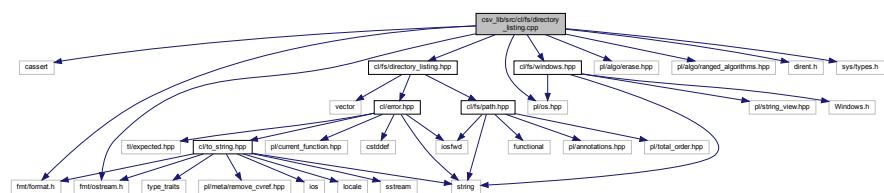


Namespaces

- cl

7.115 csv_lib/src/cl/fs/directory_listing.cpp File Reference

```
#include <cassert>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <pl/algo/erase.hpp>
#include <pl/algo/ranged_algorithms.hpp>
#include <pl/os.hpp>
#include <cl/fs/windows.hpp>
#include <dirent.h>
#include <sys/types.h>
#include <cl/fs/directory_listing.hpp>
Include dependency graph for directory_listing.cpp:
```



Namespaces

- `cl`
- `cl::fs`

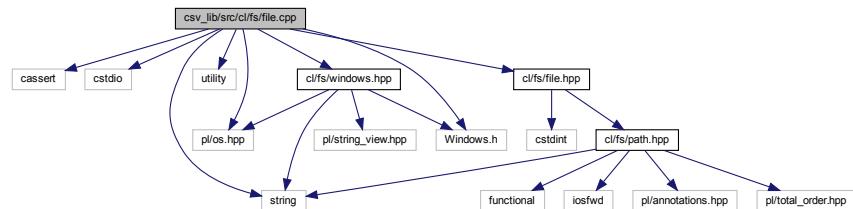
Functions

- Expected< std::vector< Path > > `cl::fs::directoryListing` (const Path &directoryPath, DirectoryListingOption directoryListingOption=DirectoryListingOption::ExcludeDotAndDotDot)

Creates a listing of the contents of a directory.

7.116 csv_lib/src/cl/fs/file.cpp File Reference

```
#include <cassert>
#include <cstdio>
#include <string>
#include <utility>
#include <pl/os.hpp>
#include "cl/fs/windows.hpp"
#include <Windows.h>
#include "cl/fs/file.hpp"
Include dependency graph for file.cpp:
```



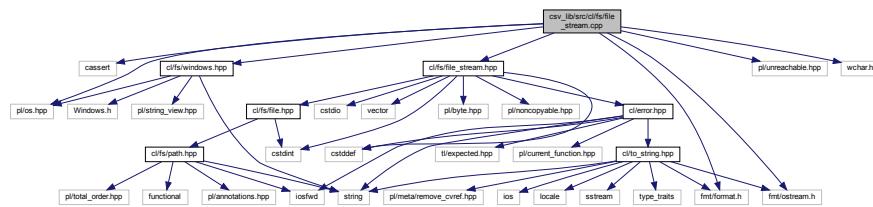
Namespaces

- `cl`
- `cl::fs`

7.117 csv_lib/src/cl/fs/file_stream.cpp File Reference

```
#include <cassert>
#include <pl/os.hpp>
#include <pl/unreachable.hpp>
#include "cl/fs/windows.hpp"
#include <wchar.h>
#include <fmt/format.hpp>
#include <fmt/ostream.hpp>
```

```
#include "cl/fs/file_stream.hpp"
Include dependency graph for file_stream.cpp:
```

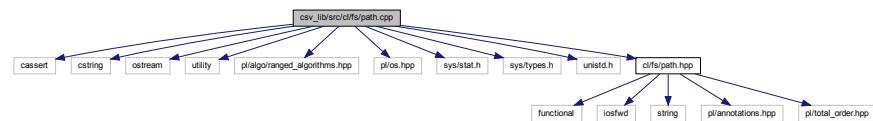


Namespaces

- `cl`
- `cl::fs`

7.118 csv_lib/src/cl/fs/path.cpp File Reference

```
#include <cassert>
#include <cstring>
#include <ostream>
#include <utility>
#include <pl/algo/ranged_algorithms.hpp>
#include <pl/os.hpp>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>
#include "cl/fs/path.hpp"
Include dependency graph for path.cpp:
```



Namespaces

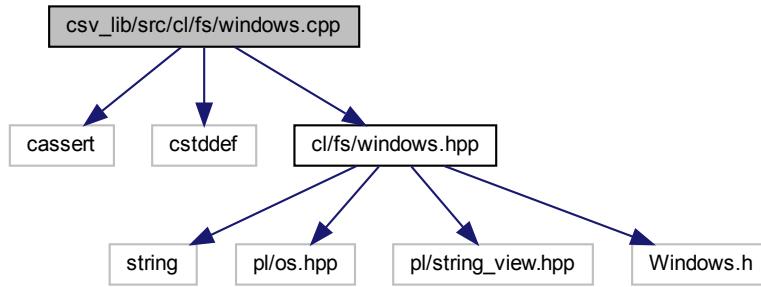
- `cl`
- `cl::fs`

Functions

- `std::ostream & cl::fs::operator<< (std::ostream &os, const Path &path)`
- `bool cl::fs::operator< (const Path &lhs, const Path &rhs) noexcept`
- `bool cl::fs::operator== (const Path &lhs, const Path &rhs) noexcept`

7.119 csv_lib/src/cl/fs/windows.cpp File Reference

```
#include <cassert>
#include <cstddef>
#include "cl/fs/windows.hpp"
Include dependency graph for windows.cpp:
```



Namespaces

- [cl](#)
- [cl::fs](#)

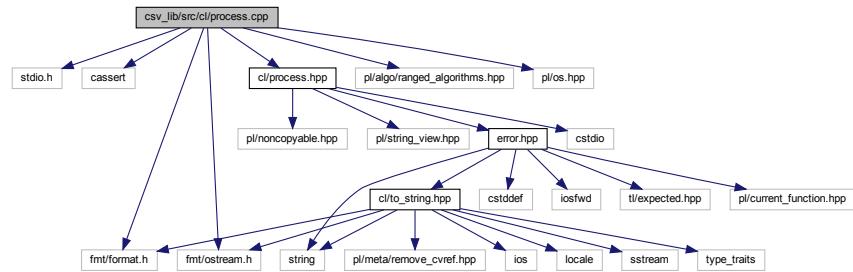
Functions

- std::wstring [cl::fs::utf8ToUtf16](#) (pl::string_view utf8)
Converts a UTF-8 encoded string to a UTF-16 encoded wstring.
- std::string [cl::fs::utf16ToUtf8](#) (pl::wstring_view utf16)
Converts a UTF-16 encoded wide character string to UTF-8 string.
- std::wstring [cl::fs::formatError](#) (DWORD errorCode)
Formats a WINAPI error code to a UTF-16 encoded wide character string.

7.120 csv_lib/src/cl/process.cpp File Reference

```
#include <stdio.h>
#include <cassert>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <pl/algo/ranged_algorithms.hpp>
#include <pl/os.hpp>
```

```
#include "cl/process.hpp"
Include dependency graph for process.cpp:
```

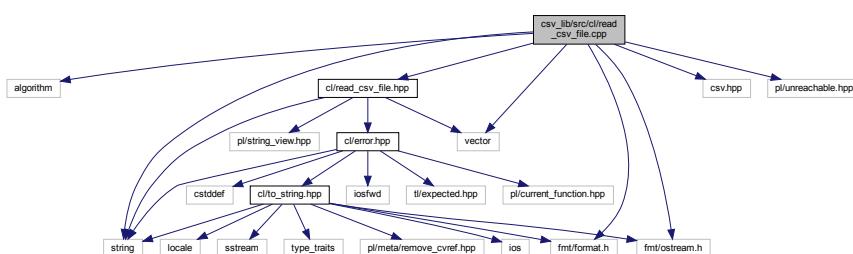


Namespaces

- `cl`

7.121 csv_lib/src/cl/read_csv_file.cpp File Reference

```
#include <algorithm>
#include <string>
#include <vector>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <csv.hpp>
#include <pl/unreachable.hpp>
#include "cl/read_csv_file.hpp"
Include dependency graph for read_csv_file.cpp:
```



Namespaces

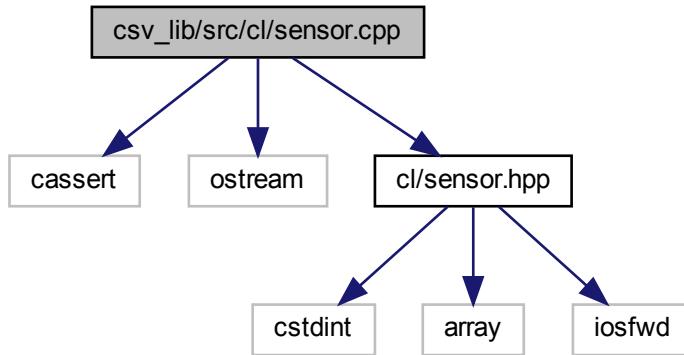
- `cl`

Functions

- `Expected< std::vector< std::vector< std::string > > > cl::readCsvFile(pl::string_view csvFilePath, std::vector< std::string > *columnNames=nullptr, CsvFileKind csvFileKind=CsvFileKind::Fixed) noexcept`

7.122 csv_lib/src/cl/sensor.cpp File Reference

```
#include <cassert>
#include <ostream>
#include "cl/sensor.hpp"
Include dependency graph for sensor.cpp:
```



Namespaces

- `cl`

Macros

- `#define CL_SENSOR_X(enumerator, value) case Sensor::enumerator: return os << #enumerator;`

Functions

- `std::ostream & cl::operator<< (std::ostream &os, Sensor sensor)`

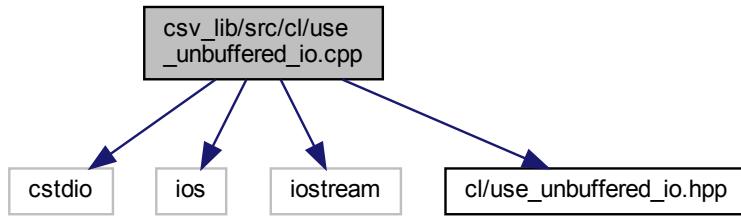
7.122.1 Macro Definition Documentation

7.122.1.1 CL_SENSOR_X

```
#define CL_SENSOR_X(
    enumerator,
    value ) case Sensor::enumerator: return os << #enumerator;
```

7.123 csv_lib/src/cl/use_unbuffered_io.cpp File Reference

```
#include <cstdio>
#include <iostream>
#include <iostream>
#include "cl/use_unbuffered_io.hpp"
Include dependency graph for use_unbuffered_io.cpp:
```



Namespaces

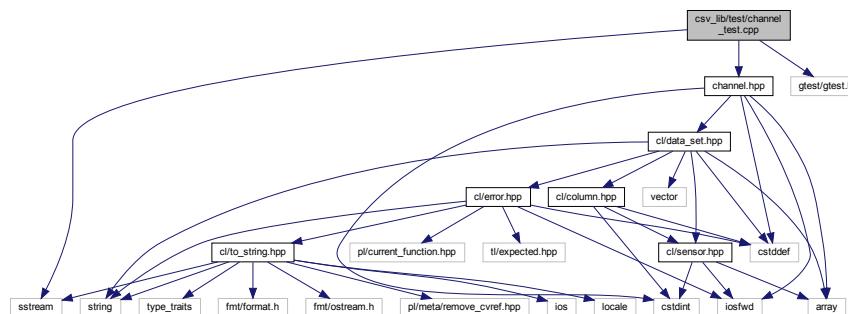
- `cl`

Functions

- `void cl::useUnbufferedIo ()`

7.124 csv_lib/test/channel_test.cpp File Reference

```
#include <sstream>
#include "gtest/gtest.h"
#include "channel.hpp"
Include dependency graph for channel_test.cpp:
```



Functions

- `TEST (channel, shouldHaveCorrectCount)`
- `TEST (channel, shouldHaveCorrectValues)`
- `TEST (channel, shouldPrintCorrectly)`
- `TEST (channel, shouldMapToCorrectDataSetAccessors)`

7.124.1 Function Documentation

7.124.1.1 TEST() [1/4]

```
TEST (
    channel ,
    shouldHaveCorrectCount )
```

Definition at line 7 of file channel_test.cpp.

7.124.1.2 TEST() [2/4]

```
TEST (
    channel ,
    shouldHaveCorrectValues )
```

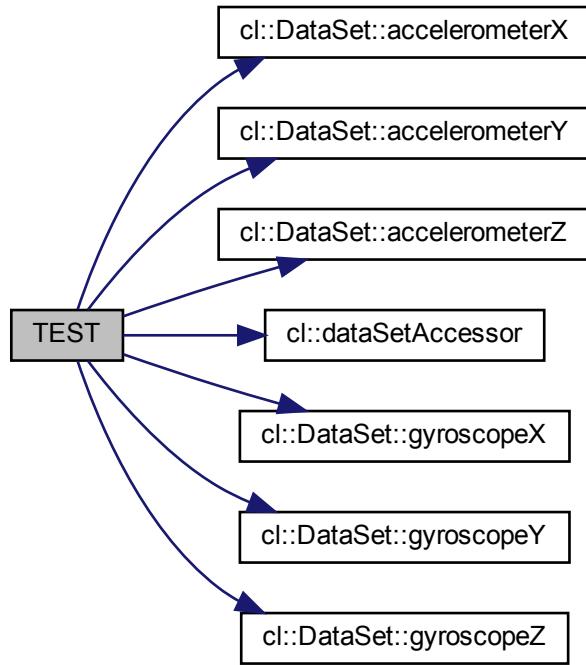
Definition at line 9 of file channel_test.cpp.

7.124.1.3 TEST() [3/4]

```
TEST (
    channel ,
    shouldMapToCorrectDataSetAccessors )
```

Definition at line 35 of file channel_test.cpp.

