

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	13
5.1.1.1 column_type	13
5.1.1.2 Expected	13
5.1.2 Enumeration Type Documentation	14
5.1.2.1 Channel	14
5.1.2.2 Column	14
5.1.2.3 CsvFileKind	15
5.1.2.4 Sensor	15
5.1.3 Function Documentation	15
5.1.3.1 CL_SPECIALIZE_COL_TRAITS() [1/11]	15
5.1.3.2 CL_SPECIALIZE_COL_TRAITS() [2/11]	15
5.1.3.3 CL_SPECIALIZE_COL_TRAITS() [3/11]	16
5.1.3.4 CL_SPECIALIZE_COL_TRAITS() [4/11]	16
5.1.3.5 CL_SPECIALIZE_COL_TRAITS() [5/11]	16
5.1.3.6 CL_SPECIALIZE_COL_TRAITS() [6/11]	16
5.1.3.7 CL_SPECIALIZE_COL_TRAITS() [7/11]	16
5.1.3.8 CL_SPECIALIZE_COL_TRAITS() [8/11]	16
5.1.3.9 CL_SPECIALIZE_COL_TRAITS() [9/11]	17
5.1.3.10 CL_SPECIALIZE_COL_TRAITS() [10/11]	17
5.1.3.11 CL_SPECIALIZE_COL_TRAITS() [11/11]	17
5.1.3.12 dataSetAccessor()	17
5.1.3.13 dos2unix()	18
5.1.3.14 isAccelerometer()	19
5.1.3.15 isGyroscope()	19
5.1.3.16 operator<<() [1/4]	20
5.1.3.17 operator<<() [2/4]	20
5.1.3.18 operator<<() [3/4]	21
5.1.3.19 operator<<() [4/4]	22
5.1.3.20 readCsvFile()	23
5.1.3.21 s2n()	23

5.1.3.22 threshold()	24
5.1.3.23 to_string()	25
5.1.3.24 useUnbufferedIo()	26
5.1.4 Variable Documentation	26
5.1.4.1 accelerometerThreshold	27
5.1.4.2 channelCount	27
5.1.4.3 channels	27
5.1.4.4 column_index	27
5.1.4.5 data_set_accessor_v	28
5.1.4.6 gyroscopeThreshold	28
5.1.4.7 sensors	28
5.2 cl::fs Namespace Reference	28
5.2.1 Enumeration Type Documentation	29
5.2.1.1 DirectoryListingOption	29
5.2.2 Function Documentation	29
5.2.2.1 directoryListing()	29
5.2.2.2 formatError()	30
5.2.2.3 operator<()	31
5.2.2.4 operator<<()	31
5.2.2.5 operator==()	32
5.2.2.6 utf16ToUtf8()	32
5.2.2.7 utf8ToUtf16()	33
5.3 cm Namespace Reference	34
5.3.1 Enumeration Type Documentation	36
5.3.1.1 DataSetIdentifier	36
5.3.1.2 Imu	36
5.3.2 Function Documentation	37
5.3.2.1 closestOne()	37
5.3.2.2 CM_SORTER() [1/4]	38
5.3.2.3 CM_SORTER() [2/4]	38
5.3.2.4 CM_SORTER() [3/4]	38
5.3.2.5 CM_SORTER() [4/4]	38
5.3.2.6 confusionMatrixBestConfigs()	39
5.3.2.7 createSegmentationResults()	40
5.3.2.8 distance()	41
5.3.2.9 distanceScore()	42
5.3.2.10 fetch()	43
5.3.2.11 interpolatedDataSetPaths()	44
5.3.2.12 operator"!="()	45
5.3.2.13 operator<() [1/2]	45
5.3.2.14 operator<() [2/2]	46
5.3.2.15 operator<<() [1/6]	46

5.3.2.16 operator<<() [2/6]	46
5.3.2.17 operator<<() [3/6]	46
5.3.2.18 operator<<() [4/6]	47
5.3.2.19 operator<<() [5/6]	48
5.3.2.20 operator<<() [6/6]	48
5.3.2.21 operator==() [1/2]	49
5.3.2.22 operator==() [2/2]	49
5.3.2.23 orderConfigurationsByQuality()	50
5.3.2.24 pythonOutput()	50
5.3.2.25 segment()	51
5.3.2.26 splitString()	53
5.3.2.27 toDataSetIdentifier()	53
5.3.3 Variable Documentation	54
5.3.3.1 addTrueSubtractFalseSorter	54
5.3.3.2 disregardTrueNegativesSorter	55
5.3.3.3 imuCount	55
5.3.3.4 imus	55
5.4 cs Namespace Reference	55
5.4.1 Enumeration Type Documentation	57
5.4.1.1 FilterKind	57
5.4.1.2 Mode	57
5.4.1.3 SegmentationKind	57
5.4.2 Function Documentation	58
5.4.2.1 CS_SPECIALIZE_DATA_SET_INFO() [1/20]	58
5.4.2.2 CS_SPECIALIZE_DATA_SET_INFO() [2/20]	58
5.4.2.3 CS_SPECIALIZE_DATA_SET_INFO() [3/20]	58
5.4.2.4 CS_SPECIALIZE_DATA_SET_INFO() [4/20]	58
5.4.2.5 CS_SPECIALIZE_DATA_SET_INFO() [5/20]	59
5.4.2.6 CS_SPECIALIZE_DATA_SET_INFO() [6/20]	59
5.4.2.7 CS_SPECIALIZE_DATA_SET_INFO() [7/20]	59
5.4.2.8 CS_SPECIALIZE_DATA_SET_INFO() [8/20]	59
5.4.2.9 CS_SPECIALIZE_DATA_SET_INFO() [9/20]	59
5.4.2.10 CS_SPECIALIZE_DATA_SET_INFO() [10/20]	59
5.4.2.11 CS_SPECIALIZE_DATA_SET_INFO() [11/20]	60
5.4.2.12 CS_SPECIALIZE_DATA_SET_INFO() [12/20]	60
5.4.2.13 CS_SPECIALIZE_DATA_SET_INFO() [13/20]	60
5.4.2.14 CS_SPECIALIZE_DATA_SET_INFO() [14/20]	60
5.4.2.15 CS_SPECIALIZE_DATA_SET_INFO() [15/20]	60
5.4.2.16 CS_SPECIALIZE_DATA_SET_INFO() [16/20]	60
5.4.2.17 CS_SPECIALIZE_DATA_SET_INFO() [17/20]	61
5.4.2.18 CS_SPECIALIZE_DATA_SET_INFO() [18/20]	61
5.4.2.19 CS_SPECIALIZE_DATA_SET_INFO() [19/20]	61

5.4.2.20 CS_SPECIALIZE_DATA_SET_INFO() [20/20]	61
5.4.2.21 logFiles()	61
5.4.2.22 operator"!="()	62
5.4.2.23 operator<<() [1/4]	63
5.4.2.24 operator<<() [2/4]	63
5.4.2.25 operator<<() [3/4]	63
5.4.2.26 operator<<() [4/4]	64
5.4.2.27 operator==()	64
5.4.2.28 parseMode()	65
5.4.2.29 PL_DEFINE_EXCEPTION_TYPE()	65
5.4.2.30 repetitionCount()	65
5.4.3 Variable Documentation	66
5.4.3.1 logPath	66
5.4.3.2 oldLogPath	67
5.5 ctg Namespace Reference	67
5.5.1 Function Documentation	67
5.5.1.1 aboveThreshold()	67
5.5.1.2 averageComparisonValueCalculator()	69
5.5.1.3 halfMaximumComparisonValueCalculator()	70
5.5.1.4 isRelevant()	71
5.5.1.5 percentageOf()	72
5.5.1.6 runAboveThreshold()	73
5.6 fmc Namespace Reference	73
5.6.1 Function Documentation	74
5.6.1.1 adjustHardwareTimestamp()	74
5.6.1.2 convertToUnixLineEndings()	75
5.6.1.3 createBackupFile()	76
5.6.1.4 deleteNonBoschSensors()	76
5.6.1.5 deleteOutOfBoundsValues()	77
5.6.1.6 removeZerosFromField()	78
5.6.1.7 restoreFromBackup()	78
5.6.1.8 writeFile()	79
6 Class Documentation	81
6.1 cm::Configuration::Builder Class Reference	81
6.1.1 Detailed Description	81
6.1.2 Constructor & Destructor Documentation	82
6.1.2.1 Builder()	82
6.1.3 Member Function Documentation	82
6.1.3.1 build()	82
6.1.3.2 deleteTooClose()	83
6.1.3.3 deleteTooLowVariance()	84

6.1.3.4 filterKind()	85
6.1.3.5 imu()	86
6.1.3.6 segmentationKind()	87
6.1.3.7 skipWindow()	88
6.1.3.8 windowSize()	89
6.2 cl::col_traits< Col > Struct Template Reference	90
6.2.1 Detailed Description	90
6.3 cm::Configuration Class Reference	91
6.3.1 Detailed Description	92
6.3.2 Constructor & Destructor Documentation	92
6.3.2.1 Configuration()	92
6.3.3 Member Function Documentation	93
6.3.3.1 createFilePath()	93
6.3.3.2 deleteTooClose()	93
6.3.3.3 deleteTooCloseOptions()	94
6.3.3.4 deleteTooLowVariance()	94
6.3.3.5 deleteTooLowVarianceOptions()	95
6.3.3.6 filterKind()	95
6.3.3.7 filterKindOptions()	96
6.3.3.8 importSegmentationPoints()	96
6.3.3.9 imu()	97
6.3.3.10 imuOptions()	98
6.3.3.11 isInitialized()	98
6.3.3.12 segmentationKind()	99
6.3.3.13 segmentationKindOptions()	99
6.3.3.14 serializeSegmentationPoints()	99
6.3.3.15 skipWindow()	100
6.3.3.16 skipWindowOptions()	101
6.3.3.17 windowSize()	101
6.3.3.18 windowSizeOptions()	102
6.3.4 Friends And Related Function Documentation	102
6.3.4.1 Builder	102
6.3.4.2 operator<	102
6.3.4.3 operator<<	103
6.3.4.4 operator==	103
6.3.4.5 std::hash< Configuration >	104
6.4 cm::ConfigWithDistanceScore Struct Reference	104
6.4.1 Detailed Description	104
6.4.2 Constructor & Destructor Documentation	104
6.4.2.1 ConfigWithDistanceScore()	105
6.4.3 Member Data Documentation	105
6.4.3.1 config	105

6.4.3.2 distScore	105
6.5 cm::ConfigWithTotalConfusionMatrix Struct Reference	105
6.5.1 Detailed Description	106
6.5.2 Constructor & Destructor Documentation	106
6.5.2.1 ConfigWithTotalConfusionMatrix() [1/2]	106
6.5.2.2 ConfigWithTotalConfusionMatrix() [2/2]	106
6.5.3 Friends And Related Function Documentation	107
6.5.3.1 operator<<	107
6.5.4 Member Data Documentation	107
6.5.4.1 config	107
6.5.4.2 matrix	107
6.6 cm::ConfusionMatrix Class Reference	108
6.6.1 Detailed Description	108
6.6.2 Member Typedef Documentation	108
6.6.2.1 this_type	109
6.6.3 Constructor & Destructor Documentation	109
6.6.3.1 ConfusionMatrix()	109
6.6.4 Member Function Documentation	109
6.6.4.1 falseNegatives()	109
6.6.4.2 falsePositives()	109
6.6.4.3 incrementFalseNegatives()	110
6.6.4.4 incrementFalsePositives()	110
6.6.4.5 incrementTrueNegatives()	111
6.6.4.6 incrementTruePositives()	111
6.6.4.7 operator+=()	111
6.6.4.8 totalCount()	112
6.6.4.9 trueNegatives()	112
6.6.4.10 truePositives()	113
6.7 cm::CsvFileInfo Class Reference	113
6.7.1 Detailed Description	113
6.7.2 Constructor & Destructor Documentation	113
6.7.2.1 CsvFileInfo()	113
6.7.3 Member Function Documentation	114
6.7.3.1 hardwareTimestamps()	114
6.8 cs::CsvLineBuilder Class Reference	114
6.8.1 Detailed Description	115
6.8.2 Member Typedef Documentation	115
6.8.2.1 this_type	115
6.8.3 Constructor & Destructor Documentation	116
6.8.3.1 CsvLineBuilder()	116
6.8.4 Member Function Documentation	116
6.8.4.1 build()	116

6.8.4.2 dataSet()	117
6.8.4.3 deleteLowVariance()	117
6.8.4.4 deleteTooClose()	118
6.8.4.5 filter()	119
6.8.4.6 isOld()	120
6.8.4.7 kind()	121
6.8.4.8 repetitions()	122
6.8.4.9 segmentationPoints()	123
6.8.4.10 sensor()	124
6.8.4.11 skipWindow()	125
6.8.4.12 windowSize()	126
6.9 cl::data_set_accessor< Chan > Struct Template Reference	127
6.9.1 Detailed Description	127
6.10 cs::data_set_info< Tag > Struct Template Reference	128
6.10.1 Detailed Description	128
6.11 cl::DataPoint Class Reference	128
6.11.1 Detailed Description	129
6.11.2 Constructor & Destructor Documentation	129
6.11.2.1 DataPoint()	129
6.11.3 Member Function Documentation	130
6.11.3.1 channel()	130
6.11.3.2 fileName()	131
6.11.3.3 sensor()	131
6.11.3.4 time()	132
6.11.3.5 value()	132
6.11.4 Friends And Related Function Documentation	133
6.11.4.1 operator<<	133
6.12 cl::DataSet Class Reference	133
6.12.1 Detailed Description	134
6.12.2 Member Typedef Documentation	135
6.12.2.1 ChannelAccessor	135
6.12.2.2 size_type	135
6.12.3 Member Function Documentation	135
6.12.3.1 accelerometerAverage()	135
6.12.3.2 accelerometerMaximum()	136
6.12.3.3 accelerometerX()	137
6.12.3.4 accelerometerY()	137
6.12.3.5 accelerometerZ()	138
6.12.3.6 create()	138
6.12.3.7 extractId()	140
6.12.3.8 fileName()	140
6.12.3.9 gyroscopeAverage()	141

6.12.3.10 gyroscopeMaximum()	142
6.12.3.11 gyroscopeX()	142
6.12.3.12 gyroscopeY()	143
6.12.3.13 gyroscopeZ()	143
6.12.3.14 hardwareTimestamp()	145
6.12.3.15 rowCount()	146
6.12.3.16 time()	146
6.12.3.17 trigger()	147
6.13 cl::Error Class Reference	147
6.13.1 Detailed Description	148
6.13.2 Member Enumeration Documentation	148
6.13.2.1 Kind	148
6.13.3 Constructor & Destructor Documentation	149
6.13.3.1 Error()	149
6.13.4 Member Function Documentation	149
6.13.4.1 file()	149
6.13.4.2 function()	150
6.13.4.3 kind()	150
6.13.4.4 line()	151
6.13.4.5 message()	151
6.13.4.6 raise()	151
6.13.4.7 to_string()	152
6.13.5 Friends And Related Function Documentation	152
6.13.5.1 operator<<	152
6.14 cl::Exception Class Reference	152
6.14.1 Detailed Description	154
6.14.2 Member Typedef Documentation	154
6.14.2.1 base_type	154
6.14.3 Constructor & Destructor Documentation	154
6.14.3.1 Exception() [1/2]	154
6.14.3.2 Exception() [2/2]	155
6.14.4 Member Function Documentation	155
6.14.4.1 file()	155
6.14.4.2 function()	156
6.14.4.3 line()	156
6.15 cl::fs::File Class Reference	157
6.15.1 Detailed Description	157
6.15.2 Constructor & Destructor Documentation	157
6.15.2.1 File()	157
6.15.3 Member Function Documentation	158
6.15.3.1 copyTo()	158
6.15.3.2 create()	159

6.15.3.3 exists()	159
6.15.3.4 moveTo()	160
6.15.3.5 path()	161
6.15.3.6 remove()	161
6.15.3.7 size()	162
6.16 cl::fs::FileStream Class Reference	162
6.16.1 Detailed Description	163
6.16.2 Member Typedef Documentation	163
6.16.2.1 this_type	163
6.16.3 Member Enumeration Documentation	163
6.16.3.1 OpenMode	163
6.16.4 Constructor & Destructor Documentation	164
6.16.4.1 FileStream()	164
6.16.4.2 ~FileStream()	164
6.16.5 Member Function Documentation	164
6.16.5.1 create()	164
6.16.5.2 operator=()	165
6.16.5.3 PL_NONCOPYABLE()	166
6.16.5.4 readAll()	166
6.16.5.5 write()	166
6.17 std::hash<::cl::fs::Path> Struct Reference	167
6.17.1 Detailed Description	167
6.17.2 Member Function Documentation	167
6.17.2.1 operator()	167
6.18 std::hash<::cm::Configuration> Struct Reference	167
6.18.1 Detailed Description	167
6.18.2 Member Function Documentation	167
6.18.2.1 operator()	168
6.19 cs::LogInfo Class Reference	168
6.19.1 Detailed Description	169
6.19.2 Constructor & Destructor Documentation	169
6.19.2.1 LogInfo()	169
6.19.3 Member Function Documentation	169
6.19.3.1 create()	169
6.19.3.2 deleteLowVariance()	170
6.19.3.3 deleteTooClose()	170
6.19.3.4 filterKind()	171
6.19.3.5 isInitialized()	171
6.19.3.6 logFilePath()	171
6.19.3.7 segmentationKind()	172
6.19.3.8 sensor()	172
6.19.3.9 skipWindow()	172

6.19.3.10 <code>windowSize()</code>	173
6.19.4 Friends And Related Function Documentation	173
6.19.4.1 <code>operator"!="</code>	173
6.19.4.2 <code>operator<<</code>	173
6.19.4.3 <code>operator==</code>	174
6.19.5 Member Data Documentation	174
6.19.5.1 <code>invalidSensor</code>	174
6.20 <code>cs::LogLine</code> Class Reference	174
6.20.1 Detailed Description	175
6.20.2 Member Function Documentation	175
6.20.2.1 <code>fileName()</code>	175
6.20.2.2 <code>filePath()</code>	176
6.20.2.3 <code>parse()</code>	177
6.20.2.4 <code>segmentationPointCount()</code>	177
6.20.2.5 <code>sensor()</code>	178
6.20.3 Member Data Documentation	178
6.20.3.1 <code>invalidSensor</code>	178
6.21 <code>cm::ManualSegmentationPoint</code> Class Reference	178
6.21.1 Detailed Description	179
6.21.2 Constructor & Destructor Documentation	179
6.21.2.1 <code>ManualSegmentationPoint()</code>	179
6.21.3 Member Function Documentation	180
6.21.3.1 <code>asMilliseconds()</code>	180
6.21.3.2 <code>convertToHardwareTimestamps()</code>	180
6.21.3.3 <code>frame()</code>	181
6.21.3.4 <code>hour()</code>	182
6.21.3.5 <code>minute()</code>	183
6.21.3.6 <code>readCsvFile()</code>	183
6.21.3.7 <code>second()</code>	184
6.21.4 Friends And Related Function Documentation	184
6.21.4.1 <code>operator"!="</code>	184
6.21.4.2 <code>operator<<</code>	185
6.21.4.3 <code>operator==</code>	185
6.22 <code>cl::fs::Path</code> Class Reference	186
6.22.1 Detailed Description	186
6.22.2 Constructor & Destructor Documentation	187
6.22.2.1 <code>Path() [1/3]</code>	187
6.22.2.2 <code>Path() [2/3]</code>	187
6.22.2.3 <code>Path() [3/3]</code>	187
6.22.3 Member Function Documentation	187
6.22.3.1 <code>exists()</code>	188
6.22.3.2 <code>isDirectory()</code>	188

6.22.3.3 <code>isFile()</code>	189
6.22.3.4 <code>str()</code>	190
6.22.4 Friends And Related Function Documentation	190
6.22.4.1 <code>operator<</code>	190
6.22.4.2 <code>operator<<</code>	191
6.22.4.3 <code>operator==</code>	191
6.23 <code>cl::Process</code> Class Reference	192
6.23.1 Detailed Description	192
6.23.2 Member Typedef Documentation	192
6.23.2.1 <code>this_type</code>	192
6.23.3 Constructor & Destructor Documentation	193
6.23.3.1 <code>Process()</code>	193
6.23.3.2 <code>~Process()</code>	193
6.23.4 Member Function Documentation	193
6.23.4.1 <code>create()</code>	193
6.23.4.2 <code>file() [1/2]</code>	194
6.23.4.3 <code>file() [2/2]</code>	194
6.23.4.4 <code>operator=()</code>	194
6.23.4.5 <code>PL_NONCOPYABLE()</code>	195
7 File Documentation	197
7.1 <code>compare_segmentation/CMakeLists.txt</code> File Reference	197
7.1.1 Function Documentation	197
7.1.1.1 <code>set()</code>	197
7.2 <code>compare_segmentation/test/CMakeLists.txt</code> File Reference	197
7.2.1 Function Documentation	197
7.2.1.1 <code>include()</code>	198
7.3 <code>counting/CMakeLists.txt</code> File Reference	198
7.3.1 Function Documentation	198
7.3.1.1 <code>set()</code>	198
7.4 <code>counting/test/CMakeLists.txt</code> File Reference	198
7.4.1 Function Documentation	198
7.4.1.1 <code>include()</code>	198
7.5 <code>csv_lib/CMakeLists.txt</code> File Reference	199
7.5.1 Function Documentation	199
7.5.1.1 <code>set()</code>	199
7.6 <code>csv_lib/test/CMakeLists.txt</code> File Reference	199
7.6.1 Function Documentation	199
7.6.1.1 <code>include()</code>	199
7.7 <code>fix_csv/CMakeLists.txt</code> File Reference	200
7.7.1 Function Documentation	200
7.7.1.1 <code>set()</code>	200

7.8 fix_csv/test/CMakeLists.txt File Reference	200
7.8.1 Function Documentation	200
7.8.1.1 include()	200
7.9 confusion_matrix/CMakeLists.txt File Reference	201
7.9.1 Function Documentation	201
7.9.1.1 set()	201
7.10 confusion_matrix/test/CMakeLists.txt File Reference	201
7.10.1 Function Documentation	201
7.10.1.1 include()	201
7.11 compare_segmentation/include/csv_line.hpp File Reference	202
7.12 compare_segmentation/include/data_set_info.hpp File Reference	203
7.12.1 Macro Definition Documentation	204
7.12.1.1 CS_SPECIALIZE_DATA_SET_INFO	204
7.13 compare_segmentation/include/filter_kind.hpp File Reference	205
7.14 compare_segmentation/include/log_files.hpp File Reference	206
7.15 compare_segmentation/include/log_info.hpp File Reference	206
7.16 compare_segmentation/include/log_line.hpp File Reference	207
7.17 compare_segmentation/include/mode.hpp File Reference	208
7.17.1 Macro Definition Documentation	209
7.17.1.1 CS_MODE	209
7.17.1.2 CS_MODE_X	209
7.18 compare_segmentation/include/paths.hpp File Reference	210
7.19 compare_segmentation/include/segmentation_kind.hpp File Reference	211
7.20 compare_segmentation/src/csv_line.cpp File Reference	212
7.21 compare_segmentation/src/data_set_info.cpp File Reference	212
7.22 compare_segmentation/src/filter_kind.cpp File Reference	213
7.23 compare_segmentation/src/log_files.cpp File Reference	213
7.24 compare_segmentation/src/log_info.cpp File Reference	214
7.25 compare_segmentation/src/log_line.cpp File Reference	215
7.26 compare_segmentation/src/main.cpp File Reference	215
7.26.1 Function Documentation	216
7.26.1.1 main()	216
7.27 compare_segmentation/test/main.cpp File Reference	218
7.27.1 Function Documentation	218
7.27.1.1 main()	218
7.28 counting/src/main.cpp File Reference	218
7.28.1 Function Documentation	219
7.28.1.1 main()	219
7.29 counting/test/main.cpp File Reference	220
7.29.1 Function Documentation	221
7.29.1.1 main()	221
7.30 csv_lib/test/main.cpp File Reference	221

7.30.1 Function Documentation	222
7.30.1.1 main()	222
7.31 fix_csv/src/main.cpp File Reference	222
7.31.1 Function Documentation	223
7.31.1.1 main()	223
7.32 fix_csv/test/main.cpp File Reference	223
7.32.1 Function Documentation	224
7.32.1.1 main()	224
7.33 confusion_matrix/src/main.cpp File Reference	224
7.33.1 Macro Definition Documentation	225
7.33.1.1 SORT_PRINT	225
7.33.2 Function Documentation	225
7.33.2.1 main()	225
7.34 confusion_matrix/test/main.cpp File Reference	226
7.34.1 Function Documentation	227
7.34.1.1 main()	227
7.35 compare_segmentation/src/mode.cpp File Reference	227
7.35.1 Macro Definition Documentation	228
7.35.1.1 CS_MODE_X [1/2]	228
7.35.1.2 CS_MODE_X [2/2]	228
7.36 compare_segmentation/src/segmentation_kind.cpp File Reference	229
7.37 compare_segmentation/test/csv_line_test.cpp File Reference	229
7.37.1 Function Documentation	230
7.37.1.1 TEST()	230
7.38 compare_segmentation/test/data_set_info_test.cpp File Reference	230
7.38.1 Function Documentation	230
7.38.1.1 TEST()	231
7.39 compare_segmentation/test/log_files_test.cpp File Reference	231
7.39.1 Function Documentation	231
7.39.1.1 TEST() [1/3]	232
7.39.1.2 TEST() [2/3]	232
7.39.1.3 TEST() [3/3]	233
7.40 compare_segmentation/test/log_info_test.cpp File Reference	233
7.40.1 Function Documentation	234
7.40.1.1 TEST() [1/19]	234
7.40.1.2 TEST() [2/19]	234
7.40.1.3 TEST() [3/19]	235
7.40.1.4 TEST() [4/19]	235
7.40.1.5 TEST() [5/19]	236
7.40.1.6 TEST() [6/19]	236
7.40.1.7 TEST() [7/19]	237
7.40.1.8 TEST() [8/19]	237

7.40.1.9 TEST() [9/19]	238
7.40.1.10 TEST() [10/19]	238
7.40.1.11 TEST() [11/19]	239
7.40.1.12 TEST() [12/19]	239
7.40.1.13 TEST() [13/19]	240
7.40.1.14 TEST() [14/19]	240
7.40.1.15 TEST() [15/19]	241
7.40.1.16 TEST() [16/19]	241
7.40.1.17 TEST() [17/19]	242
7.40.1.18 TEST() [18/19]	242
7.40.1.19 TEST() [19/19]	243
7.41 compare_segmentation/test/log_line_test.cpp File Reference	243
7.41.1 Function Documentation	243
7.41.1.1 TEST() [1/4]	244
7.41.1.2 TEST() [2/4]	244
7.41.1.3 TEST() [3/4]	245
7.41.1.4 TEST() [4/4]	245
7.42 compare_segmentation/test/mode_test.cpp File Reference	245
7.42.1 Function Documentation	246
7.42.1.1 TEST() [1/2]	246
7.42.1.2 TEST() [2/2]	247
7.43 confusion_matrix/include/closest_one.hpp File Reference	247
7.44 confusion_matrix/include/configuration.hpp File Reference	248
7.45 confusion_matrix/include/confusion_matrix.hpp File Reference	249
7.46 confusion_matrix/include/confusion_matrix_best_configs.hpp File Reference	250
7.46.1 Macro Definition Documentation	252
7.46.1.1 CM_SORTER	252
7.47 confusion_matrix/include/create_segmentation_results.hpp File Reference	252
7.48 confusion_matrix/include/csv_file_info.hpp File Reference	253
7.49 confusion_matrix/include/data_set_identifier.hpp File Reference	254
7.49.1 Macro Definition Documentation	256
7.49.1.1 CM_DATA_SET_IDENTIFIER	256
7.49.1.2 CM_DATA_SET_IDENTIFIER_X	256
7.50 confusion_matrix/include/distance.hpp File Reference	256
7.51 confusion_matrix/include/distance_score.hpp File Reference	257
7.52 confusion_matrix/include/fetch.hpp File Reference	258
7.53 confusion_matrix/include imu.hpp File Reference	259
7.53.1 Macro Definition Documentation	260
7.53.1.1 CM_IMU	261
7.53.1.2 CM_IMU_X [1/3]	261
7.53.1.3 CM_IMU_X [2/3]	261
7.53.1.4 CM_IMU_X [3/3]	261

7.54 confusion_matrix/include/interpolated_data_set_paths.hpp File Reference	262
7.55 confusion_matrix/include/manual_segmentation_point.hpp File Reference	263
7.56 confusion_matrix/include/order_configurations_by_quality.hpp File Reference	264
7.57 confusion_matrix/include/python_output.hpp File Reference	265
7.58 confusion_matrix/include/segment.hpp File Reference	266
7.59 confusion_matrix/include/split_string.hpp File Reference	267
7.60 confusion_matrix/src/closest_one.cpp File Reference	267
7.61 confusion_matrix/src/configuration.cpp File Reference	268
7.61.1 Macro Definition Documentation	269
7.61.1.1 CM_ENSURE_CONTAINS	269
7.61.1.2 CM_ENSURE_HAS_VALUE	269
7.62 confusion_matrix/src/confusion_matrix.cpp File Reference	270
7.63 confusion_matrix/src/confusion_matrix_best_configs.cpp File Reference	270
7.64 confusion_matrix/src/create_segmentation_results.cpp File Reference	271
7.65 confusion_matrix/src/csv_file_info.cpp File Reference	272
7.66 confusion_matrix/src/data_set_identifier.cpp File Reference	272
7.66.1 Macro Definition Documentation	273
7.66.1.1 CM_DATA_SET_IDENTIFIER_X	273
7.66.1.2 DSI	273
7.67 confusion_matrix/src/distance.cpp File Reference	274
7.68 confusion_matrix/src/distance_score.cpp File Reference	274
7.69 confusion_matrix/src/imu.cpp File Reference	275
7.69.1 Macro Definition Documentation	275
7.69.1.1 CM_IMU_X	276
7.70 confusion_matrix/src/interpolated_data_set_paths.cpp File Reference	276
7.71 confusion_matrix/src/manual_segmentation_point.cpp File Reference	276
7.71.1 Macro Definition Documentation	277
7.71.1.1 DSI	277
7.72 confusion_matrix/src/order_configurations_by_quality.cpp File Reference	278
7.73 confusion_matrix/src/python_output.cpp File Reference	278
7.73.1 Macro Definition Documentation	279
7.73.1.1 CM_DEV_NULL	279
7.73.1.2 CM_SEGMENTOR	279
7.74 confusion_matrix/src/segment.cpp File Reference	279
7.75 confusion_matrix/src/split_string.cpp File Reference	280
7.76 confusion_matrix/test/data_set_identifier_test.cpp File Reference	281
7.76.1 Macro Definition Documentation	281
7.76.1.1 DSI	281
7.76.2 Function Documentation	281
7.76.2.1 TEST()	282
7.77 confusion_matrix/test/interpolated_data_set_paths_test.cpp File Reference	282
7.77.1 Function Documentation	282

7.77.1.1 TEST()	283
7.78 confusion_matrix/test/manual_segmentation_point_test.cpp File Reference	283
7.78.1 Macro Definition Documentation	284
7.78.1.1 DS1	284
7.78.2 Function Documentation	284
7.78.2.1 TEST() [1/11]	284
7.78.2.2 TEST() [2/11]	285
7.78.2.3 TEST() [3/11]	285
7.78.2.4 TEST() [4/11]	285
7.78.2.5 TEST() [5/11]	285
7.78.2.6 TEST() [6/11]	285
7.78.2.7 TEST() [7/11]	286
7.78.2.8 TEST() [8/11]	286
7.78.2.9 TEST() [9/11]	286
7.78.2.10 TEST() [10/11]	286
7.78.2.11 TEST() [11/11]	286
7.79 confusion_matrix/test/segment_test.cpp File Reference	287
7.79.1 Macro Definition Documentation	287
7.79.1.1 EXPECT_SEGMENTATION_POINTS	287
7.79.2 Function Documentation	287
7.79.2.1 TEST()	288
7.80 confusion_matrix/test/split_string_test.cpp File Reference	288
7.80.1 Function Documentation	289
7.80.1.1 TEST()	289
7.81 counting/include/above_threshold.hpp File Reference	290
7.82 counting/include/average_comparison_value_calculator.hpp File Reference	291
7.83 counting/include/half_maximum_comparison_value_calculator.hpp File Reference	292
7.84 counting/include/is_relevant.hpp File Reference	293
7.85 counting/include/percentage_of.hpp File Reference	294
7.86 counting/include/run_above_threshold.hpp File Reference	295
7.87 counting/src/above_threshold.cpp File Reference	295
7.87.1 Macro Definition Documentation	296
7.87.1.1 CL_CHANNEL_X	296
7.87.2 Variable Documentation	296
7.87.2.1 channel	297
7.87.2.2 channelAccessor	297
7.88 counting/src/average_comparison_value_calculator.cpp File Reference	297
7.89 counting/src/half_maximum_comparison_value_calculator.cpp File Reference	298
7.90 counting/src/run_above_threshold.cpp File Reference	298
7.91 counting/test/above_threshold_test.cpp File Reference	299
7.91.1 Macro Definition Documentation	299
7.91.1.1 EXPECT_LONG_DOUBLE_EQ	300

7.91.2 Function Documentation	300
7.91.2.1 TEST()	300
7.92 counting/test/percentage_of_test.cpp File Reference	301
7.92.1 Macro Definition Documentation	301
7.92.1.1 EXPECT_LONG_DOUBLE_EQ	301
7.92.2 Function Documentation	301
7.92.2.1 TEST()	302
7.93 csv_lib/include/cl/channel.hpp File Reference	302
7.93.1 Macro Definition Documentation	304
7.93.1.1 CL_CHANNEL	304
7.93.1.2 CL_CHANNEL_X [1/4]	304
7.93.1.3 CL_CHANNEL_X [2/4]	304
7.93.1.4 CL_CHANNEL_X [3/4]	304
7.93.1.5 CL_CHANNEL_X [4/4]	305
7.94 csv_lib/include/cl/column.hpp File Reference	305
7.94.1 Detailed Description	307
7.94.2 Macro Definition Documentation	307
7.94.2.1 CL_SPECIALIZE_COL_TRAITS	307
7.95 csv_lib/include/cl/data_point.hpp File Reference	307
7.96 csv_lib/include/cl/data_set.hpp File Reference	308
7.97 csv_lib/include/cl/dos2unix.hpp File Reference	309
7.98 csv_lib/include/cl/error.hpp File Reference	310
7.98.1 Detailed Description	311
7.98.2 Macro Definition Documentation	311
7.98.2.1 CL_ERROR_KIND	311
7.98.2.2 CL_ERROR_KIND_X	311
7.98.2.3 CL_UNEXPECTED	311
7.99 csv_lib/include/cl/exception.hpp File Reference	312
7.99.1 Detailed Description	312
7.99.2 Macro Definition Documentation	313
7.99.2.1 CL_THROW	313
7.99.2.2 CL_THROW_FMT	313
7.100 csv_lib/include/cl/fs/directory_listing.hpp File Reference	313
7.101 csv_lib/include/cl/fs/file.hpp File Reference	314
7.102 csv_lib/include/cl/fs/file_stream.hpp File Reference	315
7.103 csv_lib/include/cl/fs/path.hpp File Reference	316
7.104 csv_lib/include/cl/fs/separator.hpp File Reference	317
7.104.1 Macro Definition Documentation	317
7.104.1.1 CL_FS_SEPARATOR	317
7.105 csv_lib/include/cl/fs/windows.hpp File Reference	318
7.105.1 Detailed Description	319
7.106 csv_lib/include/cl/process.hpp File Reference	319

7.107 csv_lib/include/cl/read_csv_file.hpp File Reference	320
7.108 csv_lib/include/cl/s2n.hpp File Reference	321
7.109 csv_lib/include/cl/sensor.hpp File Reference	321
7.109.1 Macro Definition Documentation	323
7.109.1.1 CL_SENSOR	323
7.109.1.2 CL_SENSOR_X [1/2]	323
7.109.1.3 CL_SENSOR_X [2/2]	323
7.110 csv_lib/include/cl/to_string.hpp File Reference	324
7.111 csv_lib/include/cl/use_unbuffered_io.hpp File Reference	324
7.112 csv_lib/src/cl/channel.cpp File Reference	325
7.112.1 Macro Definition Documentation	326
7.112.1.1 CL_CHANNEL_X [1/2]	326
7.112.1.2 CL_CHANNEL_X [2/2]	326
7.113 csv_lib/src/cl/data_point.cpp File Reference	326
7.113.1 Function Documentation	327
7.113.1.1 channel()	327
7.113.1.2 fileName()	327
7.113.1.3 sensor()	327
7.113.1.4 time()	328
7.113.1.5 value()	328
7.114 csv_lib/src/cl/data_set.cpp File Reference	329
7.115 csv_lib/src/cl/dos2unix.cpp File Reference	330
7.116 csv_lib/src/cl/error.cpp File Reference	331
7.116.1 Macro Definition Documentation	331
7.116.1.1 CL_ERROR_KIND_X	331
7.117 csv_lib/src/cl/exception.cpp File Reference	332
7.118 csv_lib/src/cl/fs/directory_listing.cpp File Reference	332
7.119 csv_lib/src/cl/fs/file.cpp File Reference	333
7.120 csv_lib/src/cl/fs/file_stream.cpp File Reference	333
7.121 csv_lib/src/cl/fs/path.cpp File Reference	334
7.122 csv_lib/src/cl/fs/windows.cpp File Reference	334
7.123 csv_lib/src/cl/process.cpp File Reference	335
7.124 csv_lib/src/cl/read_csv_file.cpp File Reference	336
7.125 csv_lib/src/cl/sensor.cpp File Reference	336
7.125.1 Macro Definition Documentation	337
7.125.1.1 CL_SENSOR_X	337
7.126 csv_lib/src/cl/use_unbuffered_io.cpp File Reference	338
7.127 csv_lib/test/channel_test.cpp File Reference	338
7.127.1 Function Documentation	339
7.127.1.1 TEST() [1/4]	339
7.127.1.2 TEST() [2/4]	339
7.127.1.3 TEST() [3/4]	339

7.127.1.4 TEST() [4/4]	340
7.128 csv_lib/test/column_test.cpp File Reference	340
7.128.1 Function Documentation	341
7.128.1.1 TEST() [1/2]	341
7.128.1.2 TEST() [2/2]	341
7.129 csv_lib/test/data_point_test.cpp File Reference	342
7.129.1 Function Documentation	342
7.129.1.1 TEST() [1/2]	342
7.129.1.2 TEST() [2/2]	343
7.129.2 Variable Documentation	343
7.129.2.1 dp	343
7.130 csv_lib/test/data_set_test.cpp File Reference	344
7.130.1 Macro Definition Documentation	344
7.130.1.1 EXPECT_LONG_DOUBLE_EQ	344
7.130.2 Function Documentation	344
7.130.2.1 TEST() [1/4]	345
7.130.2.2 TEST() [2/4]	345
7.130.2.3 TEST() [3/4]	346
7.130.2.4 TEST() [4/4]	347
7.131 csv_lib/test/directory_listing_test.cpp File Reference	348
7.131.1 Function Documentation	349
7.131.1.1 TEST() [1/3]	349
7.131.1.2 TEST() [2/3]	349
7.131.1.3 TEST() [3/3]	350
7.132 csv_lib/test/error_test.cpp File Reference	350
7.132.1 Function Documentation	351
7.132.1.1 TEST() [1/4]	351
7.132.1.2 TEST() [2/4]	351
7.132.1.3 TEST() [3/4]	351
7.132.1.4 TEST() [4/4]	351
7.132.2 Variable Documentation	351
7.132.2.1 error	352
7.133 csv_lib/test/exception_test.cpp File Reference	352
7.133.1 Function Documentation	352
7.133.1.1 TEST()	352
7.134 csv_lib/test/read_csv_file_test.cpp File Reference	353
7.134.1 Function Documentation	353
7.134.1.1 TEST() [1/2]	353
7.134.1.2 TEST() [2/2]	354
7.135 csv_lib/test/s2n_test.cpp File Reference	354
7.135.1 Function Documentation	355
7.135.1.1 TEST() [1/3]	355

7.135.1.2 TEST() [2/3]	355
7.135.1.3 TEST() [3/3]	356
7.136 csv_lib/test/sensor_test.cpp File Reference	356
7.136.1 Function Documentation	357
7.136.1.1 TEST() [1/2]	357
7.136.1.2 TEST() [2/2]	357
7.137 csv_lib/test/to_string_test.cpp File Reference	357
7.137.1 Function Documentation	358
7.137.1.1 TEST()	358
7.138 fix_csv/include/adjust_hardware_timestamp.hpp File Reference	358
7.139 fix_csv/include/convert_to_unix_line_endings.hpp File Reference	359
7.140 fix_csv/include/create_backup_file.hpp File Reference	360
7.141 fix_csv/include/delete_non_bosch_sensors.hpp File Reference	361
7.142 fix_csv/include/delete_out_of_bounds_values.hpp File Reference	362
7.143 fix_csv/include/remove_zeros_from_field.hpp File Reference	363
7.144 fix_csv/include/restore_from_backup.hpp File Reference	364
7.145 fix_csv/include/write_file.hpp File Reference	365
7.146 fix_csv/src/adjust_hardware_timestamp.cpp File Reference	366
7.147 fix_csv/src/convert_to_unix_line_endings.cpp File Reference	367
7.148 fix_csv/src/create_backup_file.cpp File Reference	368
7.149 fix_csv/src/delete_non_bosch_sensors.cpp File Reference	368
7.149.1 Macro Definition Documentation	369
7.149.1.1 CL_SENSOR_X	369
7.150 fix_csv/src/delete_out_of_bounds_values.cpp File Reference	369
7.151 fix_csv/src/remove_zeros_from_field.cpp File Reference	370
7.152 fix_csv/src/restore_from_backup.cpp File Reference	371
7.153 fix_csv/src/write_file.cpp File Reference	371
7.154 fix_csv/test/adjust_hardware_timestamp_test.cpp File Reference	372
7.154.1 Function Documentation	372
7.154.1.1 TEST() [1/5]	373
7.154.1.2 TEST() [2/5]	373
7.154.1.3 TEST() [3/5]	374
7.154.1.4 TEST() [4/5]	374
7.154.1.5 TEST() [5/5]	374
7.155 fix_csv/test/remove_zeros_from_field_test.cpp File Reference	375
7.155.1 Function Documentation	375
7.155.1.1 TEST() [1/6]	375
7.155.1.2 TEST() [2/6]	376
7.155.1.3 TEST() [3/6]	376
7.155.1.4 TEST() [4/6]	377
7.155.1.5 TEST() [5/6]	377
7.155.1.6 TEST() [6/6]	378

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

cl	11
cl::fs	28
cm	34
cs	55
ctg	67
fmc	73

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

cm::Configuration::Builder	81
cl::col_traits< Col >	90
cm::Configuration	91
cm::ConfigWithDistanceScore	104
cm::ConfigWithTotalConfusionMatrix	105
cm::ConfusionMatrix	108
cm::CsvFileInfo	113
cs::CsvLineBuilder	114
cl::data_set_accessor< Chan >	127
cs::data_set_info< Tag >	128
cl::DataPoint	128
cl::DataSet	133
cl::Error	147
std::exception	
std::runtime_error	
cl::Exception	152
cl::fs::File	157
cl::fs::FileStream	162
std::hash<::cl::fs::Path >	167
std::hash<::cm::Configuration >	167
cs::LogInfo	168
cs::LogLine	174
cm::ManualSegmentationPoint	178
cl::fs::Path	186
cl::Process	192

Chapter 3

Class Index

3.1 Class List

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

<code>cm::Configuration::Builder</code>	
<code>Builder</code> type for <code>Configuration</code>	81
<code>cl::col_traits< Col ></code>	
Column traits for the columns of the old, non-preprocessed CSV files. Includes the index (0 based) of the column and the C++ data type to be used for the cell contents in that column	90
<code>cm::Configuration</code>	
Represents a possible configuration for the Python segmentor	91
<code>cm::ConfigWithDistanceScore</code>	104
<code>cm::ConfigWithTotalConfusionMatrix</code>	
A <code>Configuration</code> with a <code>ConfusionMatrix</code>	105
<code>cm::ConfusionMatrix</code>	
Type to represent a confusion matrix	108
<code>cm::CsvFileInfo</code>	
Type to hold the hardware timestamps of a CSV file	113
<code>cs::CsvLineBuilder</code>	
Builder for a CSV line	114
<code>cl::data_set_accessor< Chan ></code>	
Maps a <code>Channel</code> to its associated accessor member function pointer of the <code>DataSet</code> type	127
<code>cs::data_set_info< Tag ></code>	
Meta function for data set tags	128
<code>cl::DataPoint</code>	
Type to represent a data point in a data set	128
<code>cl::DataSet</code>	
Type to represent a data set	133
<code>cl::Error</code>	
Type used to represent different kinds of errors	147
<code>cl::Exception</code>	
The exception type for this code base	152
<code>cl::fs::File</code>	
Represents a file	157
<code>cl::fs::FileStream</code>	
A binary file stream	162
<code>std::hash<::cl::fs::Path ></code>	167
<code>std::hash<::cm::Configuration ></code>	167
<code>cs::LogInfo</code>	
Information about a log file	168

cs::LogLine	174
A line out of a log file	
cm::ManualSegmentationPoint	178
Type used to represent a manual segmentation point	
cl::fs::Path	186
A filesystem path	
cl::Process	192
Type representing an operating system process	

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

compare_segmentation/include/csv_line.hpp	202
compare_segmentation/include/data_set_info.hpp	203
compare_segmentation/include/filter_kind.hpp	205
compare_segmentation/include/log_files.hpp	206
compare_segmentation/include/log_info.hpp	206
compare_segmentation/include/log_line.hpp	207
compare_segmentation/include/mode.hpp	208
compare_segmentation/include/paths.hpp	210
compare_segmentation/include/segmentation_kind.hpp	211
compare_segmentation/src/csv_line.cpp	212
compare_segmentation/src/data_set_info.cpp	212
compare_segmentation/src/filter_kind.cpp	213
compare_segmentation/src/log_files.cpp	213
compare_segmentation/src/log_info.cpp	214
compare_segmentation/src/log_line.cpp	215
compare_segmentation/src/main.cpp	215
compare_segmentation/src/mode.cpp	227
compare_segmentation/src/segmentation_kind.cpp	229
compare_segmentation/test/csv_line_test.cpp	229
compare_segmentation/test/data_set_info_test.cpp	230
compare_segmentation/test/log_files_test.cpp	231
compare_segmentation/test/log_info_test.cpp	233
compare_segmentation/test/log_line_test.cpp	243
compare_segmentation/test/main.cpp	218
compare_segmentation/test/mode_test.cpp	245
confusion_matrix/include/closest_one.hpp	247
confusion_matrix/include/configuration.hpp	248
confusion_matrix/include/confusion_matrix.hpp	249
confusion_matrix/include/confusion_matrix_best_configs.hpp	250
confusion_matrix/include/create_segmentation_results.hpp	252
confusion_matrix/include/csv_file_info.hpp	253
confusion_matrix/include/data_set_identifier.hpp	254
confusion_matrix/include/distance.hpp	256
confusion_matrix/include/distance_score.hpp	257
confusion_matrix/include/fetch.hpp	258

confusion_matrix/include/imu.hpp	259
confusion_matrix/include/interpolated_data_set_paths.hpp	262
confusion_matrix/include/manual_segmentation_point.hpp	263
confusion_matrix/include/order_configurations_by_quality.hpp	264
confusion_matrix/include/python_output.hpp	265
confusion_matrix/include/segment.hpp	266
confusion_matrix/include/split_string.hpp	267
confusion_matrix/src/closest_one.cpp	267
confusion_matrix/src/configuration.cpp	268
confusion_matrix/src/confusion_matrix.cpp	270
confusion_matrix/src/confusion_matrix_best_configs.cpp	270
confusion_matrix/src/create_segmentation_results.cpp	271
confusion_matrix/src/csv_file_info.cpp	272
confusion_matrix/src/data_set_identifier.cpp	272
confusion_matrix/src/distance.cpp	274
confusion_matrix/src/distance_score.cpp	274
confusion_matrix/src imu.cpp	275
confusion_matrix/src/interpolated_data_set_paths.cpp	276
confusion_matrix/src/main.cpp	224
confusion_matrix/src/manual_segmentation_point.cpp	276
confusion_matrix/src/order_configurations_by_quality.cpp	278
confusion_matrix/src/python_output.cpp	278
confusion_matrix/src/segment.cpp	279
confusion_matrix/src/split_string.cpp	280
confusion_matrix/test/data_set_identifier_test.cpp	281
confusion_matrix/test/interpolated_data_set_paths_test.cpp	282
confusion_matrix/test/main.cpp	226
confusion_matrix/test/manual_segmentation_point_test.cpp	283
confusion_matrix/test/segment_test.cpp	287
confusion_matrix/test/split_string_test.cpp	288
counting/include/above_threshold.hpp	290
counting/include/average_comparison_value_calculator.hpp	291
counting/include/half_maximum_comparison_value_calculator.hpp	292
counting/include/is_relevant.hpp	293
counting/include/percentage_of.hpp	294
counting/include/run_above_threshold.hpp	295
counting/src/above_threshold.cpp	295
counting/src/average_comparison_value_calculator.cpp	297
counting/src/half_maximum_comparison_value_calculator.cpp	298
counting/src/main.cpp	218
counting/src/run_above_threshold.cpp	298
counting/test/above_threshold_test.cpp	299
counting/test/main.cpp	220
counting/test/percentage_of_test.cpp	301
csv_lib/include/cl/channel.hpp	302
csv_lib/include/cl/column.hpp	
Contains definitions regarding the columns of the 'old' CSV files, i.e., the CSV files that have not been preprocessed using MATLAB	305
csv_lib/include/cl/data_point.hpp	307
csv_lib/include/cl/data_set.hpp	308
csv_lib/include/cl/dos2unix.hpp	309
csv_lib/include/cl/error.hpp	
Exports the Error type and related utilities	310
csv_lib/include/cl/exception.hpp	
Exports the cl::Exception type and related utility macros	312
csv_lib/include/cl/process.hpp	319
csv_lib/include/cl/read_csv_file.hpp	320
csv_lib/include/cl/s2n.hpp	321

csv_lib/include/cl/ sensor.hpp	321
csv_lib/include/cl/ to_string.hpp	324
csv_lib/include/cl/ use_unbuffered_io.hpp	324
csv_lib/include/cl/fs/ directory_listing.hpp	313
csv_lib/include/cl/fs/ file.hpp	314
csv_lib/include/cl/fs/ file_stream.hpp	315
csv_lib/include/cl/fs/ path.hpp	316
csv_lib/include/cl/fs/ separator.hpp	317
csv_lib/include/cl/fs/ windows.hpp	
Contains Microsoft Windows specific functions	318
csv_lib/src/cl/ channel.cpp	325
csv_lib/src/cl/ data_point.cpp	326
csv_lib/src/cl/ data_set.cpp	329
csv_lib/src/cl/ dos2unix.cpp	330
csv_lib/src/cl/ error.cpp	331
csv_lib/src/cl/ exception.cpp	332
csv_lib/src/cl/ process.cpp	335
csv_lib/src/cl/ read_csv_file.cpp	336
csv_lib/src/cl/ sensor.cpp	336
csv_lib/src/cl/ use_unbuffered_io.cpp	338
csv_lib/src/cl/fs/ directory_listing.cpp	332
csv_lib/src/cl/fs/ file.cpp	333
csv_lib/src/cl/fs/ file_stream.cpp	333
csv_lib/src/cl/fs/ path.cpp	334
csv_lib/src/cl/fs/ windows.cpp	334
csv_lib/test/ channel_test.cpp	338
csv_lib/test/ column_test.cpp	340
csv_lib/test/ data_point_test.cpp	342
csv_lib/test/ data_set_test.cpp	344
csv_lib/test/ directory_listing_test.cpp	348
csv_lib/test/ error_test.cpp	350
csv_lib/test/ exception_test.cpp	352
csv_lib/test/ main.cpp	221
csv_lib/test/ read_csv_file_test.cpp	353
csv_lib/test/ s2n_test.cpp	354
csv_lib/test/ sensor_test.cpp	356
csv_lib/test/ to_string_test.cpp	357
fix_csv/include/ adjust_hardware_timestamp.hpp	358
fix_csv/include/ convert_to_unix_line_endings.hpp	359
fix_csv/include/ create_backup_file.hpp	360
fix_csv/include/ delete_non_bosch_sensors.hpp	361
fix_csv/include/ delete_out_of_bounds_values.hpp	362
fix_csv/include/ remove_zeros_from_field.hpp	363
fix_csv/include/ restore_from_backup.hpp	364
fix_csv/include/ write_file.hpp	365
fix_csv/src/ adjust_hardware_timestamp.cpp	366
fix_csv/src/ convert_to_unix_line_endings.cpp	367
fix_csv/src/ create_backup_file.cpp	368
fix_csv/src/ delete_non_bosch_sensors.cpp	368
fix_csv/src/ delete_out_of_bounds_values.cpp	369
fix_csv/src/ main.cpp	222
fix_csv/src/ remove_zeros_from_field.cpp	370
fix_csv/src/ restore_from_backup.cpp	371
fix_csv/src/ write_file.cpp	371
fix_csv/test/ adjust_hardware_timestamp_test.cpp	372
fix_csv/test/ main.cpp	223
fix_csv/test/ remove_zeros_from_field_test.cpp	375

Chapter 5

Namespace Documentation

5.1 cl Namespace Reference

Namespaces

- [fs](#)

Classes

- struct [col_traits](#)

Column traits for the columns of the old, non-preprocessed CSV files. Includes the index (0 based) of the column and the C++ data type to be used for the cell contents in that column.

- struct [data_set_accessor](#)

Maps a Channel to its associated accessor member function pointer of the [DataSet](#) type.

- class [DataPoint](#)

Type to represent a data point in a data set.

- class [DataSet](#)

Type to represent a data set.

- class [Error](#)

Type used to represent different kinds of errors.

- class [Exception](#)

The exception type for this code base.

- class [Process](#)

Type representing an operating system process.

Typedefs

- template<Column Col>
using [column_type](#) = typename [col_traits](#)< Col >::type

Meta function for the type to use for the given Column.

- template<typename Ty >
using [Expected](#) = tl::expected< Ty, [Error](#) >

Type alias for tl::expected using [cl::Error](#) as the error type.

Enumerations

- enum `Channel` : std::uint64_t { `Channel::CL_CHANNEL_X`, `Channel::CL_CHANNEL_Y` }
Scoped enum for the 6 sensor channels.
- 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` }
A scoped enum for the columns of the old, non-preprocessed CSV files.
- enum `CsvFileKind` { `CsvFileKind::Raw`, `CsvFileKind::Fixed` }
Scoped enum for the CSV file kinds.
- enum `Sensor` : std::uint64_t { `Sensor::CL_SENSOR_X`, `Sensor::CL_SENSOR_Y` }
Scoped enum for the sensors.

Functions

- `DataSet::ChannelAccessor dataSetAccessor (Channel channel)`
Returns the data set accessor for the Channel given.
- `std::ostream & operator<< (std::ostream &os, Channel channel)`
Prints a given Channel to os.
- `bool isAccelerometer (Channel channel)`
Checks whether a given Channel is an accelerometer channel.
- `bool isGyroscope (Channel channel)`
Checks whether a given Channel is a gyroscope channel.
- `long double threshold (Channel channel)`
Returns the threshold value for 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`
Reads a CSV file.
- `template<typename Integer> Expected< Integer > s2n (const std::string &str, std::size_t *pos=nullptr, [[maybe_unused]] int base=10)`
Converts a string to an integer type.
- `std::ostream & operator<< (std::ostream &os, Sensor sensor)`
Prints a sensor to an ostream.
- `template<typename Ty> std::string to_string (const Ty &ty)`
Converts something to a string (if possible).
- `void useUnbufferedIo ()`
Routine to activate unbuffered I/O.
- `std::ostream & operator<< (std::ostream &os, const DataPoint &dataPoint)`
- `std::ostream & operator<< (std::ostream &os, const Error &error)`

Variables

- `constexpr std::size_t channelCount`
The amount of sensor channels (6).
- `constexpr std::array< Channel, channelCount > channels`
An array of the sensor channel enumerators.
- `template<Channel Chan>`
`constexpr CL_CHANNEL DataSet::ChannelAccessor data_set_accessor_v = data_set_accessor<Chan>::f`
- `constexpr long double accelerometerThreshold {1.99L}`
Constant for the maximum acceptable accelerometer value (in positive and negative direction).
- `constexpr long double gyroscopeThreshold {1999.99L}`
Constant for the maximum acceptable gyroscope value (in positive and negative direction).
- `template<Column Col>`
`constexpr std::size_t column_index = col_traits<Col>::index`
Meta function for the 0 based index of the given Column.
- `constexpr std::array< Sensor, 4 > sensors`
The 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
```

Meta function for the type to use for the given Column.

Template Parameters

<code>Col</code>	The Column enumerator to use.
------------------	-------------------------------

Definition at line 71 of file column.hpp.

5.1.1.2 `Expected`

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

Type alias for tl::expected using `cl::Error` as the error type.

Template Parameters

<code>Ty</code>	The expected type.
-----------------	--------------------

Definition at line 123 of file error.hpp.

5.1.2 Enumeration Type Documentation

5.1.2.1 Channel

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

Scoped enum for the 6 sensor channels.

Enumerator

CL_CHANNEL←_X	
CL_CHANNEL	

Definition at line 23 of file channel.hpp.

5.1.2.2 Column

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

A scoped enum for the columns of the old, non-preprocessed CSV files.

Enumerator

Time	
HardwareTimestamp	
ExtractId	
Trigger	
AccelerometerX	
AccelerometerY	
AccelerometerZ	
GyroscopeX	
GyroscopeY	
GyroscopeZ	
SamplingRate	

Definition at line 17 of file column.hpp.

5.1.2.3 CsvFileKind

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

Scoped enum for the CSV file kinds.

Enumerator

Raw	A raw CSV file
Fixed	A CSV file that's been fixed by the fix_csv application

Definition at line 14 of file read_csv_file.hpp.

5.1.2.4 Sensor

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

Scoped enum for the sensors.

Enumerator

CL_SENSOR	←	
	_X	
CL_SENSOR		

Definition at line 18 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 )
```

Returns the data set accessor for the Channel given.

Parameters

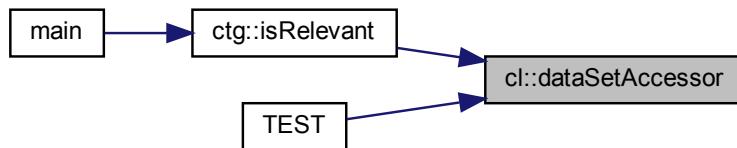
<i>channel</i>	The Channel to get the data set accessor for.
----------------	---

Returns

The data set accessor for 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 )
```

Checks whether a given Channel is an accelerometer channel.

Parameters

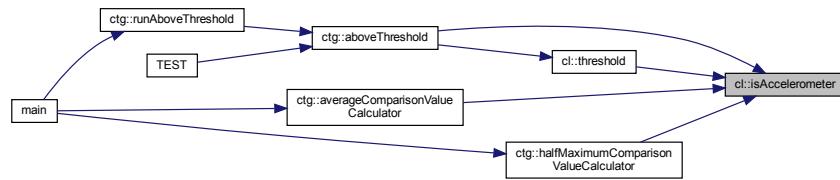
<i>channel</i>	The Channel to check.
----------------	-----------------------

Returns

true if *channel* is an accelerometer sensor channel; false otherwise.

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 )
```

Checks whether a given Channel is a gyroscope channel.

Parameters

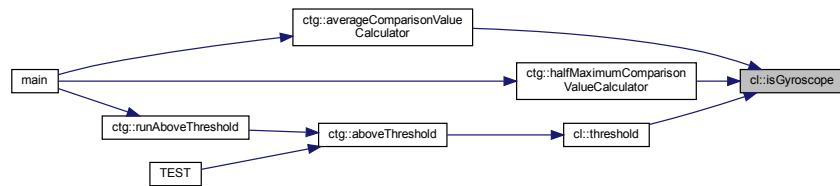
<i>channel</i>	The Channel to check.
----------------	-----------------------

Returns

true if *channel* is a gyroscope sensor channel; false otherwise.

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 )
```

Prints a given Channel to os.

Parameters

<i>os</i>	The ostream to print to.
<i>channel</i>	The Channel to print.

Returns

os.

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 )
```

Parameters

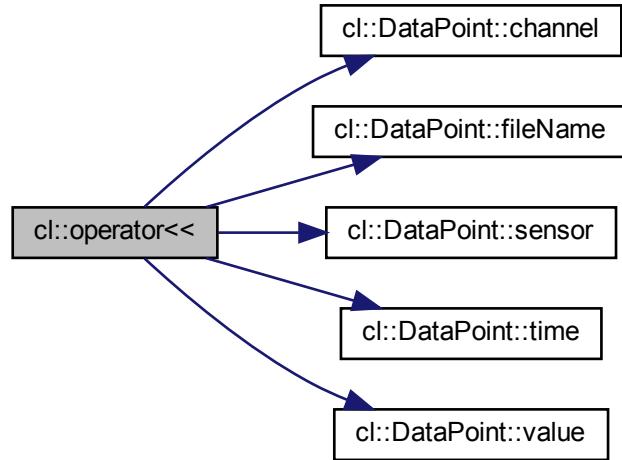
<i>os</i>	The ostream to print to.
<i>dataPoint</i>	The DataPoint to print.

Returns

os

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 )
```

Parameters

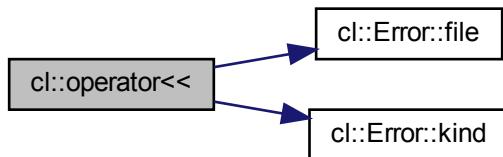
<code>os</code>	The ostream to print to.
<code>error</code>	The <code>Error</code> to print.

Returns

os.

Definition at line 35 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 )
```

Prints a sensor to an ostream.

Parameters

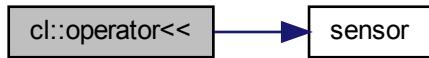
<i>os</i>	The ostream to print to.
<i>sensor</i>	The sensor to print.

Returns

os.

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]
```

Reads a CSV file.

Parameters

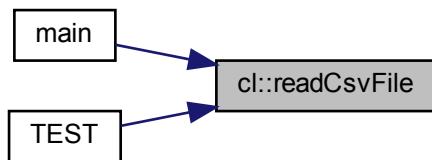
<code>csvFilePath</code>	The path to the CSV file.
<code>columnNames</code>	An output pointer to write the column names (headers) into. Can be nullptr if the column names should not be extracted.
<code>csvFileKind</code>	The kind of CSV file.

Returns

The resulting matrix or an error.

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]
```

Converts a string to an integer type.

Template Parameters

<code>Integer</code>	The integer type to convert to.
----------------------	---------------------------------

Parameters

<i>str</i>	The string to convert.
<i>pos</i>	Address of an integer to store the number of characters processed (may be nullptr).
<i>base</i>	The base to use (not used for floating point types).

Returns

The resulting number or on error.

Definition at line 25 of file s2n.hpp.

5.1.3.22 threshold()

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

Returns the threshold value for `channel`.

Parameters

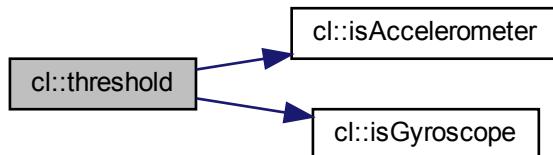
<i>channel</i>	The <code>Channel</code> to get the threshold value of.
----------------	---

Returns

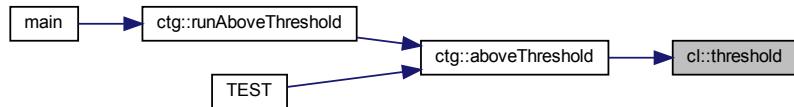
The threshold value for `channel`.

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]
```

Converts something to a string (if possible).

Template Parameters

<i>Ty</i>	The type of the object to convert to a string.
-----------	--

Parameters

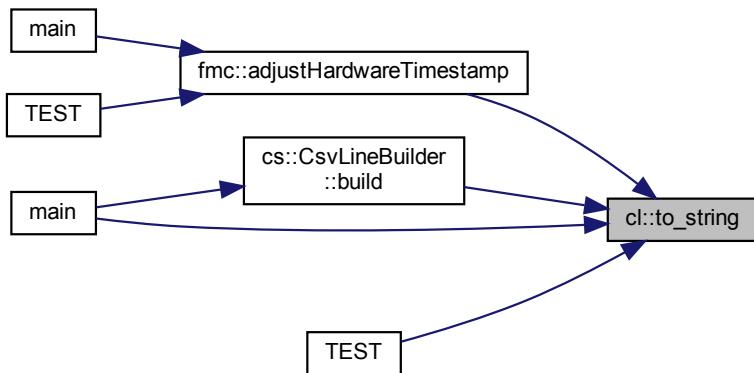
<i>ty</i>	The object to convert to a string.
-----------	------------------------------------

Returns

The resulting string.

Definition at line 22 of file `to_string.hpp`.

Here is the caller graph for this function:



5.1.3.24 `useUnbufferedIo()`

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

Routine to activate unbuffered I/O.

Note

This is only really useful for debugging purposes as the output to output streams that would normally be buffered (e.g., `stdout`) are unbuffered so that the output written to those output streams shows up even if the application crashes rather than leaving some output in those buffers.

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]
```

Constant for the maximum acceptable accelerometer value (in positive and negative direction).

Definition at line 102 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  
}
```

The amount of sensor channels (6).

Definition at line 32 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  
{}
```

An array of the sensor channel enumerators.

Definition at line 41 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]
```

Meta function for the 0 based index of the given Column.

Template Parameters

<i>Col</i>	The Column enumerator to use.
------------	-------------------------------

Definition at line 64 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 65 of file channel.hpp.

5.1.4.6 `gyroscopeThreshold`

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

Constant for the maximum acceptable gyroscope value (in positive and negative direction).

Definition at line 108 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
} }
```

The sensors.

Definition at line 27 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.

Returns

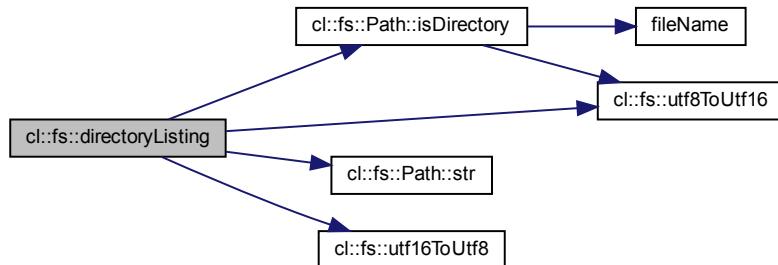
A vector of the directory entries or an error on failure.

Warning

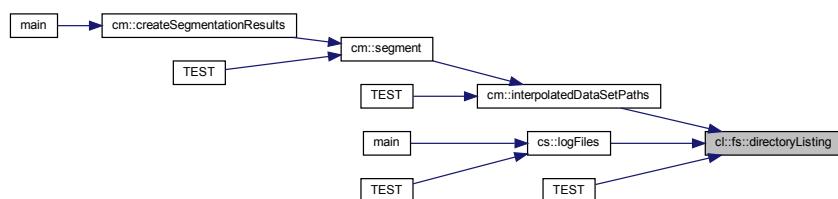
The paths returned are not complete paths to the files, but merely the identifiers of the directory's contents.

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

<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.

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.

Returnstrue 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

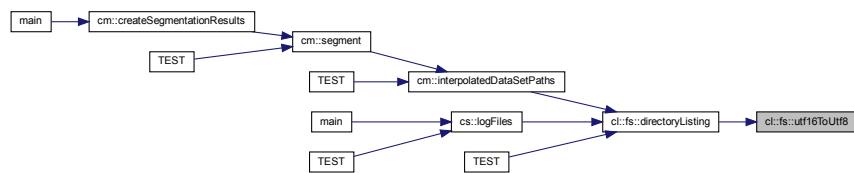
Most WINAPIs expect UTF-16 encoded wide character strings, but we don't want to pollute the code base with UTF-16 strings.

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

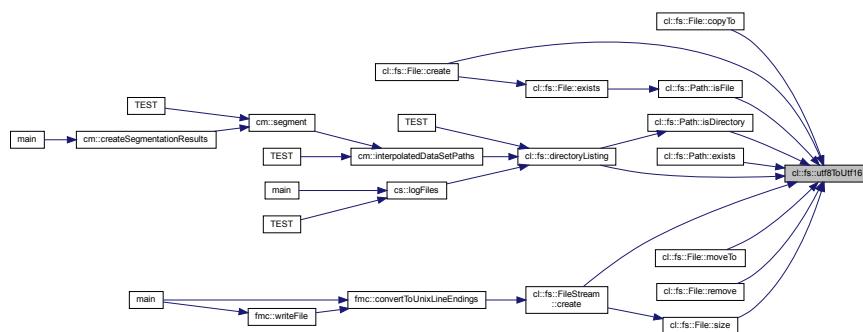
Most WINAPIs expect UTF-16 encoded wide character strings, but we don't want to pollute the code base with UTF-16 strings.

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](#)
A Configuration with a ConfusionMatrix.
- class [ConfusionMatrix](#)
Type to represent a confusion matrix.
- 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](#) }
Scoped enum type for the data set identifiers.
- 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, >)
Sorts ConfigWithTotalConfusionMatrix objects by true positives (highest first)

- **CM_SORTER** (`trueNegatives, >`)

Sorts `ConfigWithTotalConfusionMatrix` objects by true negatives (highest first)
- **CM_SORTER** (`falsePositives,<`)

Sorts `ConfigWithTotalConfusionMatrix` objects by false positives (lowest first)
- **CM_SORTER** (`falseNegatives,<`)

Sorts `ConfigWithTotalConfusionMatrix` objects by false negatives (lowest first)
- `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, const std::function< bool(const ConfigWithTotalConfusionMatrix &, const ConfigWithTotalConfusionMatrix &)> &sorter)`

Determines the 'best' configurations.
- `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)`

This old approach of distance scores didn't work too well for the confusion matrices.
- 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)`
- `std::ostream & operator<< (std::ostream &os, const ConfigWithTotalConfusionMatrix &obj)`
- `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

- struct {
 } [disregardTrueNegativesSorter](#)

Sorter to sort [ConfigWithTotalConfusionMatrix](#) objects by the count of true positives minus the count of false positives minus the count of false negatives.

- struct {
 } [addTrueSubtractFalseSorter](#)

Sorter to sort [ConfigWithTotalConfusionMatrix](#) objects by the sum of true positives and true negatives minus the false positives minus the false negatives.

- 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]
```

Scoped enum type for the data set identifiers.

Enumerator

CM_DATA_SET_IDENTIFIER	_X	
CM_DATA_SET_IDENTIFIER		

Definition at line 33 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

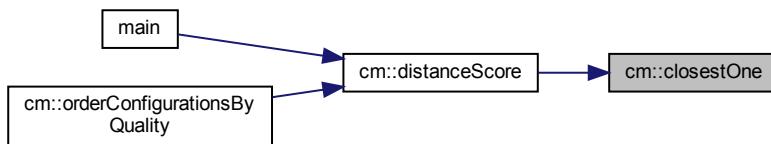
<code>cl::Exception</code>	if no segmentation point was found.
----------------------------	-------------------------------------

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 )
```

Sorts [ConfigWithTotalConfusionMatrix](#) objects by false negatives (lowest first)

5.3.2.3 CM_SORTER() [2/4]

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

Sorts [ConfigWithTotalConfusionMatrix](#) objects by false positives (lowest first)

5.3.2.4 CM_SORTER() [3/4]

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

Sorts [ConfigWithTotalConfusionMatrix](#) objects by true negatives (highest first)

5.3.2.5 CM_SORTER() [4/4]

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

Sorts [ConfigWithTotalConfusionMatrix](#) objects by true positives (highest first)

5.3.2.6 confusionMatrixBestConfigs()

```
std::vector< ConfigWithTotalConfusionMatrix > cm::confusionMatrixBestConfigs (
    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,
    const std::function< bool(const ConfigWithTotalConfusionMatrix &, const ConfigWithTotalConfusionMatrix &) >& sorter )
```

Determines the 'best' configurations.

Parameters

<i>manualSegmentationPoints</i>	The manual segmentation points (ground truth).
<i>algorithmicallyDeterminedSegmentationPoints</i>	The segmentation points found by the Python application.
<i>sorter</i>	The sorter to use.

Returns

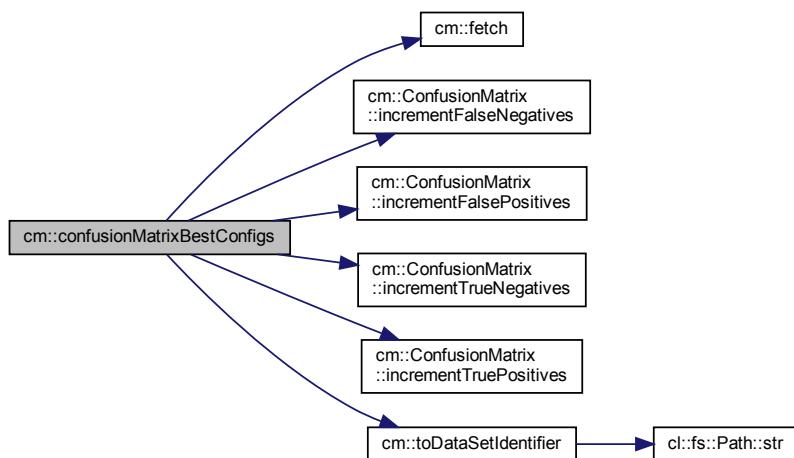
A vector of `ConfigWithTotalConfusionMatrix` objects sorted by `sorter`.

Exceptions

<code>cl::Exception</code>	if <code>sorter</code> does not contain a valid target.
----------------------------	---

Definition at line 102 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

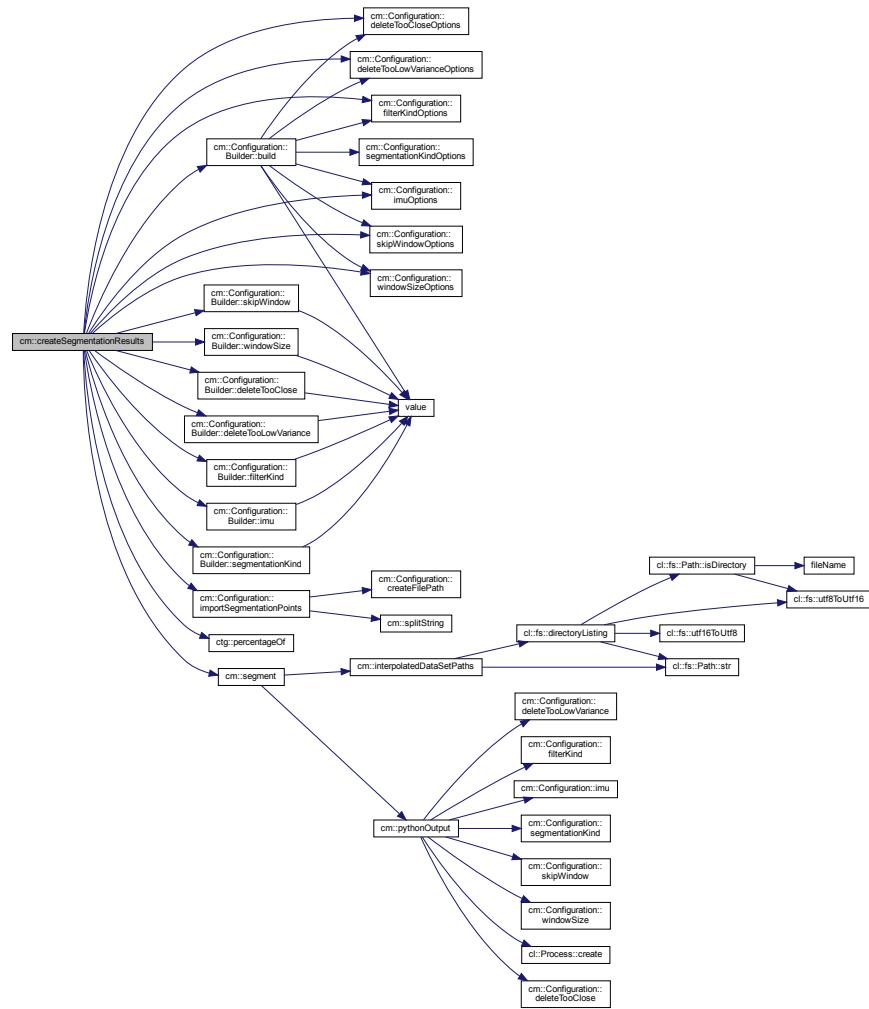
A map that maps the configurations to maps that map CSV file paths to segmentation points.

Exceptions

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

Definition at line 61 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

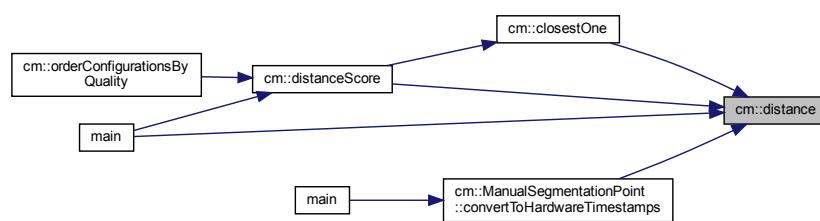
a	The first parameter.
b	The second parameter.

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()

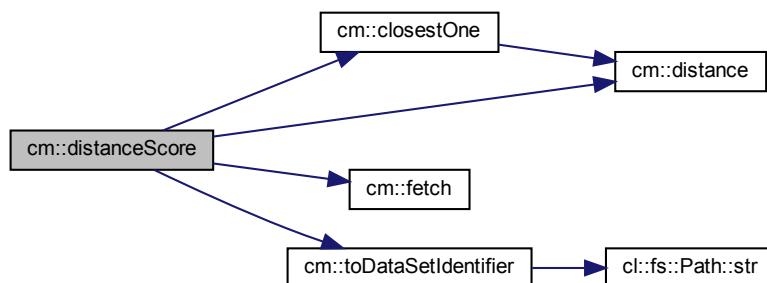
```

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 )
  
```

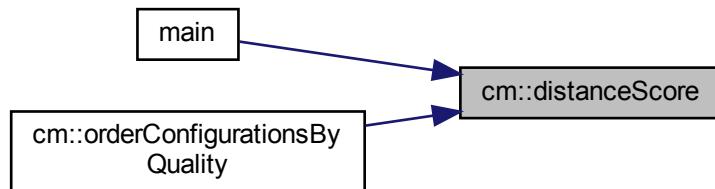
This old approach of distance scores didn't work too well for the confusion matrices.

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

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

Parameters

<i>map</i>	The map to fetch from.
<i>key</i>	The key to find the value for in <i>map</i> .

Returns

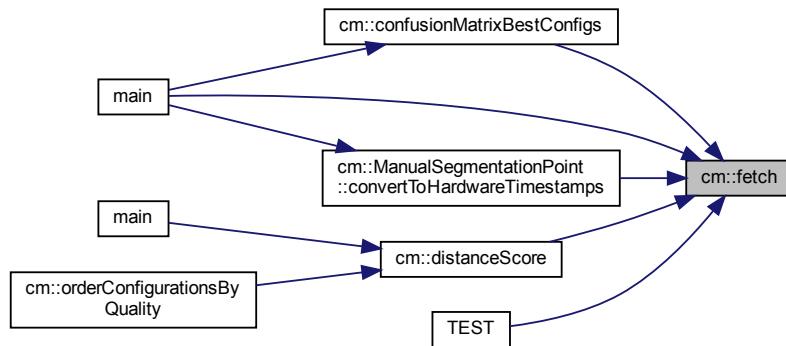
The value for *key* in *map*.

Exceptions

cl::Exception 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

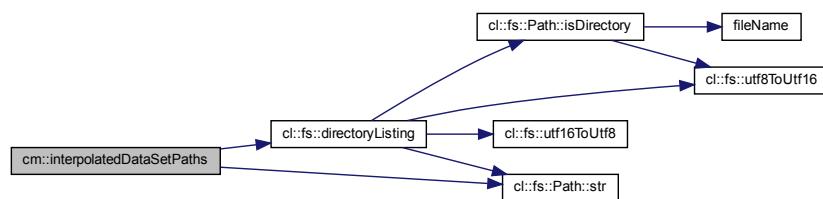
The interpolated data set paths.

Exceptions

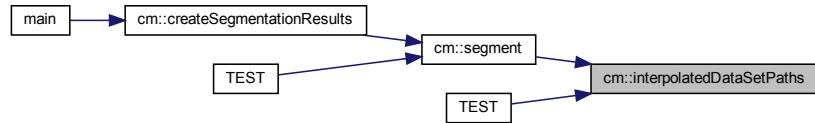
<i>cl::Exception</i>	on error.
----------------------	-----------

Definition at line 62 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"!=()

```
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 189 of file manual_segmentation_point.cpp.

5.3.2.13 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* is considered less than *rhs*; otherwise false.

Definition at line 218 of file configuration.cpp.

5.3.2.14 operator<() [2/2]

```
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<<() [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 238 of file configuration.cpp.

5.3.2.16 operator<<() [2/6]

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

Definition at line 27 of file order_configurations_by_quality.cpp.

5.3.2.17 operator<<() [3/6]

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

Parameters

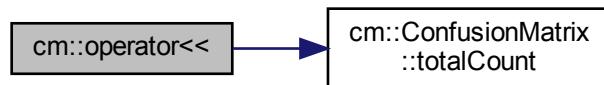
<i>os</i>	The ostream to print to.
<i>obj</i>	The ConfigWithTotalConfusionMatrix to print.

Returns

os

Definition at line 70 of file confusion_matrix_best_configs.cpp.

Here is the call graph for this function:

**5.3.2.18 operator<<() [4/6]**

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

Parameters

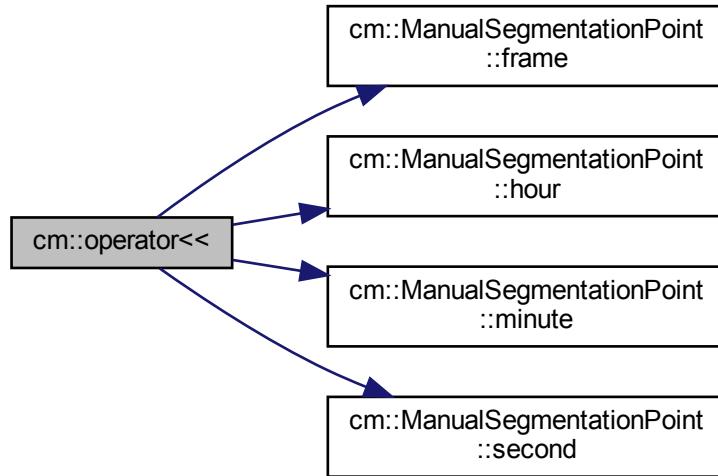
<i>os</i>	The ostream to print to
<i>manualSegmentationPoint</i>	The ManualSegmentationPoint to print.

Returns

os

Definition at line 196 of file manual_segmentation_point.cpp.

Here is the call graph for this function:



5.3.2.19 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.

5.3.2.20 operator<<() [6/6]

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

Prints *imu* to *os*.

Parameters

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

Returns

```
os
```

Definition at line 36 of file imu.cpp.

5.3.2.21 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 to be equal.

Definition at line 198 of file configuration.cpp.

5.3.2.22 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 equal to *rhs*; false otherwise.

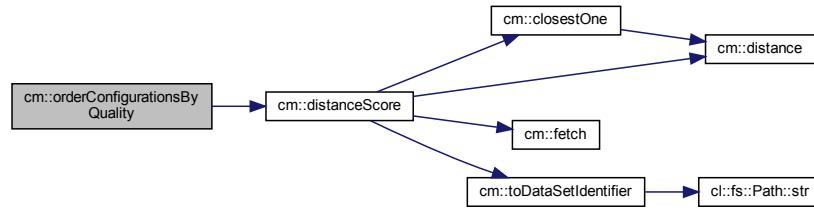
Definition at line 181 of file manual_segmentation_point.cpp.

5.3.2.23 orderConfigurationsByQuality()

```
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 )
```

Definition at line 38 of file `order_configurations_by_quality.cpp`.

Here is the call graph for this function:



5.3.2.24 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

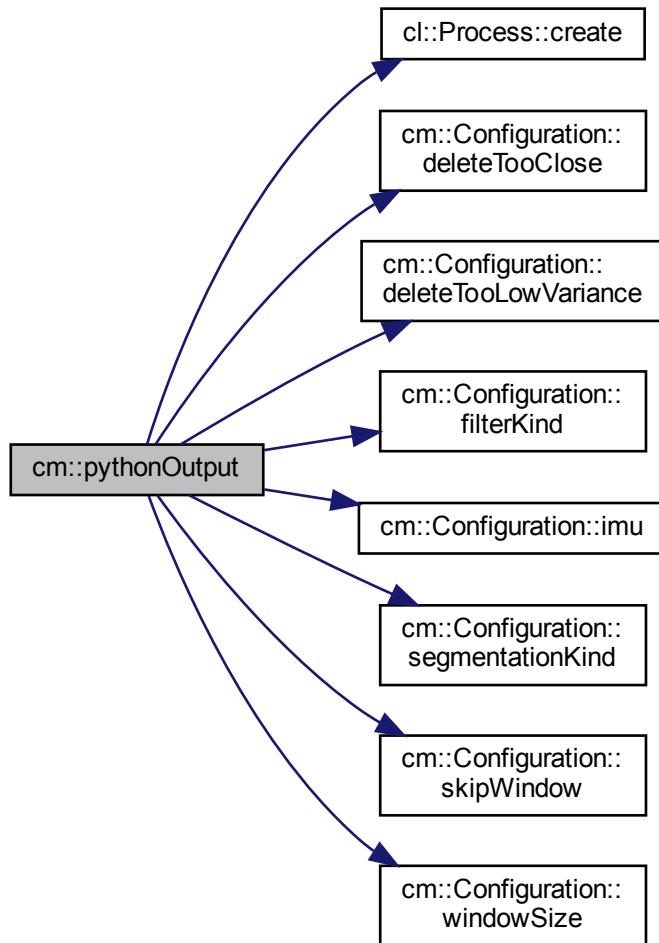
The output of the Python application.

Exceptions

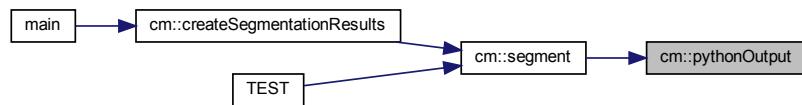
<code>cl::Exception</code>	if creating the process failed.
----------------------------	---------------------------------

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:



5.3.2.25 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

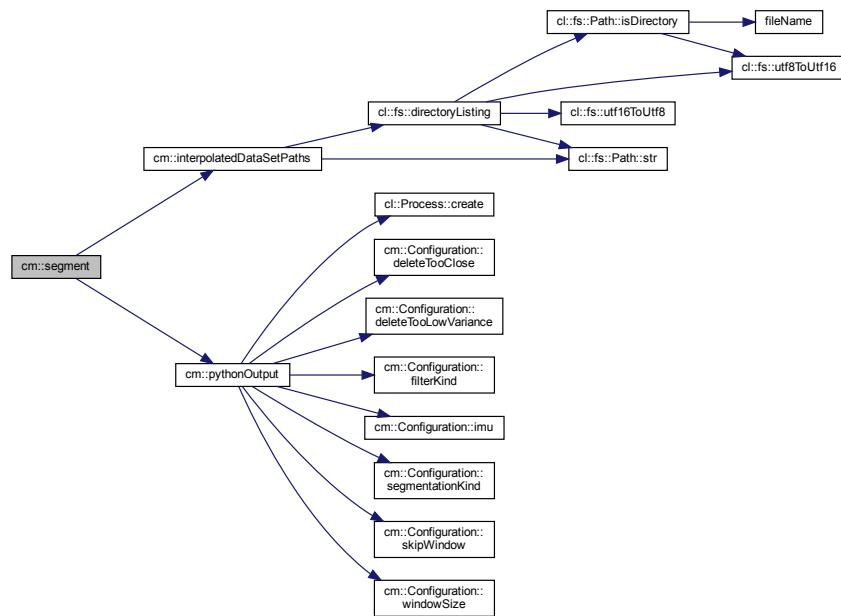
A map that maps the paths to the interpolated data sets to vectors of the hardware timestamps (in milliseconds) that are segmentation points.

Exceptions

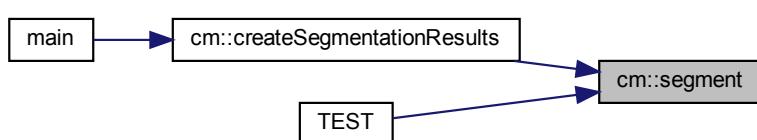
<i>cl::Exception</i>	if an error occurs.
--------------------------------------	---------------------

Definition at line 65 of file segment.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.3.2.26 splitString()

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

Splits `string` by `splitBy`.

Parameters

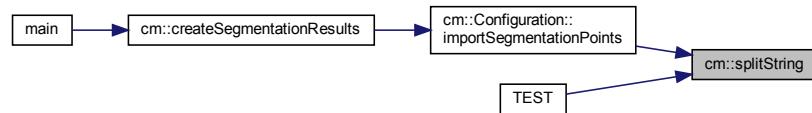
<code>string</code>	The string to split.
<code>splitBy</code>	What to split string by.

Returns

The resulting strings.

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

Here is the caller graph for this function:



5.3.2.27 toDataSetIdentifier()

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

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

Parameters

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

Returns

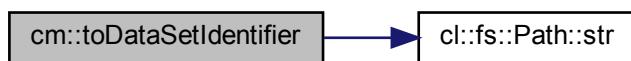
The resulting `DataSetIdentifier`.

Exceptions

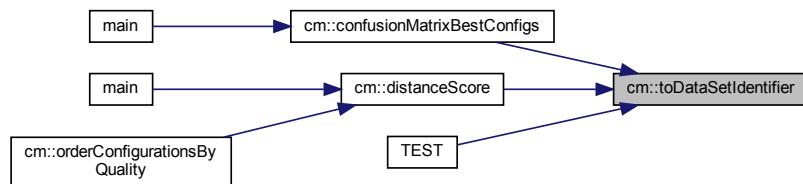
`cl::Exception` if path is unrecognized.

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 addTrueSubtractFalseSorter

```
constexpr { ... } cm::addTrueSubtractFalseSorter
```

Sorter to sort `ConfigWithTotalConfusionMatrix` objects by the sum of true positives and true negatives minus the false positives minus the false negatives.

Note

This one works the best.

5.3.3.2 disregardTrueNegativesSorter

```
constexpr { ... } cm::disregardTrueNegativesSorter
```

Sorter to sort [ConfigWithTotalConfusionMatrix](#) objects by the count of true positives minus the count of false positives minus the count of false negatives.

5.3.3.3 imuCount

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

Initial value:

```
{0  
#define CM_IMU_X(enm)  
    CM_IMU  
}
```

The amount of IMUs.

Definition at line 26 of file [imu.hpp](#).

5.3.3.4 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 `Mode` { `Mode::CS_MODE_X`, `Mode::CS_MODE` }

Enumerator type for the different modes of the compare_segmentation application.
- enum `SegmentationKind` : `pl::byte` { `SegmentationKind::Minima` = `0b0000'0001`, `SegmentationKind::Maxima` = `0b0000'0010`, `SegmentationKind::Both` = `Minima | Maxima` }

The segmentation kind (bitflag type)

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`, `Mode mode`)

Prints a Mode to an ostream.
- `cl::Expected< Mode > parseMode` (`const char *szCmdArg`)

Parses a null-terminated byte character string as a Mode.
- `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 Mode

`enum cs::Mode [strong]`

Enumerator type for the different modes of the compare_segmentation application.

Enumerator

CS_MODE	↔	
CS_MODE	X	

Definition at line 19 of file mode.hpp.

5.4.1.3 SegmentationKind

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

The segmentation kind (bitflag type)

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

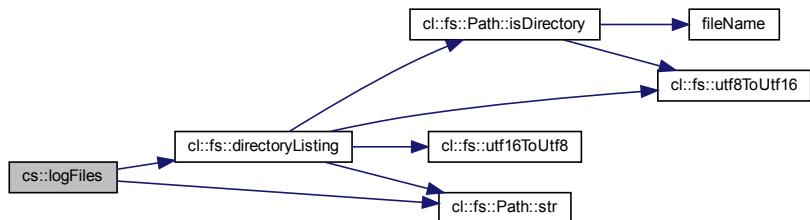
<i>directoryPath</i>	The path to a directory to search for log files.
----------------------	--

Returns

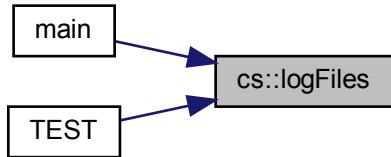
The log files found or an error.

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<<() [1/4]

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

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

```
std::ostream & cs::operator<< (
    std::ostream & os,
    Mode mode )
```

Prints a Mode to an ostream.

Parameters

<i>os</i>	The ostream to print to.
<i>mode</i>	The Mode to print.

Returns

os

Definition at line 13 of file mode.cpp.

5.4.2.26 operator<<() [4/4]

```
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.

5.4.2.27 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.

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

Definition at line 264 of file log_info.cpp.

5.4.2.28 parseMode()

```
cl::Expected< Mode > cs::parseMode (
    const char * szCmdArg )
```

Parses a null-terminated byte character string as a Mode.

Parameters

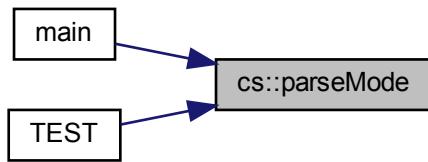
szCmdArg	A null-terminated byte character string containing a textual representation of a Mode.
----------	--

Returns

The mode parsed out of szCmdArg or an error.

Definition at line 25 of file mode.cpp.

Here is the caller graph for this function:



5.4.2.29 PL_DEFINE_EXCEPTION_TYPE()

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

5.4.2.30 repetitionCount()

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

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

Parameters

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

Returns

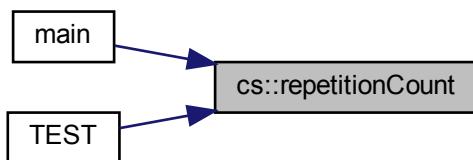
The repetition count of *dataSet*.

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.

Note

The working directory is assumed to be the root mogasens_csv path. The bash / batch scripts should already take care of that.

Definition at line 11 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.

Note

The working directory is assumed to be the root mogasens_csv path. The bash / batch scripts should already take care of that.

Definition at line 18 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)

Returns all the data points that are above / below the threshold.
- long double [averageComparisonValueCalculator](#) ([cl::Sensor](#) sensor, [cl::Channel](#) channel, const [cl::DataSet](#) &dataSet)

Calculates the average value for the given sensor, channel and data set.
- long double [halfMaximumComparisonValueCalculator](#) ([cl::Sensor](#) sensor, [cl::Channel](#) channel, const [cl::DataSet](#) &dataSet)

Returns half of the maximum for the given sensor, channel and data set.
- template<typename ComparisonValueCalculator>
 bool [isRelevant](#) ([cl::Sensor](#) sensor, [cl::Channel](#) channel, const [cl::DataSet](#) &dataSet, ComparisonValueCalculator comparisonValueCalculator)

Checks if a channel is relevant for a sensor in a data set.
- constexpr long double [percentageOf](#) (std::size_t amount, std::size_t totalCount) noexcept

Calculates what percentage amount is of totalCount.
- void [runAboveThreshold](#) (std::ostream &aboveThresholdLogFileStream, const [cl::DataSet](#) &dataSet)

Routine to find the above threshold values.

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 )
```

Returns all the data points that are above / below the threshold.