Here is the call graph for this function:



7.124.1.4 TEST() [4/4]

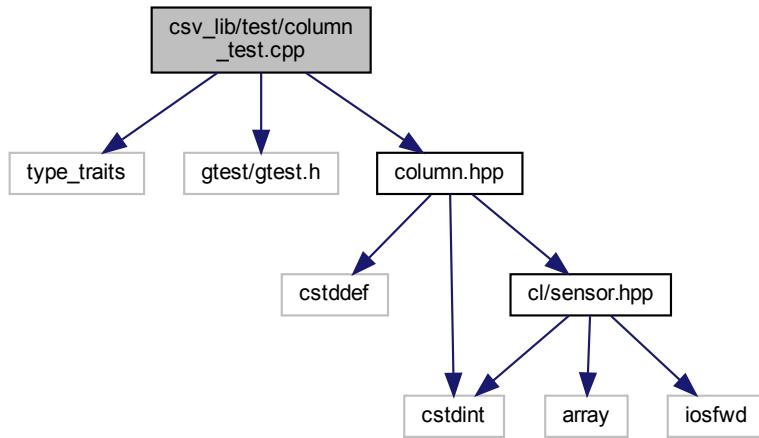
```
TEST (
    channel ,
    shouldPrintCorrectly )
```

Definition at line 19 of file channel_test.cpp.

7.125 csv_lib/test/column_test.cpp File Reference

```
#include <type_traits>
#include "gtest/gtest.h"
```

```
#include "column.hpp"
Include dependency graph for column_test.cpp:
```



Functions

- `TEST` (`column`, `shouldHaveCorrectIndex`)
- `TEST` (`column`, `shouldHaveCorrectColumnType`)

7.125.1 Function Documentation

7.125.1.1 TEST() [1/2]

```
TEST (
    column ,
    shouldHaveCorrectColumnType )
```

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

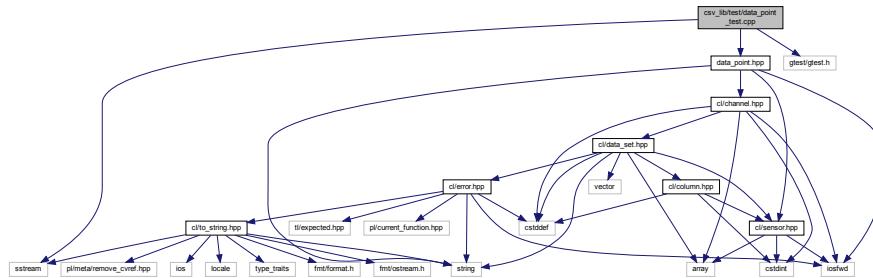
7.125.1.2 TEST() [2/2]

```
TEST (
    column ,
    shouldHaveCorrectIndex )
```

Definition at line 7 of file `column_test.cpp`.

7.126 csv_lib/test/data_point_test.cpp File Reference

```
#include <sstream>
#include "gtest/gtest.h"
#include "data_point.hpp"
Include dependency graph for data_point_test.cpp:
```



Functions

- [TEST](#) (`DataPoint`, `shouldPrintCorrectly`)
- [TEST](#) (`DataPoint`, `shouldGetValuesCorrectly`)

Variables

- const `cl::DataPoint dp`

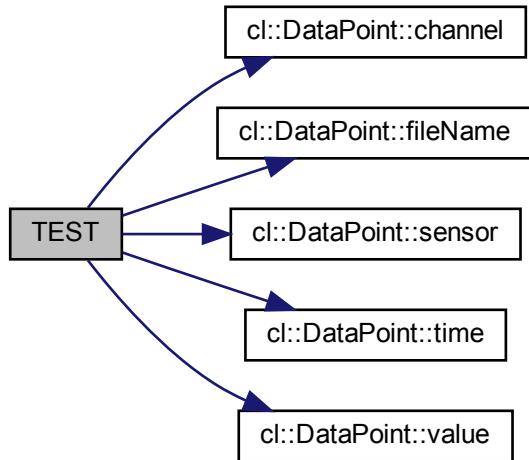
7.126.1 Function Documentation

7.126.1.1 TEST() [1/2]

```
TEST (
    DataPoint ,
    shouldGetValuesCorrectly )
```

Definition at line 23 of file `data_point_test.cpp`.

Here is the call graph for this function:



7.126.1.2 TEST() [2/2]

```
TEST (
    DataPoint ,
    shouldPrintCorrectly )
```

Definition at line 14 of file data_point_test.cpp.

7.126.2 Variable Documentation

7.126.2.1 dp

```
const cl::DataPoint dp
```

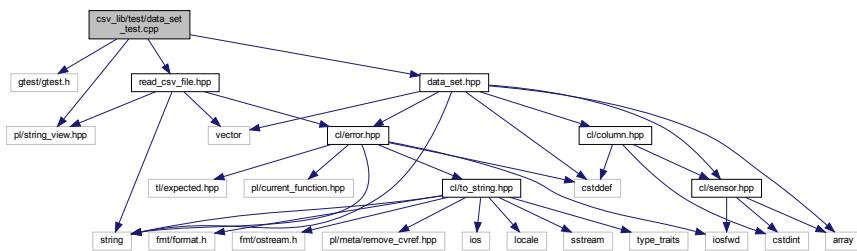
Initial value:

```
{
    "file.csv",
    0.01,
    cl::Sensor::Chest,
    cl::Channel::AccelerometerX,
    50.01}
```

Definition at line 7 of file data_point_test.cpp.

7.127 csv_lib/test/data_set_test.cpp File Reference

```
#include "gtest/gtest.h"
#include <pl/string_view.hpp>
#include "data_set.hpp"
#include "read_csv_file.hpp"
Include dependency graph for data_set_test.cpp:
```



Macros

- `#define EXPECT_LONG_DOUBLE_EQ(a, b) EXPECT_DOUBLE_EQ(static_cast<double>(a), static_cast<double>(b))`

Functions

- `TEST(DataSet, shouldBeAbleToCreateFromValidData)`
- `TEST(DataSet, shouldNotBeAbleToCreateFromEmptyMatrix)`
- `TEST(DataSet, shouldNotBeAbleToCreateFromJaggedMatrix)`
- `TEST(DataSet, shouldNotBeAbleToCreateFromInvalidData)`

7.127.1 Macro Definition Documentation

7.127.1.1 EXPECT_LONG_DOUBLE_EQ

```
#define EXPECT_LONG_DOUBLE_EQ(
    a,
    b ) EXPECT_DOUBLE_EQ(static_cast<double>(a), static_cast<double>(b))
```

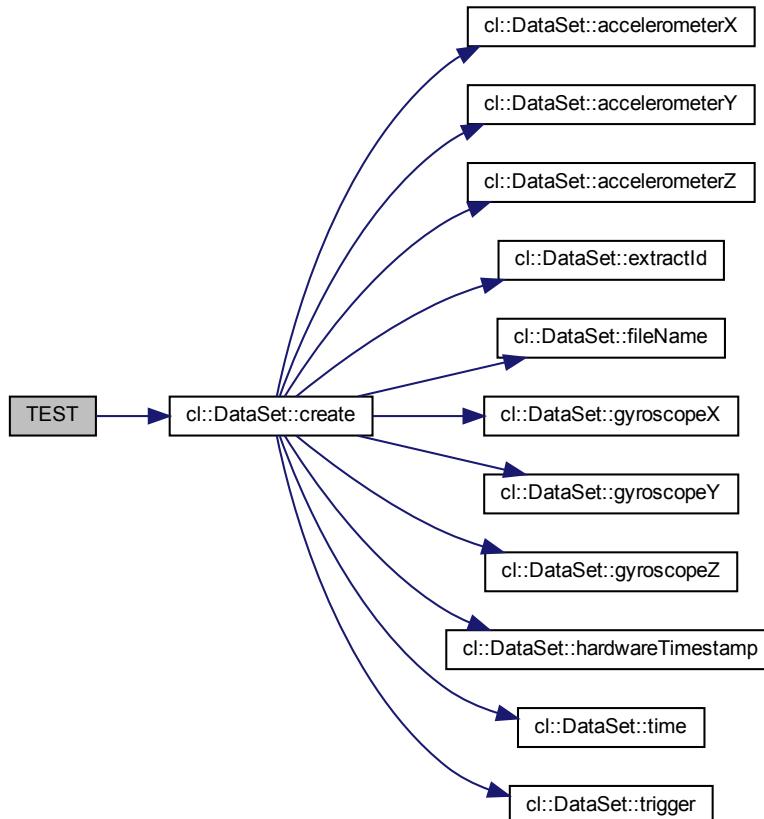
7.127.2 Function Documentation

7.127.2.1 TEST() [1/4]

```
TEST (
    DataSet ,
    shouldBeAbleToCreateFromValidData )
```

Definition at line 17 of file data_set_test.cpp.

Here is the call graph for this function:

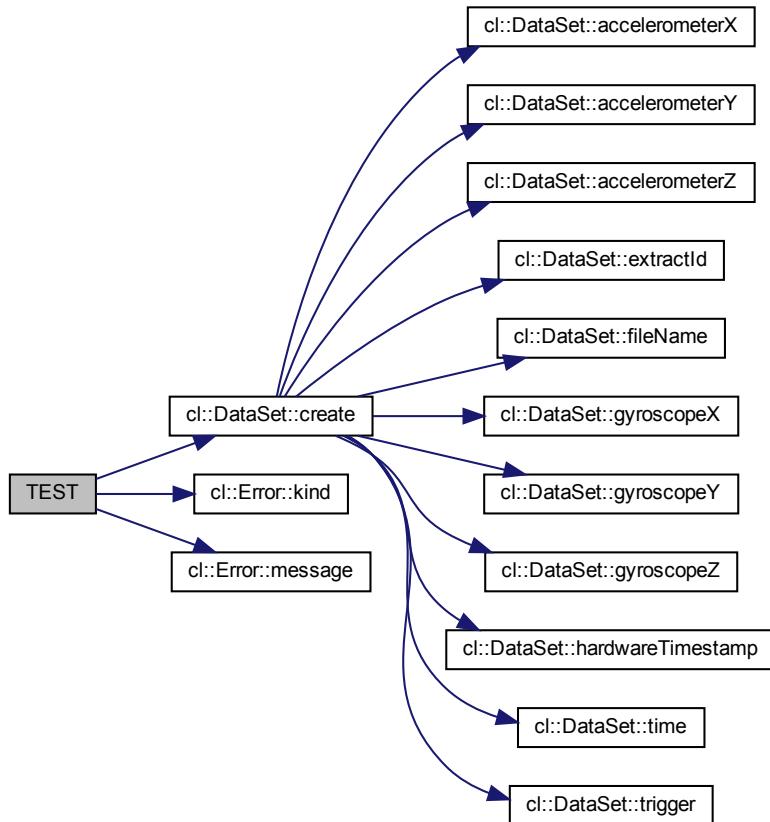


7.127.2.2 TEST() [2/4]

```
TEST (
    DataSet ,
    shouldNotBeAbleToCreateFromEmptyMatrix )
```

Definition at line 68 of file data_set_test.cpp.

Here is the call graph for this function:

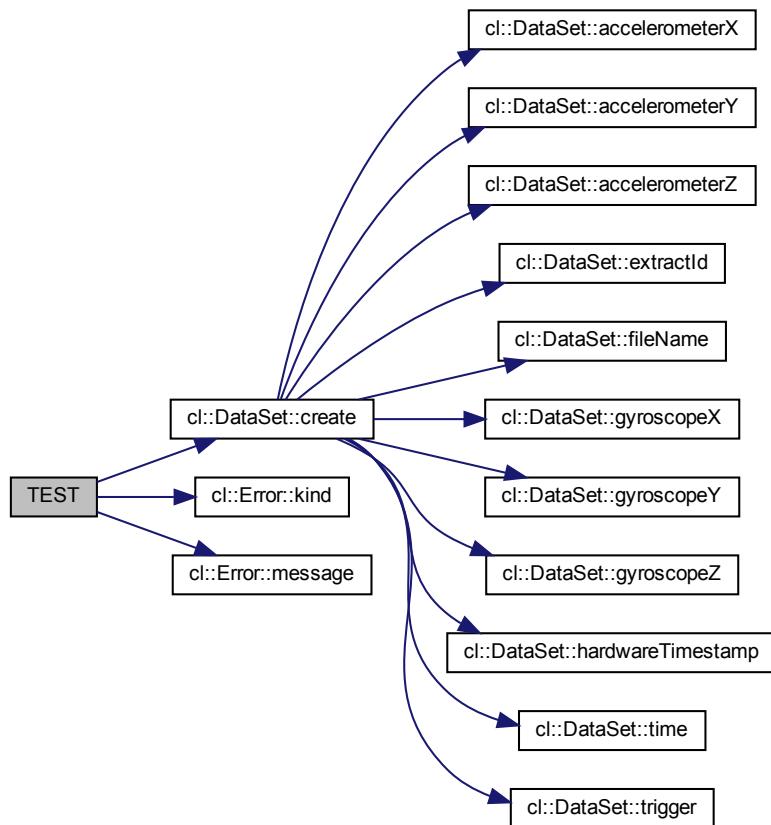


7.127.2.3 TEST() [3/4]

```
TEST (
    DataSet ,
    shouldNotBeAbleToCreateFromInvalidData )
```

Definition at line 108 of file `data_set_test.cpp`.