Parameters

<i>dataSet</i>	The data set.
<i>accelerometerThreshold</i>	The accelerometer threshold (must be positive)
<i>gyroscopeThreshold</i>	The gyroscope threshold (must be positive)

Returns

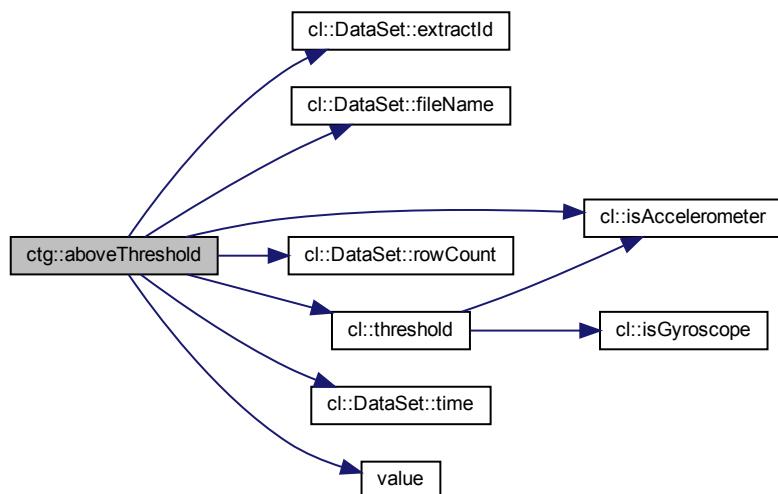
The data points that are invalid.

Note

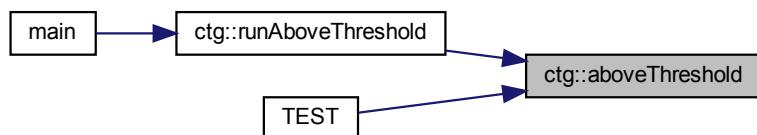
The unary operator operator- is applied to the two thresholds given to determine the minimum threshold.

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 )
```

Calculates the average value for the given sensor, channel and data set.

Parameters

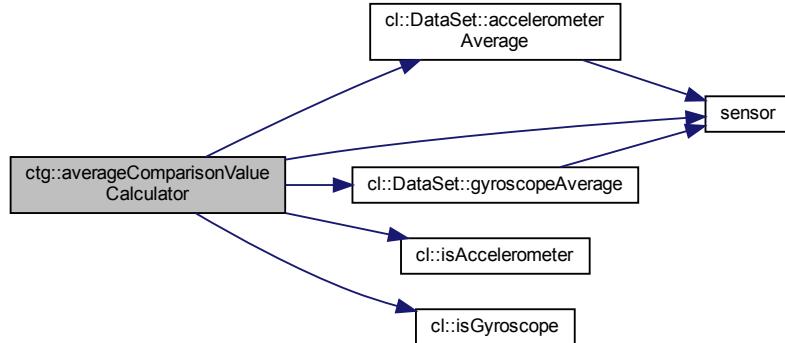
<i>sensor</i>	The sensor.
<i>channel</i>	The channel.
<i>dataSet</i>	The data set.

Returns

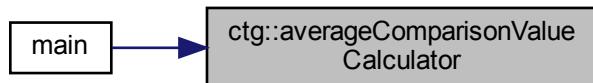
The average value.

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 )
```

Returns half of the maximum for the given sensor, channel and data set.

Parameters

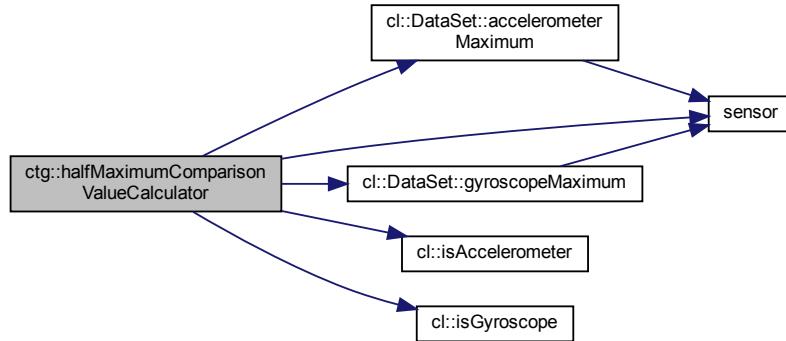
<i>sensor</i>	The sensor.
<i>channel</i>	The channel.
<i>dataSet</i>	The data set.

Returns

Half of the maximum.

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 )
```

Checks if a channel is relevant for a sensor in a data set.

Template Parameters

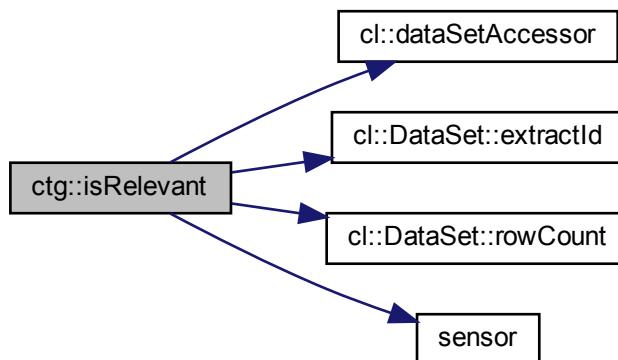
<i>ComparisonValueCalculator</i>	The type of the ComparisonValueCalculator function to use.
----------------------------------	--

Parameters

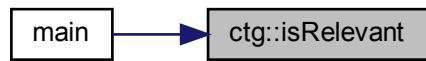
<i>sensor</i>	The sensor.
<i>channel</i>	The channel.
<i>dataSet</i>	The data set.
<i>comparisonValueCalculator</i>	The function to use to get the value to compare to.

Definition at line 21 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]
```

Calculates what percentage `amount` is of `totalCount`.

Parameters

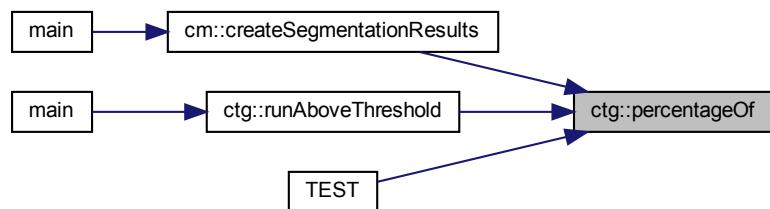
<code>amount</code>	The first argument.
<code>totalCount</code>	The second argument.

Returns

The percentage.

Definition at line 12 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 )
```

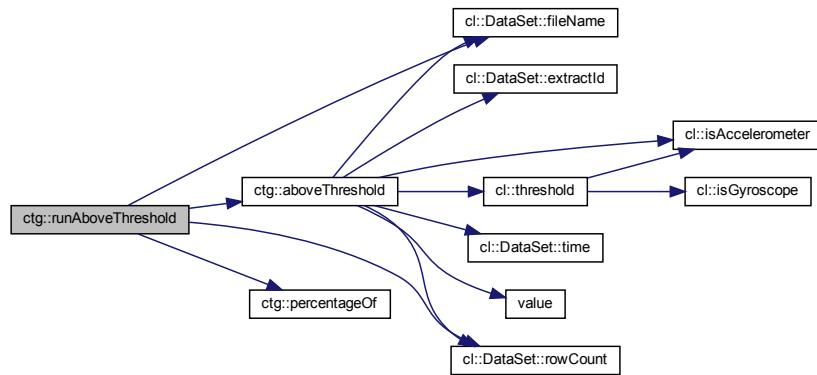
Routine to find the above threshold values.

Parameters

<i>aboveThresholdLogFileStream</i>	The stream to write to.
<i>dataSet</i>	The data set.

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`)

- Adjust the hardware timestamp.*
- `bool convertToUnixLineEndings (const std::string &csvPath)`
Convert a CSV file to UNIX line endings.
 - `bool createBackupFile (const std::string &csvFilePath, const std::string &backupFilePath)`
Creates a backup of a file.
 - `void deleteNonBoschSensors (std::vector< std::vector< std::string >> *data)`
Routine to delete anything that isn't a Bosch sensor.
 - `cl::Expected< void > deleteOutOfBoundsValues (std::vector< std::vector< std::string >> *data)`
Deletes out of bounds values from the raw CSV file read.
 - `void removeZerosFromField (std::string *field)`
Deletes extraneous zeros from a cell value.
 - `bool restoreFromBackup (const std::string &csvFilePath, const std::string &backupFilePath)`
Restore a file from a previously created backup file.
 - `bool writeFile (pl::string_view csvPath, pl::string_view csvFileExtension, const std::vector< std::string > &columnNames, const std::vector< std::vector< std::string >> &data)`
Writes a 'fixed' CSV file.

5.6.1 Function Documentation

5.6.1.1 `adjustHardwareTimestamp()`

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

Adjust the hardware timestamp.

Parameters

<code>cellContent</code>	Pointer to the string of a cell containing a hardware timestamp.
<code>nextRowHardwareTimestamp</code>	The hardware timestamp in the next row.
<code>overflowCount</code>	Pointer to the uint64_t that counts the overflows.

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 )
```

Convert a CSV file to UNIX line endings.

Parameters

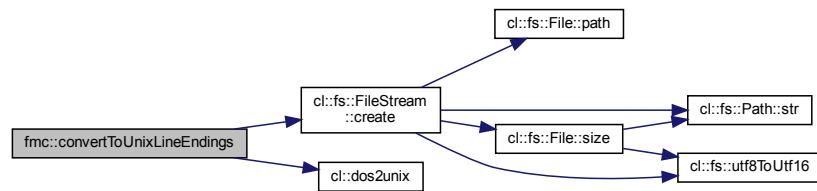
<i>csvPath</i>	The path to the CSV file.
----------------	---------------------------

Returns

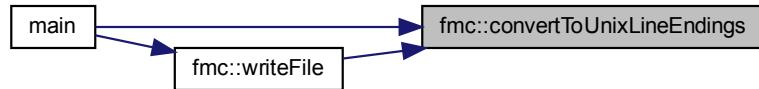
true on success; false otherwise.

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 )
```

Creates a backup of a file.

Parameters

<code>csvFilePath</code>	The path to the file to create a backup of.
<code>backupFilePath</code>	Where to copy the file to.

Returns

true on success; false otherwise.

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 )
```

Routine to delete anything that isn't a Bosch sensor.

Parameters

<i>data</i>	Pointer to the matrix read from the raw CSV file.
-------------	---

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 )
```

Deletes out of bounds values from the raw CSV file read.

Parameters

<i>data</i>	Pointer to the matrix read from the raw CSV file.
-------------	---

Returns

An empty tl::expected on success or an error on error.

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 )
```

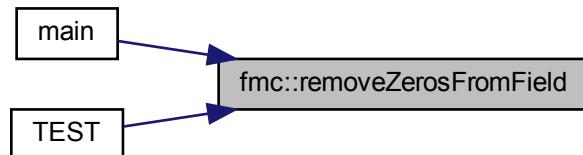
Deletes extraneous zeros from a cell value.

Parameters

<i>field</i>	Pointer to a string of a cell from a CSV file.
--------------	--

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 )
```

Restore a file from a previously created backup file.

Parameters

<i>csvFilePath</i>	The path to the modified file that shall be restored.
<i>backupFilePath</i>	The path to the backup of the original file.

Returns

true on success; otherwise false.

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 )
```

Writes a 'fixed' CSV file.

Parameters

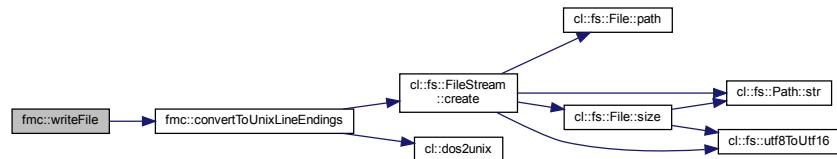
<i>csvPath</i>	Path to which to write the new 'fixed' CSV file.
<i>csvFileExtension</i>	The file extension to use, should probably be ".csv".
<i>columnNames</i>	The column headers.
<i>data</i>	The matrix to write.

Returns

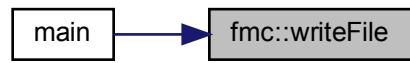
true on success; otherwise false.

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 41 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

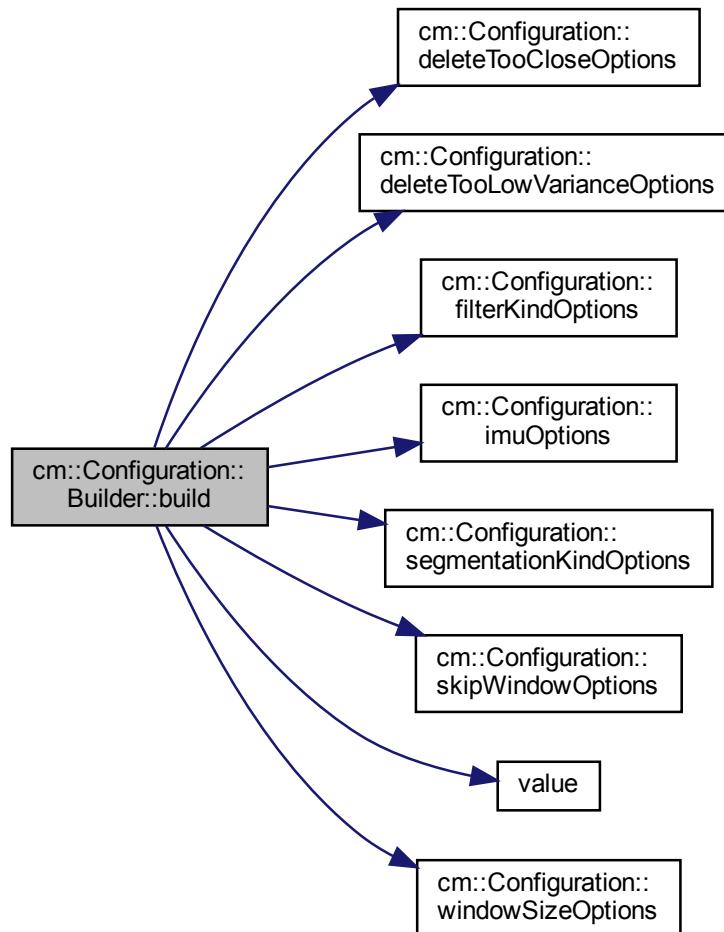
The [Configuration](#) built.

Exceptions

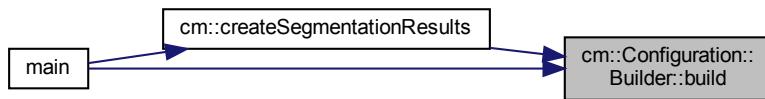
cl::Exception	if one of the properties has not been set or is invalid.
-------------------------------	--

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

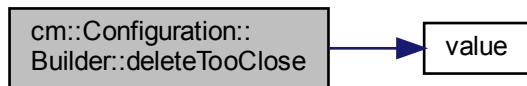
<code>value</code>	The value to use.
--------------------	-------------------

Returns

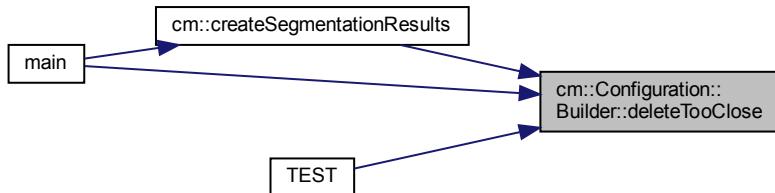
`*this`

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

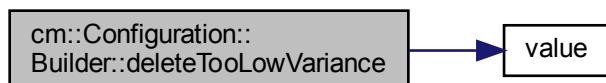
<code>value</code>	The value to use.
--------------------	-------------------

Returns

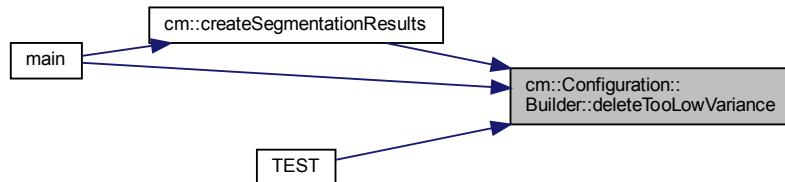
*this

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

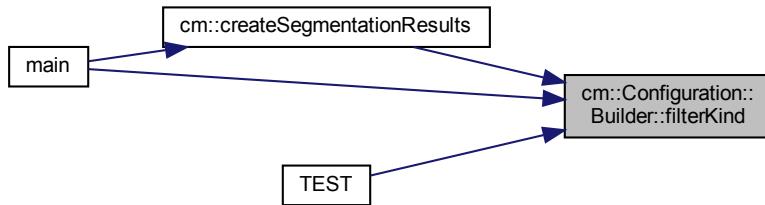
*this

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

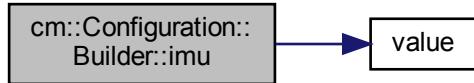
<code>value</code>	The value to use.
--------------------	-------------------

Returns

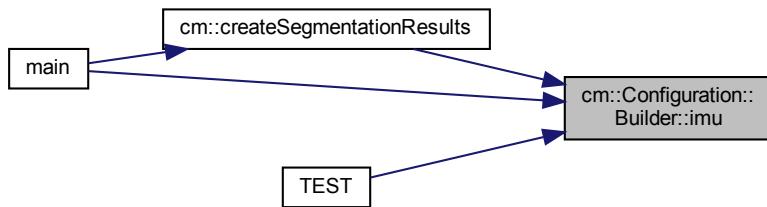
`*this`

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

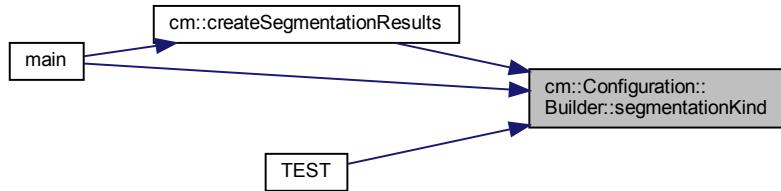
`*this`

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

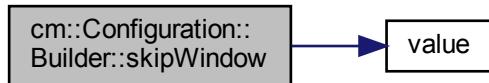
<code>value</code>	The value to use.
--------------------	-------------------

Returns

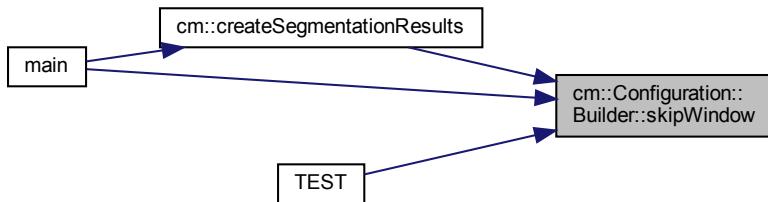
`*this`

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

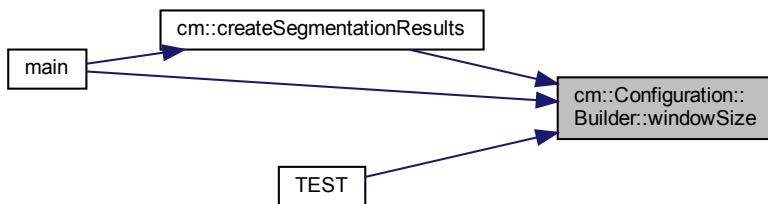
*this

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

Column traits for the columns of the old, non-preprocessed CSV files. Includes the index (0 based) of the column and the C++ data type to be used for the cell contents in that column.

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

6.2.1 Detailed Description

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

Column traits for the columns of the old, non-preprocessed CSV files. Includes the index (0 based) of the column and the C++ data type to be used for the cell contents in that column.

Template Parameters

<i>Col</i>	The Column enumerator.
------------	------------------------

Definition at line 38 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
Less than compares two Configurations.
- 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 33 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 259 of file configuration.cpp.

6.3.3 Member Function Documentation

6.3.3.1 createFilePath()

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

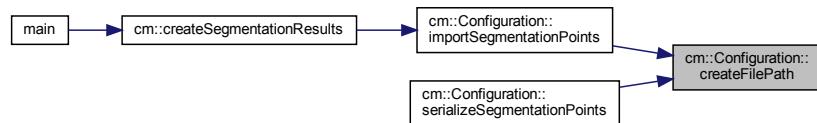
Create a file path for this kind of Configuration.

Returns

The file path for this kind of Configuration.

Definition at line 296 of file configuration.cpp.

Here is the caller graph for this function:



6.3.3.2 deleteTooClose()

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

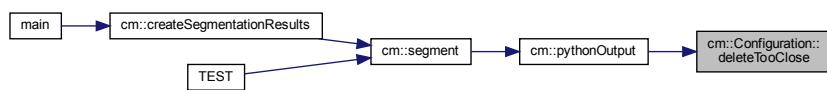
Read accessor for the `deleteTooClose` property.

Returns

The `deleteTooClose` option.

Definition at line 273 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]
```

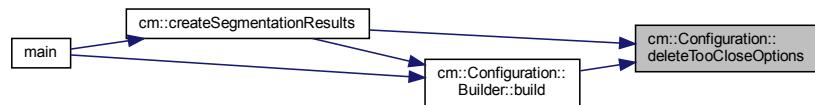
Returns the possible deleteTooClose options.

Returns

The deleteTooClose options.

Definition at line 156 of file configuration.cpp.

Here is the caller graph for this function:



6.3.3.4 deleteTooLowVariance()

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

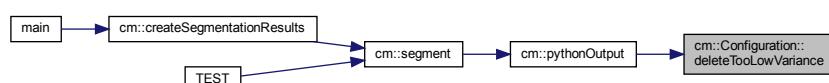
Read accessor for the deleteTooLowVariance property.

Returns

The deleteTooLowVariance option.

Definition at line 275 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]
```

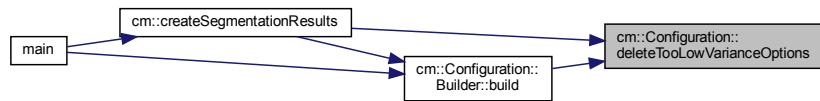
Returns the possible deleteTooLowVariance options.

Returns

The deleteTooLowVariance options.

Definition at line 162 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]
```

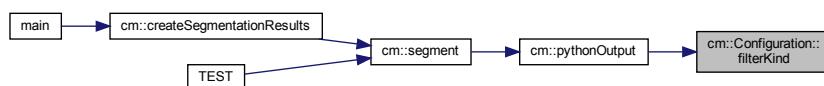
Read accessor for the filterKind property.

Returns

The filterKind option.

Definition at line 289 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]
```

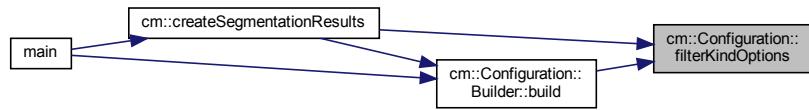
Returns the possible filterKind options.

Returns

The filterKind options.

Definition at line 191 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

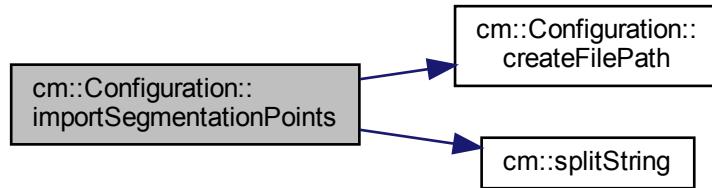
The imported segmentation points.

Exceptions

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

Definition at line 335 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]`

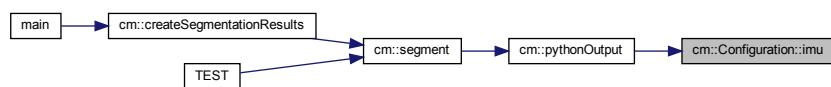
Read accessor for the `imu` property.

Returns

The `imu` option.

Definition at line 280 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]
```

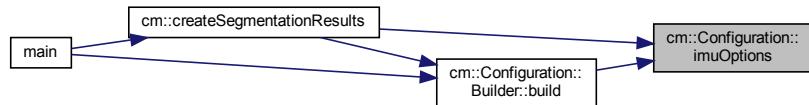
Returns the possible imu options.

Returns

The imu options.

Definition at line 168 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 294 of file configuration.cpp.

6.3.3.12 segmentationKind()

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

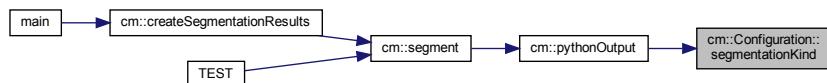
Read accessor for the segmentationKind property.

Returns

The segmentationKind option.

Definition at line 282 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]
```

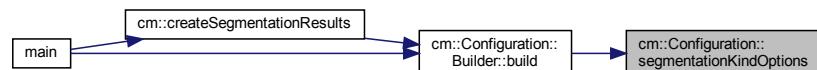
Returns the possible segmentationKind options.

Returns

The segmentationKind options.

Definition at line 175 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

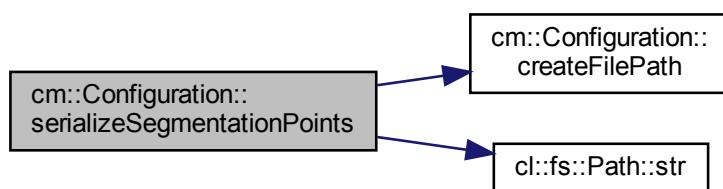
<i>segmentationPointsMap</i>	The map to serialize.
------------------------------	-----------------------

Returns

true on success; false otherwise.

Definition at line 314 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 271 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]
```

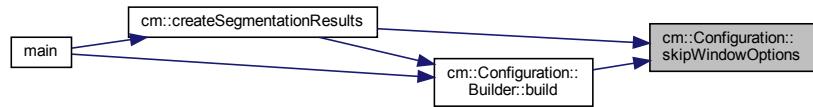
Returns the possible skipWindow options.

Returns

The skipWindow options.

Definition at line 150 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]
```

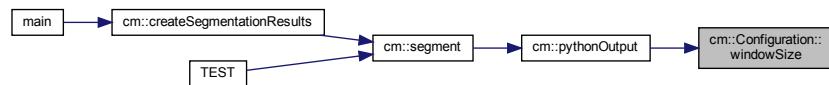
Read accessor for the windowHeight property.

Returns

The windowHeight option.

Definition at line 287 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]
```

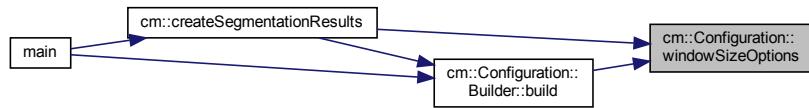
Returns the possible `windowSize` options.

Returns

The `windowSize` options.

Definition at line 182 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 35 of file `configuration.hpp`.

6.3.4.2 `operator<`

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

`Less than` compares two `Configurations`.

Parameters

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

Returns

true if *lhs* is considered less than *rhs*; otherwise false.

Definition at line 218 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 238 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 198 of file configuration.cpp.

6.3.4.5 std::hash< Configuration >

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

Definition at line 36 of file configuration.hpp.

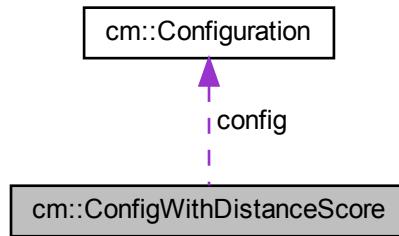
The documentation for this class was generated from the following files:

- 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 18 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 21 of file order_configurations_by_quality.hpp.

6.4.3.2 distScore

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

Definition at line 22 of file order_configurations_by_quality.hpp.

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

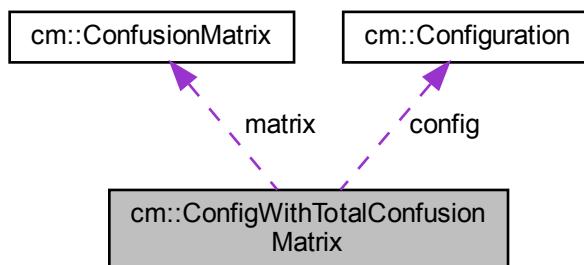
- confusion_matrix/include/order_configurations_by_quality.hpp
- confusion_matrix/src/order_configurations_by_quality.cpp

6.5 cm::ConfigWithTotalConfusionMatrix Struct Reference

A Configuration with a ConfusionMatrix.

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

Collaboration diagram for cm::ConfigWithTotalConfusionMatrix:



Public Member Functions

- `ConfigWithTotalConfusionMatrix ()=default`
Default constructor.
- `ConfigWithTotalConfusionMatrix (Configuration p_config, ConfusionMatrix p_matrix)`
Constructor.

Public Attributes

- `Configuration config`
- `ConfusionMatrix matrix`

Friends

- `std::ostream & operator<< (std::ostream &os, const ConfigWithTotalConfusionMatrix &obj)`
Prints a `ConfigWithTotalConfusionMatrix` to os.

6.5.1 Detailed Description

A `Configuration` with a `ConfusionMatrix`.

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

6.5.2 Constructor & Destructor Documentation

6.5.2.1 ConfigWithTotalConfusionMatrix() [1/2]

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

Default constructor.

6.5.2.2 ConfigWithTotalConfusionMatrix() [2/2]

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

Constructor.

Parameters

<code>p_config</code>	The <code>Configuration</code> to use.
<code>p_matrix</code>	The <code>ConfusionMatrix</code> to use.

Definition at line 95 of file confusion_matrix_best_configs.cpp.

6.5.3 Friends And Related Function Documentation

6.5.3.1 operator<<

```
std::ostream& operator<< (
    std::ostream & os,
    const ConfigWithTotalConfusionMatrix & obj ) [friend]
```

Prints a [ConfigWithTotalConfusionMatrix](#) to os.

Parameters

<i>os</i>	The ostream to print to.
<i>obj</i>	The ConfigWithTotalConfusionMatrix to print.

Returns

os

Definition at line 70 of file confusion_matrix_best_configs.cpp.

6.5.4 Member Data Documentation

6.5.4.1 config

[Configuration](#) cm::ConfigWithTotalConfusionMatrix::config

The [Configuration](#)

Definition at line 41 of file confusion_matrix_best_configs.hpp.

6.5.4.2 matrix

[ConfusionMatrix](#) cm::ConfigWithTotalConfusionMatrix::matrix

The associated [ConfusionMatrix](#)

Definition at line 42 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

Type to represent a confusion matrix.

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

Public Types

- using `this_type = ConfusionMatrix`

Public Member Functions

- `ConfusionMatrix ()`
Default constructs a `ConfusionMatrix` initializing all data members with 0.
- `std::uint64_t truePositives () const noexcept`
Read accessor for the true positive count.
- `std::uint64_t trueNegatives () const noexcept`
Read accessor for the true negative count.
- `std::uint64_t falsePositives () const noexcept`
Read accessor for the false positive count.
- `std::uint64_t falseNegatives () const noexcept`
Read accessor for the false negative count.
- `std::uint64_t totalCount () const noexcept`
Read accessor for the total count.
- `this_type & incrementTruePositives () noexcept`
Increments the true positive count and the total count.
- `this_type & incrementTrueNegatives () noexcept`
Increments the true negative count and the total count.
- `this_type & incrementFalsePositives () noexcept`
Increments the false positive count and the total count.
- `this_type & incrementFalseNegatives () noexcept`
Increments the false negative count and the total count.
- `this_type & operator+= (const ConfusionMatrix &other) noexcept`
*Accumulates `other` into `*this`.*

6.6.1 Detailed Description

Type to represent a confusion matrix.

Definition at line 9 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 11 of file confusion_matrix.hpp.

6.6.3 Constructor & Destructor Documentation

6.6.3.1 ConfusionMatrix()

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

Default constructs a [ConfusionMatrix](#) initializing all data members with 0.

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]
```

Read accessor for the false negative count.

Returns

The false negative count.

Definition at line 28 of file confusion_matrix.cpp.

6.6.4.2 falsePositives()

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

Read accessor for the false positive count.

Returns

The false positive count.

Definition at line 23 of file confusion_matrix.cpp.

6.6.4.3 incrementFalseNegatives()

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

Increments the false negative count and the total count.

Returns

*this

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]
```

Increments the false positive count and the total count.

Returns

*this

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]
```

Increments the true negative count and the total count.

Returns

*this

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]
```

Increments the true positive count and the total count.

Returns

*this

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]
```

Accumulates other into *this.

Parameters

<code>other</code>	The other ConfusionMatrix to add to this ConfusionMatrix .
--------------------	--

Returns

`*this`

Definition at line 66 of file `confusion_matrix.cpp`.

6.6.4.8 `totalCount()`

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

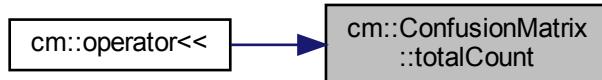
Read accessor for the total count.

Returns

The total count.

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]
```

Read accessor for the true negative count.

Returns

The true negative count.

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

6.6.4.10 truePositives()

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

Read accessor for the true positive count.

Returns

The true positive count.

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 57 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

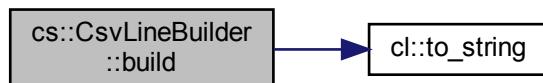
The resulting vector of strings.

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

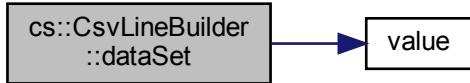
<code>value</code>	The value to use.
--------------------	-------------------

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 (\n    bool value )
```

Write accessor for the delete too close property.

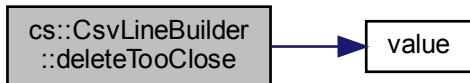
Parameters

<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.

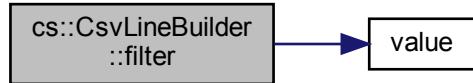
Parameters

<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

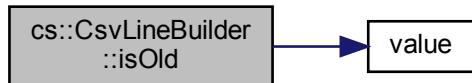
<code>value</code>	The value to use.
--------------------	-------------------

Returns

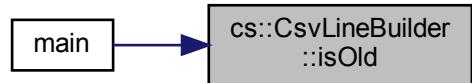
`*this`

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

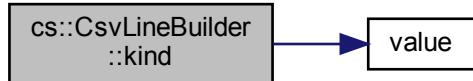
<code>value</code>	The value to use.
--------------------	-------------------

Returns

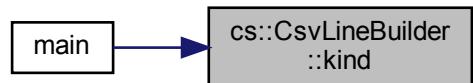
`*this`

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.

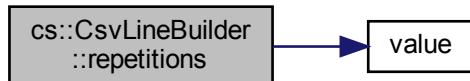
Parameters

<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.
--------------------	-------------------

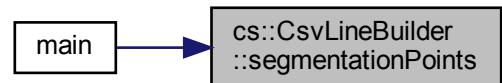
Returns`*this`

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.

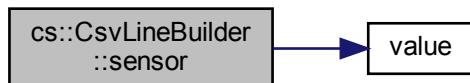
Parameters

<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.

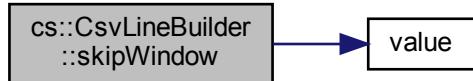
Parameters

<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

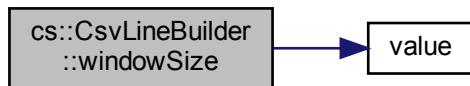
<code>value</code>	The value to use.
--------------------	-------------------

Returns

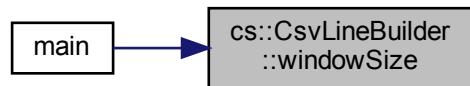
*this

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

Maps a Channel to its associated accessor member function pointer of the DataSet type.

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

6.9.1 Detailed Description

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

Maps a Channel to its associated accessor member function pointer of the DataSet type.

Template Parameters

<i>Chan</i>	The Channel to map to its data set accessor.
-------------	--

Definition at line 53 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

Type to represent a data point in a data set.

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

Public Member Functions

- `DataPoint (std::string fileName, long double time, Sensor sensor, Channel channel, long double value) noexcept`
Creates a DataPoint.
- `const std::string & fileName () const noexcept`
Read accessor for the file name.
- `long double time () const noexcept`
- `Sensor sensor () const noexcept`
Read accessor for the sensor.
- `Channel channel () const noexcept`
Read accessor for the channel.
- `long double value () const noexcept`
Read accessor for the DataPoint value.

Friends

- `std::ostream & operator<< (std::ostream &os, const DataPoint &dataPoint)`
Prints a DataPoint to os.

6.11.1 Detailed Description

Type to represent a data point in a data set.

Definition at line 13 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]
```

Creates a DataPoint.

Parameters

<code>fileName</code>	The file name of the CSV file that contains this DataPoint.
<code>time</code>	The time.
<code>sensor</code>	The sensor.
<code>channel</code>	The channel.
<code>value</code>	The value.

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]
```

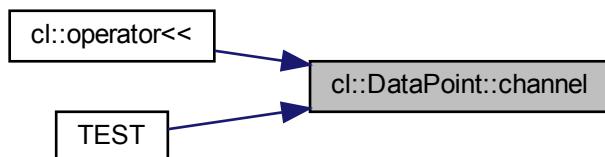
Read accessor for the channel.

Returns

The channel.

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]
```

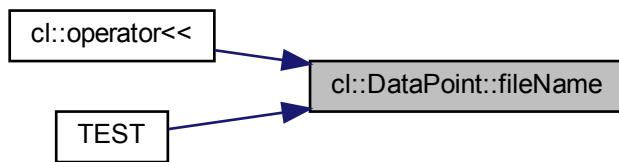
Read accessor for the file name.

Returns

The file name.

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]
```

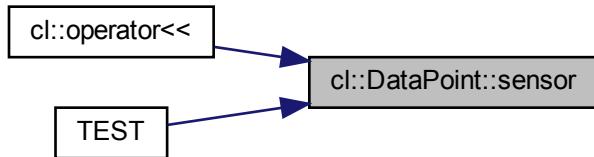
Read accessor for the sensor.

Returns

The sensor.

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]
```

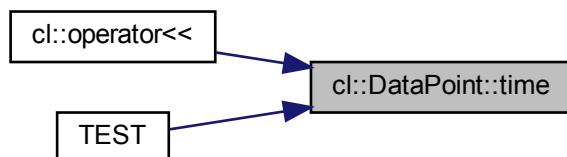
\brief Read accessor for the time.

Returns

The time.

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]
```

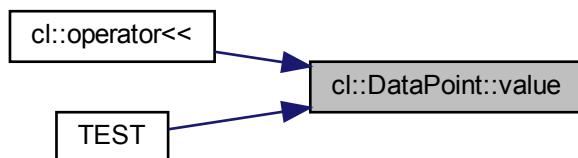
Read accessor for the [DataPoint](#) value.

Returns

The value.

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]
```

Prints a [DataPoint](#) to os.

Parameters

<i>os</i>	The ostream to print to.
<i>dataPoint</i>	The DataPoint to print.

Returns

os

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

Type to represent a data set.

```
#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`
The row count.
- `const std::string & fileName () const noexcept`
The file name.
- `column_type< Column::Time > time (size_type index) const`
The time value at row index.
- `column_type< Column::HardwareTimestamp > hardwareTimestamp (size_type index) const`
The hardware timestamp value at row index.
- `column_type< Column::ExtractId > extractId (size_type index) const`
The extract ID value at row index.
- `column_type< Column::Trigger > trigger (size_type index) const`
The trigger value at row index.
- `column_type< Column::AccelerometerX > accelerometerX (size_type index) const`
The accelerometer X value at row index.
- `column_type< Column::AccelerometerY > accelerometerY (size_type index) const`
The accelerometer Y value at row index.
- `column_type< Column::AccelerometerZ > accelerometerZ (size_type index) const`
The accelerometer Z value at row index.
- `column_type< Column::GyroscopeX > gyroscopeX (size_type index) const`
The gyroscope X value at row index.
- `column_type< Column::GyroscopeY > gyroscopeY (size_type index) const`
The gyroscope Y value at row index.
- `column_type< Column::GyroscopeZ > gyroscopeZ (size_type index) const`
The gyroscope Z value at row index.
- `long double accelerometerAverage (Sensor sensor) const`
Average accelerometer reading for the given sensor.
- `long double gyroscopeAverage (Sensor sensor) const`
Average gyroscope reading for the given sensor.
- `long double accelerometerMaximum (Sensor sensor) const`
The maximum accelerometer reading for the given sensor.
- `long double gyroscopeMaximum (Sensor sensor) const`
The maximum gyroscope reading for the given sensor.

Static Public Member Functions

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

6.12.1 Detailed Description

Type to represent a data set.

Warning

Note that this refers to the old 'fixed' data set CSV files created using the fix_csv C++ application.

Definition at line 19 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 22 of file data_set.hpp.

6.12.2.2 size_type

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

Definition at line 21 of file data_set.hpp.

6.12.3 Member Function Documentation

6.12.3.1 accelerometerAverage()

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

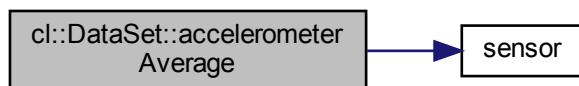
Average accelerometer reading for the given sensor.

Parameters

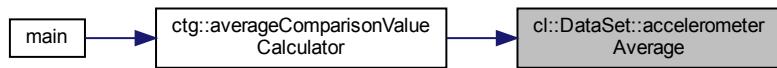
<i>sensor</i>	The sensor.
---------------	-------------

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
```

The maximum accelerometer reading for the given `sensor`.

Parameters

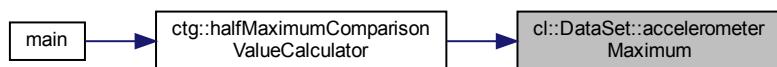
<code>sensor</code>	The sensor.
---------------------	-------------

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
```

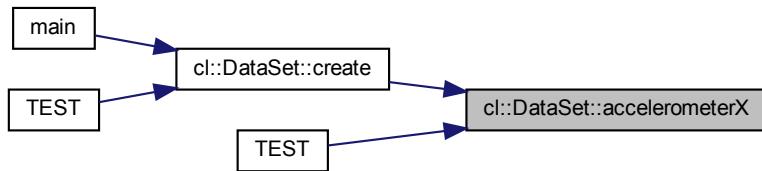
The accelerometer X value at row `index`.

Parameters

<code>index</code>	The row of which to get the accelerometer X value.
--------------------	--

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
```

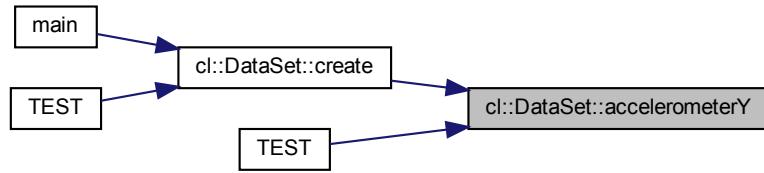
The accelerometer Y value at row `index`.

Parameters

<code>index</code>	The row of which to get the accelerometer Y value.
--------------------	--

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
```

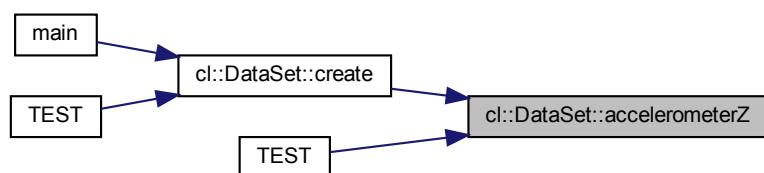
The accelerometer Z value at row `index`.

Parameters

<code>index</code>	The row of which to get the accelerometer Z value.
--------------------	--

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]
```

Creates a [DataSet](#).

Parameters

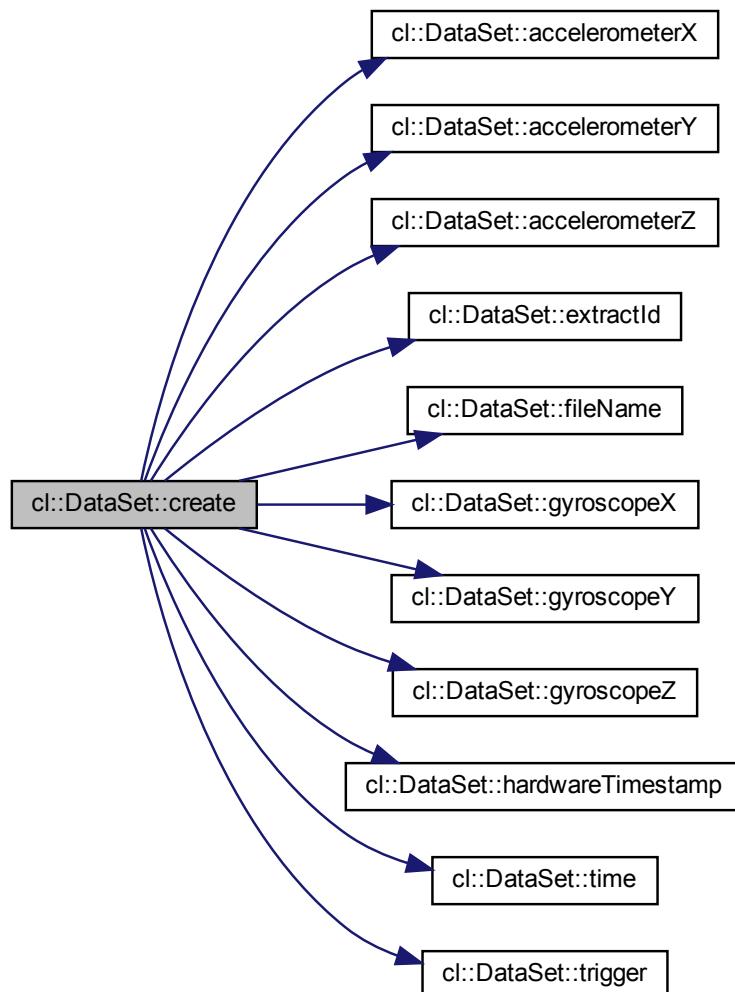
<code>fileName</code>	The file name.
<code>matrix</code>	The matrix as read with cl::readCsvFile using the Fixed setting.

Returns

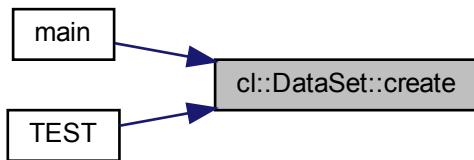
The data set.

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
```

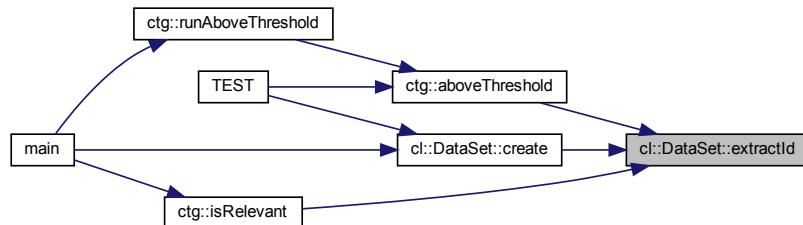
The extract ID value at row `index`.