Here is the call graph for this function:

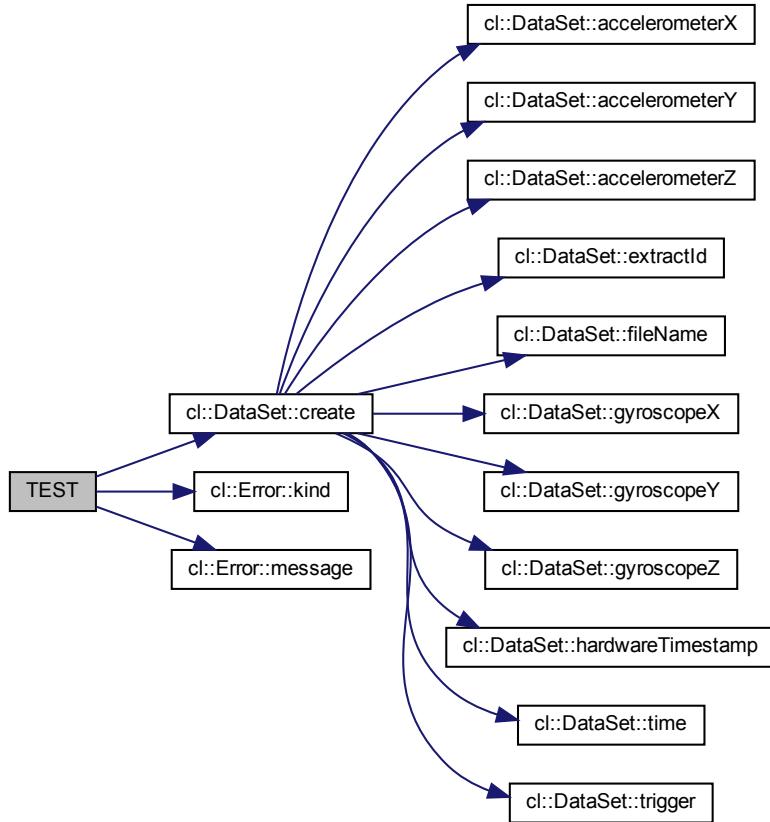


7.127.2.4 TEST() [4/4]

```
TEST (
    DataSet ,
    shouldNotBeAbleToCreateFromJaggedMatrix )
```

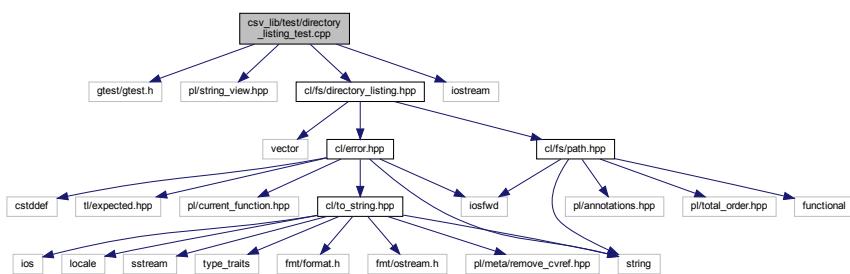
Definition at line 80 of file `data_set_test.cpp`.

Here is the call graph for this function:



7.128 csv_lib/test/directory_listing_test.cpp File Reference

```
#include "gtest/gtest.h"
#include <pl/string_view.hpp>
#include <cl/fs/directory_listing.hpp>
#include <iostream>
Include dependency graph for directory_listing_test.cpp:
```



Functions

- `TEST` (`directoryListing`, `shouldFindFiles`)
- `TEST` (`directoryListing`, `shouldFindFilesWithDotAndDotDot`)
- `TEST` (`directoryListing`, `shouldReturnErrorWhenPathDoesNotExist`)

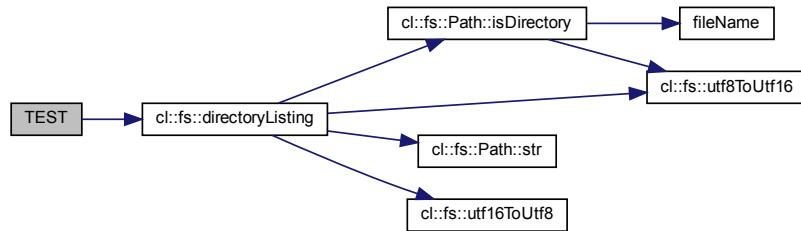
7.128.1 Function Documentation

7.128.1.1 TEST() [1/3]

```
TEST (
    directoryListing ,
    shouldFindFiles )
```

Definition at line 13 of file `directory_listing_test.cpp`.

Here is the call graph for this function:

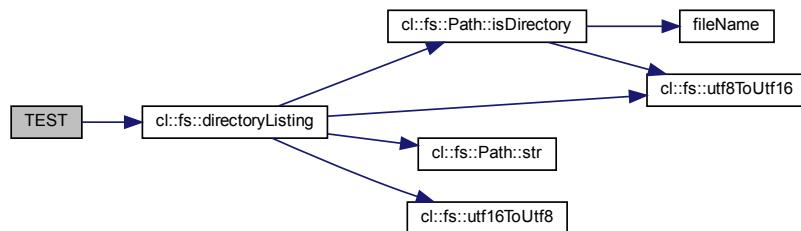


7.128.1.2 TEST() [2/3]

```
TEST (
    directoryListing ,
    shouldFindFilesWithDotAndDotDot )
```

Definition at line 28 of file `directory_listing_test.cpp`.

Here is the call graph for this function:

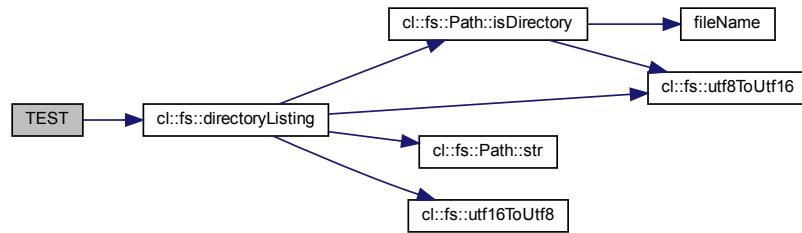


7.128.1.3 TEST() [3/3]

```
TEST (
    directoryListing ,
    shouldReturnErrorWhenPathDoesNotExist )
```

Definition at line 46 of file directory_listing_test.cpp.

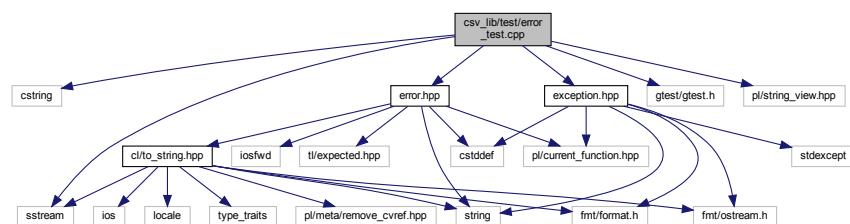
Here is the call graph for this function:



7.129 csv_lib/test/error_test.cpp File Reference

```
#include <cstring>
#include <sstream>
#include "gtest/gtest.h"
#include <pl/string_view.hpp>
#include "error.hpp"
#include "exception.hpp"
```

Include dependency graph for error_test.cpp:



Functions

- `TEST(error, shouldPrint)`
- `TEST(error, shouldReturnValue)`
- `TEST(error, shouldThrowExceptionWhenRaisesCalled)`
- `TEST(error, shouldCreateExpectedWithUnexpected)`

Variables

- const `cl::Error error`

7.129.1 Function Documentation

7.129.1.1 TEST() [1/4]

```
TEST (
    error ,
    shouldCreateExpectedWithUnexpected )
```

Definition at line 59 of file `error_test.cpp`.

7.129.1.2 TEST() [2/4]

```
TEST (
    error ,
    shouldPrint )
```

Definition at line 19 of file `error_test.cpp`.

7.129.1.3 TEST() [3/4]

```
TEST (
    error ,
    shouldReturnValues )
```

Definition at line 29 of file `error_test.cpp`.

7.129.1.4 TEST() [4/4]

```
TEST (
    error ,
    shouldThrowExceptionWhenRaiseIsCalled )
```

Definition at line 37 of file `error_test.cpp`.

7.129.2 Variable Documentation

7.129.2.1 error

```
const cl::Error error
```

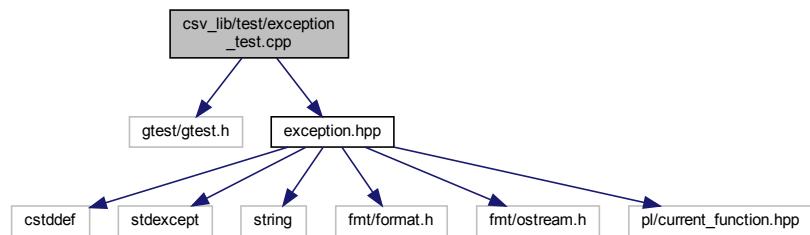
Initial value:

```
{
    cl::Error::Filesystem,
    "test_file.cpp",
    "bad_function",
    48,
    "Couldn't initialize the flux capacitor."}
```

Definition at line 12 of file error_test.cpp.

7.130 csv_lib/test/exception_test.cpp File Reference

```
#include "gtest/gtest.h"
#include "exception.hpp"
Include dependency graph for exception_test.cpp:
```



Functions

- [TEST](#) (exception, shouldWork)

7.130.1 Function Documentation

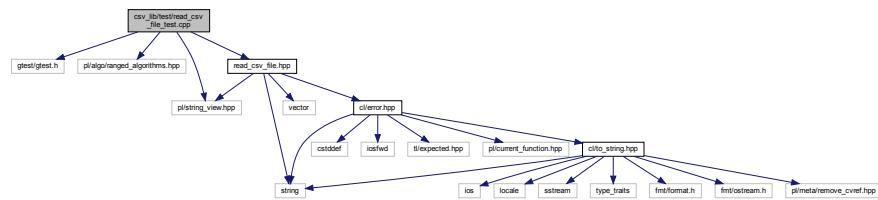
7.130.1.1 TEST()

```
TEST (
    exception ,
    shouldWork )
```

Definition at line 5 of file exception_test.cpp.

7.131 csv_lib/test/read_csv_file_test.cpp File Reference

```
#include "gtest/gtest.h"
#include <pl/algo/ranged_algorithms.hpp>
#include <pl/string_view.hpp>
#include "read_csv_file.hpp"
Include dependency graph for read_csv_file_test.cpp:
```



Functions

- [TEST](#) (`readCsvFile`, `shouldReadCsvFile`)
- [TEST](#) (`readCsvFile`, `shouldNotReadNonexistantCsvFile`)

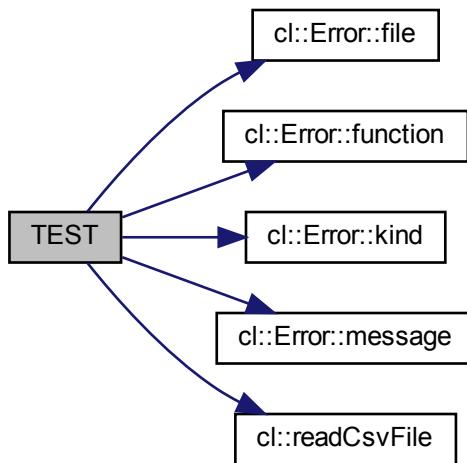
7.131.1 Function Documentation

7.131.1.1 TEST() [1/2]

```
TEST (
    readCsvFile ,
    shouldNotReadNonexistantCsvFile )
```

Definition at line 30 of file `read_csv_file_test.cpp`.

Here is the call graph for this function:

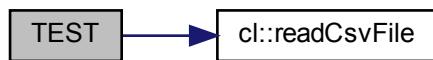


7.131.1.2 TEST() [2/2]

```
TEST (
    readCsvFile ,
    shouldReadCsvFile )
```

Definition at line 8 of file read_csv_file_test.cpp.

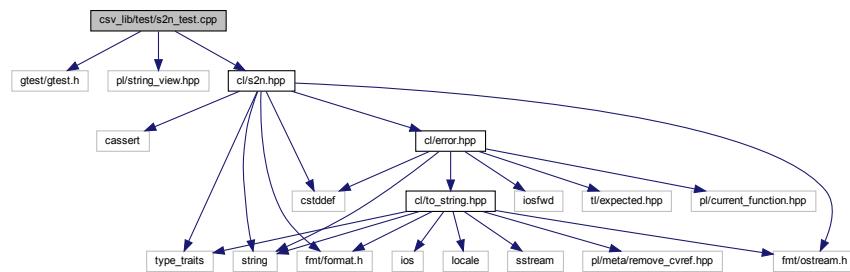
Here is the call graph for this function:



7.132 csv_lib/test/s2n_test.cpp File Reference

```
#include "gtest/gtest.h"
#include <pl/string_view.hpp>
```

```
#include "cl/s2n.hpp"
Include dependency graph for s2n_test.cpp:
```



Functions

- `TEST` (`s2n`, `shouldWork`)
- `TEST` (`s2n`, `shouldReturnInvalidArgumentErrorIfInputIsInvalid`)
- `TEST` (`s2n`, `shouldReturnOutOfRangeErrorIfInputIsOutOfRange`)

7.132.1 Function Documentation

7.132.1.1 TEST() [1/3]

```
TEST (
    s2n ,
    shouldReturnInvalidArgumentErrorIfInputIsInvalid )
```

Definition at line 21 of file `s2n_test.cpp`.

7.132.1.2 TEST() [2/3]

```
TEST (
    s2n ,
    shouldReturnOutOfRangeErrorIfInputIsOutOfRange )
```

Definition at line 29 of file `s2n_test.cpp`.

7.132.1.3 TEST() [3/3]

```
TEST (
    s2n ,
    shouldWork )
```

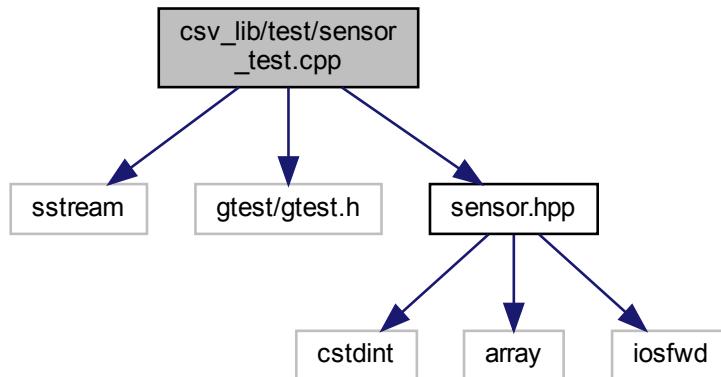
Definition at line 7 of file s2n_test.cpp.