Parameters

<code>index</code>	The row of which to get the extract ID value.
--------------------	---

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

Here is the caller graph for this function:



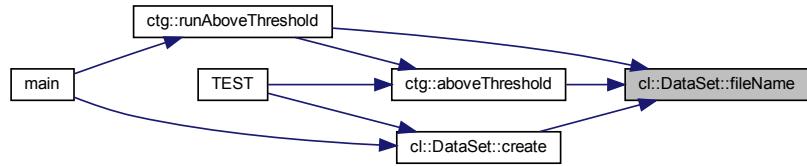
6.12.3.8 fileName()

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

The file name.

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
```

Average gyroscope reading for the given `sensor`.

Parameters

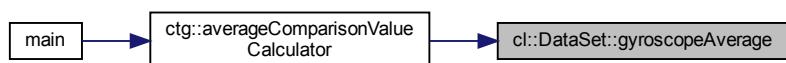
<code>sensor</code>	The sensor.
---------------------	-------------

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
```

The maximum gyroscope reading for the given sensor.

Parameters

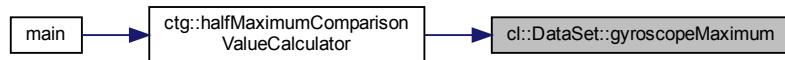
<i>sensor</i>	The sensor.
---------------	-------------

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
```

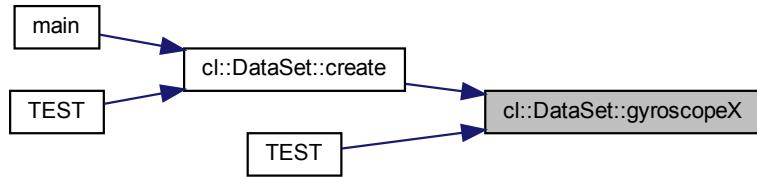
The gyroscope X value at row *index*.

Parameters

<i>index</i>	The row of which to get the gyroscope X value.
--------------	--

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
```

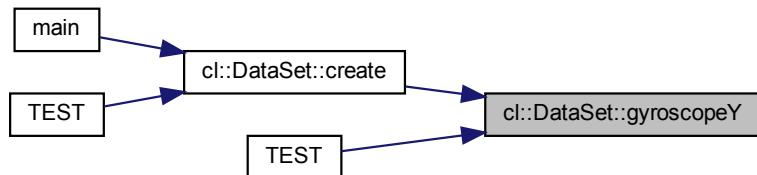
The gyroscope Y value at row `index`.

Parameters

<code>index</code>	The row of which to get the gyroscope Y value.
--------------------	--

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
```

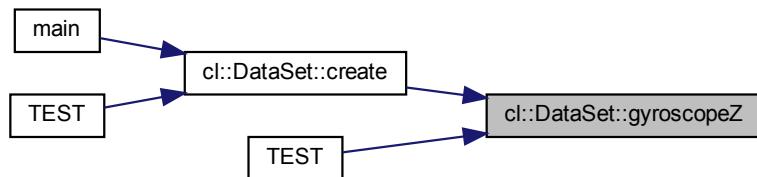
The gyroscope Z value at row `index`.

Parameters

<i>index</i>	The row of which to get the gyroscope Z value.
--------------	--

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
```

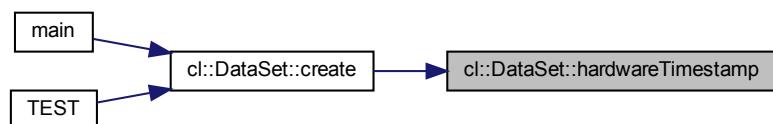
The hardware timestamp value at row `index`.

Parameters

<i>index</i>	The row of which to get the hardware timestamp value.
--------------	---

Definition at line 178 of file data_set.cpp.

Here is the caller graph for this function:



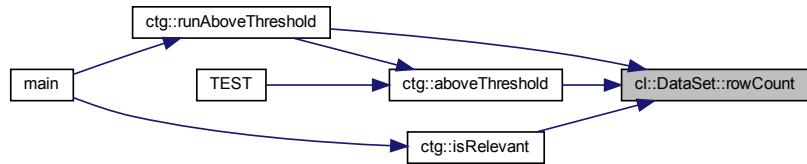
6.12.3.15 rowCount()

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

The row count.

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
```

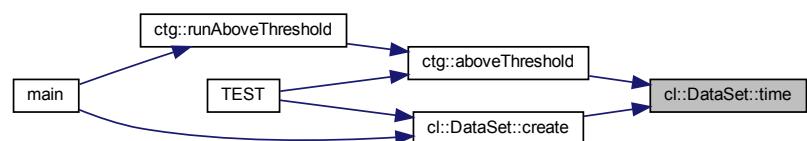
The time value at row index.

Parameters

<i>index</i>	The row of which to get the time value.
--------------	---

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
```

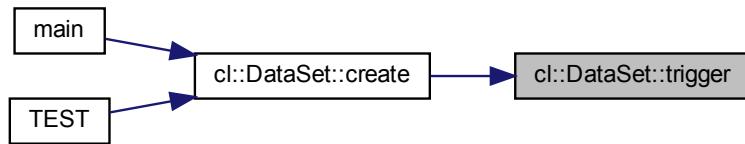
The trigger value at row index.

Parameters

<i>index</i>	The row of which to get the trigger value.
--------------	--

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

Type used to represent different kinds of errors.

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

Public Types

- enum [Kind](#) { [CL_ERROR_KIND](#) }

Unscoped enum type for the kinds of errors.

Public Member Functions

- `Error (Kind kind, std::string file, std::string function, std::size_t line, std::string message)`
Constructs an `Error` object.
- `Kind kind () const noexcept`
Read accessor for the kind enumerator.
- `const std::string & file () const noexcept`
Read accessor for the file string.
- `const std::string & function () const noexcept`
Read accessor for the function string.
- `std::size_t line () const noexcept`
Read accessor for the line number of this `Error`.
- `const std::string & message () const noexcept`
Read accesor for the error message.
- `void raise () const`
Throws this `Error` as a `cl::Exception`.
- `std::string to_string () const`
Converts this `Error` to a string.

Friends

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

6.13.1 Detailed Description

Type used to represent different kinds of errors.

Definition at line 34 of file error.hpp.

6.13.2 Member Enumeration Documentation

6.13.2.1 Kind

```
enum cl::Error::Kind
```

Unscoped enum type for the kinds of errors.

Enumerator

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

Definition at line 40 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 )
```

Constructs an [Error](#) object.

Parameters

<i>kind</i>	The enumerator corresponding to the kind of error.
<i>file</i>	The file in which the error occurred.
<i>function</i>	The function in which the error occurred.
<i>line</i>	The line in which the error occurred.
<i>message</i>	The error message of this Error .

Definition at line 46 of file error.cpp.

6.13.4 Member Function Documentation

6.13.4.1 file()

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

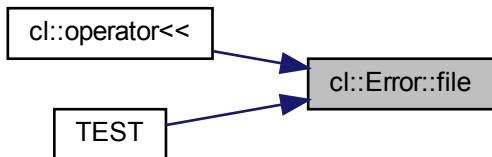
Read accessor for the file string.

Returns

The file in which this [Error](#) occurred.

Definition at line 62 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]
```

Read accessor for the function string.

Returns

The function in which this [Error](#) occurred.

Definition at line 64 of file error.cpp.

Here is the caller graph for this function:



6.13.4.3 kind()

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

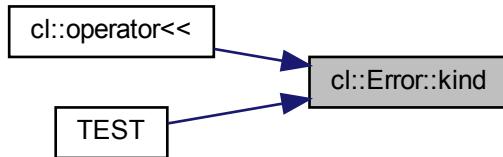
Read accessor for the kind enumerator.

Returns

The kind of this [Error](#).

Definition at line 60 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]
```

Read accessor for the line number of this [Error](#).

Returns

The line in which this [Error](#) occurred.

Definition at line 66 of file error.cpp.

6.13.4.5 message()

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

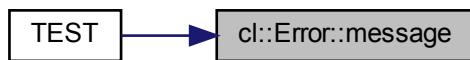
Read accessor for the error message.

Returns

The error message.

Definition at line 68 of file error.cpp.

Here is the caller graph for this function:



6.13.4.6 raise()

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

Throws this [Error](#) as a [cl::Exception](#).

Definition at line 70 of file error.cpp.

6.13.4.7 `to_string()`

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

Converts this [Error](#) to a string.

Returns

A textual representation of this [Error](#).

Definition at line 79 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]
```

Prints `error` to `os`.

Parameters

<code>os</code>	The ostream to print to.
<code>error</code>	The Error to print.

Returns

`os`.

Definition at line 35 of file error.cpp.

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

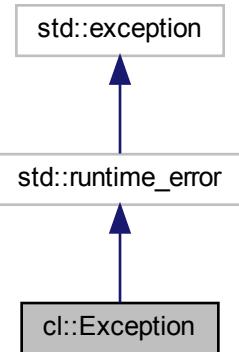
- csv_lib/include/cl/error.hpp
- csv_lib/src/cl/error.cpp

6.14 cl::Exception Class Reference

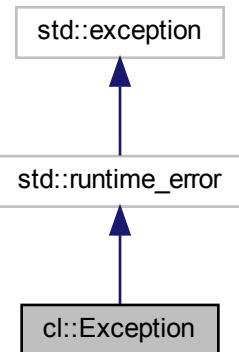
The exception type for this code base.

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

- Read accessor for the file.*
- const std::string & [function](#) () const noexcept
Read accesor for the function.
 - std::size_t [line](#) () const noexcept
Read accesor for the line.

6.14.1 Detailed Description

The exception type for this code base.

Definition at line 21 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 23 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 )
```

Constructs an [Exception](#).

Parameters

<i>file</i>	The file in which the error occurred.
<i>function</i>	The function in which the error occurred.
<i>line</i>	The line in which the error occurred.
<i>what_arg</i>	The error message.

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 )
```

Constructs an [Exception](#).

Parameters

<i>file</i>	The file in which the error occurred.
<i>function</i>	The function in which the error occurred.
<i>line</i>	The line in which the error occurred.
<i>what_arg</i>	The error message.

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]
```

Read accessor for the file.

Returns

The file where the error occurred.

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]
```

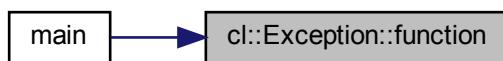
Read accessor for the function.

Returns

The function in which the error occurred.

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]
```

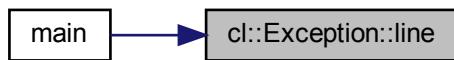
Read accessor for the line.

Returns

The line in which the error occurred.

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

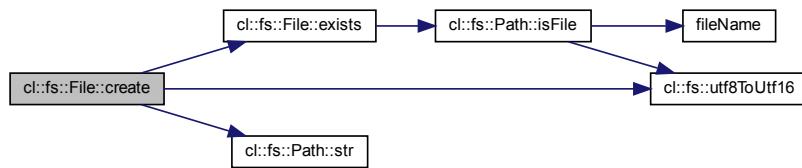
true if the file was successfully created; otherwise false.

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]
```

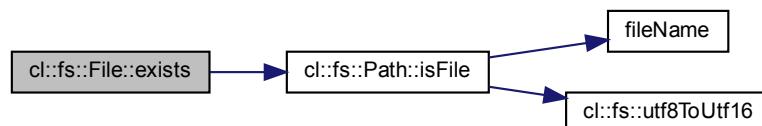
Determines if this file exists.

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

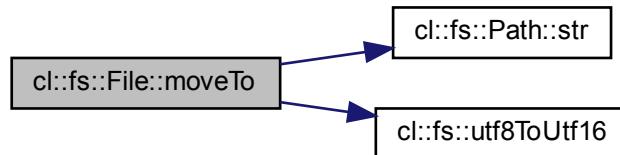
<i>newPath</i>	The path to move this file to.
----------------	--------------------------------

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]
```

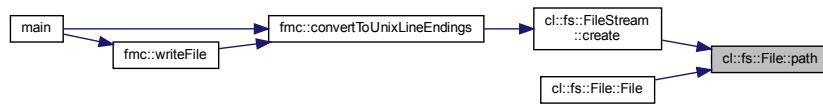
Read accessor for the path of this file.

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]
```

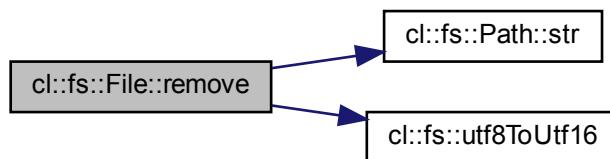
Deletes this file.

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

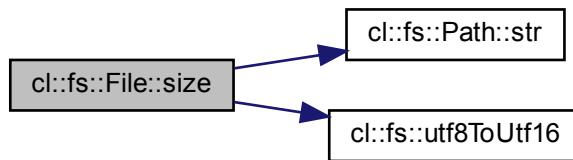
The size of this file in bytes or -1 on error.

Warning

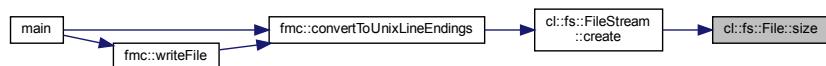
Returns -1 on error.

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

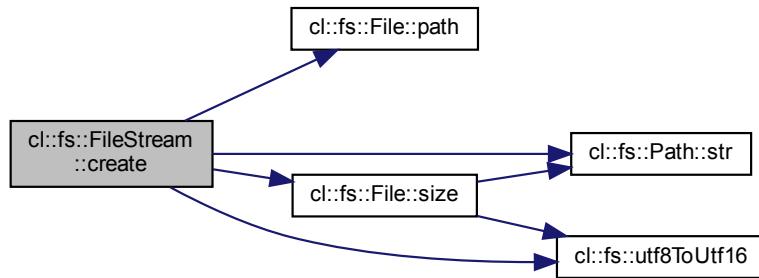
<i>file</i>	The file to open.
<i>openMode</i>	The open mode to use.

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_NONCOPYABLE()`

```
cl::fs::FileStream::PL_NONCOPYABLE (
    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 92 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 93 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 301 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 302 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

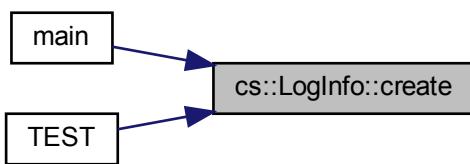
<i>logFilePath</i>	The log file path to create a LogInfo from.
--------------------	---

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\(0xFFFFFFFFFFFFFFFFF\)](#)
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
```

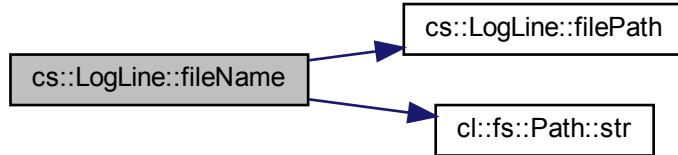
Creates the short file name for the file in the log line.

Returns

The resulting short file name or an error.

Definition at line 131 of file log_line.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



6.20.2.2 filePath()

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

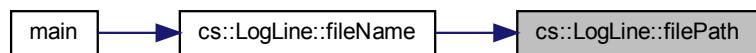
Read accessor for the file path.

Returns

The file path of the file in the log line.

Definition at line 129 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

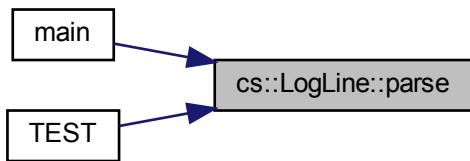
<i>line</i>	The line read.
-------------	----------------

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 124 of file `log_line.cpp`.

6.20.2.5 sensor()

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

Read accessor for the sensor.

Returns

The sensor.

Note

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

Definition at line 169 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

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

Exceptions

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

Definition at line 417 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 475 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 generated by OpenOCD that are based on hardware timestamps.

Returns

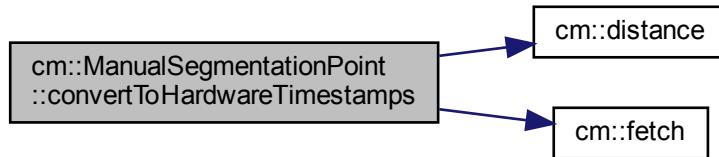
The resulting hardware timestamp based manual segementation points.

Exceptions

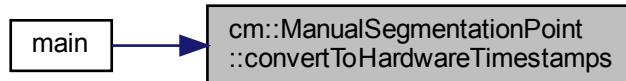
cl::Exception on error.

Definition at line 361 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 470 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]
```

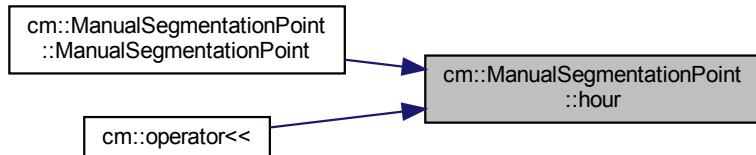
Read accessor for the hour property.

Returns

The hour.

Definition at line 458 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 460 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

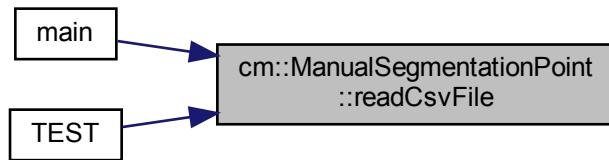
A map that maps the `DataSetIdentifier` enumerators to vectors of the corresponding manual segmentation points extracted from the CSV file.

Exceptions

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

Definition at line 211 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 465 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 189 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 196 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 181 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]
```

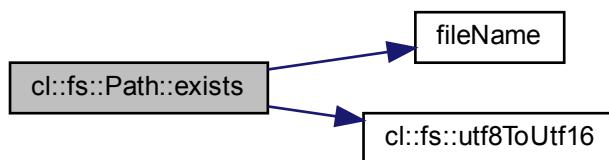
Checks if the path exists.

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]
```

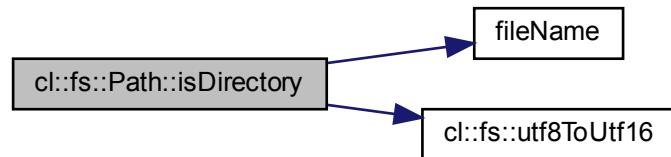
Checks if the path is a directory.

Returns

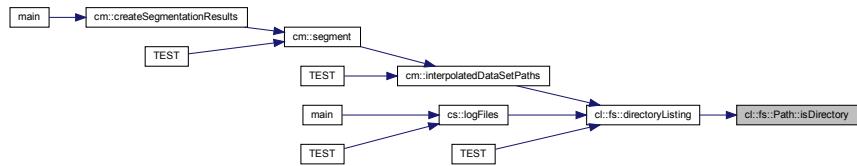
true if the path is a directory; otherwise false.

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]
```

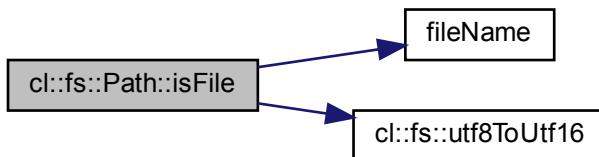
Checks if the path is a file.

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]
```

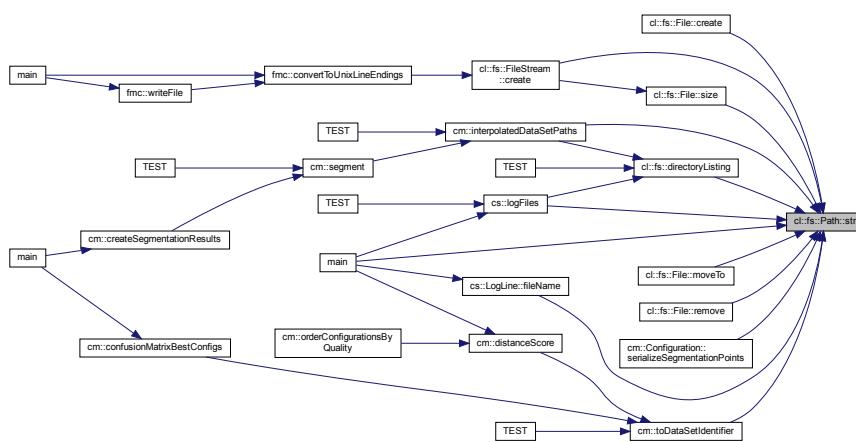
Read accessor for the underlying string.

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

Type representing an operating system process.

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

Public Types

- using `this_type = Process`

Public Member Functions

- `PL_NONCOPYABLE (Process)`
- `Process (this_type &&other) noexcept`
Move constructor.
- `this_type & operator= (this_type &&other) noexcept`
Move assignment operator.
- `~Process ()`
Destructor.
- `std::FILE * file () noexcept`
Returns the underlying std::FILE.*
- `const std::FILE * file () const noexcept`
Returns the underlying std::FILE.*

Static Public Member Functions

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

6.23.1 Detailed Description

Type representing an operating system process.

An operating system process. This type effectively wraps `popen`. This is used to invoke the Python segmentor and read its `stdout` output using C-style I/O through a `std::FILE*`.

Definition at line 19 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 23 of file `process.hpp`.

6.23.3 Constructor & Destructor Documentation

6.23.3.1 Process()

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

Move constructor.

Parameters

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

Warning

other is left in a null state.

Definition at line 58 of file process.cpp.

6.23.3.2 ~Process()

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

Destructor.

Definition at line 71 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]
```

Creates a process.

Parameters

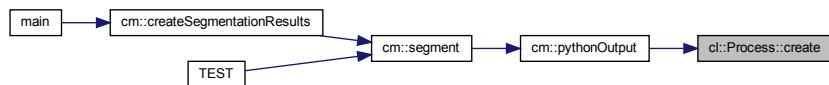
<i>command</i>	The 'command' to invoke, e.g., the application or shell command to run.
<i>mode</i>	The mode. Must be either "r" for reading or "w" for writing.

Returns

The process created.

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]
```

Returns the underlying std::FILE*.

6.23.4.3 file() [2/2]

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

Returns the underlying std::FILE*.

Definition at line 81 of file process.cpp.

6.23.4.4 operator=()

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

Move assignment operator.

Parameters

<i>other</i>	The other object to move assign from.
--------------	---------------------------------------

Returns

*this.

Warning

This process' std::FILE* handle will be handed over to `other`, leaving `other` to destroy it.

Definition at line 63 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/mode.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/mode.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 mode_test.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

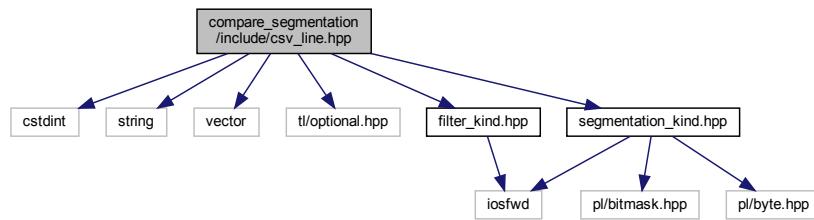
7.10.1.1 include()

```
include (  
    GoogleTest )
```

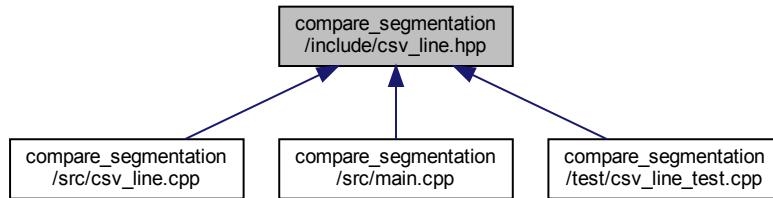
Definition at line 1 of file CMakeLists.txt.

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](#)

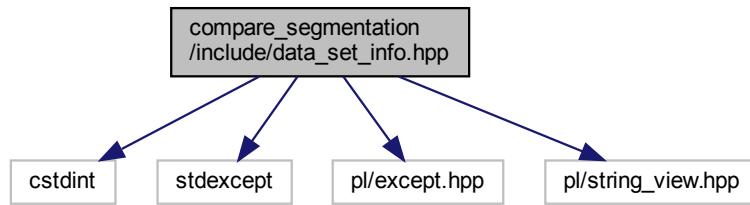
Builder for a CSV line.

Namespaces

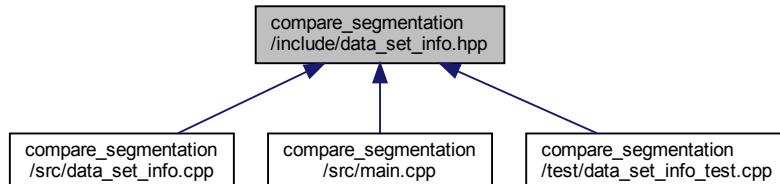
- [CS](#)

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, "Lucas_liegestuetzen1", 24)
 - cs::CS_SPECIALIZE_DATA_SET_INFO (Lucas2, "Lucas_liegestuetzen2", 19)
 - cs::CS_SPECIALIZE_DATA_SET_INFO (Lucas3, "Lucas_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(
```

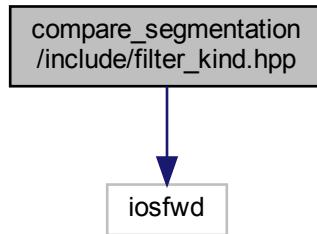
Value:

```
    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);  
    };
```

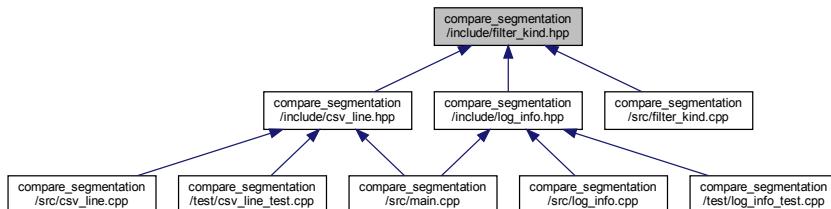
Definition at line 23 of file data_set_info.hpp.

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

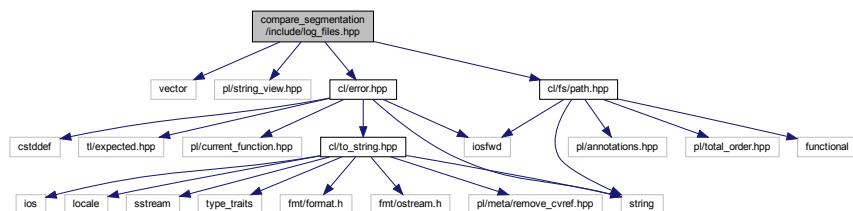
- enum `cs::FilterKind` { `cs::FilterKind::Butterworth`, `cs::FilterKind::MovingAverage` }
- Type for the different kinds of filters.*

Functions

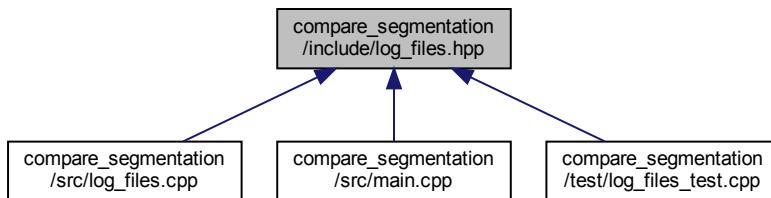
- `std::ostream & cs::operator<<` (`std::ostream &os, FilterKind filterKind`)
- Prints a FilterKind to an ostream.*

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`

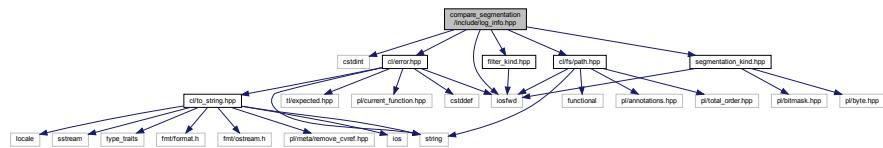
Functions

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

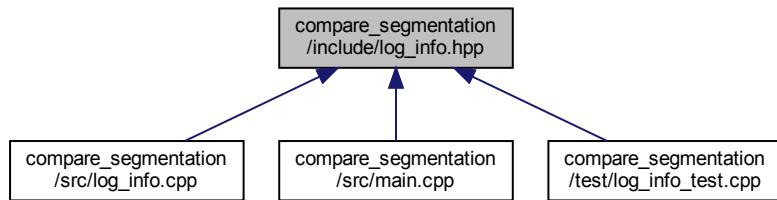
7.15 compare_segmentation/include/log_info.hpp File Reference

```
#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

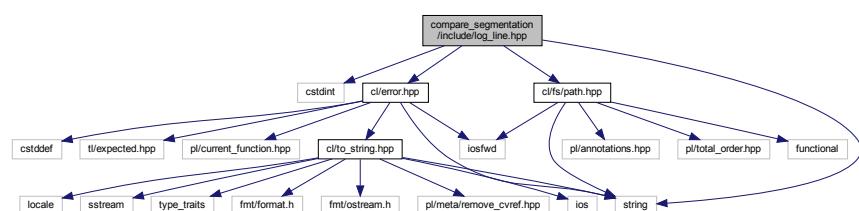
- class [cs::LogInfo](#)
Information about a log file.

Namespaces

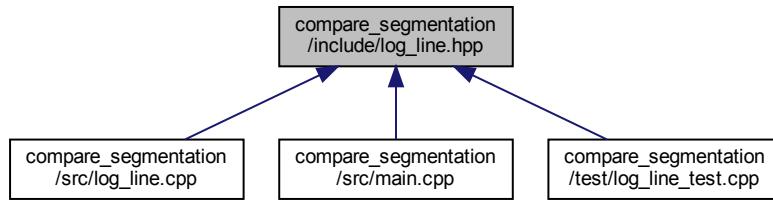
- [cs](#)

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](#)

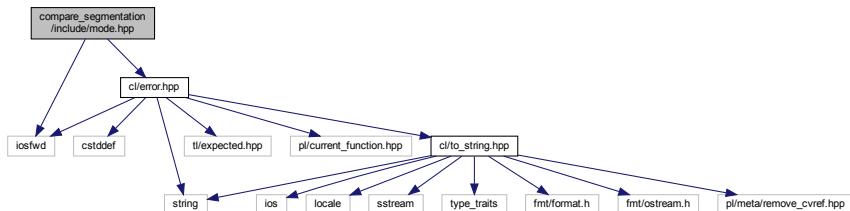
A line out of a log file.

Namespaces

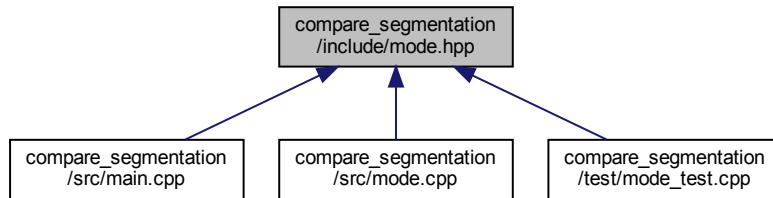
- [cs](#)

7.17 compare_segmentation/include/mode.hpp File Reference

```
#include <iostream>
#include <cl/error.hpp>
Include dependency graph for mode.hpp:
```



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



Namespaces

- `cs`

Macros

- `#define CS_MODE`
- `#define CS_MODE_X(enm) enm,`

Enumerations

- enum `cs::Mode { cs::Mode::CS_MODE_X, cs::Mode::CS_MODE }`
Enumerator type for the different modes of the compare_segmentation application.

Functions

- `std::ostream & cs::operator<< (std::ostream &os, Mode mode)`
Prints a Mode to an ostream.
- `cl::Expected< Mode > cs::parseMode (const char *szCmdArg)`
Parses a null-terminated byte character string as a Mode.

7.17.1 Macro Definition Documentation

7.17.1.1 CS_MODE

```
#define CS_MODE
```

Value:

```
CS_MODE_X(AllDataSets) \
CS_MODE_X(AllPushUps) \
CS_MODE_X(PushUps250Hz) \
CS_MODE_X(PushUps200Hz) \
CS_MODE_X(Squats)
```

Definition at line 8 of file mode.hpp.

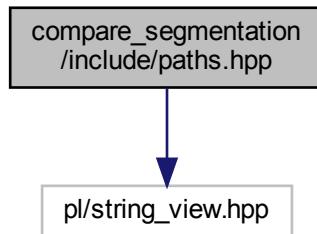
7.17.1.2 CS_MODE_X

```
#define CS_MODE_X( \
    enm ) enm,
```

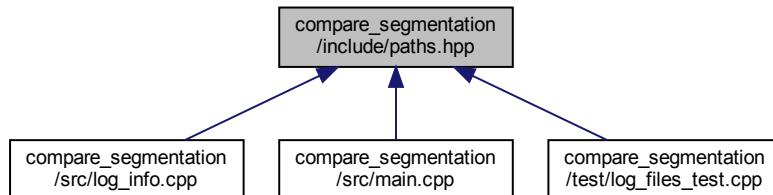
Definition at line 20 of file mode.hpp.

7.18 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

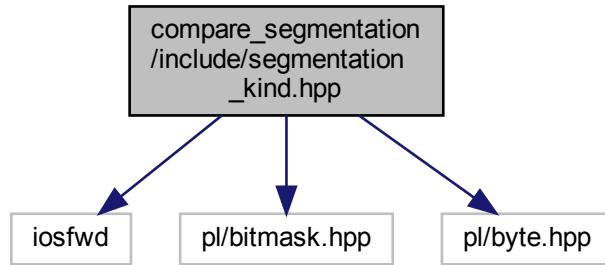
- `cs`

Variables

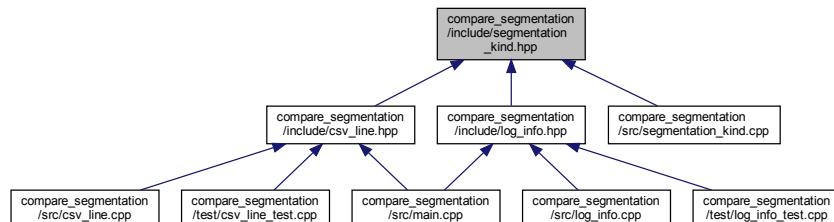
- `constexpr pl::string_view cs::logPath {"segmentation_comparison/logs"}`
Relative path to the directory containing the preprocessed log files.
- `constexpr pl::string_view cs::oldLogPath {"segmentation_comparison/logs/old"}`
Relative path to the directory containing the old log files.

7.19 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 (bitflag type)

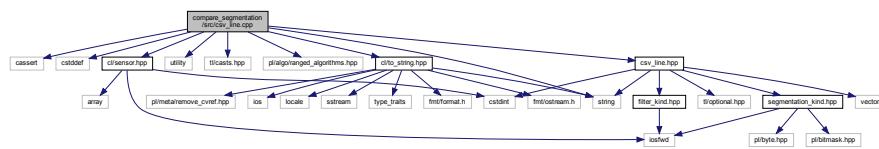
Functions

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

7.20 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:
```



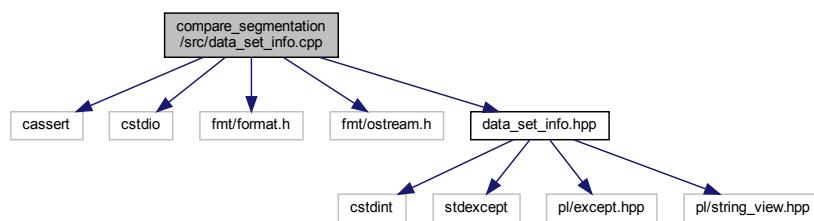
Namespaces

- [CS](#)

7.21 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

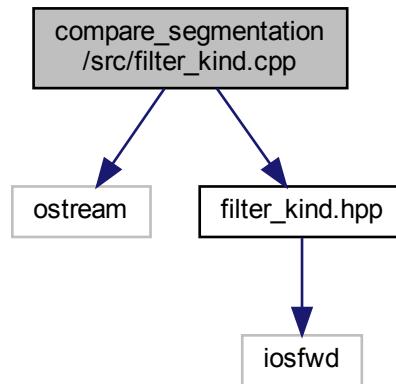
- [CS](#)

Functions

- std::uint64_t [cs::repetitionCount](#) (pl::string_view dataSet)
Fetches the repetition count for a given data set identified by its string.

7.22 compare_segmentation/src/filter_kind.cpp File Reference

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



Namespaces

- [cs](#)

Functions

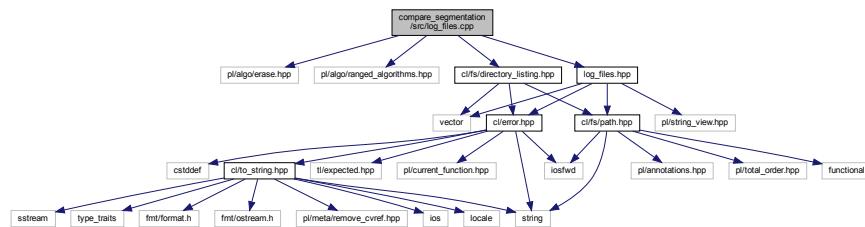
- std::ostream & [cs::operator<<](#) (std::ostream &os, FilterKind filterKind)
Prints a FilterKind to an ostream.

7.23 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

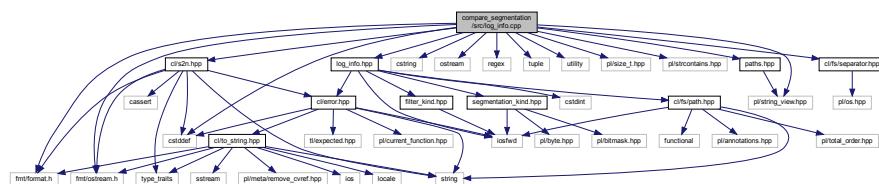
- `cs`

Functions

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

7.24 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/separatator.hpp"
#include "cl/s2n.hpp"
#include "log_info.hpp"
#include "paths.hpp"
Include dependency graph for log_info.cpp:
```



Namespaces

- CS

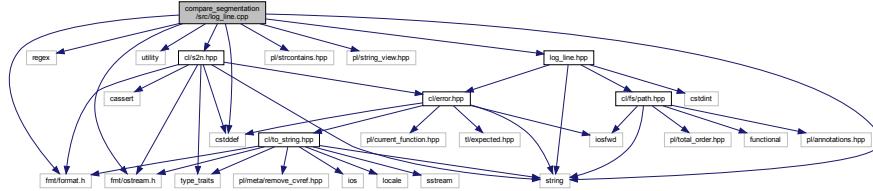
Functions

- 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)

7.25 compare_segmentation/src/log_line.cpp File Reference

```
#include <cstddef>
#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:



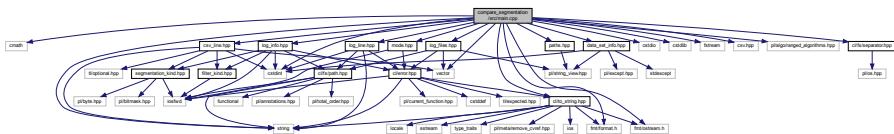
Namespaces

- CS

7.26 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 "cl/fs/separators.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 "mode.hpp"
#include "paths.hpp"
Include dependency graph for main.cpp:
```



Functions

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

The entry point.

7.26.1 Function Documentation

7.26.1.1 main()

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

The entry point.

Parameters

<code>argc</code>	The count of command line arguments.
<code>argv</code>	The command line arguments.

Returns

EXIT_SUCCESS on success or EXIT_FAILURE on failure.

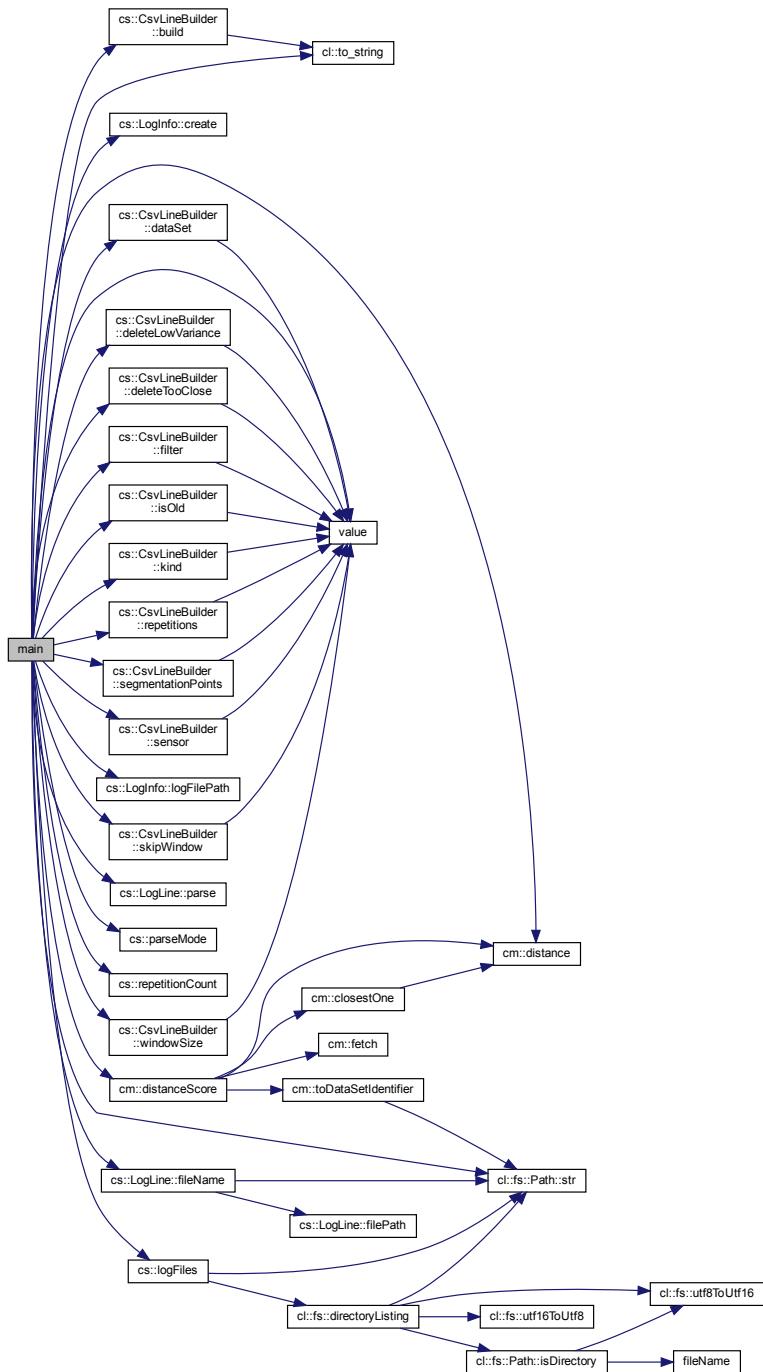
Note

The command line arguments will include this application as the 0th argument.

This compare_segmentation_app will compare the different configurations of the Python segmentation algorithm by comparing the amount of segmentation points found. The amounts of segmentation points found are compared with the expected amount. Note that the placement of the segmentation points found is not considered at all, merely the count of them is considered. See <https://git.csti.haw-hamburg.de/mogasens/dataanalyzer> for Andre's MATLAB application that preprocesses the raw CSV files.

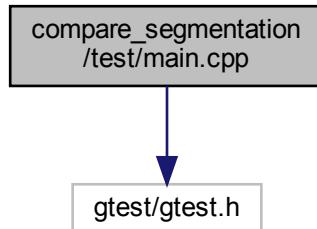
Definition at line 45 of file main.cpp.

Here is the call graph for this function:



7.27 compare_segmentation/test/main.cpp File Reference

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



Functions

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

7.27.1 Function Documentation

7.27.1.1 [main\(\)](#)

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

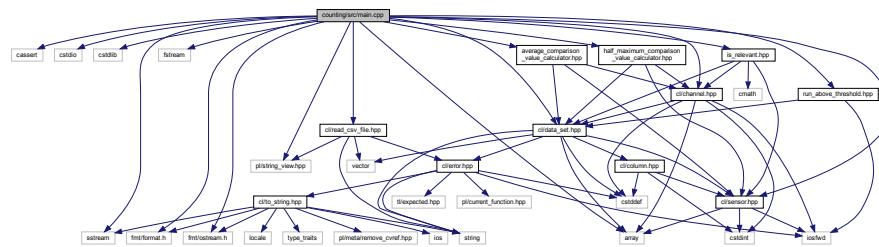
Definition at line 3 of file main.cpp.

7.28 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[])

This application counts the amount of data points above / below the maximum / minimum allowed values. And determines which channels are not relevant for which sensors in which data sets.

7.28.1 Function Documentation

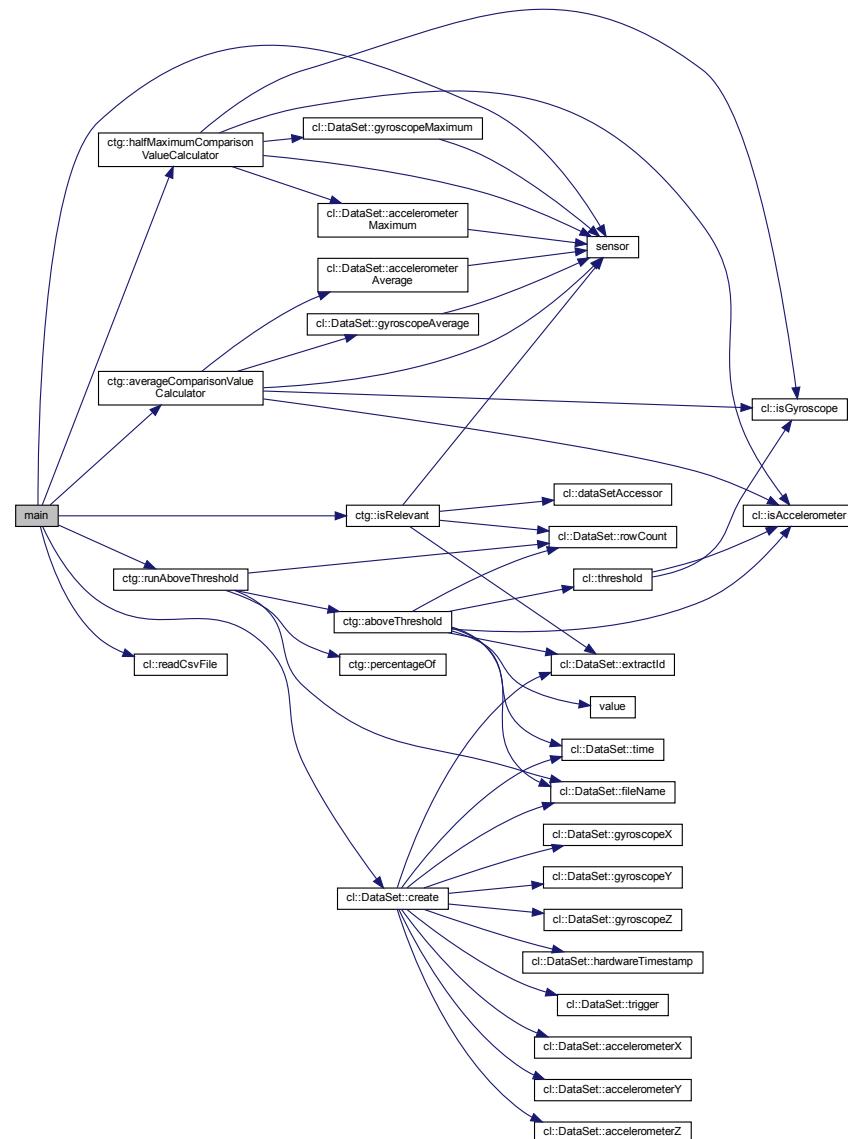
7.28.1.1 main()

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

This application counts the amount of data points above / below the maximum / minimum allowed values. And determines which channels are not relevant for which sensors in which data sets.

Definition at line 29 of file main.cpp.

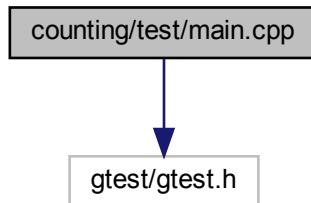
Here is the call graph for this function:



7.29 counting/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()

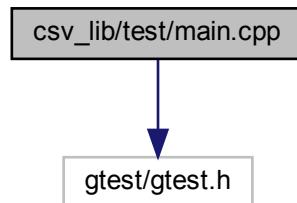
```
int main (
    int argc,
    char * argv[] )
```

Definition at line 3 of file main.cpp.

7.30 csv_lib/test/main.cpp File Reference

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

Include dependency graph for main.cpp:



Functions

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

7.30.1 Function Documentation

7.30.1.1 main()

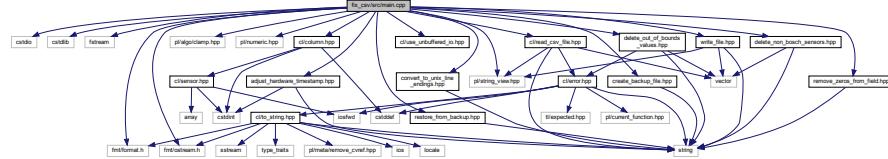
```
int main ( int argc, char * argv[ ] )
```

Definition at line 3 of file main.cpp.

7.31 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[])

Ye olde fix csv C++ application.

7.31.1 Function Documentation

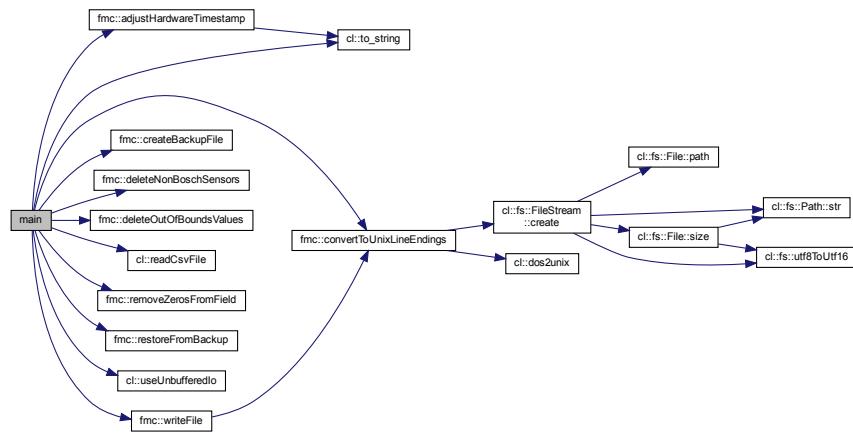
7.31.1.1 main()

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

Ye olde fix_csv C++ application.

Definition at line 29 of file main.cpp.

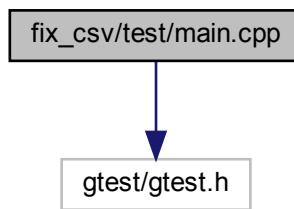
Here is the call graph for this function:



7.32 fix_csv/test/main.cpp File Reference

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

Include dependency graph for main.cpp:



Functions

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

7.32.1 Function Documentation

7.32.1.1 main()

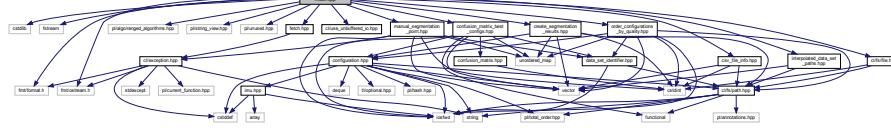
```
int main ( int argc,  
           char * argv[ ] )
```

Definition at line 3 of file main.cpp.

7.33 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:
```

include dependency graph for main.cpp



Macros

- #define SORT_PRINT(kind)

Functions

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

Entry point.