Here is the call graph for this function:



7.133 csv_lib/test/sensor_test.cpp File Reference

```
#include <sstream>
#include "gtest/gtest.h"
#include "sensor.hpp"
Include dependency graph for sensor_test.cpp:
```



Functions

- [TEST \(`sensor`, `shouldHaveCorrectValues`\)](#)
- [TEST \(`sensor`, `shouldPrintCorrely`\)](#)

7.133.1 Function Documentation

7.133.1.1 TEST() [1/2]

```
TEST (
    sensor ,
    shouldHaveCorrectValues )
```

Definition at line 7 of file sensor_test.cpp.

7.133.1.2 TEST() [2/2]

```
TEST (
    sensor ,
    shouldPrintCorrectly )
```

Definition at line 15 of file sensor_test.cpp.

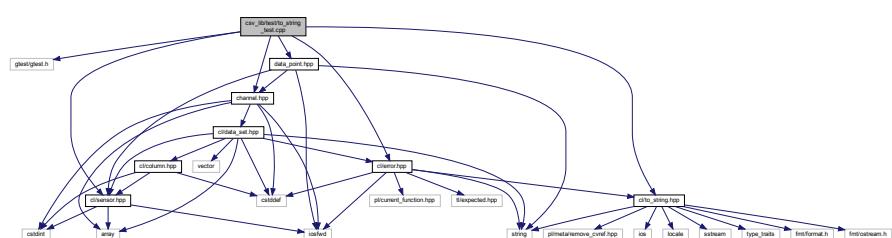
Here is the call graph for this function:



7.134 csv_lib/test/to_string_test.cpp File Reference

```
#include "gtest/gtest.h"
#include "channel.hpp"
#include "data_point.hpp"
#include "error.hpp"
#include "sensor.hpp"
#include "to_string.hpp"
```

Include dependency graph for to_string_test.cpp:



Functions

- [TEST](#) (to_string, test)

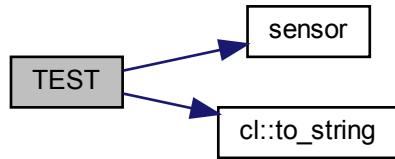
7.134.1 Function Documentation

7.134.1.1 TEST()

```
TEST (
    to_string ,
    test )
```

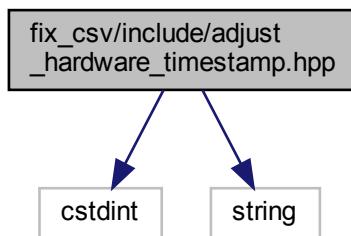
Definition at line 9 of file to_string_test.cpp.

Here is the call graph for this function:

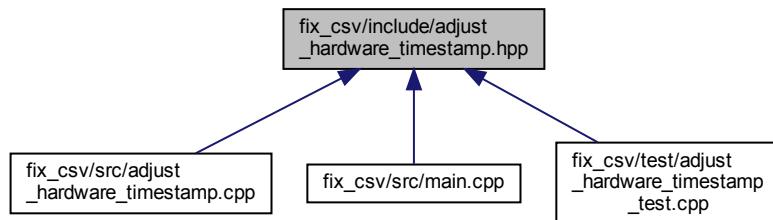


7.135 fix_csv/include/adjust_hardware_timestamp.hpp File Reference

```
#include <cstdint>
#include <string>
Include dependency graph for adjust_hardware_timestamp.hpp:
```



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



Namespaces

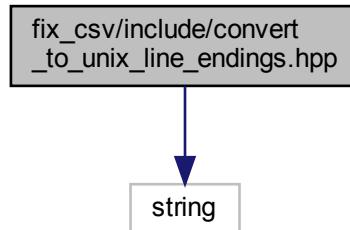
- `fmc`

Functions

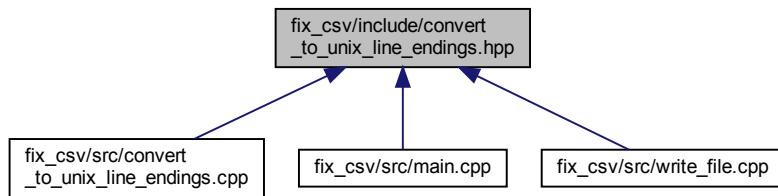
- void `fmc::adjustHardwareTimestamp` (`std::string *cellContent, const std::string &nextRowHardwareTimestamp, std::uint64_t *overflowCount)`

7.136 fix_csv/include/convert_to_unix_line_endings.hpp File Reference

```
#include <string>
Include dependency graph for convert_to_unix_line_endings.hpp:
```



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



Namespaces

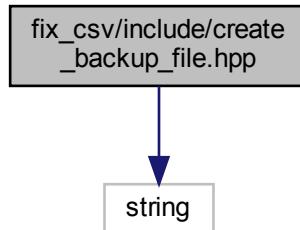
- `fmc`

Functions

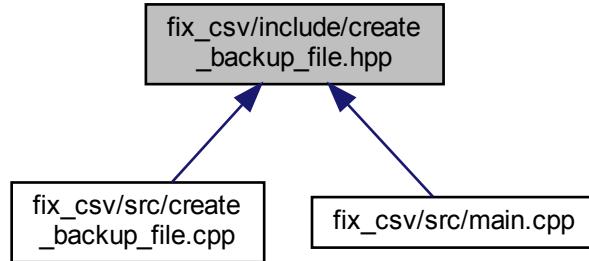
- `bool fmc::convertToUnixLineEndings (const std::string &csvPath)`

7.137 fix_csv/include/create_backup_file.hpp File Reference

```
#include <string>
Include dependency graph for create_backup_file.hpp:
```



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



Namespaces

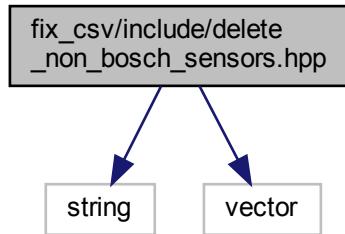
- [fmc](#)

Functions

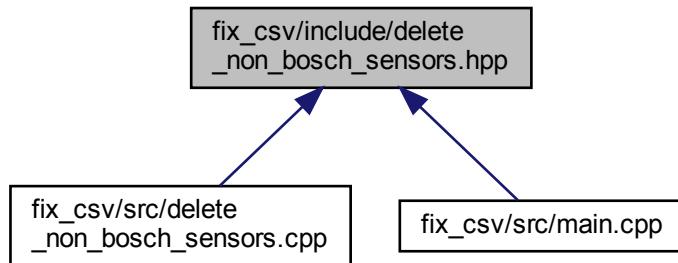
- `bool fmc::createBackupFile (const std::string &csvFilePath, const std::string &backupFilePath)`

7.138 fix_csv/include/delete_non_bosch_sensors.hpp File Reference

```
#include <string>
#include <vector>
Include dependency graph for delete_non_bosch_sensors.hpp:
```



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



Namespaces

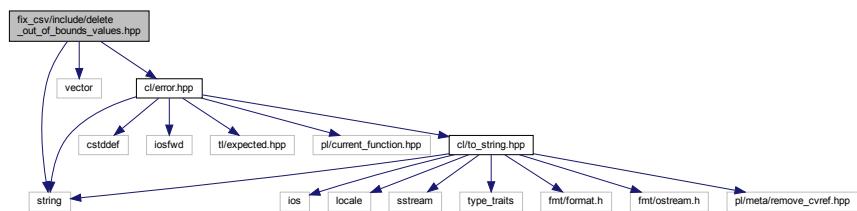
- [fmc](#)

Functions

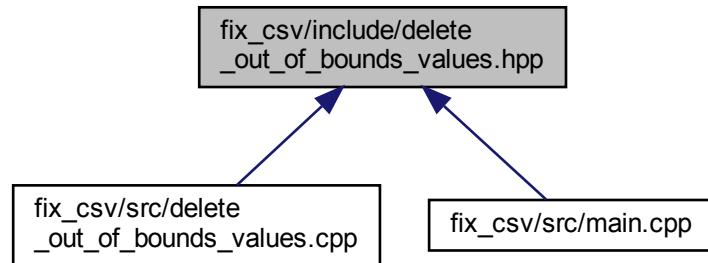
- void [fmc::deleteNonBoschSensors](#) (std::vector< std::vector< std::string >> *data)

7.139 fix_csv/include/delete_out_of_bounds_values.hpp File Reference

```
#include <string>
#include <vector>
#include "cl/error.hpp"
Include dependency graph for delete_out_of_bounds_values.hpp:
```



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



Namespaces

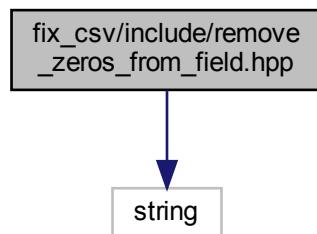
- [fmc](#)

Functions

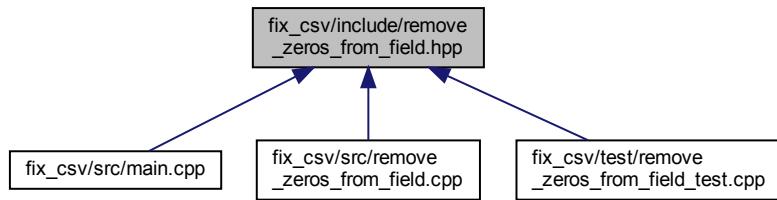
- [cl::Expected< void > fmc::deleteOutOfBoundsValues \(std::vector< std::vector< std::string >> *data\)](#)

7.140 fix_csv/include/remove_zeros_from_field.hpp File Reference

```
#include <string>
Include dependency graph for remove_zeros_from_field.hpp:
```



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



Namespaces

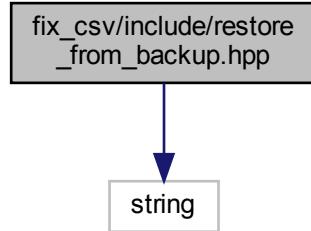
- `fmc`

Functions

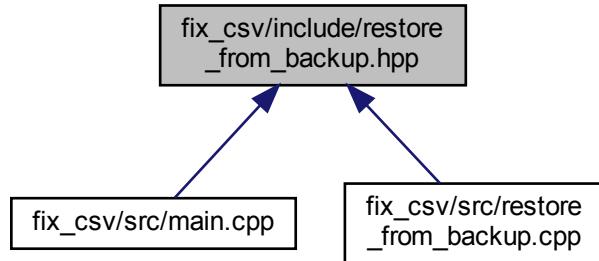
- void `fmc::removeZerosFromField` (`std::string *field`)

7.141 fix_csv/include/restore_from_backup.hpp File Reference

```
#include <string>
Include dependency graph for restore_from_backup.hpp:
```



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



Namespaces

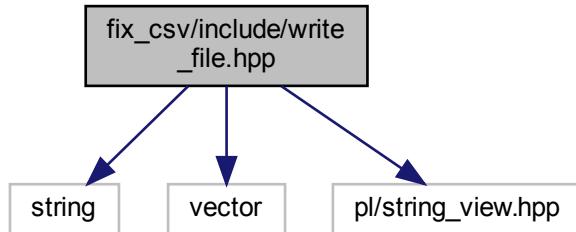
- [fmc](#)

Functions

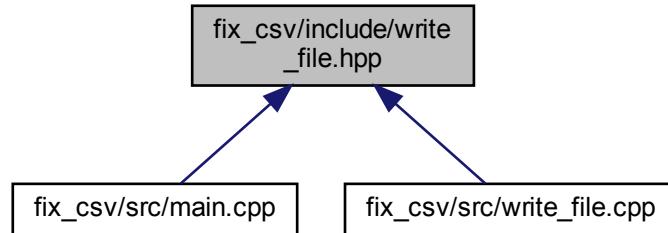
- bool [fmc::restoreFromBackup](#) (const std::string &csvFilePath, const std::string &backupFilePath)

7.142 fix_csv/include/write_file.hpp File Reference

```
#include <string>
#include <vector>
#include <pl/string_view.hpp>
Include dependency graph for write_file.hpp:
```



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



Namespaces

- `fmc`

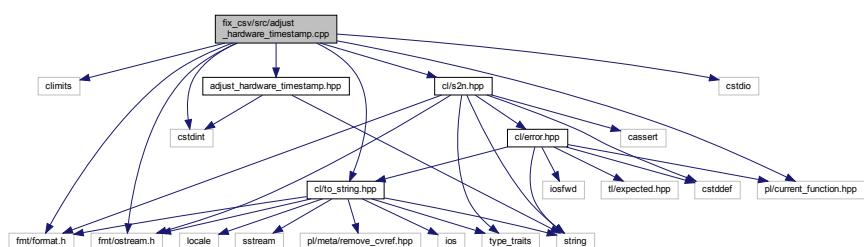
Functions

- `bool fmc::writeFile (pl::string_view csvPath, pl::string_view csvFileExtension, const std::vector< std::string > &columnNames, const std::vector< std::vector< std::string >> &data)`

7.143 fix_csv/src/adjust_hw timestamp.cpp File Reference

```

#include <climits>
#include <cstdint>
#include <cstdio>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <pl/current_function.hpp>
#include "cl/s2n.hpp"
#include "cl/to_string.hpp"
#include "adjust_hw timestamp.hpp"
Include dependency graph for adjust_hw timestamp.hpp:
  
```



Namespaces

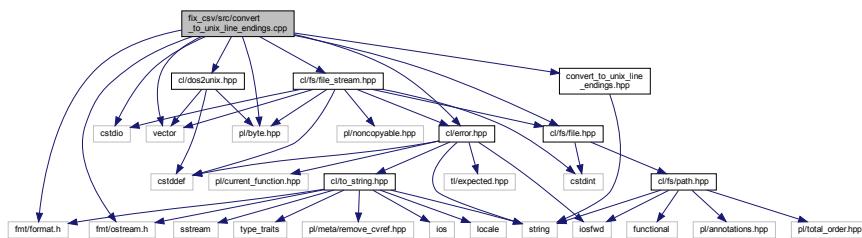
- [fmc](#)

Functions

- void [fmc::adjustHardwareTimestamp](#) (std::string *cellContent, const std::string &nextRowHardwareTimestamp, std::uint64_t *overflowCount)

7.144 fix_csv/src/convert_to_unix_line_endings.cpp File Reference

```
#include <cstdio>
#include <vector>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <pl/byte.hpp>
#include "cl/dos2unix.hpp"
#include "cl/error.hpp"
#include "cl/fs/file.hpp"
#include "cl/fs/file_stream.hpp"
#include "convert_to_unix_line_endings.hpp"
Include dependency graph for convert_to_unix_line_endings.cpp:
```



Namespaces

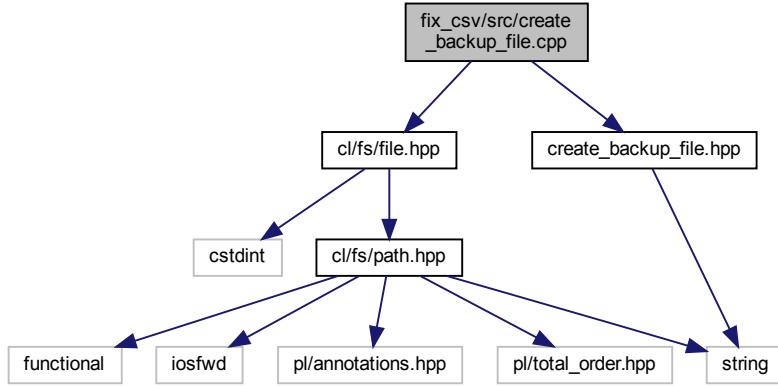
- [fmc](#)

Functions

- bool [fmc::convertToUnixLineEndings](#) (const std::string &csvPath)

7.145 fix_csv/src/create_backup_file.cpp File Reference

```
#include "cl/fs/file.hpp"
#include "create_backup_file.hpp"
Include dependency graph for create_backup_file.cpp:
```



Namespaces

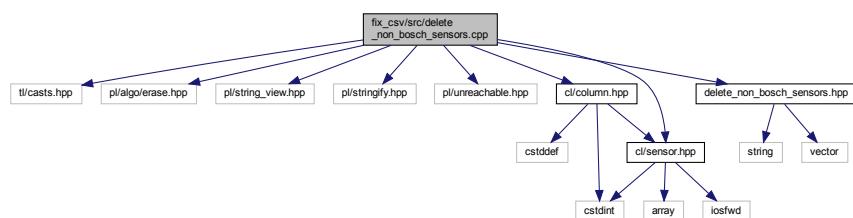
- `fmc`

Functions

- `bool fmc::createBackupFile (const std::string &csvFilePath, const std::string &backupFilePath)`

7.146 fix_csv/src/delete_non_bosch_sensors.cpp File Reference

```
#include <tl/casts.hpp>
#include <pl/algo/erase.hpp>
#include <pl/string_view.hpp>
#include <pl/stringify.hpp>
#include <pl/unreachable.hpp>
#include "cl/column.hpp"
#include "cl/sensor.hpp"
#include "delete_non_bosch_sensors.hpp"
Include dependency graph for delete_non_bosch_sensors.cpp:
```



Namespaces

- fmc

Macros

- ```
• #define CL_SENSOR_X(enm, value) case cl::Sensor::enm: return PL_STRINGIFY(value);
```

## Functions

- void **fmc::deleteNonBoschSensors** (std::vector< std::vector< std::string >> \*data)

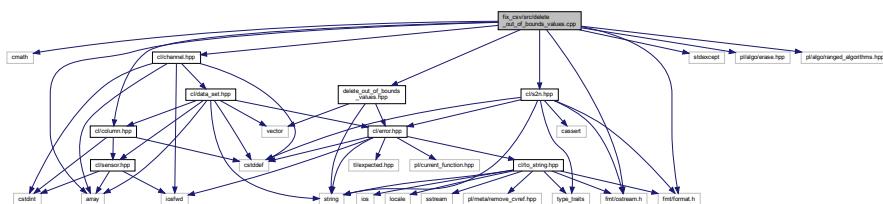
### 7.146.1 Macro Definition Documentation

#### **7.146.1.1 CL SENSOR X**

```
#define CL_SENSOR_X(enm, value) case cl::Sensor::enm: return PL_STRINGIFY(value);
```

7.147 fix\_csv/src/delete\_out\_of\_bounds\_values.cpp File Reference

```
#include <cmath>
#include <array>
#include <stdexcept>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <pl/algo/erase.hpp>
#include <pl/algo/ranged_algorithms.hpp>
#include "cl/channel.hpp"
#include "cl/column.hpp"
#include "cl/s2n.hpp"
#include "delete_out_of_bounds_values.hpp"
Include dependency graph for delete_out_of_bounds_values.cpp:
```



## Namespaces

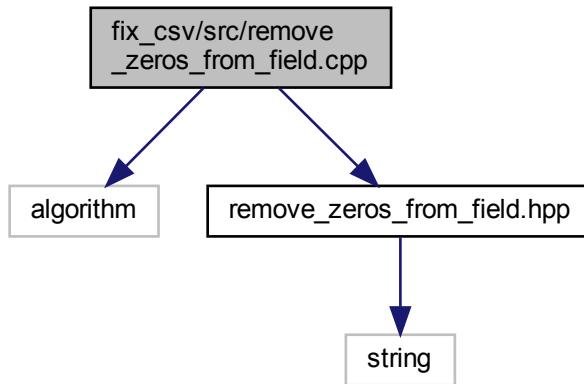
- fmc

## Functions

- `cl::Expected< void > fmc::deleteOutOfBoundsValues (std::vector< std::vector< std::string >> *data)`

## 7.148 fix\_csv/src/remove\_zeros\_from\_field.cpp File Reference

```
#include <algorithm>
#include "remove_zeros_from_field.hpp"
Include dependency graph for remove_zeros_from_field.cpp:
```



## Namespaces

- `fmc`

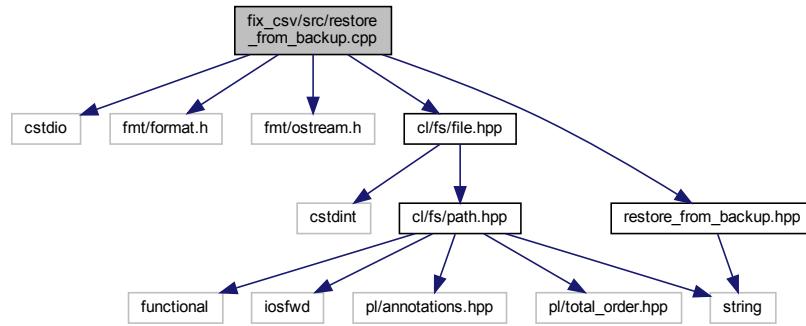
## Functions

- `void fmc::removeZerosFromField (std::string *field)`

## 7.149 fix\_csv/src/restore\_from\_backup.cpp File Reference

```
#include <cstdio>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include "cl/fs/file.hpp"
```

```
#include "restore_from_backup.hpp"
Include dependency graph for restore_from_backup.cpp:
```



## Namespaces

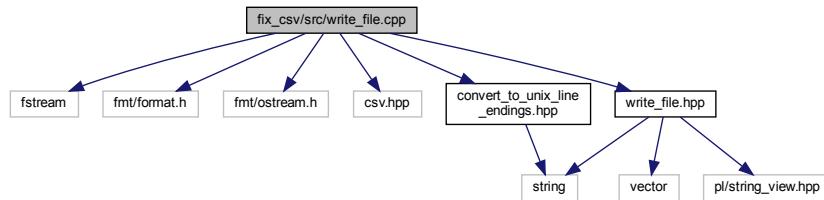
- `fmc`

## Functions

- `bool fmc::restoreFromBackup (const std::string &csvFilePath, const std::string &backupFilePath)`

## 7.150 fix\_csv/src/write\_file.cpp File Reference

```
#include <fstream>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <csv.hpp>
#include "convert_to_unix_line_endings.hpp"
#include "write_file.hpp"
Include dependency graph for write_file.cpp:
```



## Namespaces

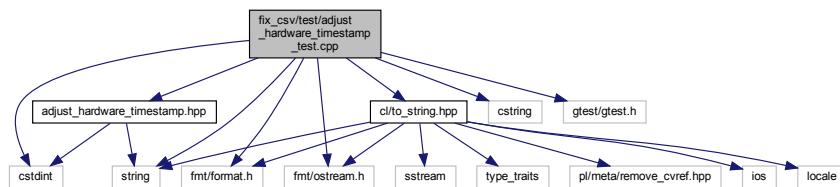
- `fmc`

## Functions

- bool `fmc::writeFile` (pl::string\_view csvPath, pl::string\_view csvFileExtension, const std::vector< std::string > &columnNames, const std::vector< std::vector< std::string >> &data)

## 7.151 fix\_csv/test/adjust\_hw\_timestamp\_test.cpp File Reference

```
#include <cstdint>
#include <cstring>
#include <string>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include "gtest/gtest.h"
#include "cl/to_string.hpp"
#include "adjust_hw_timestamp.hpp"
Include dependency graph for adjust_hw_timestamp_test.cpp:
```



## Functions

- `TEST` (`adjustHardwareTimestamp`, `shouldDoNothingForNonOverflowedValue`)
- `TEST` (`adjustHardwareTimestamp`, `shouldIncrementOverflowCount`)
- `TEST` (`adjustHardwareTimestamp`, `shouldWorkForOneRoundOfOverflow`)
- `TEST` (`adjustHardwareTimestamp`, `shouldWorkForTwoRoundsOfOverflow`)
- `TEST` (`adjustHardwareTimestamp`, `shouldWork`)

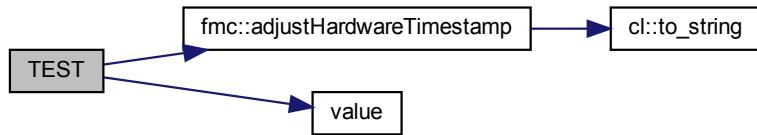
### 7.151.1 Function Documentation

#### 7.151.1.1 TEST() [1/5]

```
TEST (
 adjustHardwareTimestamp ,
 shouldDoNothingForNonOverflowedValue)
```

Definition at line 15 of file `adjust_hw_timestamp_test.cpp`.

Here is the call graph for this function:

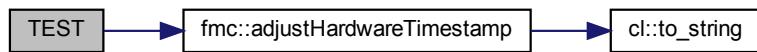


### 7.151.1.2 TEST() [2/5]

```
TEST (
 adjustHardwareTimestamp ,
 shouldIncrementOverflowCount)
```

Definition at line 26 of file `adjust_hardware_timestamp_test.cpp`.

Here is the call graph for this function:

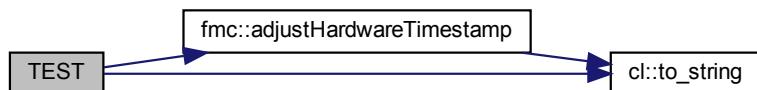


### 7.151.1.3 TEST() [3/5]

```
TEST (
 adjustHardwareTimestamp ,
 shouldWork)
```

Definition at line 132 of file `adjust_hardware_timestamp_test.cpp`.

Here is the call graph for this function:

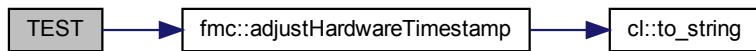


**7.151.1.4 TEST() [4/5]**

```
TEST (
 adjustHardwareTimestamp ,
 shouldWorkForOneRoundOfOverflow)
```

Definition at line 48 of file adjust\_hardware\_timestamp\_test.cpp.

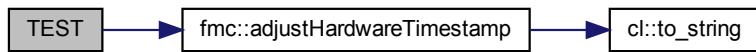
Here is the call graph for this function:

**7.151.1.5 TEST() [5/5]**

```
TEST (
 adjustHardwareTimestamp ,
 shouldWorkForTwoRoundsOfOverflow)
```

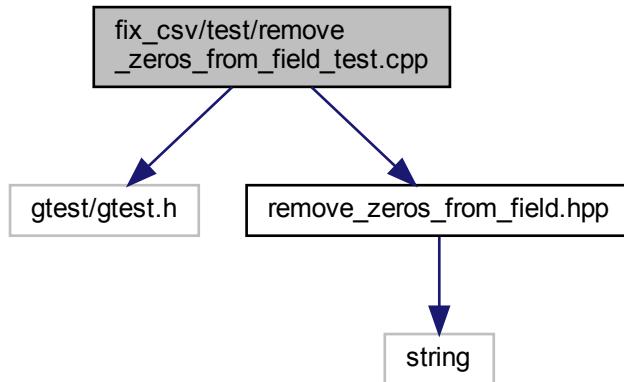
Definition at line 96 of file adjust\_hardware\_timestamp\_test.cpp.

Here is the call graph for this function:

**7.152 fix\_csv/test/remove\_zeros\_from\_field\_test.cpp File Reference**

```
#include "gtest/gtest.h"
#include "remove_zeros_from_field.hpp"
```

Include dependency graph for remove\_zeros\_from\_field\_test.cpp:



## Functions

- `TEST (removeZerosFromField, shouldRemoveDotAndZeros)`
- `TEST (removeZerosFromField, shouldNotRemovelfNonZerosFollow)`
- `TEST (removeZerosFromField, shouldNotRemovelfNoDot)`
- `TEST (removeZerosFromField, shouldDoNothingIfStringIsEmpty)`
- `TEST (removeZerosFromField, shouldDeleteStringIfStringIsSingleDot)`
- `TEST (removeZerosFromField, shouldDeleteStringIfStringIsDotAndZero)`

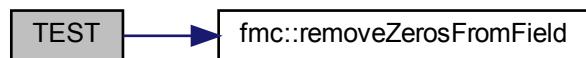
### 7.152.1 Function Documentation

#### 7.152.1.1 TEST() [1/6]

```
TEST (
 removeZerosFromField ,
 shouldDeleteStringIfStringIsDotAndZero)
```

Definition at line 53 of file `remove_zeros_from_field_test.cpp`.