7.33.1 Macro Definition Documentation

7.33.1.1 SORT_PRINT

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

Value:

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

7.33.2 Function Documentation

7.33.2.1 main()

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

Entry point.

Parameters

<i>argc</i>	Argument count.
<i>argv</i>	Argument values.

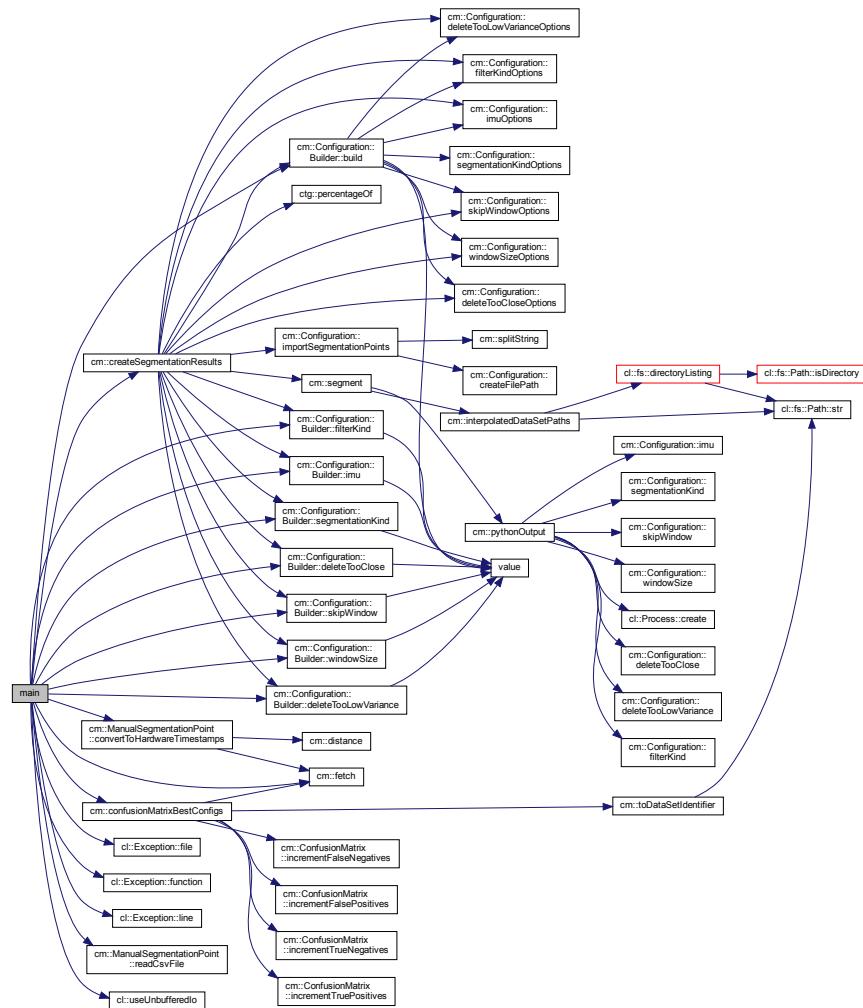
Returns

EXIT_SUCCESS on success; otherwise EXIT_FAILURE.

This confusion_matrix_app C++ application generates the confusion matrices of the different configurations of the Python segmentation algorithm and compares them. The comparison is based on the (inaccurate) ground truth that has been manually extracted from the video recordings of the exercises.

Definition at line 60 of file main.cpp.

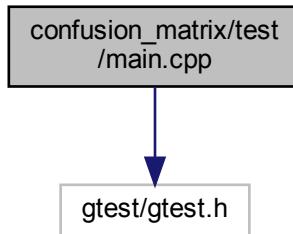
Here is the call graph for this function:



7.34 confusion_matrix/test/main.cpp File Reference

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

Include dependency graph for main.cpp:



Functions

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

7.34.1 Function Documentation

7.34.1.1 main()

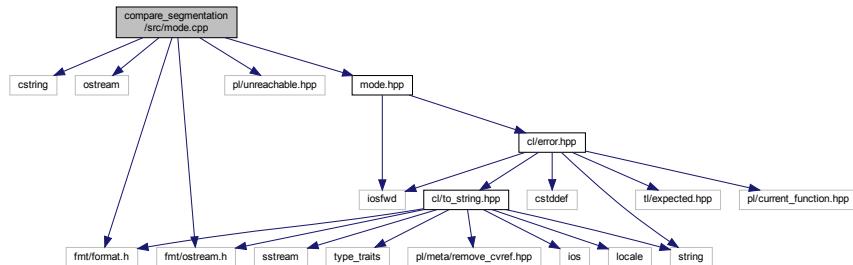
```
int main (
    int argc,
    char * argv[ ] )
```

Definition at line 3 of file main.cpp.

7.35 compare_segmentation/src/mode.cpp File Reference

```
#include <cstring>
#include <iostream>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <pl/unreachable.hpp>
#include "mode.hpp"
```

Include dependency graph for mode.cpp:



Namespaces

- [cs](#)

Macros

- `#define CS_MODE_X(enm) case Mode::enm: return os << #enm;`
- `#define CS_MODE_X(enm) if (std::strcmp(#enm, szCmdArg) == 0) { return Mode::enm; }`

Functions

- `std::ostream & cs::operator<< (std::ostream &os, Mode mode)`
Prints a Mode to an ostream.
- `cl::Expected< Mode > cs::parseMode (const char *szCmdArg)`
Parses a null-terminated byte character string as a Mode.

7.35.1 Macro Definition Documentation

7.35.1.1 CS_MODE_X [1/2]

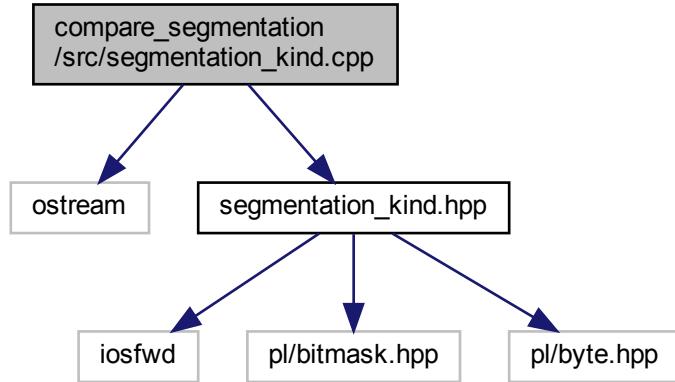
```
#define CS_MODE_X(  
    enm ) case Mode::enm:    return os << #enm;
```

7.35.1.2 CS_MODE_X [2/2]

```
#define CS_MODE_X(  
    enm ) if (std::strcmp(#enm, szCmdArg) == 0) { return Mode::enm; }
```

7.36 compare_segmentation/src/segmentation_kind.cpp File Reference

```
#include <iostream>
#include "segmentation_kind.hpp"
Include dependency graph for segmentation_kind.cpp:
```



Namespaces

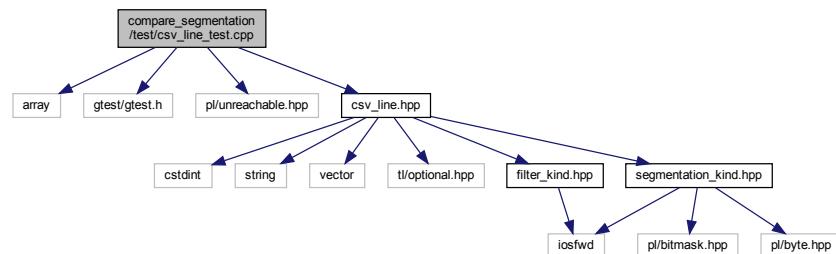
- `cs`

Functions

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

7.37 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)

7.37.1 Function Documentation

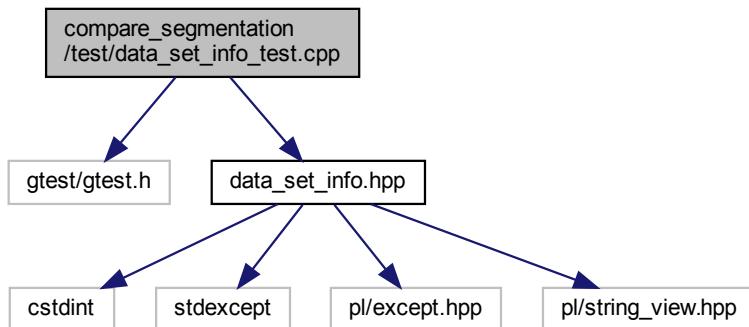
7.37.1.1 TEST()

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

Definition at line 30 of file csv_line_test.cpp.

7.38 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)

7.38.1 Function Documentation

7.38.1.1 TEST()

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

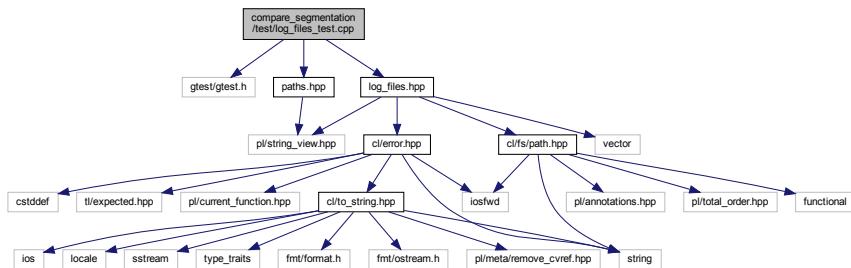
Definition at line 5 of file `data_set_info_test.cpp`.

Here is the call graph for this function:



7.39 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`)

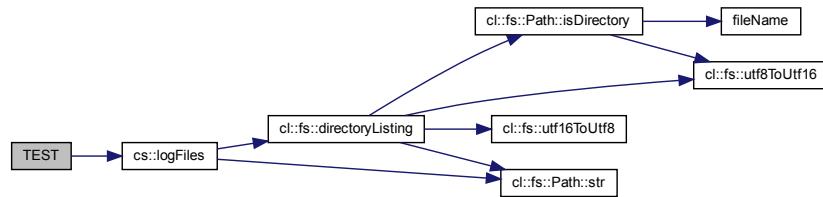
7.39.1 Function Documentation

7.39.1.1 TEST() [1/3]

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

Definition at line 6 of file log_files_test.cpp.

Here is the call graph for this function:

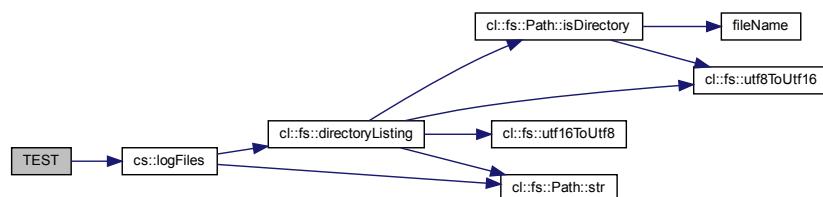


7.39.1.2 TEST() [2/3]

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

Definition at line 23 of file log_files_test.cpp.

Here is the call graph for this function:

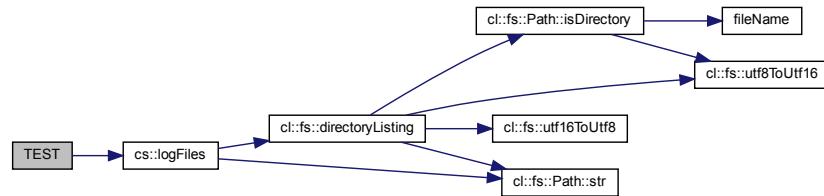


7.39.1.3 TEST() [3/3]

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

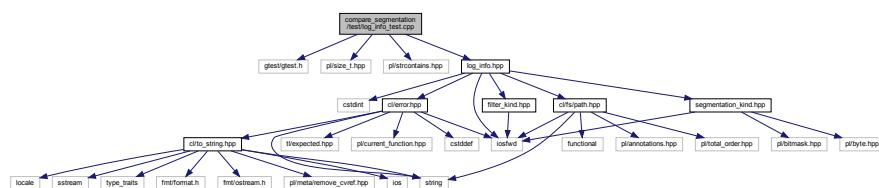
Definition at line 46 of file log_files_test.cpp.

Here is the call graph for this function:



7.40 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)

7.40.1 Function Documentation

7.40.1.1 TEST() [1/19]

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

Definition at line 388 of file log_info_test.cpp.

7.40.1.2 TEST() [2/19]

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

Definition at line 341 of file log_info_test.cpp.

Here is the call graph for this function:



7.40.1.3 TEST() [3/19]

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

Definition at line 350 of file log_info_test.cpp.

Here is the call graph for this function:



7.40.1.4 TEST() [4/19]

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

Definition at line 379 of file log_info_test.cpp.

Here is the call graph for this function:



7.40.1.5 TEST() [5/19]

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

Definition at line 359 of file log_info_test.cpp.

Here is the call graph for this function:



7.40.1.6 TEST() [6/19]

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

Definition at line 332 of file log_info_test.cpp.

Here is the call graph for this function:



7.40.1.7 TEST() [7/19]

```
TEST (
    LogInfo ,
    shouldFailIfWindowSizeIsInvalid )
```

Definition at line 368 of file log_info_test.cpp.

Here is the call graph for this function:

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

```
TEST (
    LogInfo ,
    shouldResultInErrorIfLogFilePathIsTooShort )
```

Definition at line 325 of file log_info_test.cpp.

Here is the call graph for this function:



7.40.1.9 TEST() [9/19]

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

Definition at line 8 of file log_info_test.cpp.

Here is the call graph for this function:

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

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

Definition at line 37 of file log_info_test.cpp.

Here is the call graph for this function:



7.40.1.11 TEST() [11/19]

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

Definition at line 66 of file log_info_test.cpp.

Here is the call graph for this function:

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

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

Definition at line 95 of file log_info_test.cpp.

Here is the call graph for this function:



7.40.1.13 TEST() [13/19]

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

Definition at line 124 of file log_info_test.cpp.

Here is the call graph for this function:

**7.40.1.14 TEST() [14/19]**

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

Definition at line 153 of file log_info_test.cpp.

Here is the call graph for this function:



7.40.1.15 TEST() [15/19]

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

Definition at line 182 of file log_info_test.cpp.

Here is the call graph for this function:

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

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

Definition at line 211 of file log_info_test.cpp.

Here is the call graph for this function:



7.40.1.17 TEST() [17/19]

```
TEST (
    LogInfo ,
    shouldWork9 )
```

Definition at line 240 of file log_info_test.cpp.

Here is the call graph for this function:

**7.40.1.18 TEST() [18/19]**

```
TEST (
    LogInfo ,
    shouldWorkWithPath )
```

Definition at line 269 of file log_info_test.cpp.

Here is the call graph for this function:



7.40.1.19 TEST() [19/19]

```
TEST (
    LogInfo ,
    shouldWorkWithOldPath2 )
```

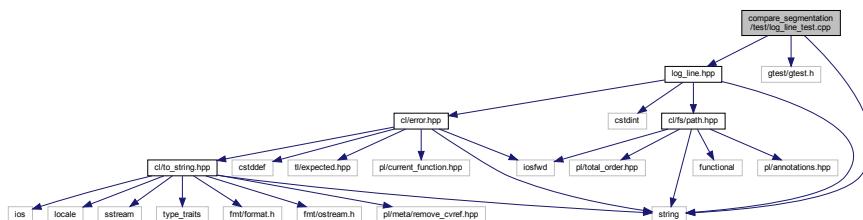
Definition at line 297 of file log_info_test.cpp.

Here is the call graph for this function:



7.41 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)

7.41.1 Function Documentation

7.41.1.1 TEST() [1/4]

```
TEST (
    LogLine ,
    shouldNotMatchGarbage   )
```

Definition at line 41 of file log_line_test.cpp.

Here is the call graph for this function:



7.41.1.2 TEST() [2/4]

```
TEST (
    LogLine ,
    shouldNotParseGarbageSensor   )
```

Definition at line 48 of file log_line_test.cpp.

Here is the call graph for this function:



7.41.1.3 TEST() [3/4]

```
TEST (
    LogLine ,
    shouldWorkWithOldLine )
```

Definition at line 25 of file log_line_test.cpp.

Here is the call graph for this function:

**7.41.1.4 TEST() [4/4]**

```
TEST (
    LogLine ,
    shouldWorkWithPreprocessedLine )
```

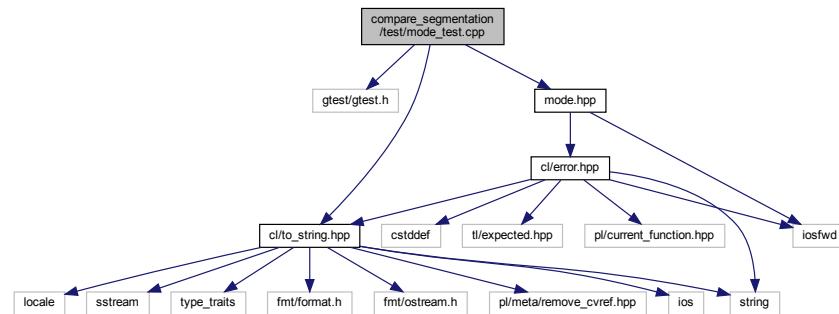
Definition at line 9 of file log_line_test.cpp.

Here is the call graph for this function:

**7.42 compare_segmentation/test/mode_test.cpp File Reference**

```
#include "gtest/gtest.h"
#include <c1/to_string.hpp>
```

```
#include "mode.hpp"
Include dependency graph for mode_test.cpp:
```



Functions

- `TEST` (`Mode, shouldPrintCorrectly`)
- `TEST` (`Mode, shouldParseCorrectly`)

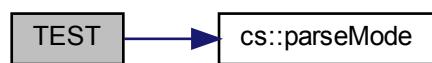
7.42.1 Function Documentation

7.42.1.1 TEST() [1/2]

```
TEST (
    Mode ,
    shouldParseCorrectly )
```

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

Here is the call graph for this function:

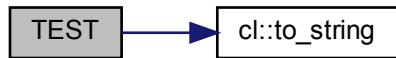


7.42.1.2 TEST() [2/2]

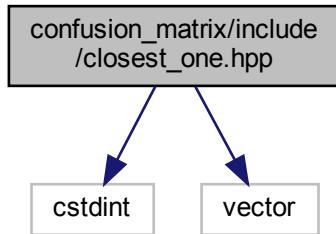
```
TEST(  
    Mode,  
    shouldPrintCorrectly)
```

Definition at line 7 of file mode_test.cpp.

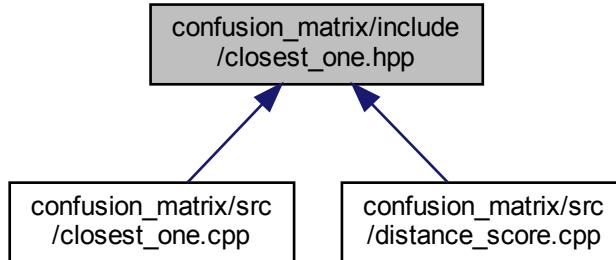
Here is the call graph for this function:

**7.43 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`

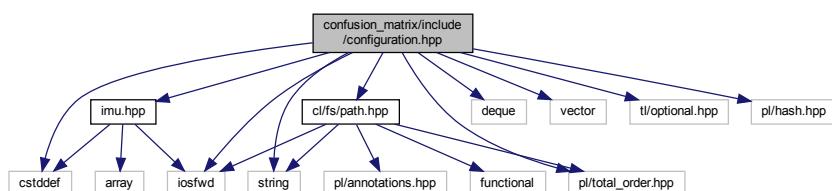
Functions

- `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.

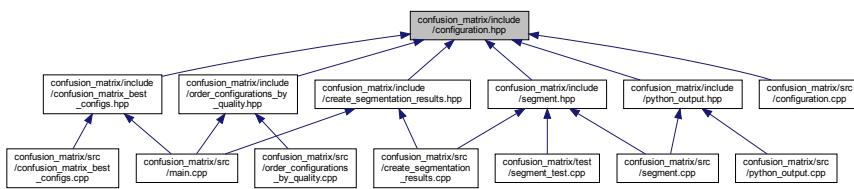
7.44 confusion_matrix/include/configuration.hpp File Reference

```
#include <cstddef>
#include <deque>
#include <iostream>
#include <string>
#include <vector>
#include <tutorial/optional.hpp>
#include <pl/hash.hpp>
#include <pl/total_order.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

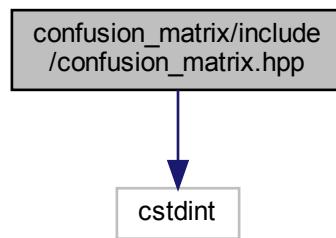
- class [cm::Configuration](#)
Represents a possible configuration for the Python segmentor.
- class [cm::Configuration::Builder](#)
Builder type for Configuration.
- struct [std::hash<::cm::Configuration >](#)

Namespaces

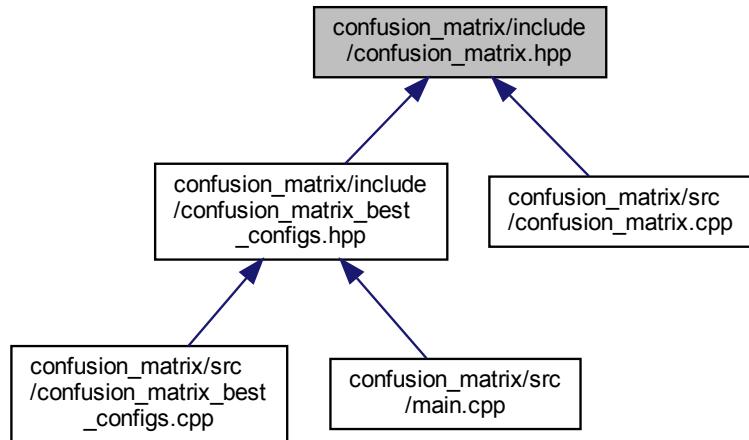
- [cm](#)

7.45 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`

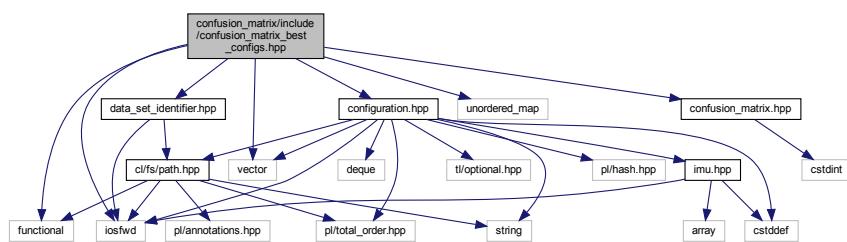
Type to represent a confusion matrix.

Namespaces

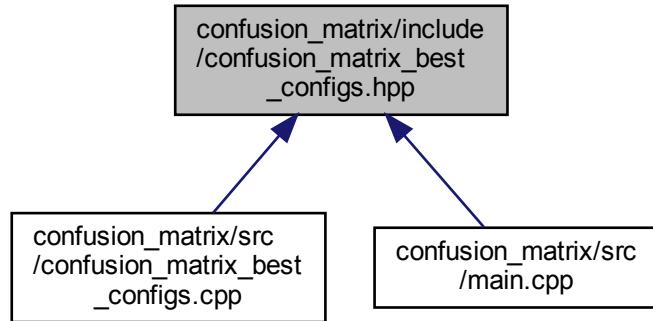
- cm

7.46 confusion_matrix/include/confusion_matrix_best_configs.hpp File Reference

```
#include <functional>
#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`
A *Configuration* with a *ConfusionMatrix*.

Namespaces

- `cm`

Macros

- `#define CM_SORTER(criterion, op)`
Macro to define a sorter based on a single criterion for `ConfigWithTotalConfusionMatrix` objects.

Functions

- `cm::CM_SORTER (truePositives, >)`
Sorts `ConfigWithTotalConfusionMatrix` objects by true positives (highest first)
- `cm::CM_SORTER (trueNegatives, >)`
Sorts `ConfigWithTotalConfusionMatrix` objects by true negatives (highest first)
- `cm::CM_SORTER (falsePositives,<)`
Sorts `ConfigWithTotalConfusionMatrix` objects by false positives (lowest first)
- `cm::CM_SORTER (falseNegatives,<)`
Sorts `ConfigWithTotalConfusionMatrix` objects by false negatives (lowest first)
- `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, const std::function< bool(const ConfigWithTotalConfusionMatrix &, const ConfigWithTotalConfusionMatrix &)> &sorter)`
Determines the 'best' configurations.

Variables

- struct {
 } [cm::disregardTrueNegativesSorter](#)

Sorter to sort [ConfigWithTotalConfusionMatrix](#) objects by the count of true positives minus the count of false positives minus the count of false negatives.

- struct {
 } [cm::addTrueSubtractFalseSorter](#)

Sorter to sort [ConfigWithTotalConfusionMatrix](#) objects by the sum of true positives and true negatives minus the false positives minus the false negatives.

7.46.1 Macro Definition Documentation

7.46.1.1 CM_SORTER

```
#define CM_SORTER(
    criterion,
    op )
```

Value:

```
inline constexpr struct {
    [[nodiscard]] bool operator()(
        const ConfigWithTotalConfusionMatrix& lhs,
        const ConfigWithTotalConfusionMatrix& rhs) const noexcept
{
    if (
        !(lhs.matrix.criterion() op rhs.matrix.criterion())
        && !(rhs.matrix.criterion() op lhs.matrix.criterion())) {
        return lhs.config < rhs.config;
    }

    return lhs.matrix.criterion() op rhs.matrix.criterion();
} criterion##Sorter
```

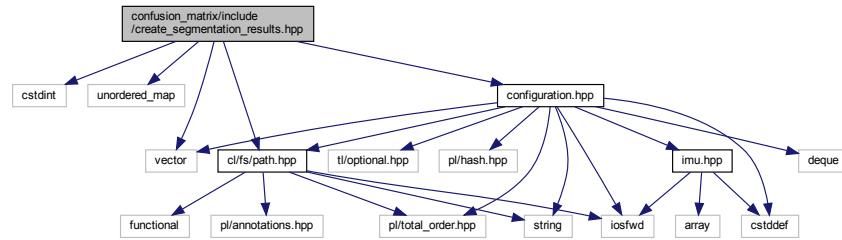
Macro to define a sorter based on a single criterion for [ConfigWithTotalConfusionMatrix](#) objects.

Definition at line 50 of file [confusion_matrix_best_configs.hpp](#).

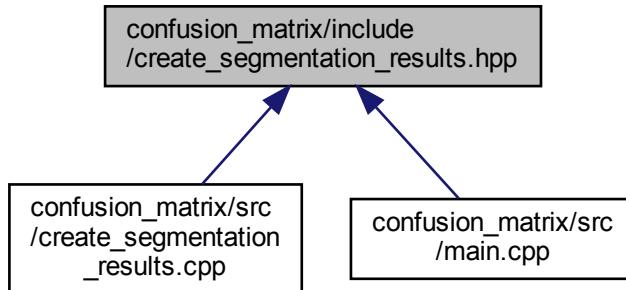
7.47 confusion_matrix/include/create_segmentation_results.hpp File Reference

```
#include <cstdint>
#include <unordered_map>
#include <vector>
#include <c1/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`

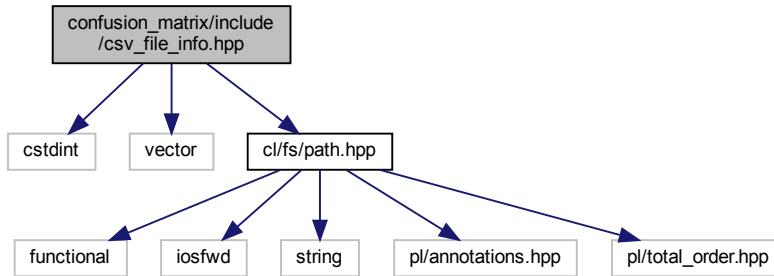
Functions

- `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.

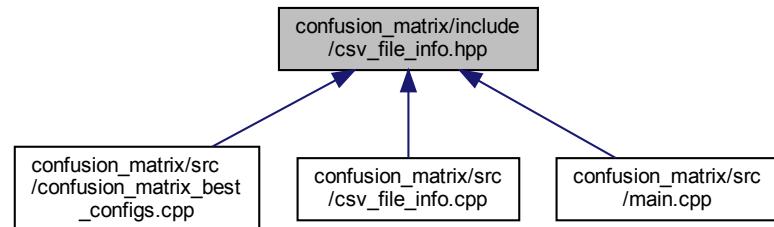
7.48 confusion_matrix/include/csv_file_info.hpp File Reference

```
#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.

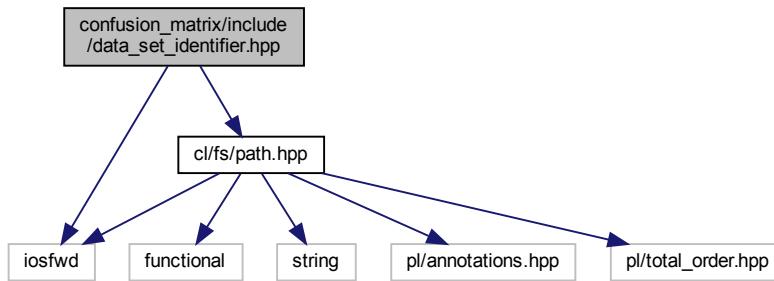
Namespaces

- [cm](#)

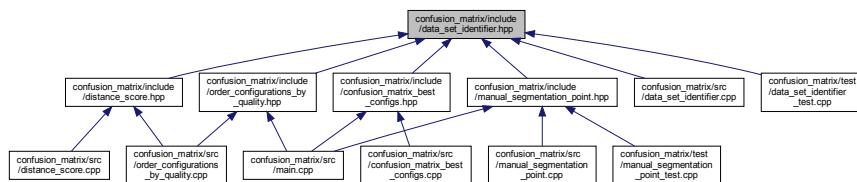
7.49 confusion_matrix/include/data_set_identifier.hpp File Reference

```
#include <iostream>
#include <cl/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_X](#) }

Scoped enum type for the data set identifiers.

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.

7.49.1 Macro Definition Documentation

7.49.1.1 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.

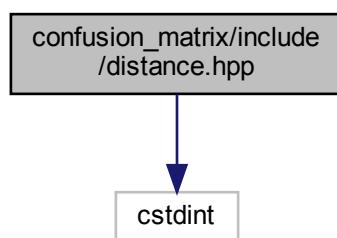
7.49.1.2 CM_DATA_SET_IDENTIFIER_X

```
#define CM_DATA_SET_IDENTIFIER_X(
    enm ) enm,
```

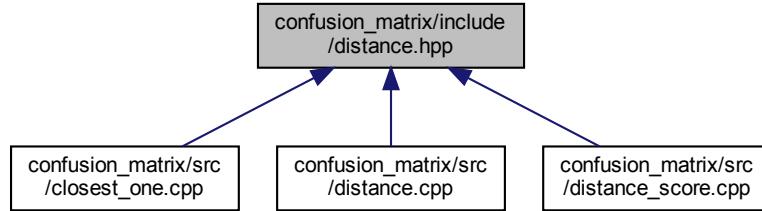
Definition at line 34 of file data_set_identifier.hpp.

7.50 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`

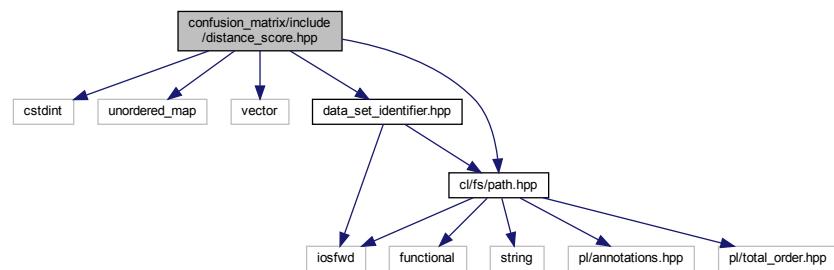
Functions

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

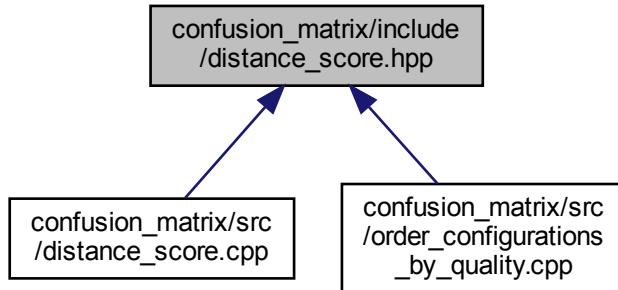
Calculates the distance between a and b.

7.51 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`

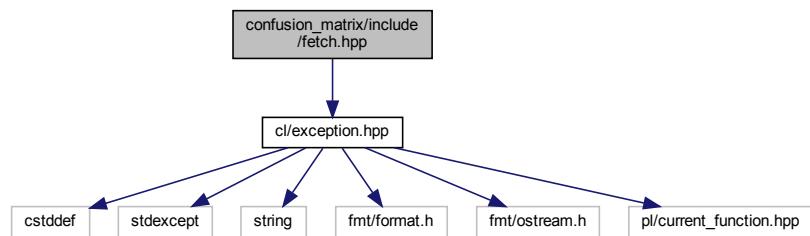
Functions

- `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)`

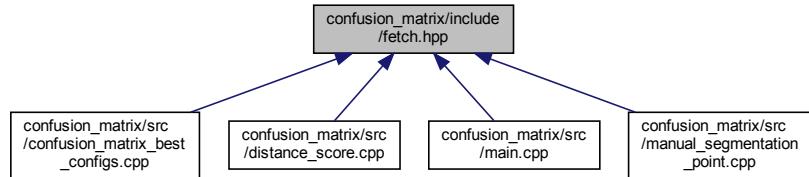
This old approach of distance scores didn't work too well for the confusion matrices.

7.52 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

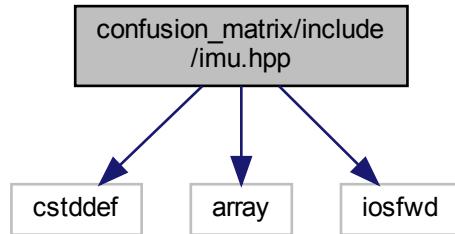
- `cm`

Functions

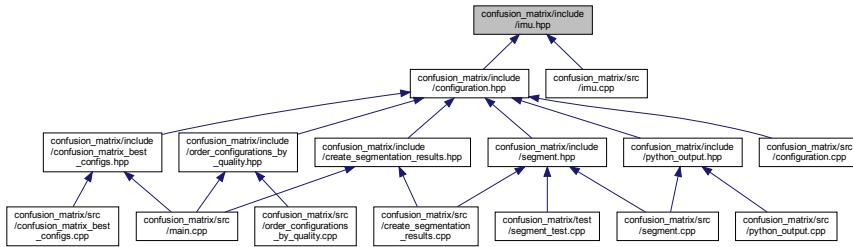
- template<typename Map , typename Key >
auto `cm::fetch` (const Map &map, const Key &key)
- Fetches a value from a map for a given key.*

7.53 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.

7.53.1 Macro Definition Documentation

7.53.1.1 CM_IMU

```
#define CM_IMU
```

Value:

```
CM_IMU_X (Accelerometer) \
CM_IMU_X (Gyroscope)
```

Definition at line 10 of file imu.hpp.

7.53.1.2 CM_IMU_X [1/3]

```
#define CM_IMU_X(
    enm ) enm,
```

Definition at line 18 of file imu.hpp.

7.53.1.3 CM_IMU_X [2/3]

```
#define CM_IMU_X(
    enm ) +1
```

Definition at line 18 of file imu.hpp.

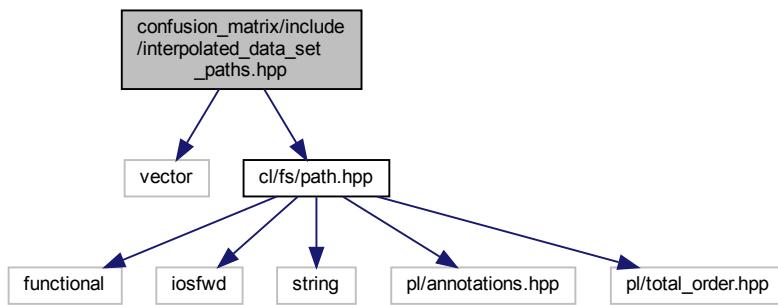
7.53.1.4 CM_IMU_X [3/3]

```
#define CM_IMU_X(
    enm ) ::cm::Imu::enm,
```

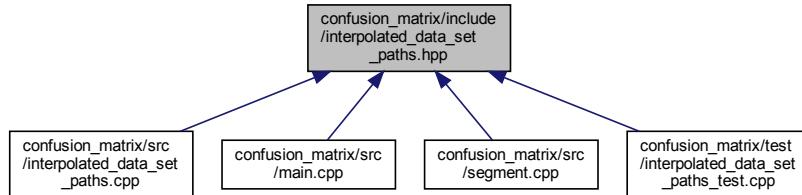
Definition at line 18 of file imu.hpp.

7.54 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`

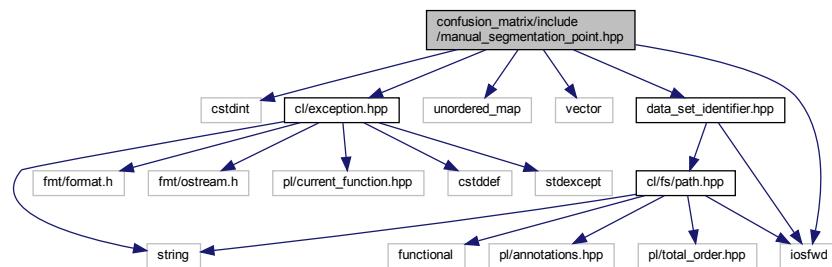
Functions

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

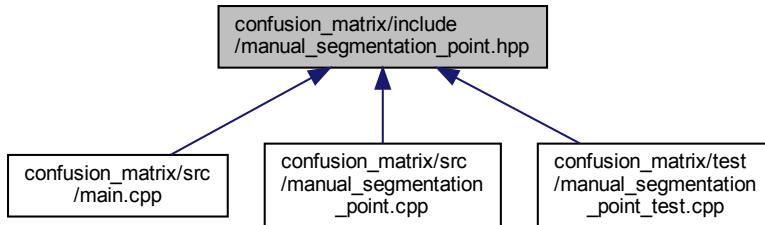
Returns the paths to the interpolated data sets.

7.55 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

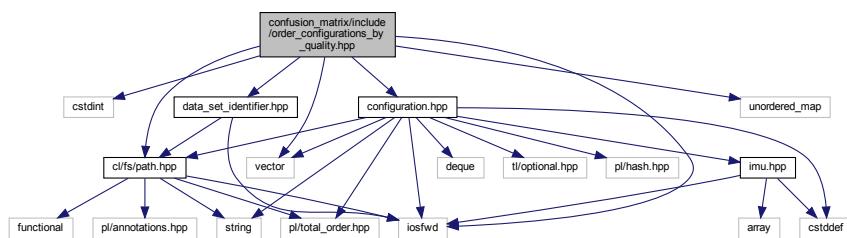
- class [cm::ManualSegmentationPoint](#)
Type used to represent a manual segmentation point.

Namespaces

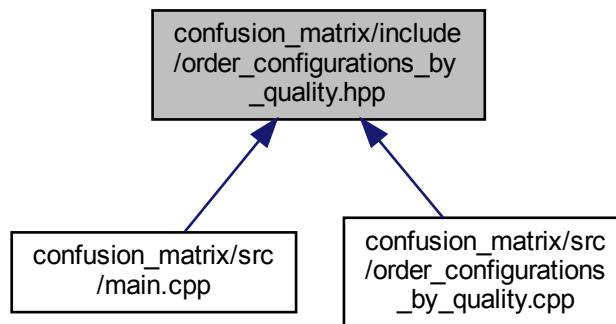
- [cm](#)

7.56 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

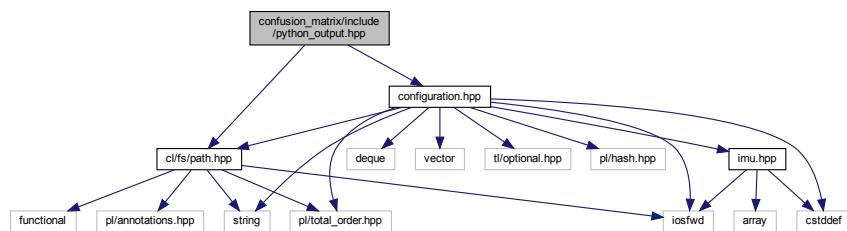
- [cm](#)

Functions

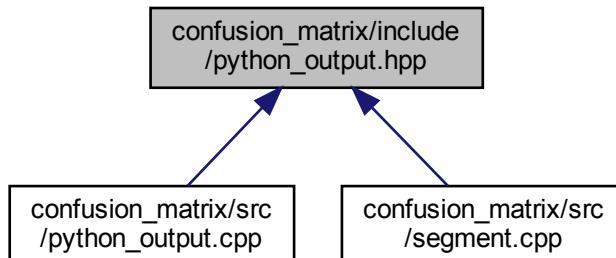
- bool `cm::operator<` (const ConfigWithDistanceScore &lhs, const ConfigWithDistanceScore &rhs) noexcept
- std::ostream & `cm::operator<<` (std::ostream &os, const ConfigWithDistanceScore &configWithDistScore)
- std::vector< ConfigWithDistanceScore > `cm::orderConfigurationsByQuality` (const std::unordered_map< DataSetIdentifier, std::vector< std::uint64_t > >> &manualSegmentationPoints, const std::unordered_map< Configuration, std::unordered_map< cl::fs::Path, std::vector< std::uint64_t > >>> &algorithmicallyDeterminedSegmentationPoints)

7.57 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

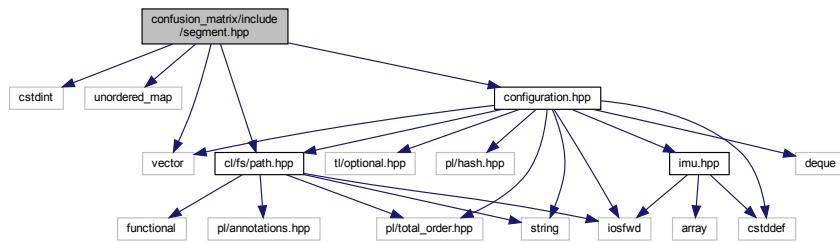
- `cm`

Functions

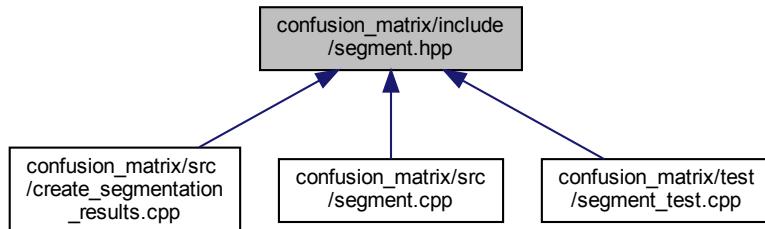
- std::string `cm::pythonOutput` (const `cl::fs::Path` &csvFilePath, const Configuration &segmentorConfiguration)
Runs the Python segmentor on path.

7.58 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

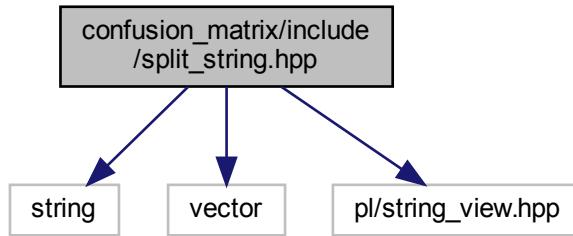
- `cm`

Functions

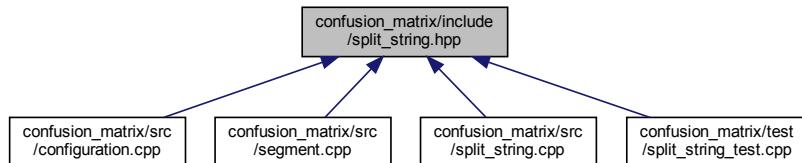
- `std::unordered_map< cl::fs::Path, std::vector< std::uint64_t > > cm::segment (const Configuration &segmentorConfiguration)`

7.59 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](#)

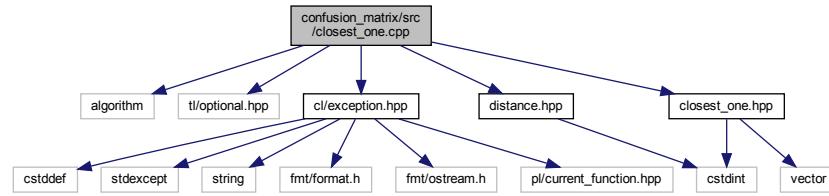
Functions

- `std::vector< std::string > cm::splitString (std::string string, pl::string_view splitBy)`
Splits string by splitBy.

7.60 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

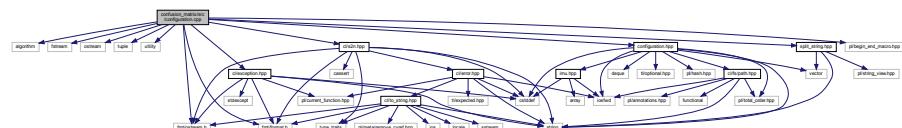
- `cm`

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.

7.61 confusion_matrix/src/configuration.cpp File Reference

```
#include <algorithm>
#include <fstream>
#include <iostream>
#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)`

7.61.1 Macro Definition Documentation

7.61.1.1 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
```

7.61.1.2 CM_ENSURE_HAS_VALUE

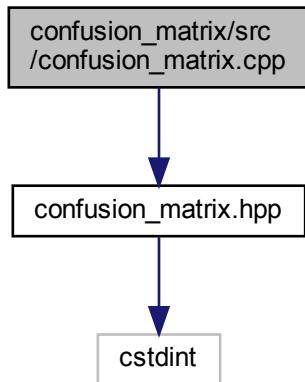
```
#define CM_ENSURE_HAS_VALUE (
    dataMember )
```

Value:

```
PL_BEGIN_MACRO
if (!dataMember.has_value()) {
    CL_THROW_FMT("\"{}\" was nullopt!", #dataMember); \
}
PL_END_MACRO
```

7.62 confusion_matrix/src/confusion_matrix.cpp File Reference

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



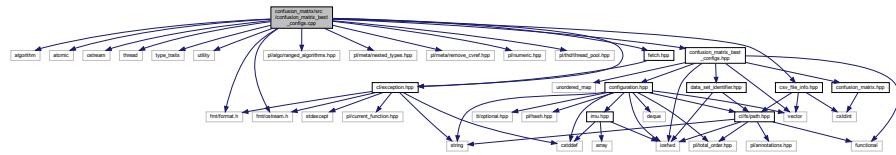
Namespaces

- [cm](#)

7.63 confusion_matrix/src/confusion_matrix_best_configs.cpp File Reference

```
#include <algorithm>  
#include <atomic>  
#include <iostream>  
#include <thread>  
#include <type_traits>  
#include <utility>  
#include <fmt/format.h>  
#include <fmt/ostream.h>  
#include <pl/algo/ranged_algorithms.hpp>  
#include <pl/meta/nested_types.hpp>  
#include <pl/meta/remove_cvref.hpp>  
#include <pl/numeric.hpp>  
#include <pl/thd/thread_pool.hpp>  
#include <cl/exception.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](#)

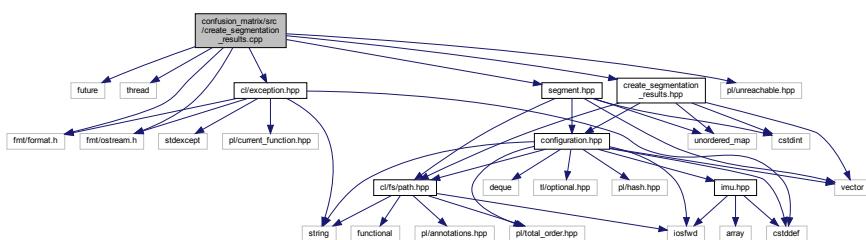
Functions

- 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, const std::function< bool(const ConfigWithTotalConfusionMatrix &, const ConfigWithTotalConfusionMatrix &) > &sorter)

Determines the 'best' configurations.