Here is the call graph for this function:

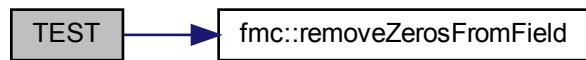


**7.152.1.2 TEST() [2/6]**

```
TEST (
 removeZerosFromField ,
 shouldDeleteStringIfStringIsSingleDot)
```

Definition at line 44 of file remove\_zeros\_from\_field\_test.cpp.

Here is the call graph for this function:

**7.152.1.3 TEST() [3/6]**

```
TEST (
 removeZerosFromField ,
 shouldDoNothingIfStringIsEmpty)
```

Definition at line 35 of file remove\_zeros\_from\_field\_test.cpp.

Here is the call graph for this function:

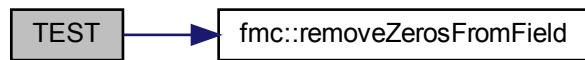


#### 7.152.1.4 TEST() [4/6]

```
TEST (
 removeZerosFromField ,
 shouldNotRemoveIfNoDot)
```

Definition at line 25 of file remove\_zeros\_from\_field\_test.cpp.

Here is the call graph for this function:



#### 7.152.1.5 TEST() [5/6]

```
TEST (
 removeZerosFromField ,
 shouldNotRemoveIfNonZerosFollow)
```

Definition at line 15 of file remove\_zeros\_from\_field\_test.cpp.

Here is the call graph for this function:



### 7.152.1.6 TEST() [6/6]

```
TEST (
 removeZerosFromField ,
 shouldRemoveDotAndZeros)
```

Definition at line 5 of file remove\_zeros\_from\_field\_test.cpp.

Here is the call graph for this function:





# Index

~FileStream  
    cl::fs::FileStream, 145

~Process  
    cl::Process, 173

above\_threshold.cpp  
    channel, 272  
    channelAccessor, 273  
    CL\_CHANNEL\_X, 272

above\_threshold\_test.cpp  
    EXPECT\_LONG\_DOUBLE\_EQ, 275  
    TEST, 276

aboveThreshold  
    ctg, 60

accelerometerAverage  
    cl::DataSet, 121

accelerometerMaximum  
    cl::DataSet, 121

accelerometerThreshold  
    cl, 22

AccelerometerX  
    cl, 13

accelerometerX  
    cl::DataSet, 122

AccelerometerY  
    cl, 13

accelerometerY  
    cl::DataSet, 122

AccelerometerZ  
    cl, 13

accelerometerZ  
    cl::DataSet, 123

adjust\_hw\_timestamp\_test.cpp  
    TEST, 347–349

adjustHardwareTimestamp  
    fmc, 65

asMilliseconds  
    cm::ManualSegmentationPoint, 161

averageComparisonValueCalculator  
    ctg, 61

base\_type  
    cl::Exception, 135

Both  
    cs, 52

build  
    cm::Configuration::Builder, 72  
    cs::CsvLineBuilder, 103

Builder  
    cm::Configuration, 92

    cm::Configuration::Builder, 72

Butterworth  
    cs, 50

Channel  
    cl, 13

channel  
    above\_threshold.cpp, 272  
    cl::DataPoint, 117  
    data\_point.cpp, 302

channel.cpp  
    CL\_CHANNEL\_X, 301

channel.hpp  
    CL\_CHANNEL, 279  
    CL\_CHANNEL\_X, 280

channel\_test.cpp  
    TEST, 314, 315

ChannelAccessor  
    cl::DataSet, 121

channelAccessor  
    above\_threshold.cpp, 273

channelCount  
    cl, 23

channels  
    cl, 23

cl, 11  
    accelerometerThreshold, 22  
    AccelerometerX, 13  
    AccelerometerY, 13  
    AccelerometerZ, 13  
    Channel, 13  
    channelCount, 23  
    channels, 23  
    CL\_CHANNEL, 13  
    CL\_CHANNEL\_X, 13  
    CL\_SENSOR, 14  
    CL\_SENSOR\_X, 14  
    CL\_SPECIALIZE\_COL\_TRAITS, 14–16  
    Column, 13  
    column\_index, 23  
    column\_type, 12  
    CsvFileKind, 13  
    data\_set\_accessor\_v, 23  
    dataSetAccessor, 16  
    dos2unix, 16  
    Expected, 12  
    ExtractId, 13  
    Fixed, 14  
    gyroscopeThreshold, 23  
    GyroscopeX, 13

GyroscopeY, 13  
 GyroscopeZ, 13  
 HardwareTimestamp, 13  
 isAccelerometer, 17  
 isGyroscope, 17  
 operator<<, 18, 19  
 Raw, 14  
 readCsvFile, 20  
 s2n, 20  
 SamplingRate, 13  
 Sensor, 14  
 sensors, 24  
 threshold, 21  
 Time, 13  
 to\_string, 21  
 Trigger, 13  
 useUnbufferedIo, 22  
 cl::col\_traits< Col >, 80  
 cl::data\_set\_accessor< Chan >, 115  
 cl::DataPoint, 116  
     channel, 117  
     DataPoint, 117  
     fileName, 117  
     operator<<, 119  
     sensor, 118  
     time, 118  
     value, 119  
 cl::DataSet, 120  
     accelerometerAverage, 121  
     accelerometerMaximum, 121  
     accelerometerX, 122  
     accelerometerY, 122  
     accelerometerZ, 123  
     ChannelAccessor, 121  
     create, 123  
     extractId, 125  
     fileName, 125  
     gyroscopeAverage, 126  
     gyroscopeMaximum, 126  
     gyroscopeX, 127  
     gyroscopeY, 127  
     gyroscopeZ, 128  
     hardwareTimestamp, 128  
     rowCount, 129  
     size\_type, 121  
     time, 129  
     trigger, 130  
 cl::Error, 130  
     CL\_ERROR\_KIND, 131  
     Error, 131  
     file, 132  
     function, 132  
     Kind, 131  
     Kind, 132  
     line, 133  
     message, 133  
     operator<<, 134  
     raise, 133  
         to\_string, 134  
     cl::Exception, 134  
         base\_type, 135  
         Exception, 136  
         file, 136  
         function, 136  
         line, 137  
 cl::fs, 24  
     directoryListing, 25  
     DirectoryListingOption, 25  
     ExcludeDotAndDotDot, 25  
     formatError, 26  
     None, 25  
     operator<, 27  
     operator<<, 27  
     operator==, 27  
     utf16ToUtf8, 28  
     utf8ToUtf16, 28  
 cl::fs::File, 137  
     copyTo, 139  
     create, 139  
     exists, 140  
     File, 138  
     moveTo, 141  
     path, 141  
     remove, 142  
     size, 142  
 cl::fs::FileStream, 143  
     ~FileStream, 145  
     create, 145  
     FileStream, 145  
     OpenMode, 144  
     operator=, 146  
     PL\_NONCOPYABLE, 147  
     Read, 145  
     readAll, 147  
     ReadWrite, 145  
     this\_type, 144  
     Write, 145  
     write, 147  
 cl::fs::Path, 167  
     exists, 168  
     isDirectory, 169  
     isFile, 170  
     operator<, 171  
     operator<<, 172  
     operator==, 172  
     Path, 168  
     str, 170  
 cl::Process, 173  
     ~Process, 173  
     create, 174  
     file, 174  
     operator=, 174  
     PL\_NONCOPYABLE, 174  
     Process, 173  
     this\_type, 173  
         CL\_CHANNEL

channel.hpp, 279  
cl, 13  
**CL\_CHANNEL\_X**  
above\_threshold.cpp, 272  
channel.cpp, 301  
channel.hpp, 280  
cl, 13  
**CL\_ERROR\_KIND**  
cl::Error, 131  
error.hpp, 286  
**CL\_ERROR\_KIND\_X**  
error.cpp, 306  
error.hpp, 287  
**CL\_FS\_SEPARATOR**  
separator.hpp, 293  
**CL\_SENSOR**  
cl, 14  
sensor.hpp, 298  
**CL\_SENSOR\_X**  
cl, 14  
delete\_non\_bosch\_sensors.cpp, 344  
sensor.cpp, 312  
sensor.hpp, 298, 299  
**CL\_SPECIALIZE\_COL\_TRAITS**  
cl, 14–16  
column.hpp, 282  
**CL\_THROW**  
exception.hpp, 288  
**CL\_THROW\_FMT**  
exception.hpp, 288  
**CL\_UNEXPECTED**  
error.hpp, 287  
closestOne  
cm, 32  
cm, 29  
closestOne, 32  
CM\_DATA\_SET\_IDENTIFIER, 31  
CM\_DATA\_SET\_IDENTIFIER\_X, 31  
CM\_IMU, 31  
CM\_IMU\_X, 31  
CM\_SORTER, 33  
confusionMatrixBestConfigs, 33  
createSegmentationResults, 34  
DataSetIdentifier, 31  
distance, 35  
distanceScore, 36  
fetch, 37  
Imu, 31  
imuCount, 48  
imus, 48  
interpolatedDataSetPaths, 38  
operator!=, 39  
operator<, 40  
operator<<, 40–42  
operator==, 43  
orderConfigurationsByQuality, 44  
pythonOutput, 44  
segment, 45  
splitString, 47  
toDataSetIdentifier, 47  
**cm::Configuration**, 81  
Builder, 92  
Configuration, 82  
createFilePath, 82  
deleteTooClose, 83  
deleteTooCloseOptions, 83  
deleteTooLowVariance, 84  
deleteTooLowVarianceOptions, 84  
filterKind, 85  
filterKindOptions, 85  
importSegmentationPoints, 86  
imu, 87  
imuOptions, 87  
isInitialized, 88  
operator!=, 92  
operator<<, 93  
operator==, 93  
segmentationKind, 88  
segmentationKindOptions, 89  
serializeSegmentationPoints, 89  
skipWindow, 90  
skipWindowOptions, 90  
std::hash< Configuration >, 93  
windowSize, 91  
windowSizeOptions, 91  
**cm::Configuration::Builder**, 71  
build, 72  
Builder, 72  
deleteTooClose, 73  
deleteTooLowVariance, 74  
filterKind, 75  
imu, 76  
segmentationKind, 77  
skipWindow, 78  
windowSize, 79  
**cm::ConfigWithDistanceScore**, 94  
config, 95  
ConfigWithDistanceScore, 94  
distScore, 95  
**cm::ConfigWithTotalConfusionMatrix**, 95  
config, 96  
ConfigWithTotalConfusionMatrix, 96  
matrix, 96  
**cm::ConfusionMatrix**, 97  
ConfusionMatrix, 98  
falseNegatives, 98  
falsePositives, 98  
incrementFalseNegatives, 98  
incrementFalsePositives, 98  
incrementTrueNegatives, 99  
incrementTruePositives, 99  
operator+=, 99  
this\_type, 97  
totalCount, 100  
trueNegatives, 100  
truePositives, 100

cm::CsvFileInfo, 101  
 CsvFileInfo, 101  
 hardwareTimestamps, 101  
 cm::ManualSegmentationPoint, 159  
 asMilliseconds, 161  
 convertToHardwareTimestamps, 161  
 frame, 162  
 hour, 163  
 ManualSegmentationPoint, 160  
 minute, 163  
 operator!=, 165  
 operator<<, 166  
 operator==, 166  
 readCsvFile, 164  
 second, 165  
**CM\_DATA\_SET\_IDENTIFIER**  
 cm, 31  
 data\_set\_identifier.hpp, 231  
**CM\_DATA\_SET\_IDENTIFIER\_X**  
 cm, 31  
 data\_set\_identifier.cpp, 249  
 data\_set\_identifier.hpp, 232  
**CM\_DEV\_NULL**  
 python\_output.cpp, 255  
**CM\_ENSURE\_CONTAINS**  
 configuration.cpp, 245  
**CM\_ENSURE\_HAS\_VALUE**  
 configuration.cpp, 245  
**CM\_IMU**  
 cm, 31  
 imu.hpp, 236  
**CM\_IMU\_X**  
 cm, 31  
 imu.cpp, 251  
 imu.hpp, 237  
**CM\_SEGMENTATOR**  
 python\_output.cpp, 255  
**CM\_SORTER**  
 cm, 33  
 confusion\_matrix\_best\_configs.hpp, 228  
 CMakeLists.txt  
 include, 177–181  
 set, 177–181  
 Column  
 cl, 13  
 column.hpp  
 CL\_SPECIALIZE\_COL\_TRAITS, 282  
 column\_index  
 cl, 23  
 column\_test.cpp  
 TEST, 316  
 column\_type  
 cl, 12  
 compare\_segmentation/CMakeLists.txt, 177  
 compare\_segmentation/include/csv\_line.hpp, 182  
 compare\_segmentation/include/data\_set\_info.hpp, 183  
 compare\_segmentation/include/filter\_kind.hpp, 185  
 compare\_segmentation/include/log\_files.hpp, 186  
 compare\_segmentation/include/log\_info.hpp, 186  
 compare\_segmentation/include/log\_line.hpp, 187  
 compare\_segmentation/include/paths.hpp, 188  
 compare\_segmentation/include/segmentation\_kind.hpp,  
 189  
 compare\_segmentation/src/csv\_line.cpp, 190  
 compare\_segmentation/src/data\_set\_info.cpp, 191  
 compare\_segmentation/src/filter\_kind.cpp, 191  
 compare\_segmentation/src/log\_files.cpp, 192  
 compare\_segmentation/src/log\_info.cpp, 193  
 compare\_segmentation/src/log\_line.cpp, 194  
 compare\_segmentation/src/main.cpp, 194  
 compare\_segmentation/src/segmentation\_kind.cpp,  
 206  
 compare\_segmentation/test/CMakeLists.txt, 177  
 compare\_segmentation/test/csv\_line\_test.cpp, 207  
 compare\_segmentation/test/data\_set\_info\_test.cpp,  
 208  
 compare\_segmentation/test/log\_files\_test.cpp, 209  
 compare\_segmentation/test/log\_info\_test.cpp, 211  
 compare\_segmentation/test/log\_line\_test.cpp, 221  
 compare\_segmentation/test/main.cpp, 196  
 config  
 cm::ConfigWithDistanceScore, 95  
 cm::ConfigWithTotalConfusionMatrix, 96  
 Configuration  
 cm::Configuration, 82  
 configuration.cpp  
 CM\_ENSURE\_CONTAINS, 245  
 CM\_ENSURE\_HAS\_VALUE, 245  
 ConfigWithDistanceScore  
 cm::ConfigWithDistanceScore, 94  
 ConfigWithTotalConfusionMatrix  
 cm::ConfigWithTotalConfusionMatrix, 96  
 confusion\_matrix/CMakeLists.txt, 181  
 confusion\_matrix/include/closest\_one.hpp, 223  
 confusion\_matrix/include/configuration.hpp, 224  
 confusion\_matrix/include/confusion\_matrix.hpp, 226  
 confusion\_matrix/include/confusion\_matrix\_best\_configs.hpp,  
 227  
 confusion\_matrix/include/create\_segmentation\_results.hpp,  
 228  
 confusion\_matrix/include/csv\_file\_info.hpp, 229  
 confusion\_matrix/include/data\_set\_identifier.hpp, 230  
 confusion\_matrix/include/distance.hpp, 232  
 confusion\_matrix/include/distance\_score.hpp, 233  
 confusion\_matrix/include/fetch.hpp, 234  
 confusion\_matrix/include/imu.hpp, 235  
 confusion\_matrix/include/interpolated\_data\_set\_paths.hpp,  
 238  
 confusion\_matrix/include/manual\_segmentation\_point.hpp,  
 239  
 confusion\_matrix/include/order\_configurations\_by\_quality.hpp,  
 240  
 confusion\_matrix/include/python\_output.hpp, 241  
 confusion\_matrix/include/segment.hpp, 242  
 confusion\_matrix/include/split\_string.hpp, 243  
 confusion\_matrix/src/closest\_one.cpp, 243

confusion\_matrix/src/configuration.cpp, 244  
confusion\_matrix/src/confusion\_matrix.cpp, 246  
confusion\_matrix/src/confusion\_matrix\_best\_configs.cpp, 246  
confusion\_matrix/src/create\_segmentation\_results.cpp, 247  
confusion\_matrix/src/csv\_file\_info.cpp, 248  
confusion\_matrix/src/data\_set\_identifier.cpp, 248  
confusion\_matrix/src/distance.cpp, 249  
confusion\_matrix/src/distance\_score.cpp, 250  
confusion\_matrix/src/imu.cpp, 250  
confusion\_matrix/src/interpolated\_data\_set\_paths.cpp, 252  
confusion\_matrix/src/main.cpp, 203  
confusion\_matrix/src/manual\_segmentation\_point.cpp, 252  
confusion\_matrix/src/order\_configurations\_by\_quality.cpp, cs, 49  
confusion\_matrix/src/python\_output.cpp, 254  
confusion\_matrix/src/segment.cpp, 255  
confusion\_matrix/src/split\_string.cpp, 256  
confusion\_matrix/test/CMakeLists.txt, 181  
confusion\_matrix/test/data\_set\_identifier\_test.cpp, 257  
confusion\_matrix/test/interpolated\_data\_set\_paths\_test.cpp, 258  
confusion\_matrix/test/main.cpp, 205  
confusion\_matrix/test/manual\_segmentation\_point\_test.cpp, 259  
confusion\_matrix/test/segment\_test.cpp, 263  
confusion\_matrix/test/split\_string\_test.cpp, 264  
confusion\_matrix\_best\_configs.hpp  
    CM\_SORTER, 228  
ConfusionMatrix  
    cm::ConfusionMatrix, 98  
confusionMatrixBestConfigs  
    cm, 33  
convertToHardwareTimestamps  
    cm::ManualSegmentationPoint, 161  
convertToUnixLineEndings  
    fmc, 65  
copyTo  
    cl::fs::File, 139  
counting/CMakeLists.txt, 178  
counting/include/above\_threshold.hpp, 266  
counting/include/average\_comparison\_value\_calculator.hpp, 267  
counting/include/half\_maximum\_comparison\_value\_calculator.hpp\_type, 103  
counting/include/is\_relevant.hpp, 268  
counting/include/percentage\_of.hpp, 269  
counting/include/run\_above\_threshold.hpp, 270  
counting/src/above\_threshold.cpp, 271  
counting/src/average\_comparison\_value\_calculator.cpp, 273  
counting/src/half\_maximum\_comparison\_value\_calculator.cpp, invalidSensor, 155  
counting/src/main.cpp, 197  
counting/src/run\_above\_threshold.cpp, 274  
counting/test/above\_threshold\_test.cpp, 275  
counting/test/CMakeLists.txt, 178  
counting/test/main.cpp, 199  
counting/test/percentage\_of\_test.cpp, 277  
create  
    cl::DataSet, 123  
    cl::fs::File, 139  
    cl::fs::FileStream, 145  
    cl::Process, 174  
    cs::LogInfo, 150  
createBackupFile  
    fmc, 66  
createFilePath  
    cm::Configuration, 82  
createSegmentationResults  
    cm, 34  
    Both, 52  
    Butterworth, 50  
    CS\_SPECIALIZE\_DATA\_SET\_INFO, 52–55  
    FilterKind, 50  
    logFiles, 55  
    logPath, 59  
    Maxima, 52  
    Minima, 52  
    MovingAverage, 50  
    oldLogPath, 59  
    operator!=, 56  
    operator<<, 57  
    operator==, 58  
    PL\_DEFINE\_EXCEPTION\_TYPE, 58  
    repetitionCount, 58  
    SegmentationKind, 50  
cs::CsvLineBuilder, 102  
    build, 103  
    CsvLineBuilder, 103  
    dataSet, 104  
    deleteLowVariance, 105  
    deleteTooClose, 106  
    filter, 107  
    isOld, 108  
    kind, 109  
    repetitions, 110  
    segmentationPoints, 111  
    sensor, 112  
    skipWindow, 113  
    windowSize, 114  
    cs::data\_set\_info< Tag >, 116  
    cs::LogInfo, 149  
        create, 150  
        deleteLowVariance, 151  
        deleteTooClose, 151  
        filterKind, 151  
        invalidSensor, 155  
        isInitialized, 152  
        logFilePath, 152  
        LogInfo, 150

operator!=, 154  
 operator<<, 154  
 operator==, 155  
 segmentationKind, 152  
 sensor, 153  
 skipWindow, 153  
 windowSize, 153  
 cs::LogLine, 155  
     fileName, 156  
     filePath, 157  
     invalidSensor, 159  
     parse, 157  
     segmentationPointCount, 158  
     sensor, 158  
 CS\_SPECIALIZE\_DATA\_SET\_INFO  
     cs, 52–55  
         data\_set\_info.hpp, 184  
 csv\_lib/CMakeLists.txt, 179  
 csv\_lib/include/cl/channel.hpp, 278  
 csv\_lib/include/cl/column.hpp, 281  
 csv\_lib/include/cl/data\_point.hpp, 283  
 csv\_lib/include/cl/data\_set.hpp, 284  
 csv\_lib/include/cl/dos2unix.hpp, 285  
 csv\_lib/include/cl/error.hpp, 286  
 csv\_lib/include/cl/exception.hpp, 287  
 csv\_lib/include/cl/fs/directory\_listing.hpp, 289  
 csv\_lib/include/cl/fs/file.hpp, 290  
 csv\_lib/include/cl/fs/file\_stream.hpp, 290  
 csv\_lib/include/cl/fs/path.hpp, 291  
 csv\_lib/include/cl/fs/separator.hpp, 292  
 csv\_lib/include/cl/fs/windows.hpp, 293  
 csv\_lib/include/cl/process.hpp, 295  
 csv\_lib/include/cl/read\_csv\_file.hpp, 295  
 csv\_lib/include/cl/s2n.hpp, 296  
 csv\_lib/include/cl/sensor.hpp, 297  
 csv\_lib/include/cl/to\_string.hpp, 299  
 csv\_lib/include/cl/use\_unbuffered\_io.hpp, 300  
 csv\_lib/src/cl/channel.cpp, 300  
 csv\_lib/src/cl/data\_point.cpp, 302  
 csv\_lib/src/cl/data\_set.cpp, 305  
 csv\_lib/src/cl/dos2unix.cpp, 305  
 csv\_lib/src/cl/error.cpp, 306  
 csv\_lib/src/cl/exception.cpp, 307  
 csv\_lib/src/cl/fs/directory\_listing.cpp, 307  
 csv\_lib/src/cl/fs/file.cpp, 308  
 csv\_lib/src/cl/fs/file\_stream.cpp, 308  
 csv\_lib/src/cl/fs/path.cpp, 309  
 csv\_lib/src/cl/fs/windows.cpp, 310  
 csv\_lib/src/cl/process.cpp, 310  
 csv\_lib/src/cl/read\_csv\_file.cpp, 311  
 csv\_lib/src/cl/sensor.cpp, 312  
 csv\_lib/src/cl/use\_unbuffered\_io.cpp, 313  
 csv\_lib/test/channel\_test.cpp, 313  
 csv\_lib/test/CMakeLists.txt, 179  
 csv\_lib/test/column\_test.cpp, 315  
 csv\_lib/test/data\_point\_test.cpp, 317  
 csv\_lib/test/data\_set\_test.cpp, 319  
 csv\_lib/test/directory\_listing\_test.cpp, 323  
 csv\_lib/test/error\_test.cpp, 325  
 csv\_lib/test/exception\_test.cpp, 327  
 csv\_lib/test/main.cpp, 200  
 csv\_lib/test/read\_csv\_file\_test.cpp, 328  
 csv\_lib/test/s2n\_test.cpp, 329  
 csv\_lib/test/sensor\_test.cpp, 331  
 csv\_lib/test/to\_string\_test.cpp, 332  
 csv\_line\_test.cpp  
     TEST, 208  
 CsvFileInfo  
     cm::CsvFileInfo, 101  
 CsvFileKind  
     cl, 13  
 CsvLineBuilder  
     cs::CsvLineBuilder, 103  
 ctg, 60  
     aboveThreshold, 60  
     averageComparisonValueCalculator, 61  
     halfMaximumComparisonValueCalculator, 61  
     isRelevant, 62  
     percentageOf, 63  
     runAboveThreshold, 64  
 data\_point.cpp  
     channel, 302  
     fileName, 302  
     sensor, 303  
     time, 303  
     value, 304  
 data\_point\_test.cpp  
     dp, 318  
     TEST, 317, 318  
 data\_set\_accessor\_v  
     cl, 23  
 data\_set\_identifier.cpp  
     CM\_DATA\_SET\_IDENTIFIER\_X, 249  
     DSI, 249  
 data\_set\_identifier.hpp  
     CM\_DATA\_SET\_IDENTIFIER, 231  
     CM\_DATA\_SET\_IDENTIFIER\_X, 232  
 data\_set\_identifier\_test.cpp  
     DSI, 257  
     TEST, 257  
 data\_set\_info.hpp  
     CS\_SPECIALIZE\_DATA\_SET\_INFO, 184  
 data\_set\_info\_test.cpp  
     TEST, 208  
 data\_set\_test.cpp  
     EXPECT\_LONG\_DOUBLE\_EQ, 319  
     TEST, 319–322  
 DataPoint  
     cl::DataPoint, 117  
 dataSet  
     cs::CsvLineBuilder, 104  
 dataSetAccessor  
     cl, 16  
 DataSetIdentifier  
     cm, 31  
 delete\_non\_bosch\_sensors.cpp