7.64 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](#)

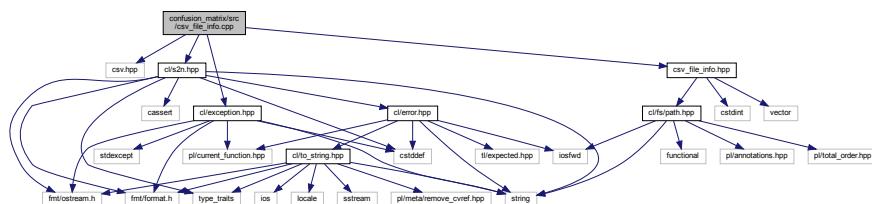
Functions

- 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.

7.65 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:
```

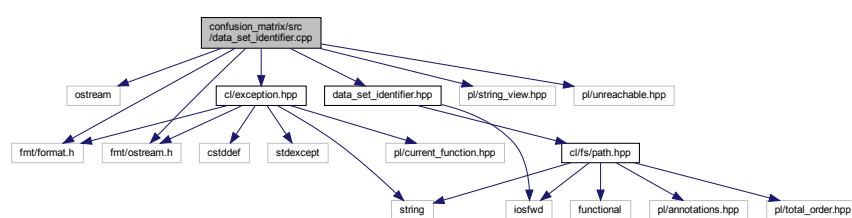


Namespaces

- `cm`

7.66 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.

7.66.1 Macro Definition Documentation

7.66.1.1 CM_DATA_SET_IDENTIFIER_X

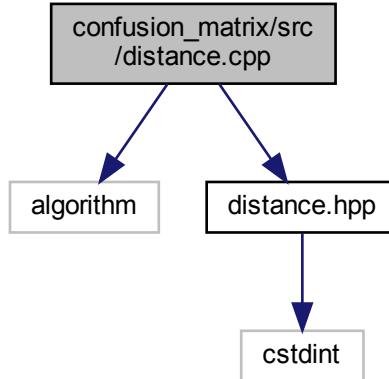
```
#define CM_DATA_SET_IDENTIFIER_X(  
    enm ) case DataSetIdentifier::enm:    return #enm; /* stringify */
```

7.66.1.2 DSI

```
#define DSI DataSetIdentifier
```

7.67 confusion_matrix/src/distance.cpp File Reference

```
#include <algorithm>
#include "distance.hpp"
Include dependency graph for distance.cpp:
```



Namespaces

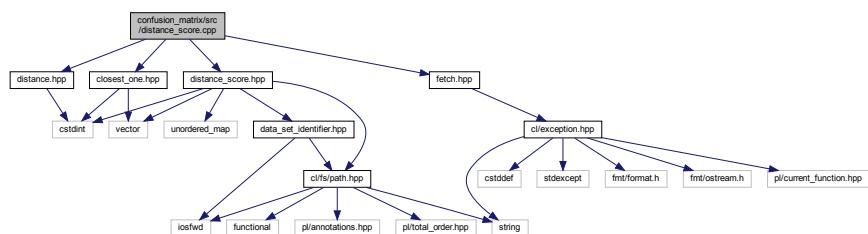
- [cm](#)

Functions

- std::uint64_t [cm::distance](#) (std::uint64_t a, std::uint64_t b)
Calculates the distance between a and b.

7.68 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](#)

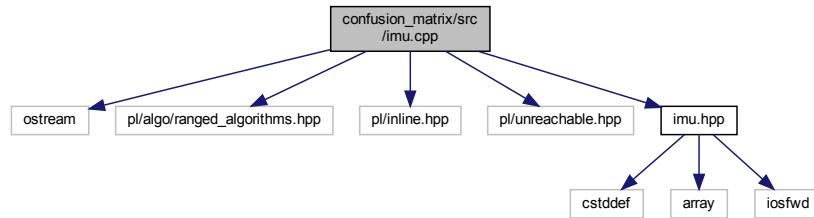
Functions

- 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)

This old approach of distance scores didn't work too well for the confusion matrices.

7.69 confusion_matrix/src/imu.cpp File Reference

```
#include <iostream>
#include <pl/algo/ranged_algorithms.hpp>
#include <pl/inline.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.

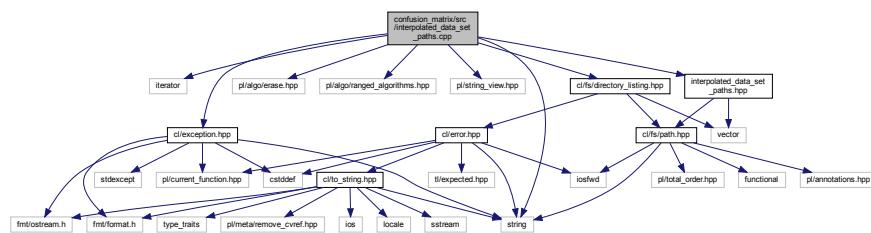
7.69.1 Macro Definition Documentation

7.69.1.1 CM_IMU_X

```
#define CM_IMU_X(
    enm ) case Imu::enm: return os << toLower(#enm);
```

7.70 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`

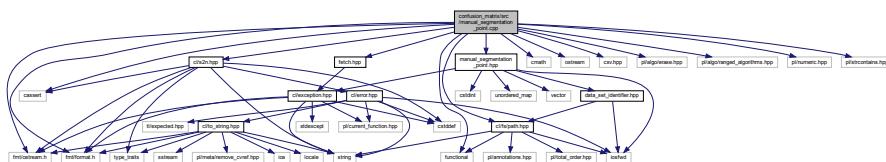
Functions

- `std::vector< cl::fs::Path > cm::interpolatedDataSetPaths ()`
Returns the paths to the interpolated data sets.

7.71 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 <c1/fs/path.hpp>
#include <c1/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)

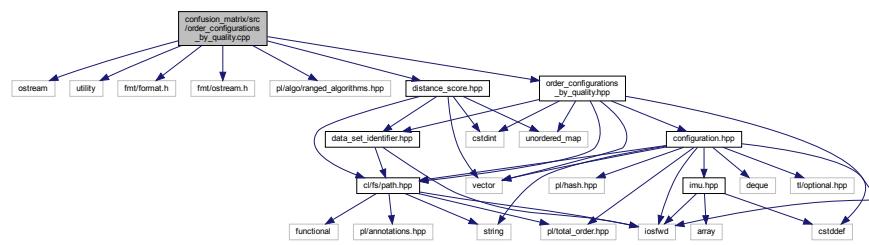
7.71.1 Macro Definition Documentation

7.71.1.1 DSI

```
#define DST DataSetIdentifier
```

7.72 confusion_matrix/src/order_configurations_by_quality.cpp File Reference

```
#include <iostream>
#include <utility>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <pl/algo/ranged_algorithms.hpp>
#include "distance_score.hpp"
#include "order_configurations_by_quality.hpp"
Include dependency graph for order_configurations_by_quality.cpp:
```



Namespaces

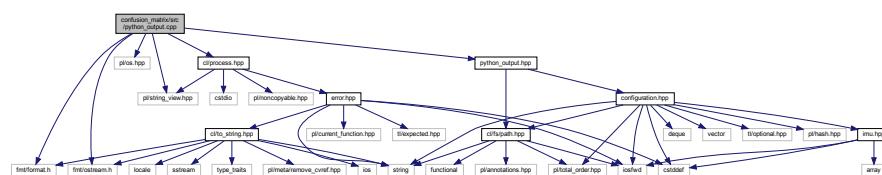
- [cm](#)

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)

7.73 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 <cl/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.

7.73.1 Macro Definition Documentation

7.73.1.1 CM DEV NULL

```
#define CM_DEV NULL "/dev/null"
```

Object like macro for /dev/null.

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

7.73.1.2 CM_SEGMENTOR

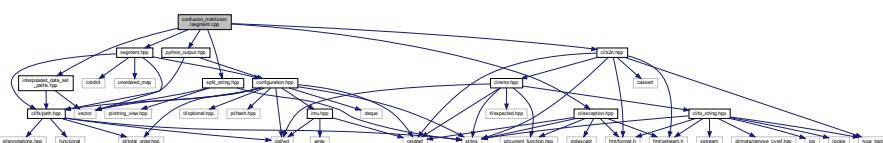
```
#define CM_SEGMENTOR "./preprocessed_segment.sh"
```

Object like macro for the segmentor script.

Definition at line 23 of file `python_output.cpp`.

7.74 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

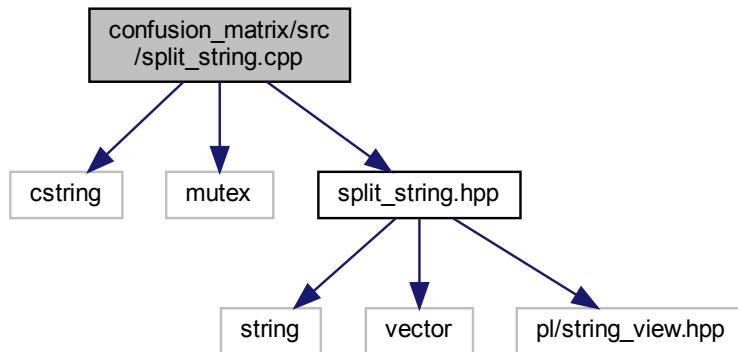
- [cm](#)

Functions

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

7.75 confusion_matrix/src/split_string.cpp File Reference

```
#include <cstring>
#include <mutex>
#include "split_string.hpp"
Include dependency graph for split_string.cpp:
```



Namespaces

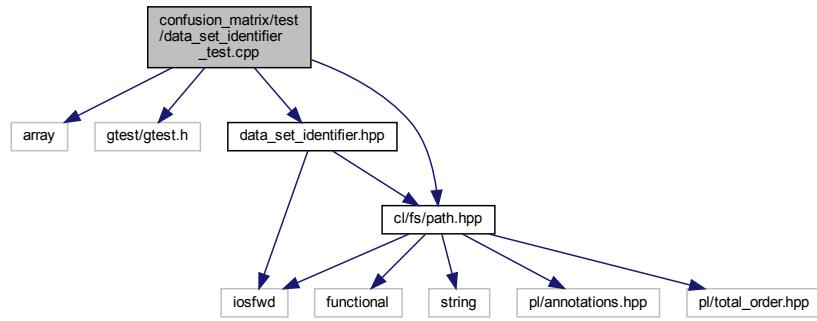
- [cm](#)

Functions

- `std::vector< std::string > cm::splitString (std::string string, pl::string_view splitBy)`
Splits string by splitBy.

7.76 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)`

7.76.1 Macro Definition Documentation

7.76.1.1 DSI

```
#define DSI ::cm::DataSetIdentifier
```

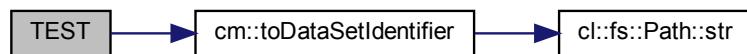
7.76.2 Function Documentation

7.76.2.1 TEST()

```
TEST (
    DataSetIdentifier ,
    shouldConvertPaths )
```

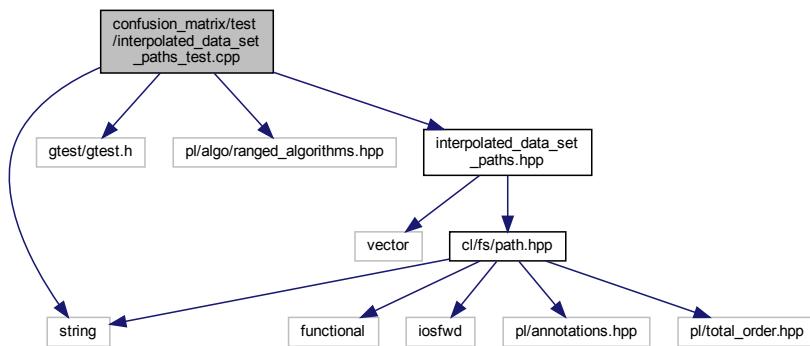
Definition at line 9 of file `data_set_identifier_test.cpp`.

Here is the call graph for this function:



7.77 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`)

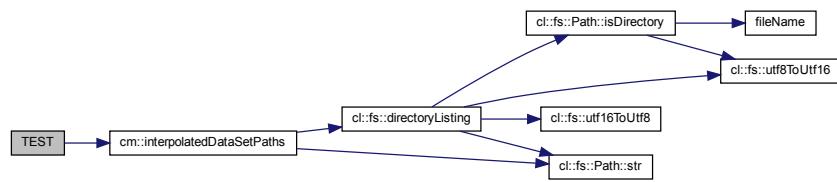
7.77.1 Function Documentation

7.77.1.1 TEST()

```
TEST (
    interpolatedDataSetPaths ,
    shouldFetchPaths )
```

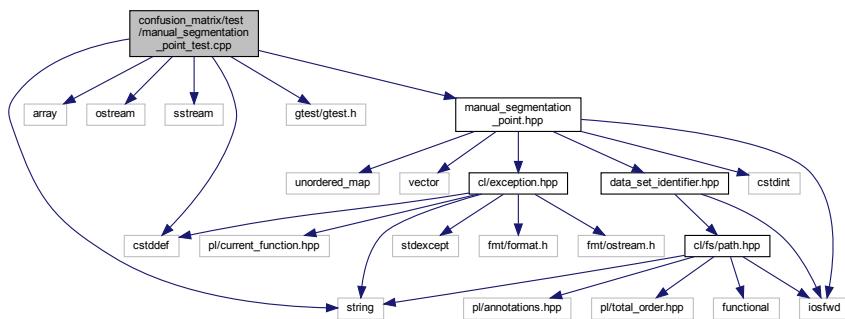
Definition at line 9 of file interpolated_data_set_paths_test.cpp.

Here is the call graph for this function:



7.78 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)`

7.78.1 Macro Definition Documentation

7.78.1.1 DSI

```
#define DSI ::cm::DataSetIdentifier
```

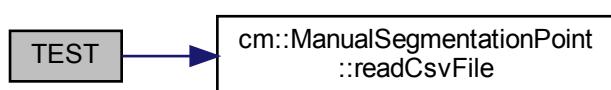
7.78.2 Function Documentation

7.78.2.1 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:



7.78.2.2 TEST() [2/11]

```
TEST (
    ManualSegmentationPoint ,
    shouldConstruct )
```

Definition at line 12 of file manual_segmentation_point_test.cpp.

7.78.2.3 TEST() [3/11]

```
TEST (
    ManualSegmentationPoint ,
    shouldConvertFramesToMilliseconds )
```

Definition at line 82 of file manual_segmentation_point_test.cpp.

7.78.2.4 TEST() [4/11]

```
TEST (
    ManualSegmentationPoint ,
    shouldConvertHourToMilliseconds )
```

Definition at line 64 of file manual_segmentation_point_test.cpp.

7.78.2.5 TEST() [5/11]

```
TEST (
    ManualSegmentationPoint ,
    shouldConvertMinuteToMilliseconds )
```

Definition at line 70 of file manual_segmentation_point_test.cpp.

7.78.2.6 TEST() [6/11]

```
TEST (
    ManualSegmentationPoint ,
    shouldConvertSecondToMilliseconds )
```

Definition at line 76 of file manual_segmentation_point_test.cpp.

7.78.2.7 TEST() [7/11]

```
TEST (
    ManualSegmentationPoint ,
    shouldConvertToMilliseconds )
```

Definition at line 58 of file manual_segmentation_point_test.cpp.

7.78.2.8 TEST() [8/11]

```
TEST (
    ManualSegmentationPoint ,
    shouldPrint )
```

Definition at line 370 of file manual_segmentation_point_test.cpp.

7.78.2.9 TEST() [9/11]

```
TEST (
    ManualSegmentationPoint ,
    shouldThrowWhenConstructingWithInvalidFrame )
```

Definition at line 46 of file manual_segmentation_point_test.cpp.

7.78.2.10 TEST() [10/11]

```
TEST (
    ManualSegmentationPoint ,
    shouldThrowWhenConstructingWithInvalidMinute )
```

Definition at line 22 of file manual_segmentation_point_test.cpp.

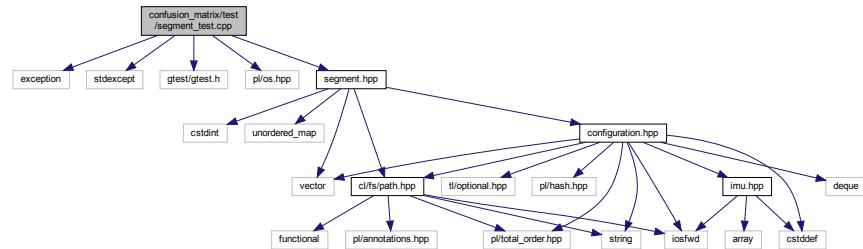
7.78.2.11 TEST() [11/11]

```
TEST (
    ManualSegmentationPoint ,
    shouldThrowWhenConstructingWithInvalidSecond )
```

Definition at line 34 of file manual_segmentation_point_test.cpp.

7.79 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)`

7.79.1 Macro Definition Documentation

7.79.1.1 EXPECT_SEGMENTATION_POINTS

```
#define EXPECT_SEGMENTATION_POINTS(
    path,
    ... ) EXPECT_EQ((std::vector<std::uint64_t>{__VA_ARGS__}), fetch(path))
```

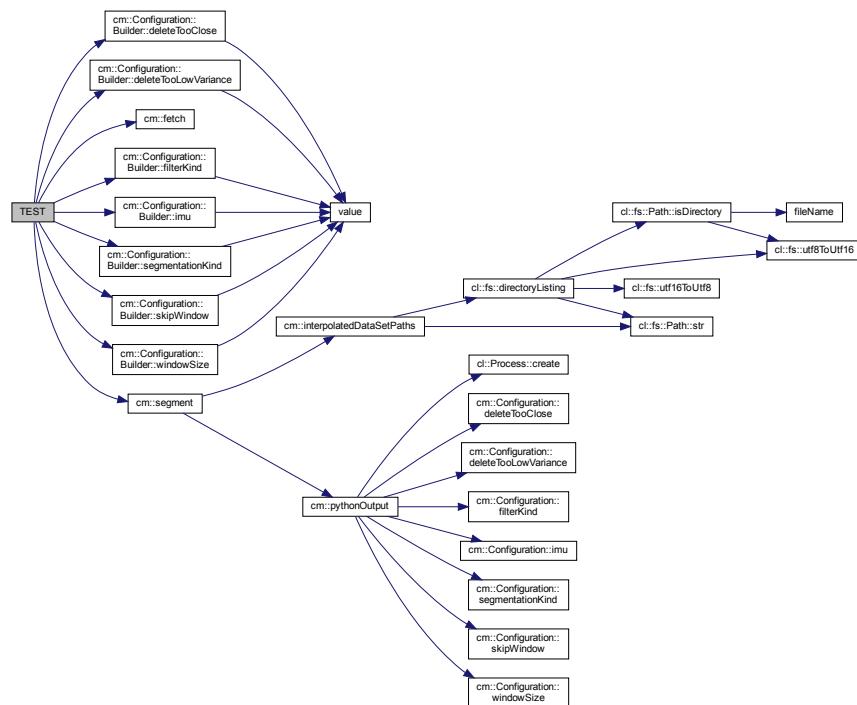
7.79.2 Function Documentation

7.79.2.1 TEST()

```
TEST (
    segment ,
    shouldGetExpectedSegmentationPointsFromPython )
```

Definition at line 11 of file segment_test.cpp.

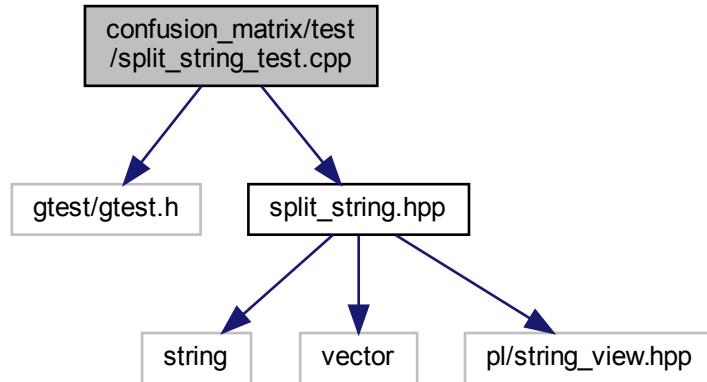
Here is the call graph for this function:



7.80 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`)

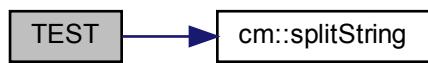
7.80.1 Function Documentation

7.80.1.1 TEST()

```
TEST (
    splitString ,
    shouldSplitString )
```

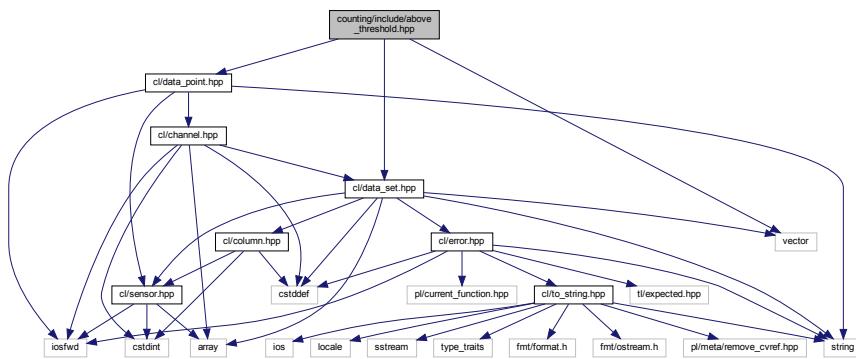
Definition at line 5 of file `split_string_test.cpp`.

Here is the call graph for this function:

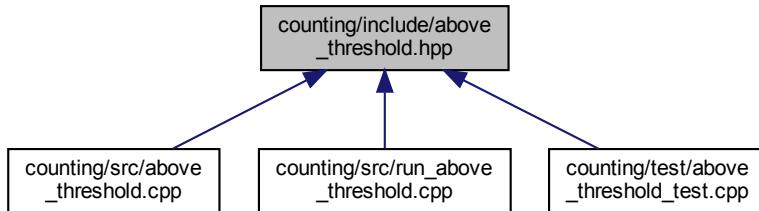


7.81 counting/include/above_threshold.hpp File Reference

```
#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`

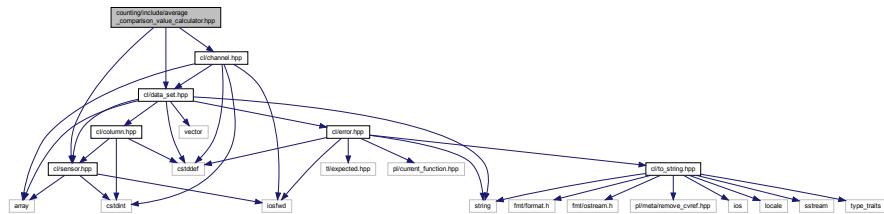
Functions

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

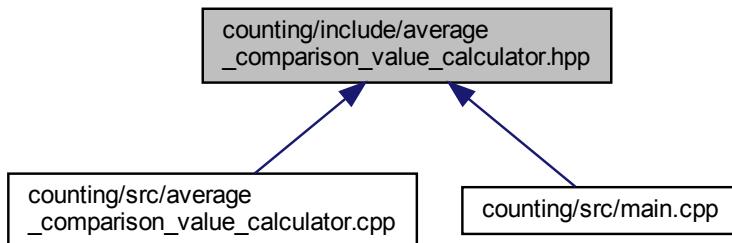
Returns all the data points that are above / below the threshold.

7.82 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`

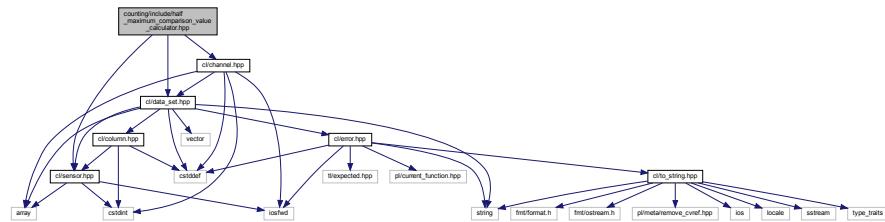
Functions

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

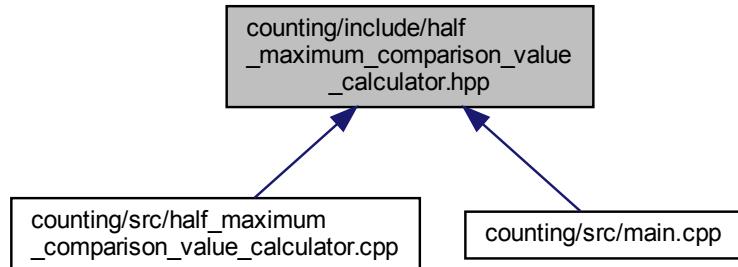
Calculates the average value for the given sensor, channel and data set.

7.83 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

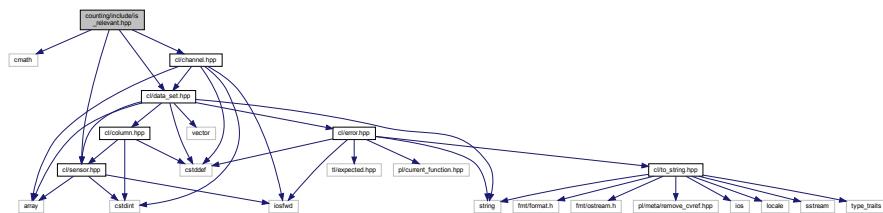
Functions

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

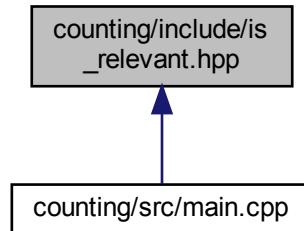
Returns half of the maximum for the given sensor, channel and data set.

7.84 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`

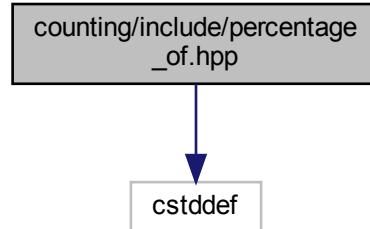
Functions

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

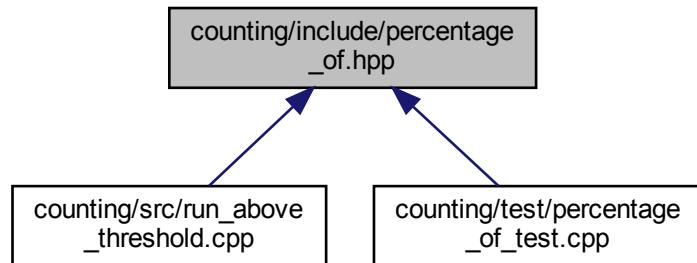
Checks if a channel is relevant for a sensor in a data set.

7.85 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

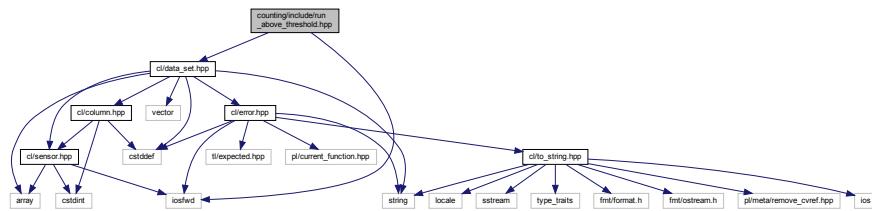
- [ctg](#)

Functions

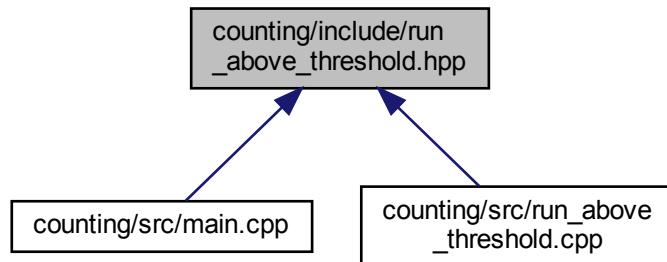
- `constexpr long double ctg::percentageOf (std::size_t amount, std::size_t totalCount) noexcept`
Calculates what percentage amount is of totalCount.

7.86 counting/include/run_above_threshold.hpp File Reference

```
#include <iostream>
#include "cl/data_set.hpp"
Include dependency graph for run_above_threshold.hpp:
```



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



Namespaces

- `ctg`

Functions

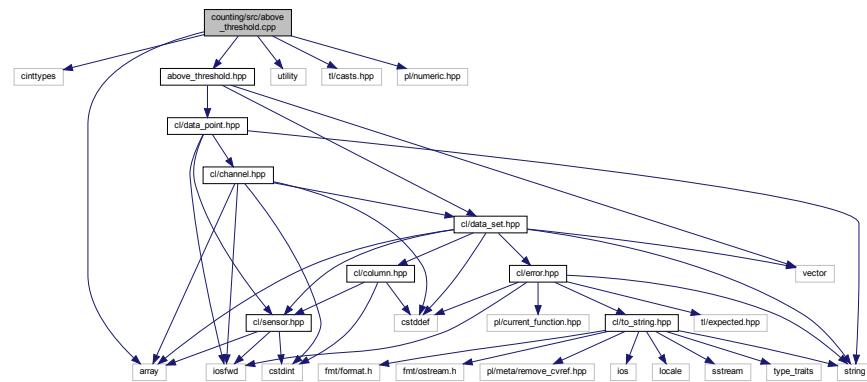
- void `ctg::runAboveThreshold` (`std::ostream &aboveThresholdLogFileStream, const cl::DataSet &dataSet)`

Routine to find the above threshold values.

7.87 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)`
Returns all the data points that are above / below the threshold.

7.87.1 Macro Definition Documentation

7.87.1.1 CL_CHANNEL_X

```
#define CL_CHANNEL_X(
    enm,
    v,
    accessor ) {accessor, cl::Channel::enm},
```

7.87.2 Variable Documentation

7.87.2.1 channel

```
cl::Channel channel
```

Definition at line 18 of file above_threshold.cpp.

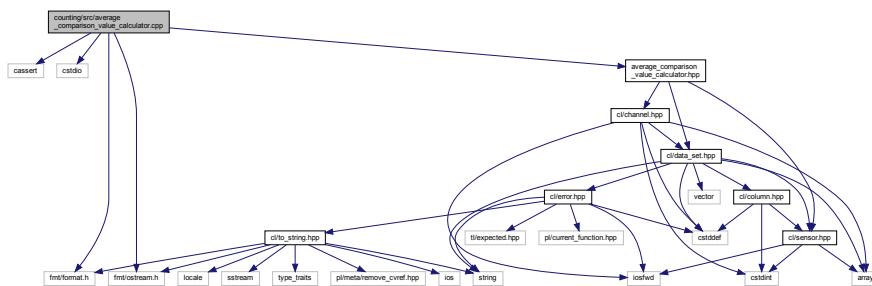
7.87.2.2 channelAccessor

```
cl::DataSet::ChannelAccessor channelAccessor
```

Definition at line 17 of file above_threshold.cpp.

7.88 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

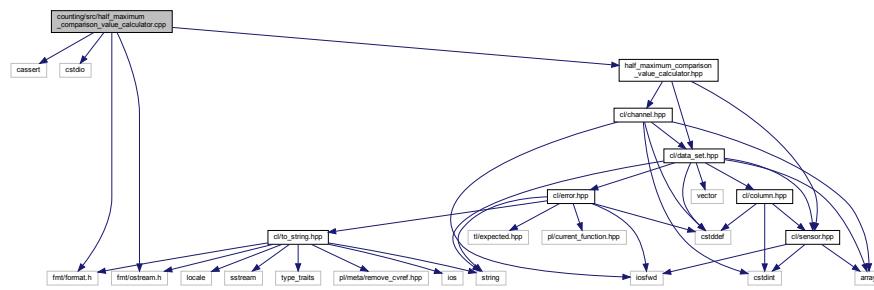
Functions

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

Calculates the average value for the given sensor, channel and data set.

7.89 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`

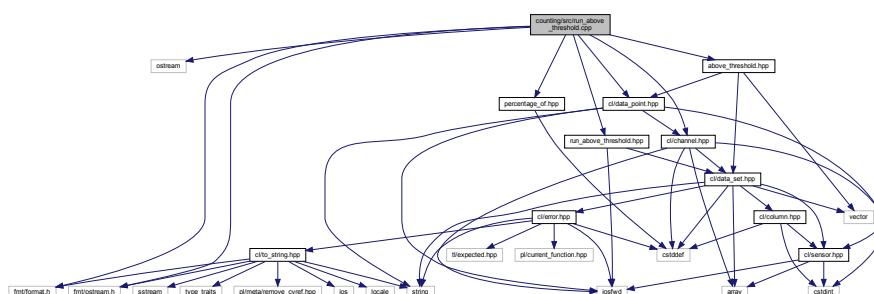
Functions

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

Returns half of the maximum for the given sensor, channel and data set.

7.90 counting/src/run_above_threshold.cpp File Reference

```
#include <ostream>
#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`

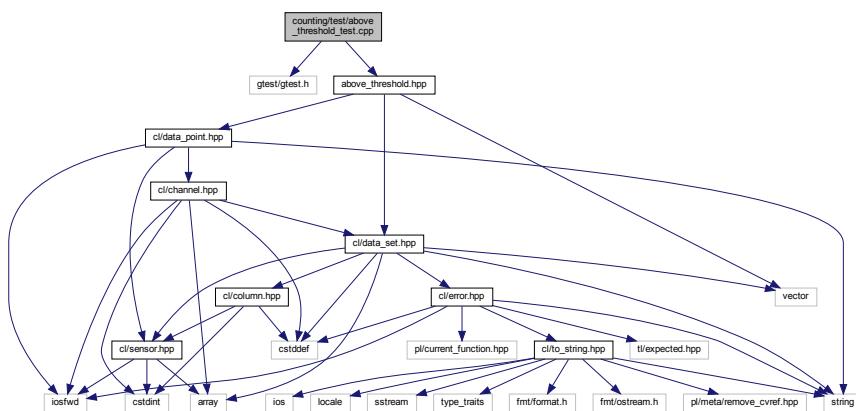
Functions

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

Routine to find the above threshold values.

7.91 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`)

7.91.1 Macro Definition Documentation

7.91.1.1 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.

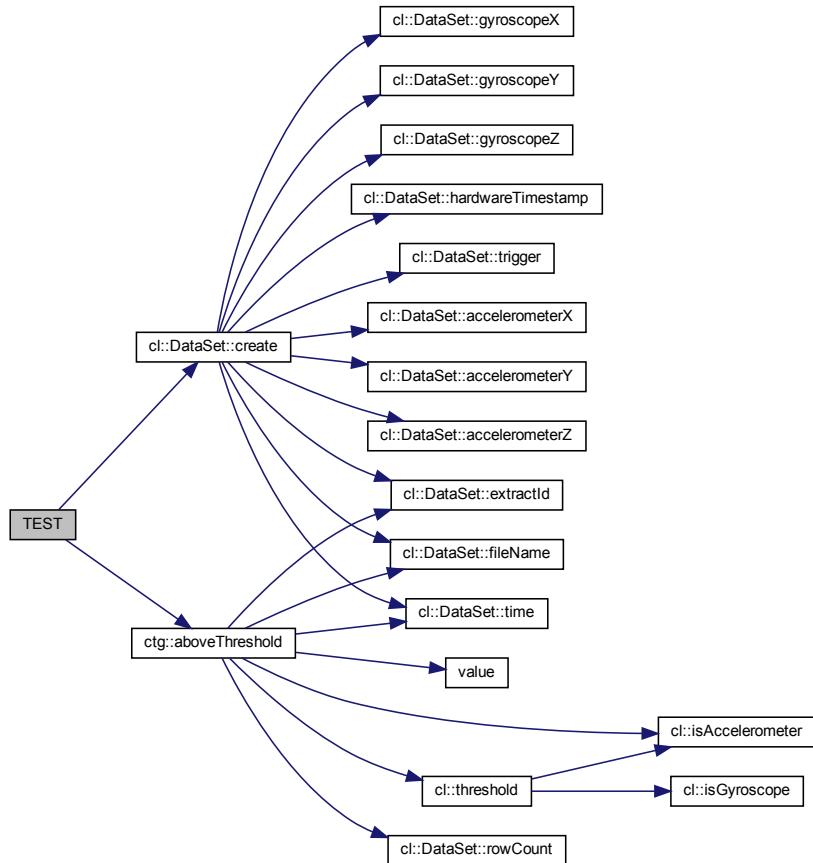
7.91.2 Function Documentation

7.91.2.1 TEST()

```
TEST( aboveThreshold , shouldFindDataPointsIfThereAreAny )
```

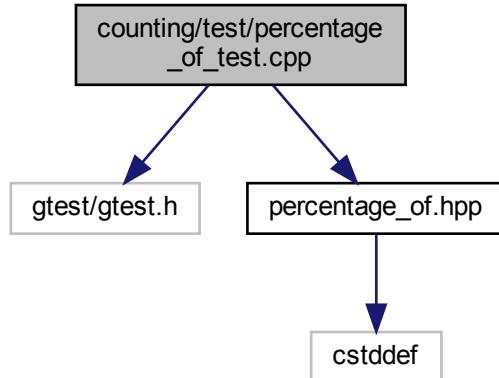
Definition at line 10 of file above_threshold_test.cpp.

Here is the call graph for this function:



7.92 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)`

7.92.1 Macro Definition Documentation

7.92.1.1 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.

7.92.2 Function Documentation

7.92.2.1 TEST()

```
TEST (
    percentageOf ,
    shouldWork )
```

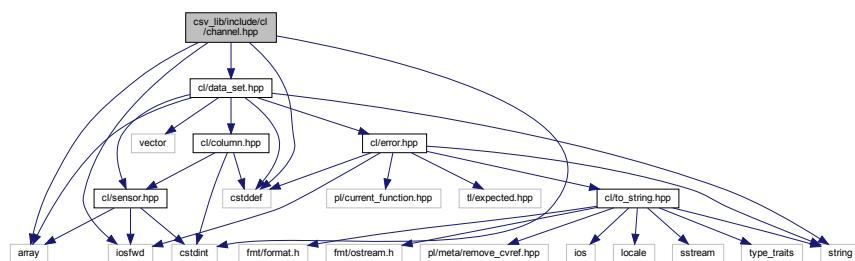
Definition at line 10 of file percentage_of_test.cpp.

Here is the call graph for this function:

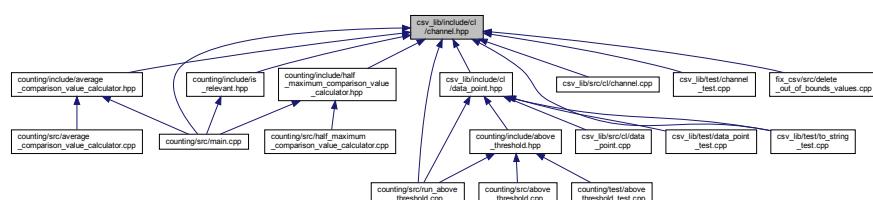


7.93 csv_lib/include/cl/channel.hpp File Reference

```
#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 >`
Maps a Channel to its associated accessor member function pointer of the DataSet type.

Namespaces

- `cl`

Macros

- `#define CL_CHANNEL`
- `#define CL_CHANNEL_X(enum, value, dataSetAccessor) enum = value,`
- `#define CL_CHANNEL_X(enum, value, dataSetAccessor) +1`
- `#define CL_CHANNEL_X(enm, v, a) ::cl::Channel::enm,`
- `#define CL_CHANNEL_X(enum, value, dataSetAccessor)`

Enumerations

- enum `cl::Channel : std::uint64_t { cl::Channel::CL_CHANNEL_X, cl::Channel::CL_CHANNEL }`
Scoped enum for the 6 sensor channels.

Functions

- `DataSet::ChannelAccessor cl::dataSetAccessor (Channel channel)`
Returns the data set accessor for the Channel given.
- `std::ostream & cl::operator<< (std::ostream &os, Channel channel)`
Prints a given Channel to os.
- `bool cl::isAccelerometer (Channel channel)`
Checks whether a given Channel is an accelerometer channel.
- `bool cl::isGyroscope (Channel channel)`
Checks whether a given Channel is a gyroscope channel.
- `long double cl::threshold (Channel channel)`
Returns the threshold value for channel.

Variables

- `constexpr std::size_t cl::channelCount`
The amount of sensor channels (6).
- `constexpr std::array< Channel, channelCount > cl::channels`
An array of the sensor channel enumerators.
- `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}`
Constant for the maximum acceptable accelerometer value (in positive and negative direction).
- `constexpr long double cl::gyroscopeThreshold {1999.99L}`
Constant for the maximum acceptable gyroscope value (in positive and negative direction).

7.93.1 Macro Definition Documentation

7.93.1.1 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.

7.93.1.2 CL_CHANNEL_X [1/4]

```
#define CL_CHANNEL_X(
    enm,
    v,
    a ) ::cl::Channel::enm,
```

Definition at line 55 of file channel.hpp.

7.93.1.3 CL_CHANNEL_X [2/4]

```
#define CL_CHANNEL_X(
    enumerator,
    value,
    dataSetAccessor ) enumerator = value,
```

Definition at line 55 of file channel.hpp.

7.93.1.4 CL_CHANNEL_X [3/4]

```
#define CL_CHANNEL_X(
    enumerator,
    value,
    dataSetAccessor ) +1
```

Definition at line 55 of file channel.hpp.

7.93.1.5 CL_CHANNEL_X [4/4]

```
#define CL_CHANNEL_X(enumerator, value, dataSetAccessor )
```

Value:

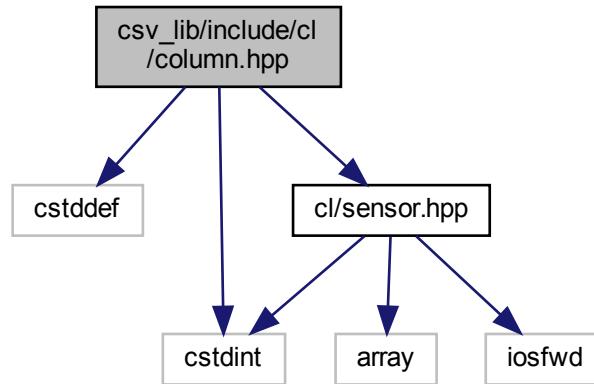
```
template<>
struct data_set_accessor<Channel::enumerator> {
    static constexpr ::cl::DataSet::ChannelAccessor f = dataSetAccessor;
};
```

Definition at line 55 of file channel.hpp.

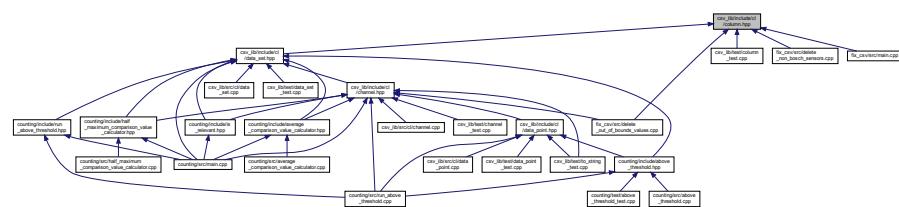
7.94 csv_lib/include/cl/column.hpp File Reference

Contains definitions regarding the columns of the 'old' CSV files, i.e., the CSV files that have not been preprocessed using MATLAB.

```
#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 >`

Column traits for the columns of the old, non-preprocessed CSV files. Includes the index (0 based) of the column and the C++ data type to be used for the cell contents in that column.