CL\_SENSOR\_X, 344  
deleteLowVariance  
  cs::CsvLineBuilder, 105  
  cs::LogInfo, 151  
deleteNonBoschSensors  
  fmc, 66  
deleteOutOfBoundsValues  
  fmc, 67  
deleteTooClose  
  cm::Configuration, 83  
  cm::Configuration::Builder, 73  
  cs::CsvLineBuilder, 106  
  cs::LogInfo, 151  
deleteTooCloseOptions  
  cm::Configuration, 83  
deleteTooLowVariance  
  cm::Configuration, 84  
  cm::Configuration::Builder, 74  
deleteTooLowVarianceOptions  
  cm::Configuration, 84  
directory\_listing\_test.cpp  
  TEST, 324  
directoryListing  
  cl::fs, 25  
DirectoryListingOption  
  cl::fs, 25  
distance  
  cm, 35  
distanceScore  
  cm, 36  
distScore  
  cm::ConfigWithDistanceScore, 95  
dos2unix  
  cl, 16  
dp  
  data\_point.cpp, 318  
DSI  
  data\_set\_identifier.cpp, 249  
  data\_set\_identifier\_test.cpp, 257  
  manual\_segmentation\_point.cpp, 253  
  manual\_segmentation\_point\_test.cpp, 260  
Error  
  cl::Error, 131  
error  
  error\_test.cpp, 326  
error.cpp  
  CL\_ERROR\_KIND\_X, 306  
error.hpp  
  CL\_ERROR\_KIND, 286  
  CL\_ERROR\_KIND\_X, 287  
  CL\_UNEXPECTED, 287  
error\_test.cpp  
  error, 326  
  TEST, 326  
Exception  
  cl::Exception, 136  
exception.hpp  
  CL\_THROW, 288  
          CL\_THROW\_FMT, 288  
exception\_test.cpp  
  TEST, 327  
ExcludeDotAndDotDot  
  cl::fs, 25  
exists  
  cl::fs::File, 140  
  cl::fs::Path, 168  
EXPECT\_LONG\_DOUBLE\_EQ  
  above\_threshold\_test.cpp, 275  
  data\_set\_test.cpp, 319  
  percentage\_of\_test.cpp, 277  
EXPECT\_SEGMENTATION\_POINTS  
  segment\_test.cpp, 263  
Expected  
  cl, 12  
ExtractId  
  cl, 13  
extractId  
  cl::DataSet, 125  
falseNegatives  
  cm::ConfusionMatrix, 98  
falsePositives  
  cm::ConfusionMatrix, 98  
fetch  
  cm, 37  
File  
  cl::fs::File, 138  
file  
  cl::Error, 132  
  cl::Exception, 136  
  cl::Process, 174  
fileName  
  cl::DataPoint, 117  
  cl::DataSet, 125  
  cs::LogLine, 156  
  data\_point.cpp, 302  
filePath  
  cs::LogLine, 157  
FileStream  
  cl::fs::FileStream, 145  
filter  
  cs::CsvLineBuilder, 107  
FilterKind  
  cs, 50  
filterKind  
  cm::Configuration, 85  
  cm::Configuration::Builder, 75  
  cs::LogInfo, 151  
filterKindOptions  
  cm::Configuration, 85  
fix\_csv/CMakeLists.txt, 180  
fix\_csv/include/adjust\_hardware\_timestamp.hpp, 333  
fix\_csv/include/convert\_to\_unix\_line\_endings.hpp, 334  
fix\_csv/include/create\_backup\_file.hpp, 335  
fix\_csv/include/delete\_non\_bosch\_sensors.hpp, 336  
fix\_csv/include/delete\_out\_of\_bounds\_values.hpp, 337  
fix\_csv/include/remove\_zeros\_from\_field.hpp, 338

fix\_csv/include/restore\_from\_backup.hpp, 339  
 fix\_csv/include/write\_file.hpp, 340  
 fix\_csv/src/adjust\_hardware\_timestamp.cpp, 341  
 fix\_csv/src/convert\_to\_unix\_line\_endings.cpp, 342  
 fix\_csv/src/create\_backup\_file.cpp, 343  
 fix\_csv/src/delete\_non\_bosch\_sensors.cpp, 343  
 fix\_csv/src/delete\_out\_of\_bounds\_values.cpp, 344  
 fix\_csv/src/main.cpp, 201  
 fix\_csv/src/remove\_zeros\_from\_field.cpp, 345  
 fix\_csv/src/restore\_from\_backup.cpp, 345  
 fix\_csv/src/write\_file.cpp, 346  
 fix\_csv/test/adjust\_hardware\_timestamp\_test.cpp, 347  
 fix\_csv/test/CMakeLists.txt, 180  
 fix\_csv/test/main.cpp, 202  
 fix\_csv/test/remove\_zeros\_from\_field\_test.cpp, 349  
**Fixed**  
 cl, 14  
**fmc**, 64  
 adjustHardwareTimestamp, 65  
 convertToUnixLineEndings, 65  
 createBackupFile, 66  
 deleteNonBoschSensors, 66  
 deleteOutOfBoundsValues, 67  
 removeZerosFromField, 67  
 restoreFromBackup, 68  
 writeFile, 68  
**formatError**  
 cl::fs, 26  
**frame**  
 cm::ManualSegmentationPoint, 162  
**function**  
 cl::Error, 132  
 cl::Exception, 136  
**gyroscopeAverage**  
 cl::DataSet, 126  
**gyroscopeMaximum**  
 cl::DataSet, 126  
**gyroscopeThreshold**  
 cl, 23  
**GyroscopeX**  
 cl, 13  
**gyroscopeX**  
 cl::DataSet, 127  
**GyroscopeY**  
 cl, 13  
**gyroscopeY**  
 cl::DataSet, 127  
**GyroscopeZ**  
 cl, 13  
**gyroscopeZ**  
 cl::DataSet, 128  
**halfMaximumComparisonValueCalculator**  
 ctg, 61  
**HardwareTimestamp**  
 cl, 13  
**hardwareTimestamp**  
 cl::DataSet, 128  
**hardwareTimestamps**  
 cm::CsvFileInfo, 101  
**hour**  
 cm::ManualSegmentationPoint, 163  
**importSegmentationPoints**  
 cm::Configuration, 86  
**Imu**  
 cm, 31  
**imu**  
 cm::Configuration, 87  
 cm::Configuration::Builder, 76  
**imu.cpp**  
 CM\_IMU\_X, 251  
**imu.hpp**  
 CM\_IMU, 236  
 CM\_IMU\_X, 237  
**imuCount**  
 cm, 48  
**imuOptions**  
 cm::Configuration, 87  
**imus**  
 cm, 48  
**include**  
 CMakeLists.txt, 177–181  
**incrementFalseNegatives**  
 cm::ConfusionMatrix, 98  
**incrementFalsePositives**  
 cm::ConfusionMatrix, 98  
**incrementTrueNegatives**  
 cm::ConfusionMatrix, 99  
**incrementTruePositives**  
 cm::ConfusionMatrix, 99  
**interpolated\_data\_set\_paths\_test.cpp**  
 TEST, 258  
**interpolatedDataSetPaths**  
 cm, 38  
**invalidSensor**  
 cs::LogInfo, 155  
 cs::LogLine, 159  
**isAccelerometer**  
 cl, 17  
**isDirectory**  
 cl::fs::Path, 169  
**isFile**  
 cl::fs::Path, 170  
**isGyroscope**  
 cl, 17  
**isInitialized**  
 cm::Configuration, 88  
 cs::LogInfo, 152  
**isOld**  
 cs::CsvLineBuilder, 108  
**isRelevant**  
 ctg, 62  
**Kind**  
 cl::Error, 131  
**kind**

cl::Error, 132  
cs::CsvLineBuilder, 109

line  
  cl::Error, 133  
  cl::Exception, 137

log\_files\_test.cpp  
  TEST, 209, 210

log\_info\_test.cpp  
  TEST, 212–220

log\_line\_test.cpp  
  TEST, 221–223

logFilePath  
  cs::LogInfo, 152

logFiles  
  cs, 55

LogInfo  
  cs::LogInfo, 150

logPath  
  cs, 59

main  
  main.cpp, 195, 197, 198, 200–204, 206

main.cpp  
  main, 195, 197, 198, 200–204, 206  
  SORT\_PRINT, 204

manual\_segmentation\_point.cpp  
  DSI, 253

manual\_segmentation\_point\_test.cpp  
  DSI, 260  
  TEST, 260–262

ManualSegmentationPoint  
  cm::ManualSegmentationPoint, 160

matrix  
  cm::ConfigWithTotalConfusionMatrix, 96

Maxima  
  cs, 52

message  
  cl::Error, 133

Minima  
  cs, 52

minute  
  cm::ManualSegmentationPoint, 163

moveTo  
  cl::fs::File, 141

MovingAverage  
  cs, 50

None  
  cl::fs, 25

oldLogPath  
  cs, 59

OpenMode  
  cl::fs::FileStream, 144

operator!=  
  cm, 39  
  cm::Configuration, 92  
  cm::ManualSegmentationPoint, 165

              cs, 56  
              cs::LogInfo, 154

operator<  
  cl::fs, 27  
  cl::fs::Path, 171  
  cm, 40

operator<<  
  cl, 18, 19  
  cl::DataPoint, 119  
  cl::Error, 134  
  cl::fs, 27  
  cl::fs::Path, 172  
  cm, 40–42  
  cm::Configuration, 93  
  cm::ManualSegmentationPoint, 166  
  cs, 57  
  cs::LogInfo, 154

operator()  
  std::hash<::cl::fs::Path >, 148  
  std::hash<::cm::Configuration >, 148

operator+=  
  cm::ConfusionMatrix, 99

operator=  
  cl::fs::FileStream, 146  
  cl::Process, 174

operator==  
  cl::fs, 27  
  cl::fs::Path, 172  
  cm, 43  
  cm::Configuration, 93  
  cm::ManualSegmentationPoint, 166  
  cs, 58  
  cs::LogInfo, 155

orderConfigurationsByQuality  
  cm, 44

parse  
  cs::LogLine, 157

Path  
  cl::fs::Path, 168

path  
  cl::fs::File, 141

percentage\_of\_test.cpp  
  EXPECT\_LONG\_DOUBLE\_EQ, 277  
  TEST, 277

percentageOf  
  ctg, 63

PL\_DEFINE\_EXCEPTION\_TYPE  
  cs, 58

PL\_NONCOPYABLE  
  cl::fs::FileStream, 147  
  cl::Process, 174

Process  
  cl::Process, 173

python\_output.cpp  
  CM\_DEV\_NULL, 255  
  CM\_SEGMENTOR, 255

pythonOutput  
  cm, 44

raise  
    cl::Error, 133

Raw  
    cl, 14

Read  
    cl::fs::FileStream, 145

read\_csv\_file\_test.cpp  
    TEST, 328, 329

readAll  
    cl::fs::FileStream, 147

readCsvFile  
    cl, 20  
        cm::ManualSegmentationPoint, 164

ReadWrite  
    cl::fs::FileStream, 145

remove  
    cl::fs::File, 142

remove\_zeros\_from\_field\_test.cpp  
    TEST, 350–352

removeZerosFromField  
    fmc, 67

repetitionCount  
    cs, 58

repetitions  
    cs::CsvLineBuilder, 110

restoreFromBackup  
    fmc, 68

rowCount  
    cl::DataSet, 129

runAboveThreshold  
    ctg, 64

s2n  
    cl, 20

s2n\_test.cpp  
    TEST, 330

SamplingRate  
    cl, 13

second  
    cm::ManualSegmentationPoint, 165

segment  
    cm, 45

segment\_test.cpp  
    EXPECT\_SEGMENTATION\_POINTS, 263  
    TEST, 263

SegmentationKind  
    cs, 50

segmentationKind  
    cm::Configuration, 88  
    cm::Configuration::Builder, 77  
    cs::LogInfo, 152

segmentationKindOptions  
    cm::Configuration, 89

segmentationPointCount  
    cs::LogLine, 158

segmentationPoints  
    cs::CsvLineBuilder, 111

Sensor  
    cl, 14

sensor  
    cl::DataPoint, 118  
    cs::CsvLineBuilder, 112  
    cs::LogInfo, 153  
    cs::LogLine, 158  
    data\_point.cpp, 303

sensor.cpp  
    CL\_SENSOR\_X, 312

sensor.hpp  
    CL\_SENSOR, 298  
    CL\_SENSOR\_X, 298, 299

sensor\_test.cpp  
    TEST, 332

sensors  
    cl, 24

separator.hpp  
    CL\_FS\_SEPARATOR, 293

serializeSegmentationPoints  
    cm::Configuration, 89

set  
    CMakeLists.txt, 177–181

size  
    cl::fs::File, 142

size\_type  
    cl::DataSet, 121

skipWindow  
    cm::Configuration, 90  
    cm::Configuration::Builder, 78  
    cs::CsvLineBuilder, 113  
    cs::LogInfo, 153

skipWindowOptions  
    cm::Configuration, 90

SORT\_PRINT  
    main.cpp, 204

split\_string\_test.cpp  
    TEST, 265

splitString  
    cm, 47

std::hash< Configuration >  
    cm::Configuration, 93

std::hash<::cl::fs::Path >, 148  
    operator(), 148

std::hash<::cm::Configuration >, 148  
    operator(), 148

str  
    cl::fs::Path, 170

TEST  
    above\_threshold\_test.cpp, 276  
    adjust\_hardware\_timestamp\_test.cpp, 347–349  
    channel\_test.cpp, 314, 315  
    column\_test.cpp, 316  
    csv\_line\_test.cpp, 208  
    data\_point\_test.cpp, 317, 318  
    data\_set\_identifier\_test.cpp, 257  
    data\_set\_info\_test.cpp, 208  
    data\_set\_test.cpp, 319–322  
    directory\_listing\_test.cpp, 324  
    error\_test.cpp, 326

exception\_test.cpp, 327  
interpolated\_data\_set\_paths\_test.cpp, 258  
log\_files\_test.cpp, 209, 210  
log\_info\_test.cpp, 212–220  
log\_line\_test.cpp, 221–223  
manual\_segmentation\_point\_test.cpp, 260–262  
percentage\_of\_test.cpp, 277  
read\_csv\_file\_test.cpp, 328, 329  
remove\_zeros\_from\_field\_test.cpp, 350–352  
s2n\_test.cpp, 330  
segment\_test.cpp, 263  
sensor\_test.cpp, 332  
split\_string\_test.cpp, 265  
to\_string\_test.cpp, 333

this\_type  
    cl::FileStream, 144  
    cl::Process, 173  
    cm::ConfusionMatrix, 97  
    cs::CsvLineBuilder, 103

threshold  
    cl, 21

Time  
    cl, 13

time  
    cl::DataPoint, 118  
    cl::DataSet, 129  
    data\_point.cpp, 303

to\_string  
    cl, 21  
    cl::Error, 134

to\_string\_test.cpp  
    TEST, 333

toDataSetIdentifier  
    cm, 47

totalCount  
    cm::ConfusionMatrix, 100

Trigger  
    cl, 13

trigger  
    cl::DataSet, 130

trueNegatives  
    cm::ConfusionMatrix, 100

truePositives  
    cm::ConfusionMatrix, 100

useUnbufferedIo  
    cl, 22

utf16ToUtf8  
    cl::fs, 28

utf8ToUtf16  
    cl::fs, 28

value  
    cl::DataPoint, 119  
    data\_point.cpp, 304

windowSize  
    cm::Configuration, 91  
    cm::Configuration::Builder, 79