Namespaces

- `cl`

Macros

- `#define CL_SPECIALIZE_COL_TRAITS(column, columnType)`

Typedefs

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

Meta function for the type to use for the given Column.

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 }`

A scoped enum for the columns of the old, non-preprocessed CSV files.

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`

Meta function for the 0 based index of the given Column.

7.94.1 Detailed Description

Contains definitions regarding the columns of the 'old' CSV files, i.e., the CSV files that have not been preprocessed using MATLAB.

7.94.2 Macro Definition Documentation

7.94.2.1 CL_SPECIALIZE_COL_TRAITS

```
#define CL_SPECIALIZE_COL_TRAITS( \
    column, \
    columnType )
```

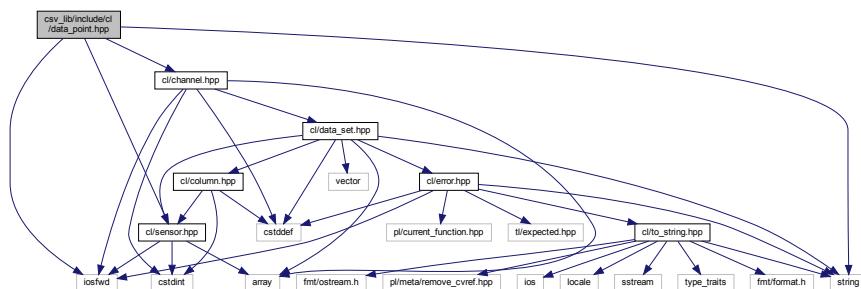
Value:

```
template<> \
struct col_traits<column> { \
    static constexpr std::size_t index = static_cast<std::size_t>(column); \
    using type = columnType; \
}
```

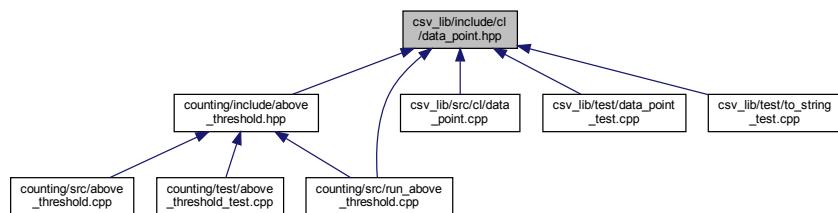
Definition at line 40 of file column.hpp.

7.95 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](#)

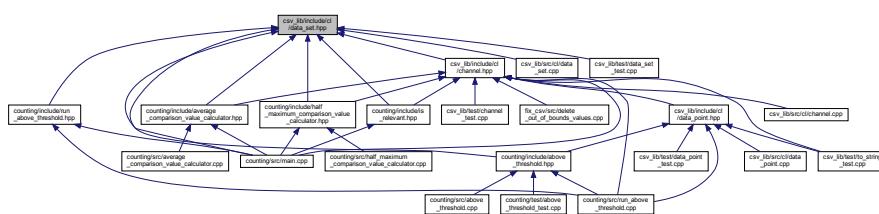
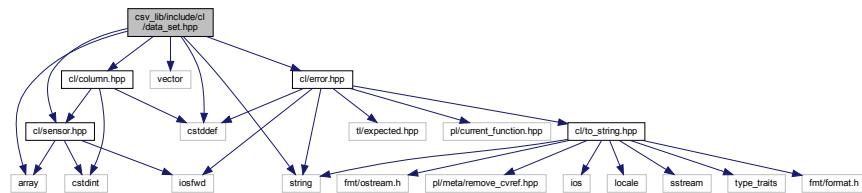
Type to represent a data point in a data set.

Namespaces

- [cl](#)

7.96 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:
```



Classes

- class [cl::DataSet](#)

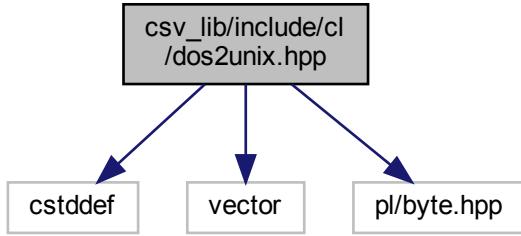
Type to represent a data set.

Namespaces

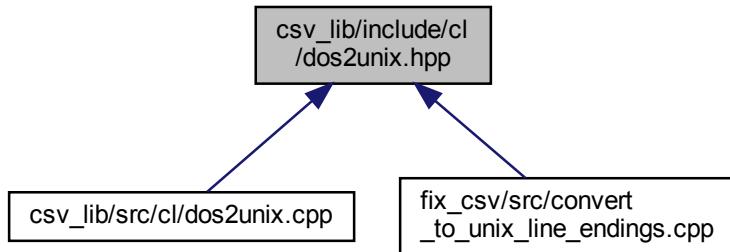
- [cl](#)

7.97 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`

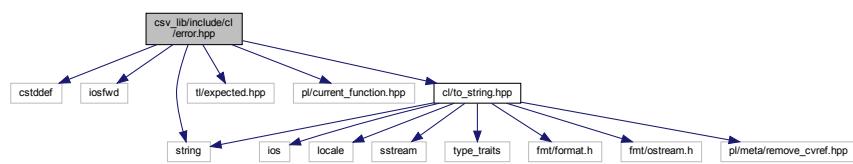
Functions

- `std::vector< pl::byte > cl::dos2unix (const void *p, std::size_t size)`
Converts DOS / Microsoft Windows line endings to UNIX line endings.

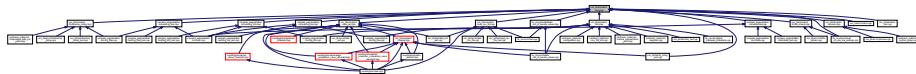
7.98 csv_lib/include/cl/error.hpp File Reference

Exports the `Error` type and related utilities.

```
#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`
Type used to represent different kinds of errors.

Namespaces

- `cl`

Macros

- `#define CL_ERROR_KIND`
X-Macro for the kinds of errors.
- `#define CL_ERROR_KIND_X(kind) kind,`
- `#define CL_UNEXPECTED(kind, message)`
Creates a `cl::Expected` object that indicates an error.

Typedefs

- template<typename Ty >
`using cl::Expected = tl::expected< Ty, Error >`
Type alias for `tl::expected` using `cl::Error` as the error type.

7.98.1 Detailed Description

Exports the `Error` type and related utilities.

7.98.2 Macro Definition Documentation

7.98.2.1 CL_ERROR_KIND

```
#define CL_ERROR_KIND
```

Value:

```
CL_ERROR_KIND_X(Filesystem) \
CL_ERROR_KIND_X(InvalidArgument) \
CL_ERROR_KIND_X(OutOfRange) \
CL_ERROR_KIND_X(Parsing) \
CL_ERROR_KIND_X(Logic) \
CL_ERROR_KIND_X(OperatingSystem)
```

X-Macro for the kinds of errors.

Definition at line 22 of file error.hpp.

7.98.2.2 CL_ERROR_KIND_X

```
#define CL_ERROR_KIND_X(
    kind ) kind,
```

Definition at line 41 of file error.hpp.

7.98.2.3 CL_UNEXPECTED

```
#define CL_UNEXPECTED (
    kind,
    message )
```

Value:

```
::t1::make_unexpected(
    ::cl::Error(kind, __FILE__, PL_CURRENT_FUNCTION, __LINE__, message))
```

Creates a `cl::Expected` object that indicates an error.

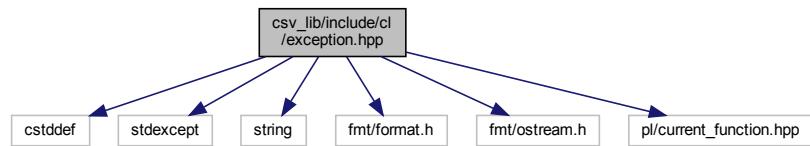
Definition at line 130 of file error.hpp.

7.99 csv_lib/include/cl/exception.hpp File Reference

Exports the `cl::Exception` type and related utility macros.

```
#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`

The exception type for this code base.

Namespaces

- `cl`

Macros

- `#define CL_THROW(what_arg) throw ::cl::Exception { __FILE__, PL_CURRENT_FUNCTION, __LINE__ ← , what_arg }`

Macro to throw a `cl::Exception` given an error message.
- `#define CL_THROW_FMT(fmt_str, ...) CL_THROW(::fmt::format(fmt_str, __VA_ARGS__))`

Convenience macro to throw a `cl::Exception` formatting the error message using `fmt::format`.

7.99.1 Detailed Description

Exports the `cl::Exception` type and related utility macros.

7.99.2 Macro Definition Documentation

7.99.2.1 CL_THROW

```
#define CL_THROW(
    what_arg ) throw ::cl::Exception { __FILE__, PL_CURRENT_FUNCTION, __LINE__←
, what_arg }
```

Macro to throw a `cl::Exception` given an error message.

Definition at line 79 of file exception.hpp.

7.99.2.2 CL_THROW_FMT

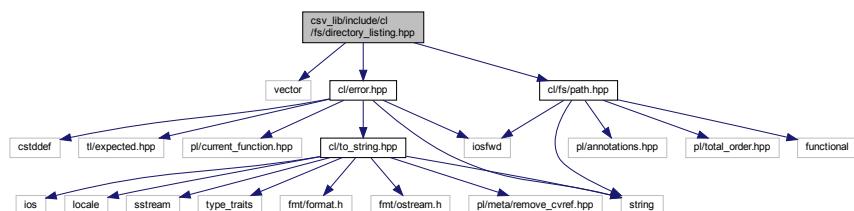
```
#define CL_THROW_FMT(
    fmt_str,
    ...
) CL_THROW(::fmt::format(fmt_str, __VA_ARGS__))
```

Convenience macro to throw a `cl::Exception` formatting the error message using `fmt::format`.

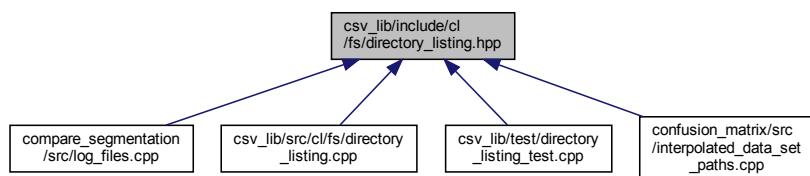
Definition at line 87 of file exception.hpp.

7.100 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

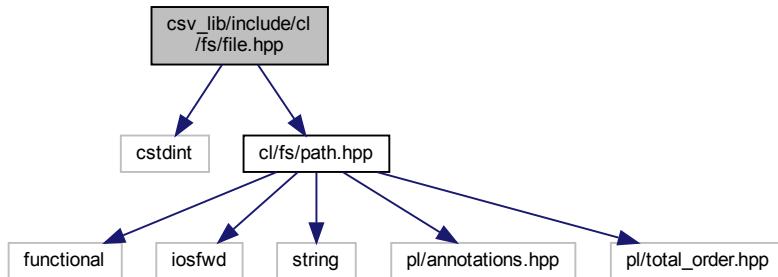
- enum `cl::fs::DirectoryListingOption` { `cl::fs::DirectoryListingOption::None`, `cl::fs::DirectoryListingOption::ExcludeDotAndDotDot` }
- Options for directoryListing.*

Functions

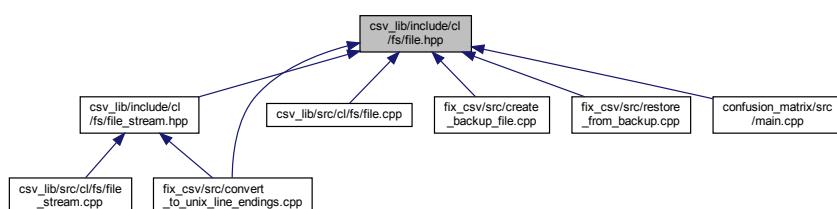
- Expected< std::vector< Path > > `cl::fs::directoryListing` (const Path &directoryPath, DirectoryListingOption directoryListingOption=DirectoryListingOption::ExcludeDotAndDotDot)
- Creates a listing of the contents of a directory.*

7.101 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](#)

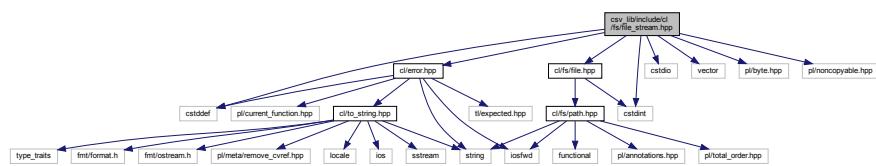
Represents a file.

Namespaces

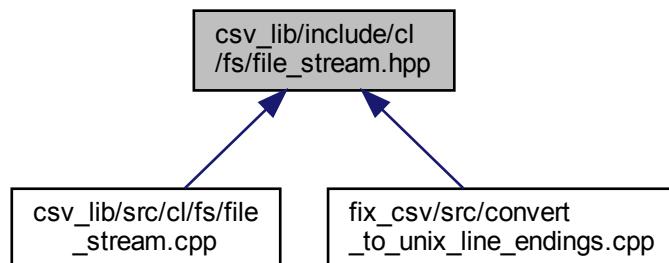
- [cl](#)
- [cl::fs](#)

7.102 csv_lib/include/cl/fs/file_stream.hpp File Reference

```
#include <cstdint>
#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](#)

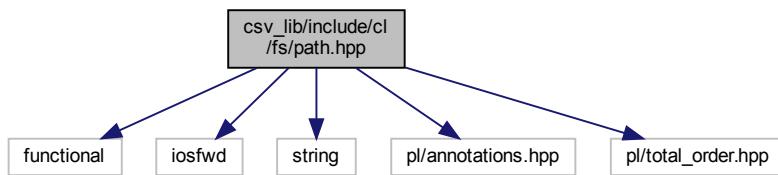
A binary file stream.

Namespaces

- [cl](#)
- [cl::fs](#)

7.103 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

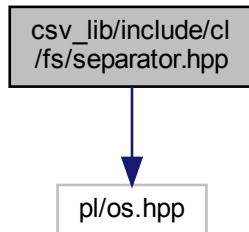
- class [cl::fs::Path](#)
A filesystem path.
- struct [std::hash<::cl::fs::Path >](#)

Namespaces

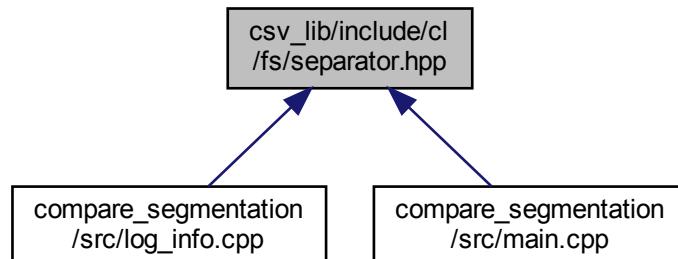
- [cl](#)
- [cl::fs](#)

7.104 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.

7.104.1 Macro Definition Documentation

7.104.1.1 CL_FS_SEPARATOR

```
#define CL_FS_SEPARATOR "\\\"
```

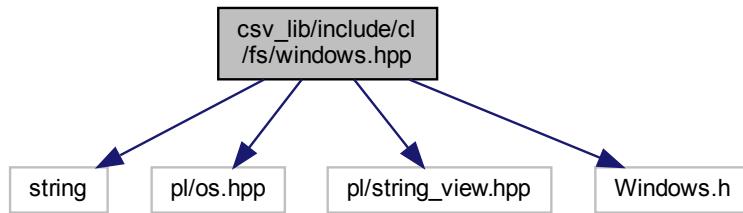
The filesystem separator of the operating system.

Definition at line 11 of file separator.hpp.

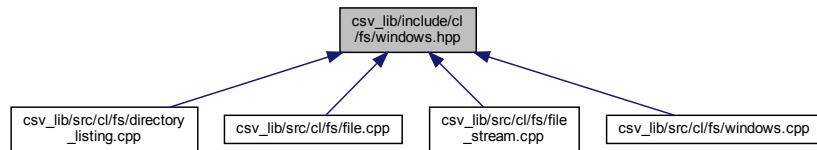
7.105 csv_lib/include/cl/fs/windows.hpp File Reference

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

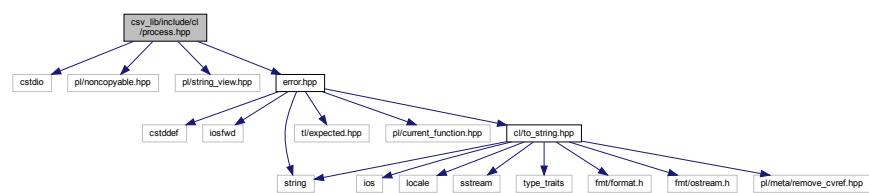
- `std::wstring cl::fs::utf8ToUtf16 (pl::string_view utf8)`
Converts a UTF-8 encoded string to a UTF-16 encoded wstring.
- `std::string cl::fs::utf16ToUtf8 (pl::wstring_view utf16)`
Converts a UTF-16 encoded wide character string to UTF-8 string.
- `std::wstring cl::fs::formatError (DWORD errorCode)`
Formats a WINAPI error code to a UTF-16 encoded wide character string.

7.105.1 Detailed Description

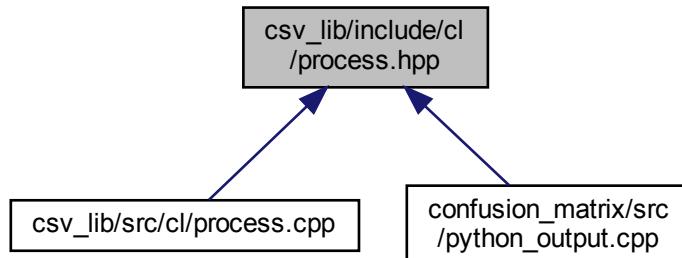
Contains Microsoft Windows specific functions.

7.106 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

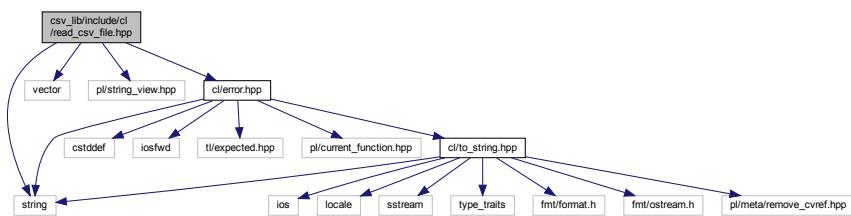
- class [cl::Process](#)
Type representing an operating system process.

Namespaces

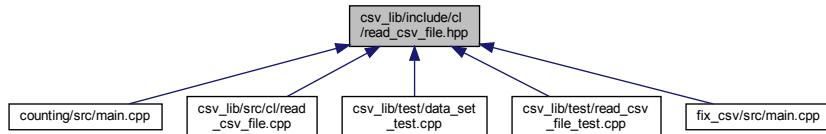
- [cl](#)

7.107 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 }`

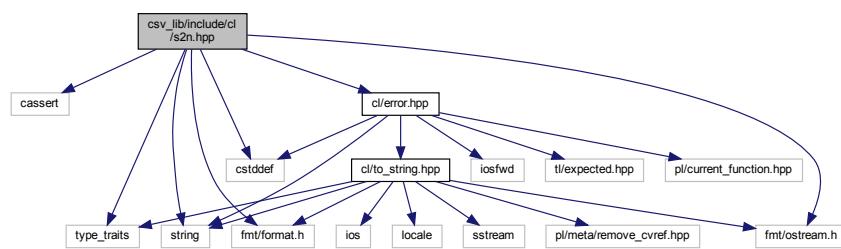
Scoped enum for the CSV file kinds.

Functions

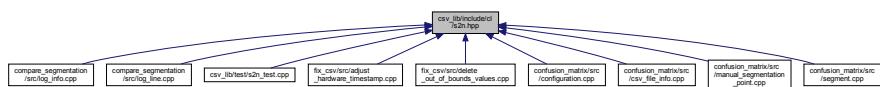
- Expected< std::vector< std::vector< std::string > > > `cl::readCsvFile` (`pl::string_view csvFilePath, std::vector< std::string > *columnNames=nullptr, CsvFileKind csvFileKind=CsvFileKind::Fixed`) noexcept
Reads a CSV file.

7.108 csv_lib/include/cl/s2n.hpp File Reference

```
#include <cassert>
#include <cstddef>
#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`

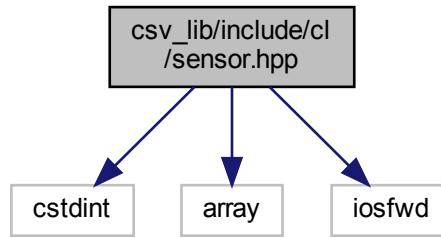
Functions

- template<typename Integer >
`Expected< Integer > cl::s2n (const std::string &str, std::size_t *pos=nullptr, [[maybe_unused]] int base=10)`
Converts a string to an integer type.

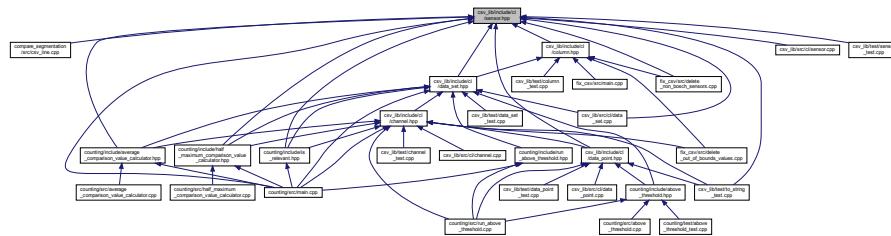
7.109 csv_lib/include/cl/sensor.hpp File Reference

```
#include <cstdint>
#include <array>
```

```
#include <iostream>
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(enum, value) enumerator = value,`
- `#define CL_SENSOR_X(enm, v) ::cl::Sensor::enm,`

Enumerations

- enum `cl::Sensor : std::uint64_t { CL_SENSOR_X, CL_SENSOR }`
Scoped enum for the sensors.

Functions

- `std::ostream & operator<< (std::ostream &os, Sensor sensor)`
Prints a sensor to an ostream.

Variables

- `constexpr std::array< Sensor, 4 > cl::sensors`

The sensors.

7.109.1 Macro Definition Documentation

7.109.1.1 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.

7.109.1.2 CL_SENSOR_X [1/2]

```
#define CL_SENSOR_X(
    enm,
    v ) ::cl::Sensor::enm,
```

Definition at line 19 of file sensor.hpp.

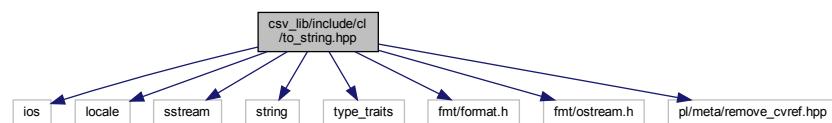
7.109.1.3 CL_SENSOR_X [2/2]

```
#define CL_SENSOR_X(
    enumerator,
    value ) enumerator = value,
```

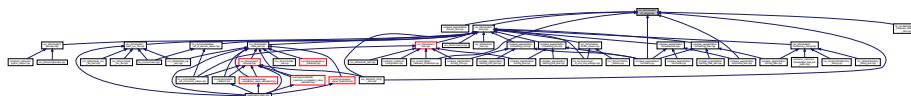
Definition at line 19 of file sensor.hpp.

7.110 csv_lib/include/cl/to_string.hpp File Reference

```
#include <iostream>
#include <locale>
#include <sstream>
#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

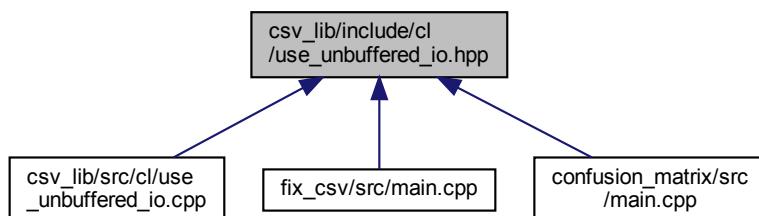
- `cl`

Functions

- template<typename Ty >
`std::string cl::to_string (const Ty &ty)`
Converts something to a string (if possible).

7.111 csv_lib/include/cl/use_unbuffered_io.hpp File Reference

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



Namespaces

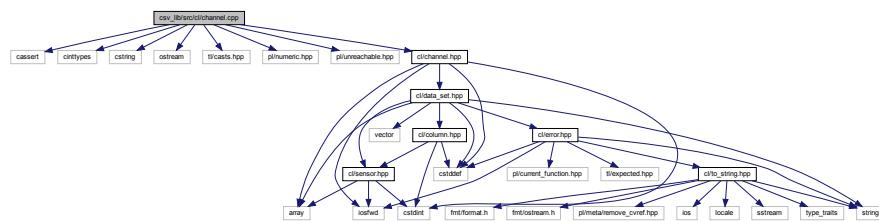
- `cl`

Functions

- `void cl::useUnbufferedIo ()`
Routine to activate unbuffered I/O.

7.112 csv_lib/src/cl/channel.cpp File Reference

```
#include <cassert>
#include <cinttypes>
#include <cstring>
#include <iostream>
#include <tl/casts.hpp>
#include <pl/numeric.hpp>
#include <pl/unreachable.hpp>
#include "cl/channel.hpp"
Include dependency graph for channel.cpp:
```



Namespaces

- `cl`

Macros

- `#define CL_CHANNEL_X(emn, v, acc) case Channel::emn: return data_set_accessor_v<Channel::emn>;`
- `#define CL_CHANNEL_X(enum, value, dataSetAccessor) case Channel::enum: return os << #enum;`

Functions

- `DataSet::ChannelAccessor cl::dataSetAccessor (Channel channel)`
Returns the data set accessor for the Channel given.
- `std::ostream & cl::operator<< (std::ostream &os, Channel channel)`
Prints a given Channel to os.
- `bool cl::isAccelerometer (Channel channel)`
Checks whether a given Channel is an accelerometer channel.
- `bool cl::isGyroscope (Channel channel)`
Checks whether a given Channel is a gyroscope channel.
- `long double cl::threshold (Channel channel)`
Returns the threshold value for channel.

7.112.1 Macro Definition Documentation

7.112.1.1 CL_CHANNEL_X [1/2]

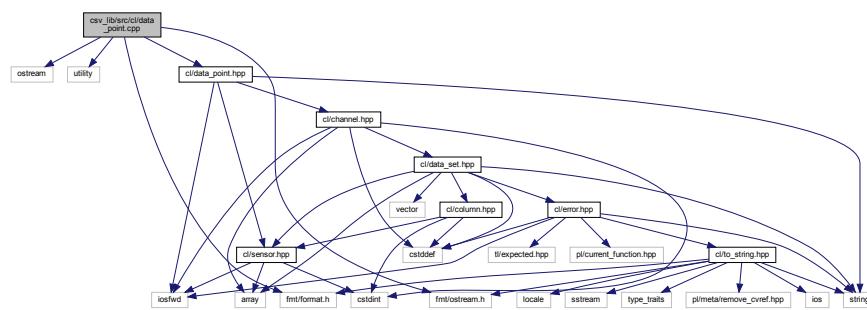
```
#define CL_CHANNEL_X(
    enm,
    v,
    acc ) case Channel::enm: return data_set_accessor_v<Channel::enm>;
```

7.112.1.2 CL_CHANNEL_X [2/2]

```
#define CL_CHANNEL_X(
    enumerator,
    value,
    dataSetAccessor ) case Channel::enumerator: return os << #enumerator;
```

7.113 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()`

7.113.1 Function Documentation

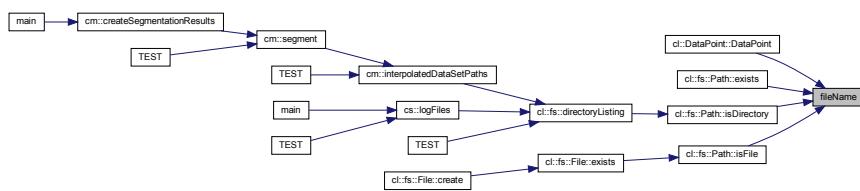
7.113.1.1 `channel()`

```
dataPoint dataPoint dataPoint channel ( )
```

7.113.1.2 `fileName()`

```
dataPoint fileName ( )
```

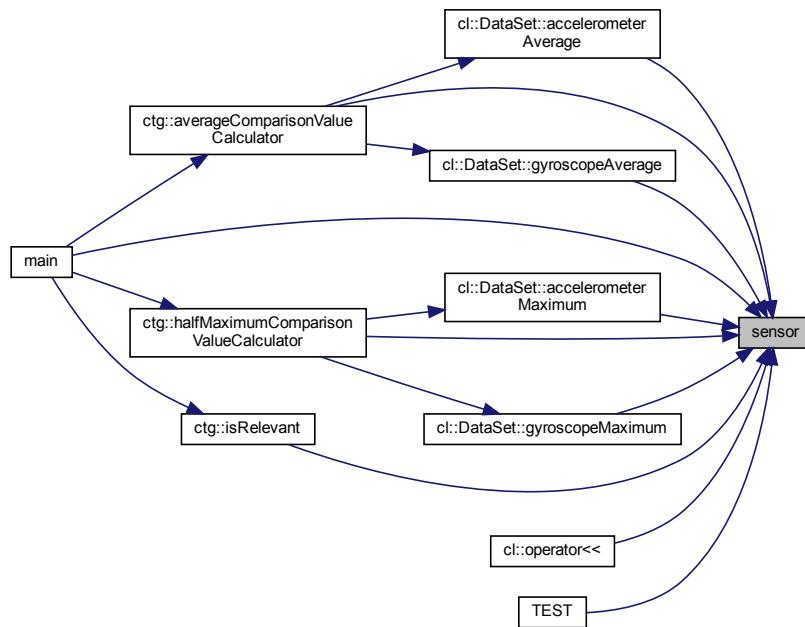
Here is the caller graph for this function:



7.113.1.3 `sensor()`

```
dataPoint dataPoint dataPoint sensor ( )
```

Here is the caller graph for this function:



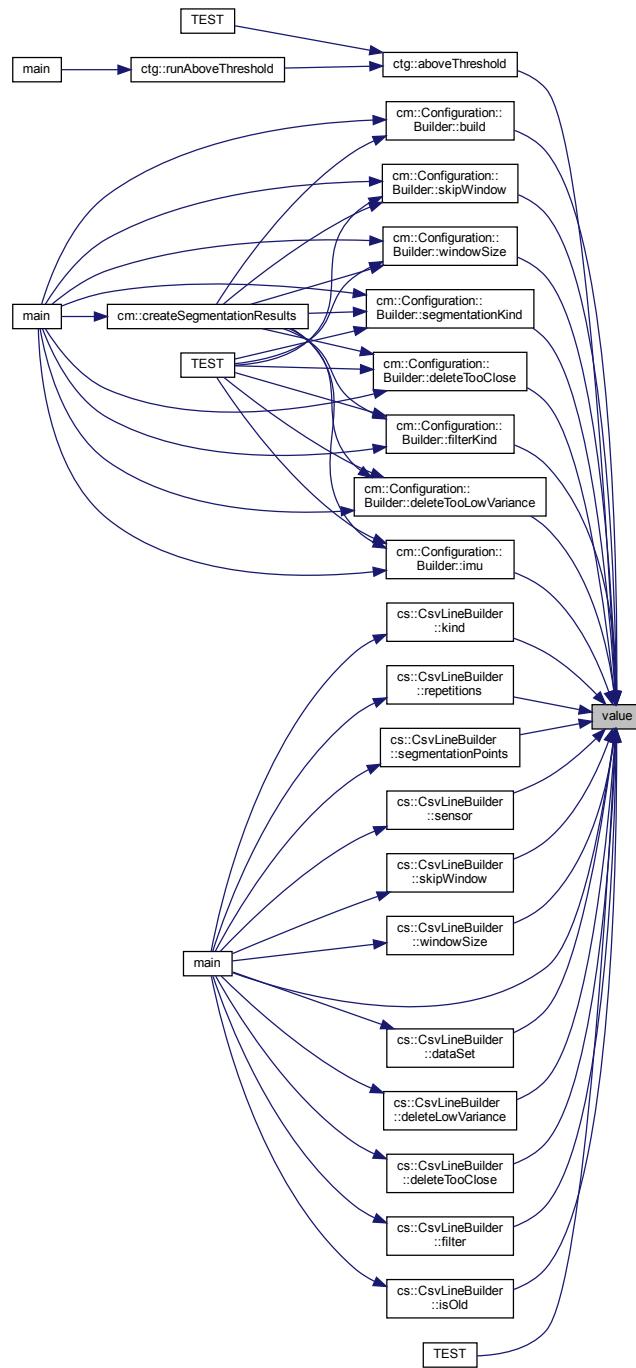
7.113.1.4 `time()`

```
dataPoint dataPoint time ( )
```

7.113.1.5 `value()`

```
dataPoint dataPoint dataPoint dataPoint dataPoint value ( )
```

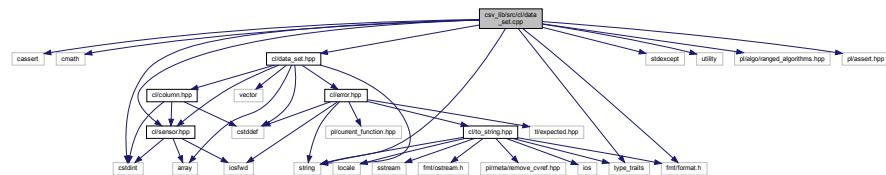
Here is the caller graph for this function:



7.114 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:
```

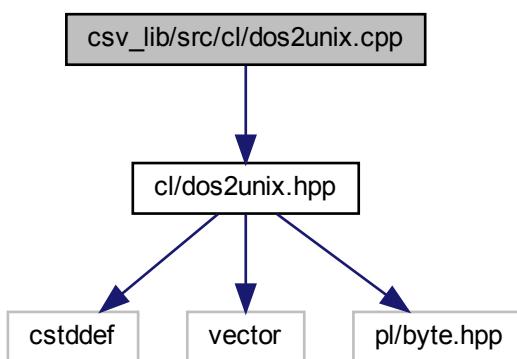


Namespaces

- `cl`

7.115 csv_lib/src/cl/dos2unix.cpp File Reference

```
#include "cl/dos2unix.hpp"
Include dependency graph for dos2unix.cpp:
```



Namespaces

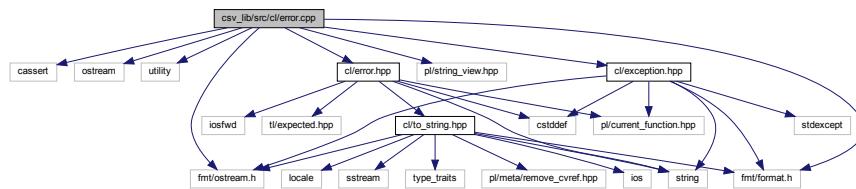
- `cl`

Functions

- std::vector< pl::byte > [cl::dos2unix](#) (const void *p, std::size_t size)
Converts DOS / Microsoft Windows line endings to UNIX line endings.

7.116 csv_lib/src/cl/error.cpp File Reference

```
#include <cassert>
#include <ostream>
#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)`

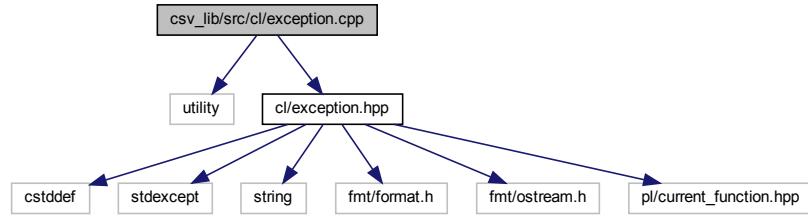
7.116.1 Macro Definition Documentation

7.116.1.1 CL_ERROR_KIND_X

```
#define CL_ERROR_KIND_X(
    kind ) case Error::kind: return #kind;
```

7.117 csv_lib/src/cl/exception.cpp File Reference

```
#include <utility>
#include "cl/exception.hpp"
Include dependency graph for exception.cpp:
```

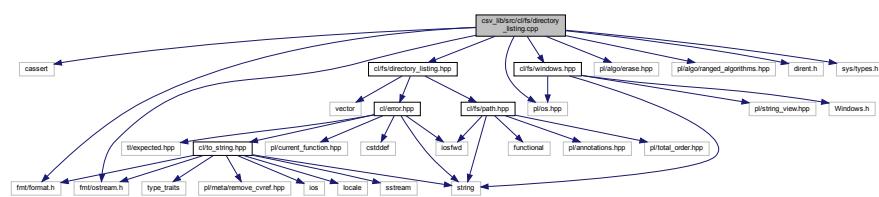


Namespaces

- `cl`

7.118 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`

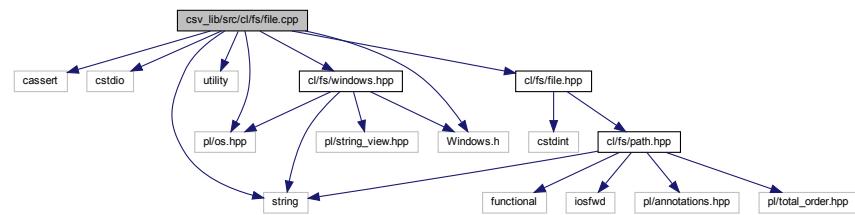
Functions

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

Creates a listing of the contents of a directory.

7.119 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:
```

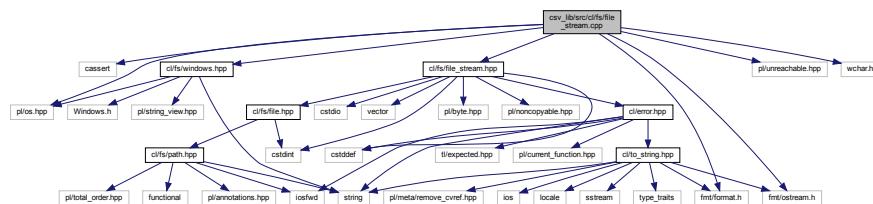


Namespaces

- [cl](#)
- [cl::fs](#)

7.120 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.h>
#include <fmt/ostream.h>
#include "cl/fs/file_stream.hpp"
Include dependency graph for file_stream.cpp:
```

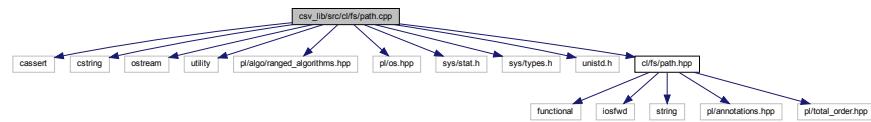


Namespaces

- [cl](#)
- [cl::fs](#)

7.121 csv_lib/src/cl/fs/path.cpp File Reference

```
#include <cassert>
#include <cstring>
#include <iostream>
#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](#)

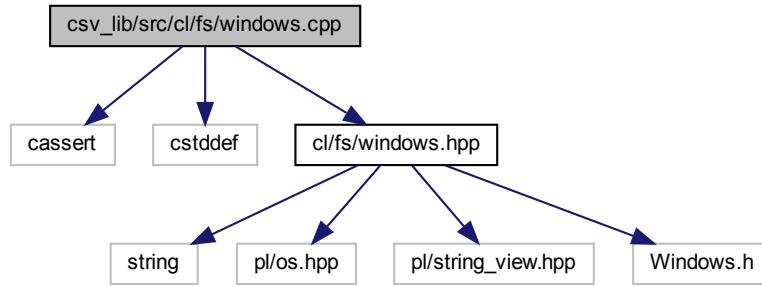
Functions

- `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`

7.122 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

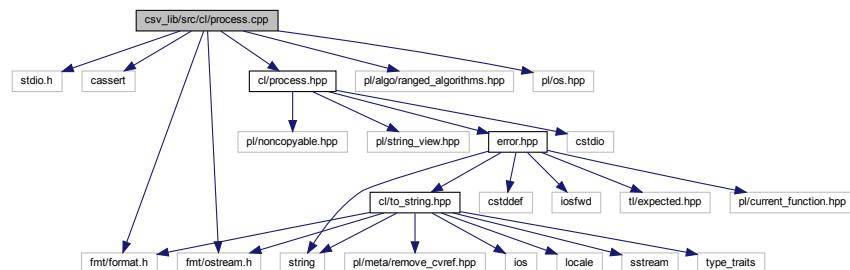
- `cl`
- `cl::fs`

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.

7.123 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:
```

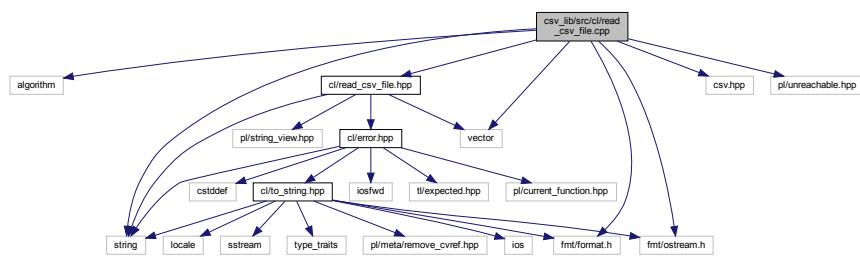


Namespaces

- [cl](#)

7.124 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](#)

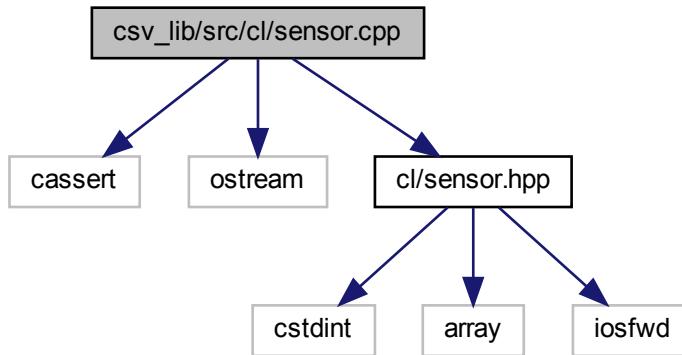
Functions

- `Expected< std::vector< std::vector< std::string > > > cl::readCsvFile(pl::string_view csvFilePath, std::vector< std::string > *columnNames=nullptr, CsvFileKind csvFileKind=CsvFileKind::Fixed) noexcept`
Reads a CSV file.

7.125 csv_lib/src/cl/sensor.cpp File Reference

```
#include <cassert>
#include <iostream>
```

```
#include "cl/sensor.hpp"
Include dependency graph for sensor.cpp:
```



Namespaces

- `cl`

Macros

- `#define CL_SENSOR_X(enum, value) case Sensor::enum: return os << #enum;`

Functions

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

Prints a sensor to an ostream.

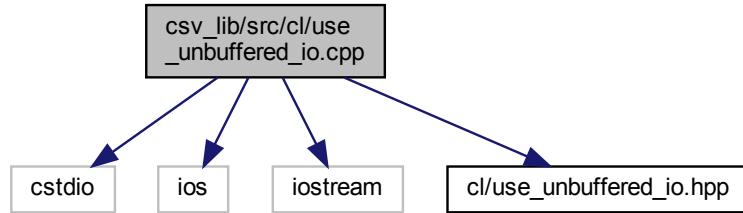
7.125.1 Macro Definition Documentation

7.125.1.1 CL_SENSOR_X

```
#define CL_SENSOR_X(
    enum,
    value ) case Sensor::enum: return os << #enum;
```

7.126 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

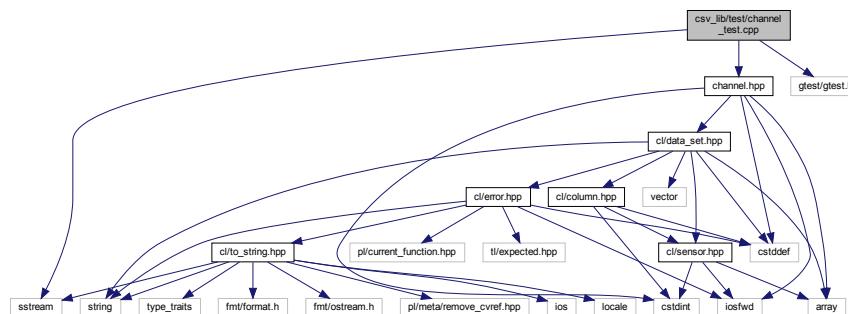
- `cl`

Functions

- `void cl::useUnbufferedIo ()`
Routine to activate unbuffered I/O.

7.127 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 \(channel, shouldMapToCorrectDataSetAccessors\)](#)

7.127.1 Function Documentation

7.127.1.1 TEST() [1/4]

```
TEST (
    channel ,
    shouldHaveCorrectCount )
```

Definition at line 7 of file channel_test.cpp.

7.127.1.2 TEST() [2/4]

```
TEST (
    channel ,
    shouldHaveCorrectValues )
```

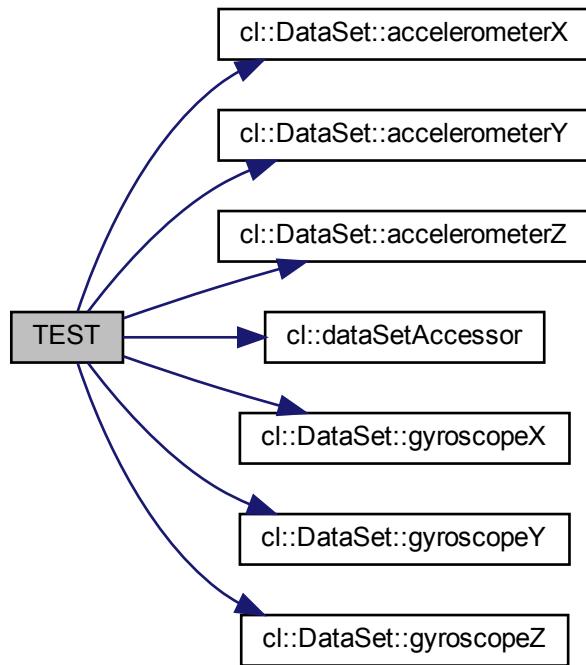
Definition at line 9 of file channel_test.cpp.

7.127.1.3 TEST() [3/4]

```
TEST (
    channel ,
    shouldMapToCorrectDataSetAccessors )
```

Definition at line 35 of file channel_test.cpp.

Here is the call graph for this function:



7.127.1.4 TEST() [4/4]

```
TEST (
    channel ,
    shouldPrintCorrectly )
```

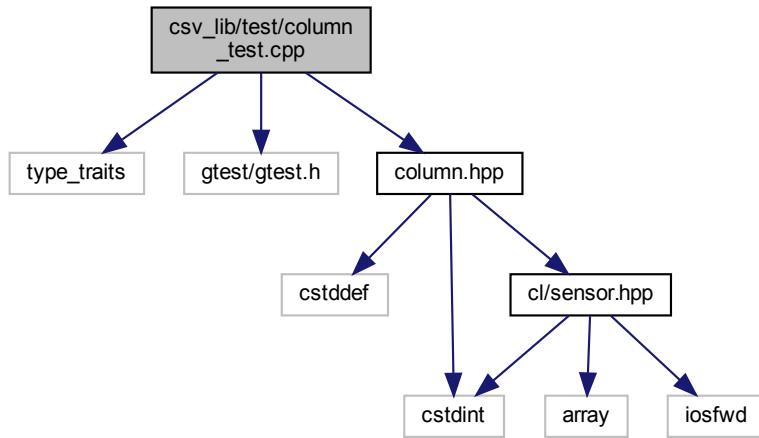
Definition at line 19 of file channel_test.cpp.

7.128 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`)

7.128.1 Function Documentation

7.128.1.1 TEST() [1/2]

```
TEST (
    column ,
    shouldHaveCorrectColumnType )
```

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

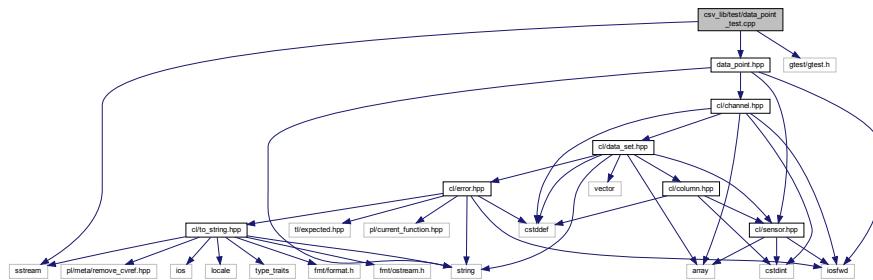
7.128.1.2 TEST() [2/2]

```
TEST (
    column ,
    shouldHaveCorrectIndex )
```

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

7.129 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`

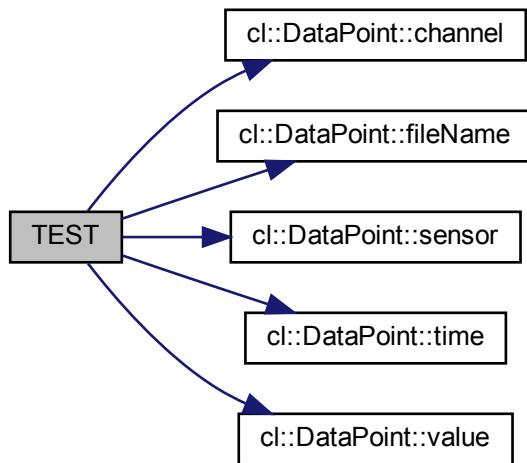
7.129.1 Function Documentation

7.129.1.1 TEST() [1/2]

```
TEST (
    DataPoint ,
    shouldGetValuesCorrectly )
```

Definition at line 23 of file `data_point_test.cpp`.

Here is the call graph for this function:



7.129.1.2 TEST() [2/2]

```
TEST (
    DataPoint ,
    shouldPrintCorrectly )
```

Definition at line 14 of file data_point_test.cpp.

7.129.2 Variable Documentation

7.129.2.1 dp

```
const cl::DataPoint dp
```

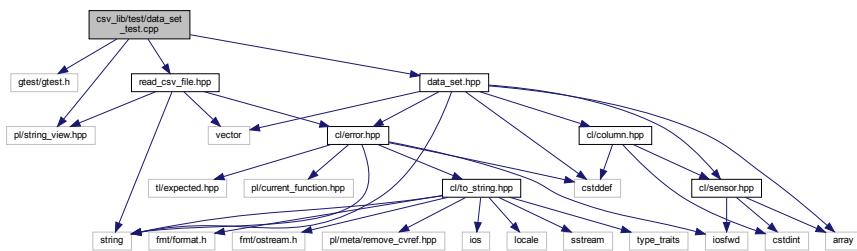
Initial value:

```
{
    "file.csv",
    0.01,
    cl::Sensor::Chest,
    cl::Channel::AccelerometerX,
    50.01}
```

Definition at line 7 of file data_point_test.cpp.

7.130 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)`

7.130.1 Macro Definition Documentation

7.130.1.1 EXPECT_LONG_DOUBLE_EQ

```
#define EXPECT_LONG_DOUBLE_EQ(
    a,
    b ) EXPECT_DOUBLE_EQ(static_cast<double>(a), static_cast<double>(b))
```

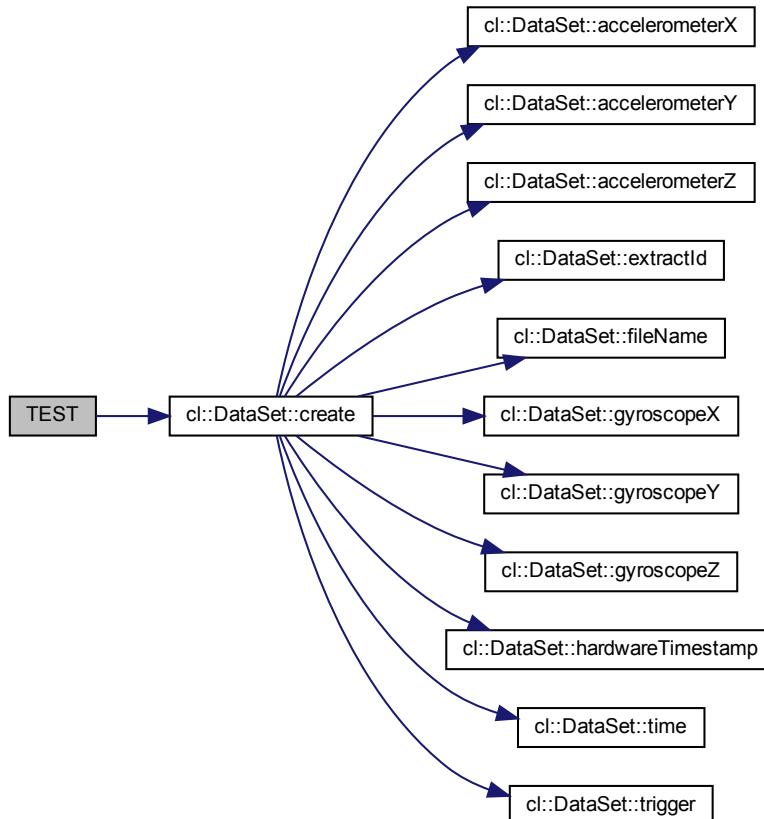
7.130.2 Function Documentation

7.130.2.1 TEST() [1/4]

```
TEST (
    DataSet ,
    shouldBeAbleToCreateFromValidData )
```

Definition at line 17 of file data_set_test.cpp.

Here is the call graph for this function:

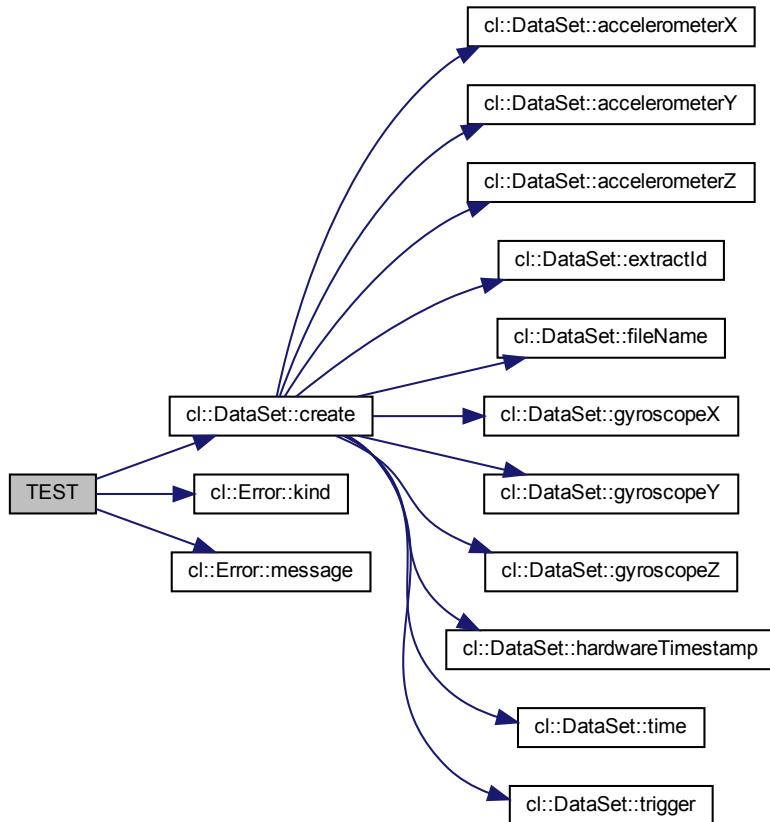


7.130.2.2 TEST() [2/4]

```
TEST (
    DataSet ,
    shouldNotBeAbleToCreateFromEmptyMatrix )
```

Definition at line 68 of file data_set_test.cpp.

Here is the call graph for this function:

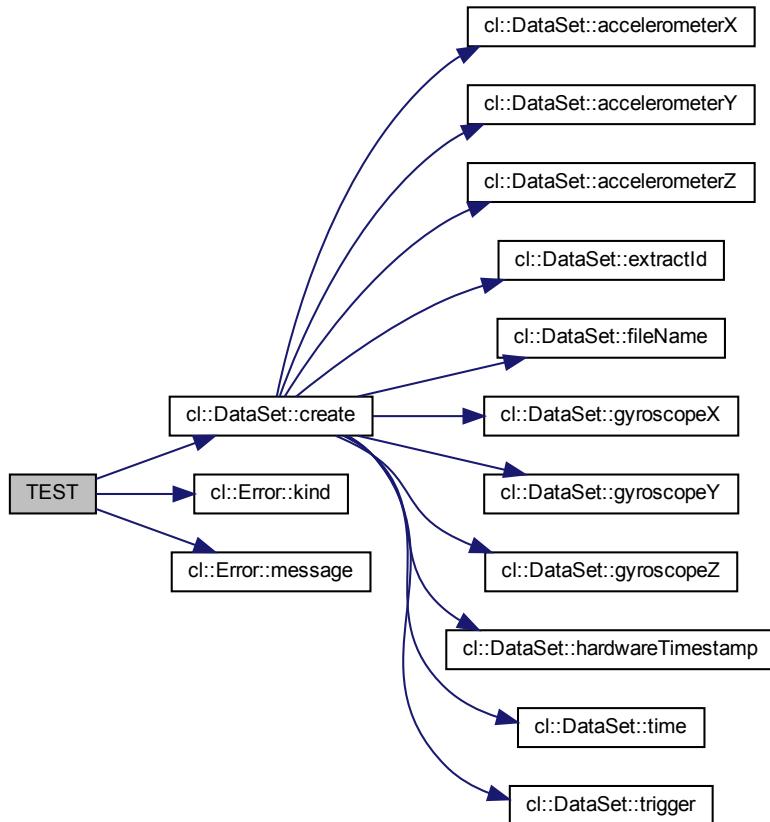


7.130.2.3 TEST() [3/4]

```
TEST (
    DataSet ,
    shouldNotBeAbleToCreateFromInvalidData )
```

Definition at line 108 of file `data_set_test.cpp`.

Here is the call graph for this function:

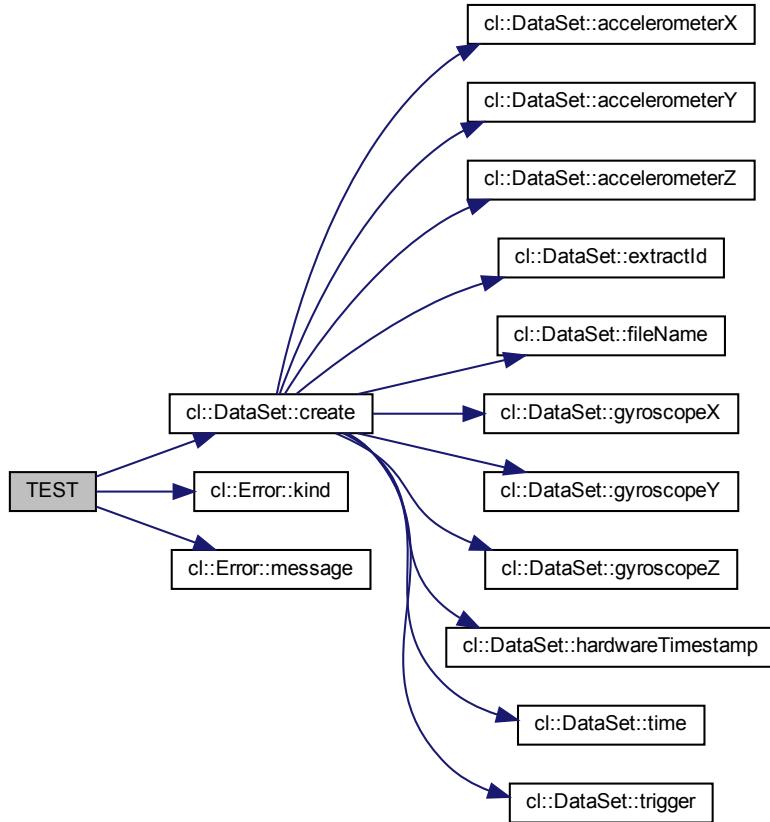


7.130.2.4 TEST() [4/4]

```
TEST (
    DataSet ,
    shouldNotBeAbleToCreateFromJaggedMatrix )
```

Definition at line 80 of file data_set_test.cpp.

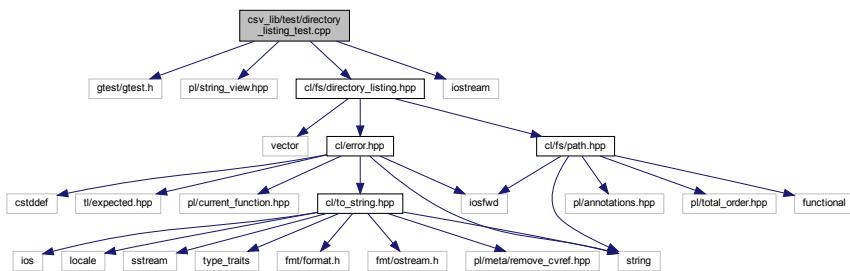
Here is the call graph for this function:



7.131 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)`

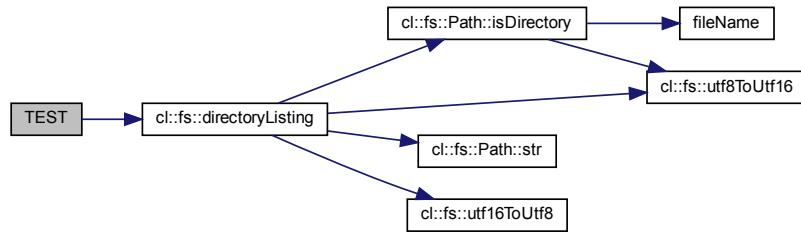
7.131.1 Function Documentation

7.131.1.1 TEST() [1/3]

```
TEST (
    directoryListing ,
    shouldFindFiles )
```

Definition at line 13 of file directory_listing_test.cpp.

Here is the call graph for this function:

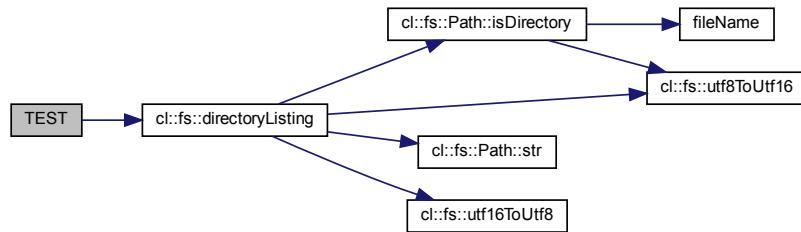


7.131.1.2 TEST() [2/3]

```
TEST (
    directoryListing ,
    shouldFindFilesWithDotAndDotDot )
```

Definition at line 28 of file directory_listing_test.cpp.

Here is the call graph for this function:

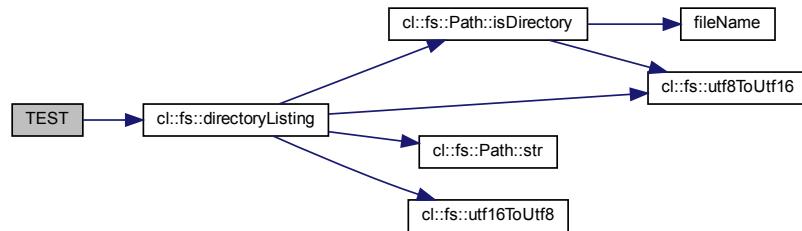


7.131.1.3 TEST() [3/3]

```
TEST (
    directoryListing ,
    shouldReturnErrorWhenPathDoesNotExist )
```

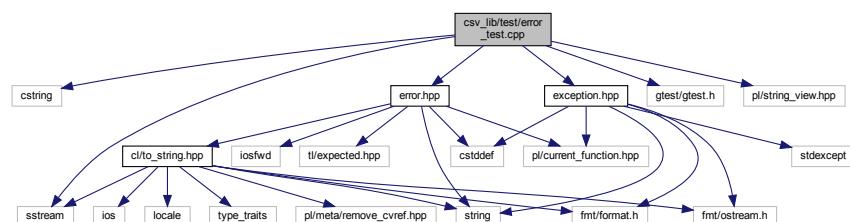
Definition at line 46 of file directory_listing_test.cpp.

Here is the call graph for this function:



7.132 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 \(error, shouldReturnValues\)](#)
- [TEST \(error, shouldThrowExceptionWhenRaisesCalled\)](#)
- [TEST \(error, shouldCreateExpectedWithUnexpected\)](#)

Variables

- const `cl::Error error`

7.132.1 Function Documentation

7.132.1.1 TEST() [1/4]

```
TEST (
    error ,
    shouldCreateExpectedWithUnexpected )
```

Definition at line 59 of file error_test.cpp.

7.132.1.2 TEST() [2/4]

```
TEST (
    error ,
    shouldPrint )
```

Definition at line 19 of file error_test.cpp.

7.132.1.3 TEST() [3/4]

```
TEST (
    error ,
    shouldReturnValues )
```

Definition at line 29 of file error_test.cpp.

7.132.1.4 TEST() [4/4]

```
TEST (
    error ,
    shouldThrowExceptionWhenRaiseIsCalled )
```

Definition at line 37 of file error_test.cpp.

7.132.2 Variable Documentation

7.132.2.1 error

```
const cl::Error error
```

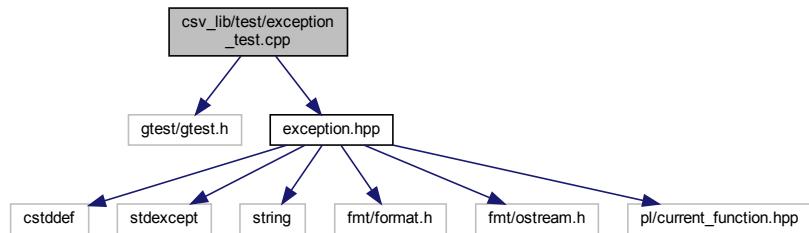
Initial value:

```
{
    cl::Error::Filesystem,
    "test_file.cpp",
    "bad_function",
    48,
    "Couldn't initialize the flux capacitor."}
```

Definition at line 12 of file error_test.cpp.

7.133 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)

7.133.1 Function Documentation

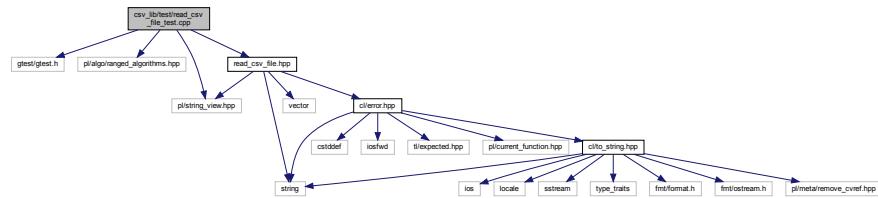
7.133.1.1 TEST()

```
TEST (
    exception ,
    shouldWork )
```

Definition at line 5 of file exception_test.cpp.

7.134 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`)

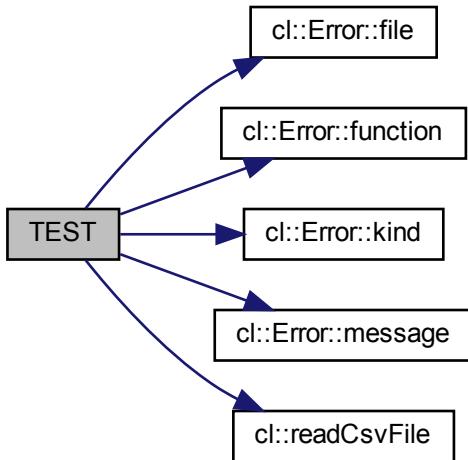
7.134.1 Function Documentation

7.134.1.1 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:

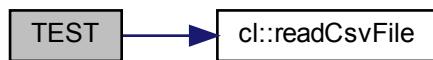


7.134.1.2 TEST() [2/2]

```
TEST (
    readCsvFile ,
    shouldReadCsvFile )
```

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

Here is the call graph for this function:

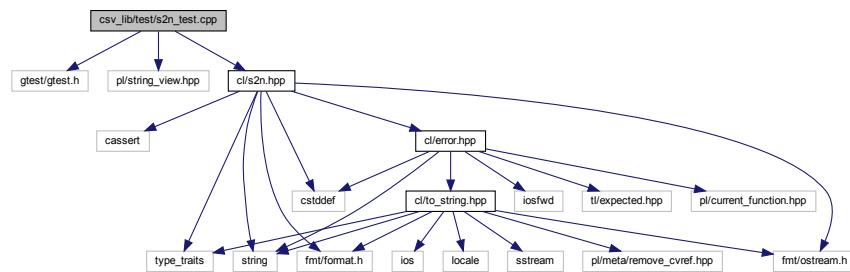


7.135 csv_lib/test/s2n_test.cpp File Reference

```
#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)

7.135.1 Function Documentation

7.135.1.1 TEST() [1/3]

```
TEST (
    s2n ,
    shouldReturnInvalidArgumentErrorIfInputIsInvalid )
```

Definition at line 21 of file s2n_test.cpp.

7.135.1.2 TEST() [2/3]

```
TEST (
    s2n ,
    shouldReturnOutOfRangeErrorIfInputIsOutOfRange )
```

Definition at line 29 of file s2n_test.cpp.

7.135.1.3 TEST() [3/3]

```
TEST (
    s2n ,
    shouldWork )
```

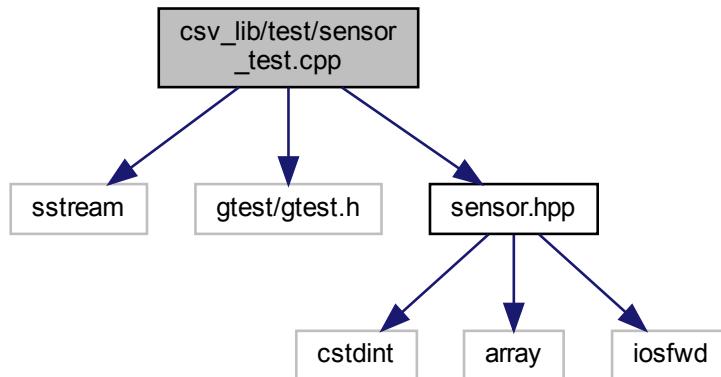
Definition at line 7 of file s2n_test.cpp.

Here is the call graph for this function:



7.136 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 \(sensor, shouldPrintCorrely\)](#)

7.136.1 Function Documentation

7.136.1.1 TEST() [1/2]

```
TEST (
    sensor ,
    shouldHaveCorrectValues )
```

Definition at line 7 of file sensor_test.cpp.

7.136.1.2 TEST() [2/2]

```
TEST (
    sensor ,
    shouldPrintCorrectly )
```

Definition at line 15 of file sensor_test.cpp.

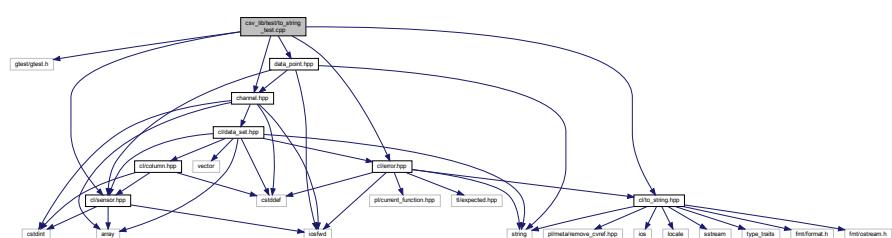
Here is the call graph for this function:



7.137 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)

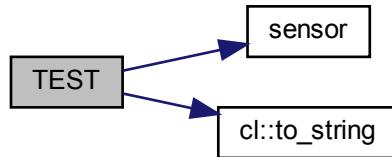
7.137.1 Function Documentation

7.137.1.1 TEST()

```
TEST (
    to_string ,
    test )
```

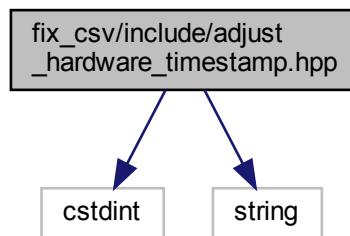
Definition at line 9 of file to_string_test.cpp.

Here is the call graph for this function:

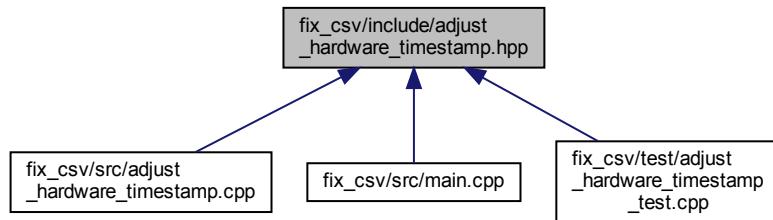


7.138 fix_csv/include/adjust_hw_timestamp.hpp File Reference

```
#include <cstdint>
#include <string>
Include dependency graph for adjust_hw_timestamp.hpp:
```



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



Namespaces

- `fmc`

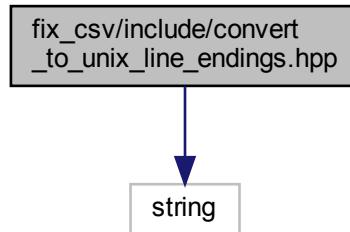
Functions

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

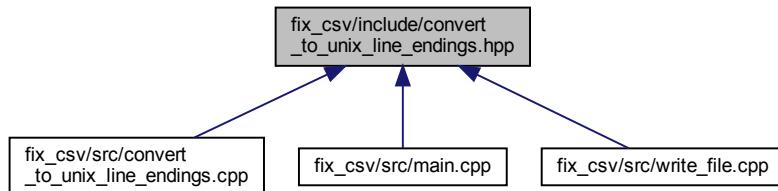
Adjust the hardware timestamp.

7.139 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

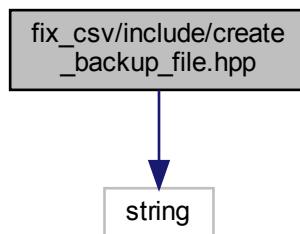
- bool `fmc::convertToUnixLineEndings (const std::string &csvPath)`

Convert a CSV file to UNIX line endings.

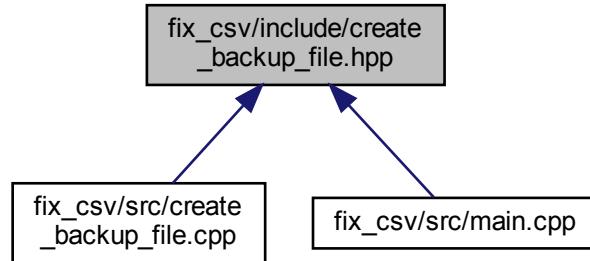
7.140 fix_csv/include/create_backup_file.hpp File Reference

```
#include <string>
```

Include dependency graph for `create_backup_file.hpp`:



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



Namespaces

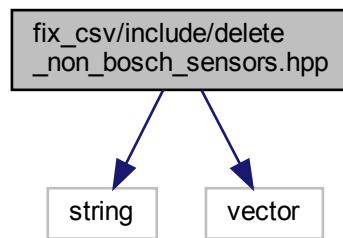
- [fmc](#)

Functions

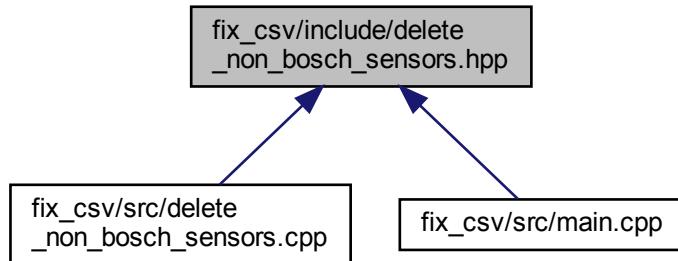
- `bool fmc::createBackupFile (const std::string &csvFilePath, const std::string &backupFilePath)`
Creates a backup of a file.

7.141 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](#)

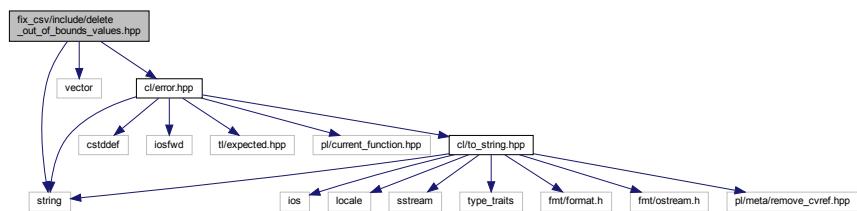
Functions

- void [fmc::deleteNonBoschSensors](#) (std::vector< std::vector< std::string >> *data)

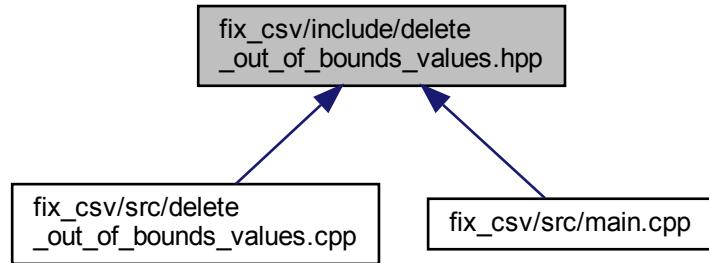
Routine to delete anything that isn't a Bosch sensor.

7.142 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

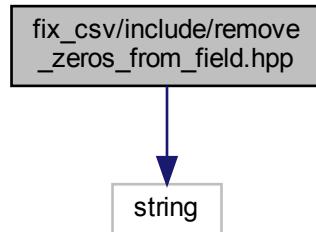
- fmc

Functions

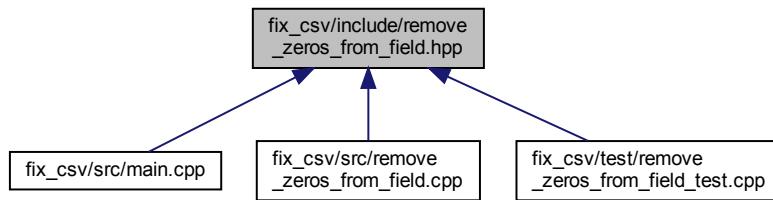
- `cl::Expected< void > fmc::deleteOutOfBoundsValues (std::vector< std::vector< std::string >> *data)`
Deletes out of bounds values from the raw CSV file read.

7.143 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`

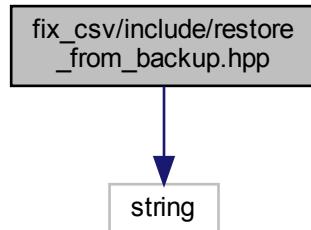
Functions

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

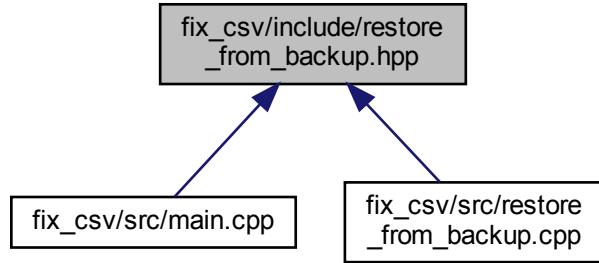
Deletes extraneous zeros from a cell value.

7.144 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

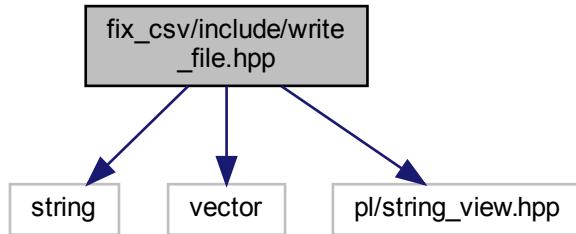
- [fmc](#)

Functions

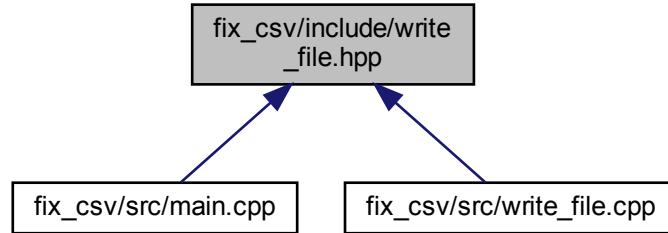
- bool [fmc::restoreFromBackup](#) (const std::string &csvFilePath, const std::string &backupFilePath)
Restore a file from a previously created backup file.

7.145 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](#)

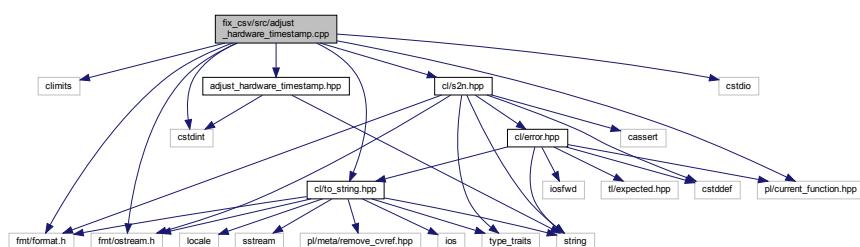
Functions

- `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)`

Writes a 'fixed' CSV file.

7.146 fix_csv/src/adjust_hardware_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_hardware_timestamp.hpp"
Include dependency graph for adjust_hardware_timestamp.cpp:
```



Namespaces

- [fmc](#)

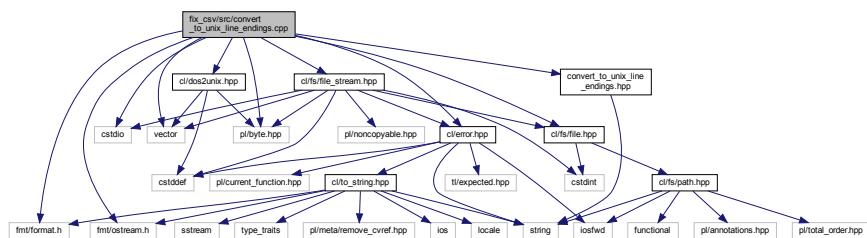
Functions

- void [fmc::adjustHardwareTimestamp](#) (std::string *cellContent, const std::string &nextRowHardwareTimestamp, std::uint64_t *overflowCount)

Adjust the hardware timestamp.

7.147 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](#)

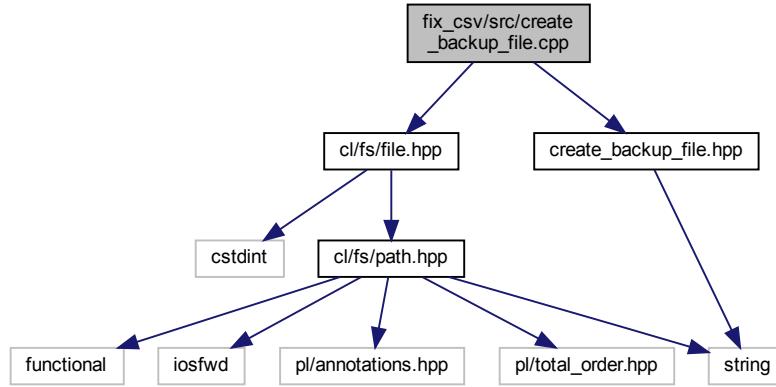
Functions

- bool [fmc::convertToUnixLineEndings](#) (const std::string &csvPath)

Convert a CSV file to UNIX line endings.

7.148 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

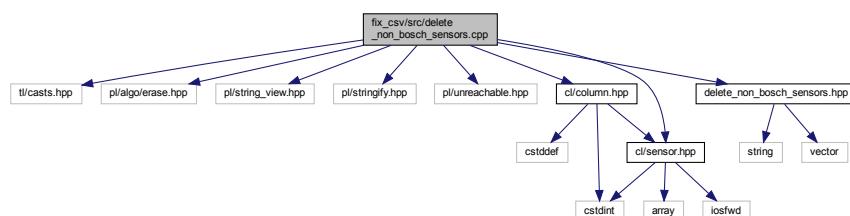
- `fmc`

Functions

- bool `fmc::createBackupFile` (const std::string &csvFilePath, const std::string &backupFilePath)
Creates a backup of a file.

7.149 fix_csv/src/delete_non_bosch_sensors.cpp File Reference

```
#include <t1/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)`
Routine to delete anything that isn't a Bosch sensor.

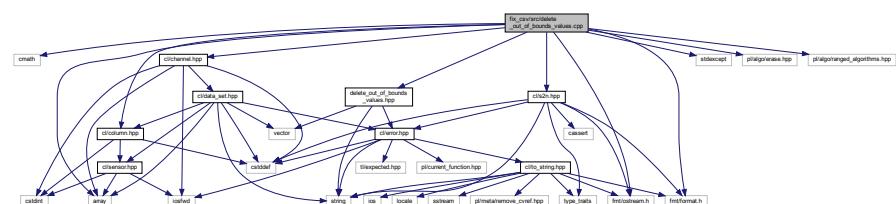
7.149.1 Macro Definition Documentation

7.149.1.1 CL_SENSOR_X

```
#define CL_SENSOR_X(
    enm,
    value ) case cl::Sensor::enm: return PL_STRINGIFY(value);
```

7.150 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

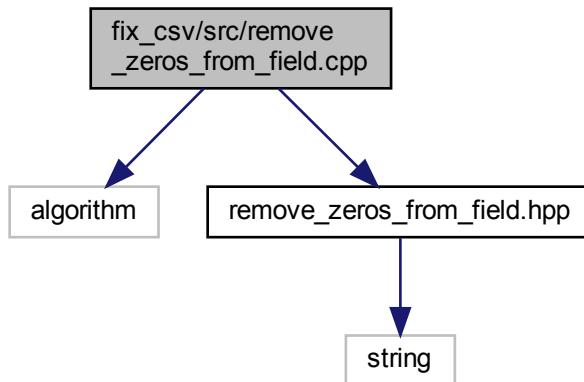
- fmc

Functions

- cl::Expected< void > [fmc::deleteOutOfBoundsValues](#) (std::vector< std::vector< std::string >> *data)
Deletes out of bounds values from the raw CSV file read.

7.151 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

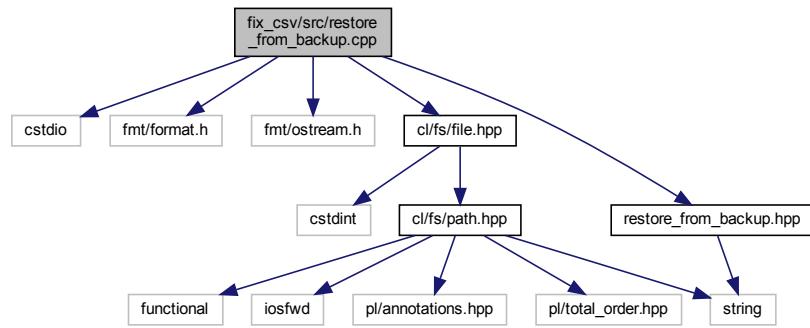
- fmc

Functions

- void [fmc::removeZerosFromField](#) (std::string *field)
Deletes extraneous zeros from a cell value.

7.152 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

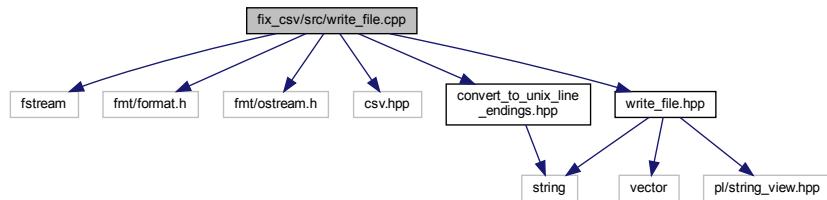
- `fmc`

Functions

- `bool fmc::restoreFromBackup (const std::string &csvFilePath, const std::string &backupFilePath)`
Restore a file from a previously created backup file.

7.153 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](#)

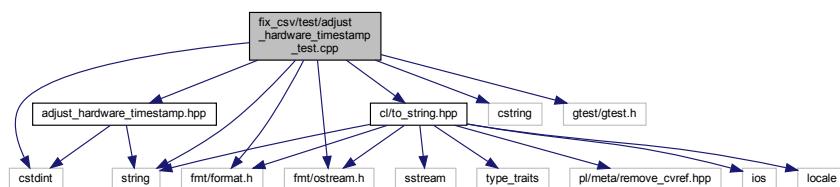
Functions

- 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)

Writes a 'fixed' CSV file.

7.154 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`)

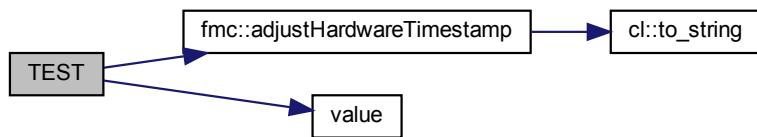
7.154.1 Function Documentation

7.154.1.1 TEST() [1/5]

```
TEST (
    adjustHardwareTimestamp ,
    shouldDoNothingForNonOverflowedValue )
```

Definition at line 15 of file adjust_hardware_timestamp_test.cpp.

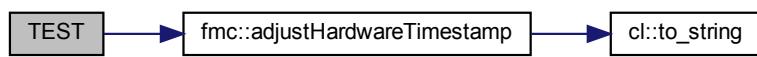
Here is the call graph for this function:

**7.154.1.2 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:

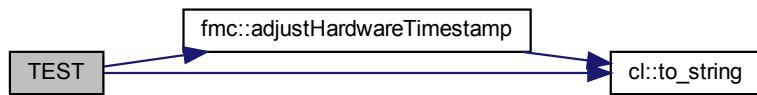


7.154.1.3 TEST() [3/5]

```
TEST (
    adjustHardwareTimestamp ,
    shouldWork   )
```

Definition at line 132 of file `adjust_hardware_timestamp_test.cpp`.

Here is the call graph for this function:

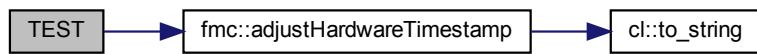


7.154.1.4 TEST() [4/5]

```
TEST (
    adjustHardwareTimestamp ,
    shouldWorkForOneRoundOfOverflow   )
```

Definition at line 48 of file `adjust_hardware_timestamp_test.cpp`.

Here is the call graph for this function:

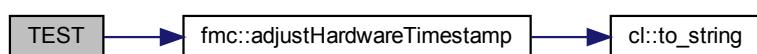


7.154.1.5 TEST() [5/5]

```
TEST (
    adjustHardwareTimestamp ,
    shouldWorkForTwoRoundsOfOverflow   )
```

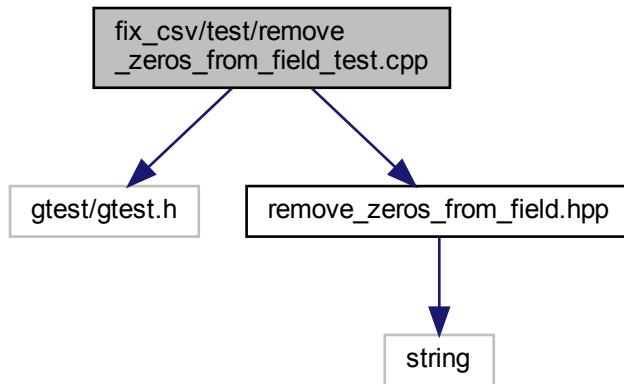
Definition at line 96 of file `adjust_hardware_timestamp_test.cpp`.

Here is the call graph for this function:



7.155 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 \(removeZerosFromField, shouldDeleteStringIfStringIsDotAndZero\)](#)

7.155.1 Function Documentation

7.155.1.1 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:



7.155.1.2 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:

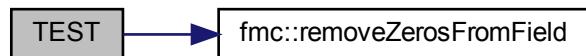


7.155.1.3 TEST() [3/6]

```
TEST (
    removeZerosFromField ,
    shouldDoNothingIfStringIsEmpty )
```

Definition at line 35 of file remove_zeros_from_field_test.cpp.

Here is the call graph for this function:

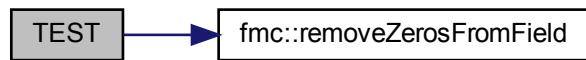


7.155.1.4 TEST() [4/6]

```
TEST (
    removeZerosFromField ,
    shouldNotRemoveIfNoDot )
```

Definition at line 25 of file remove_zeros_from_field_test.cpp.

Here is the call graph for this function:

**7.155.1.5 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:



7.155.1.6 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, 164

~Process
 cl::Process, 193

above_threshold.cpp
 channel, 296
 channelAccessor, 297
 CL_CHANNEL_X, 296

above_threshold_test.cpp
 EXPECT_LONG_DOUBLE_EQ, 299
 TEST, 300

aboveThreshold
 ctg, 67

accelerometerAverage
 cl::DataSet, 135

accelerometerMaximum
 cl::DataSet, 136

accelerometerThreshold
 cl, 26

AccelerometerX
 cl, 14

accelerometerX
 cl::DataSet, 136

AccelerometerY
 cl, 14

accelerometerY
 cl::DataSet, 137

AccelerometerZ
 cl, 14

accelerometerZ
 cl::DataSet, 138

addTrueSubtractFalseSorter
 cm, 54

adjust_hardware_timestamp_test.cpp
 TEST, 372–374

adjustHardwareTimestamp
 fmc, 74

asMilliseconds
 cm::ManualSegmentationPoint, 180

averageComparisonValueCalculator
 ctg, 68

base_type
 cl::Exception, 154

Both
 cs, 58

build
 cm::Configuration::Builder, 82
 cs::CsvLineBuilder, 116

Builder
 cm::Configuration, 102
 cm::Configuration::Builder, 82

Butterworth
 cs, 57

Channel
 cl, 14

channel
 above_threshold.cpp, 296
 cl::DataPoint, 130
 data_point.cpp, 327

channel.cpp
 CL_CHANNEL_X, 326

channel.hpp
 CL_CHANNEL, 304
 CL_CHANNEL_X, 304

channel_test.cpp
 TEST, 339, 340

ChannelAccessor
 cl::DataSet, 135

channelAccessor
 above_threshold.cpp, 297

channelCount
 cl, 27

channels
 cl, 27

cl, 11
 accelerometerThreshold, 26
 AccelerometerX, 14
 AccelerometerY, 14
 AccelerometerZ, 14
 Channel, 14
 channelCount, 27
 channels, 27
 CL_CHANNEL, 14
 CL_CHANNEL_X, 14
 CL_SENSOR, 15
 CL_SENSOR_X, 15
 CL_SPECIALIZE_COL_TRAITS, 15–17
 Column, 14
 column_index, 27
 column_type, 13
 CsvFileKind, 14
 data_set_accessor_v, 28
 dataSetAccessor, 17
 dos2unix, 18
 Expected, 13
 ExtractId, 14
 Fixed, 15

gyroscopeThreshold, 28
 GyroscopeX, 14
 GyroscopeY, 14
 GyroscopeZ, 14
 HardwareTimestamp, 14
 isAccelerometer, 18
 isGyroscope, 19
 operator<<, 20–22
 Raw, 15
 readCsvFile, 22
 s2n, 23
 SamplingRate, 14
 Sensor, 15
 sensors, 28
 threshold, 24
 Time, 14
 to_string, 25
 Trigger, 14
 useUnbufferedIo, 26
 cl::col_traits< Col >, 90
 cl::data_set_accessor< Chan >, 127
 cl::DataPoint, 128
 channel, 130
 DataPoint, 129
 fileName, 130
 operator<<, 133
 sensor, 131
 time, 131
 value, 132
 cl::DataSet, 133
 accelerometerAverage, 135
 accelerometerMaximum, 136
 accelerometerX, 136
 accelerometerY, 137
 accelerometerZ, 138
 ChannelAccessor, 135
 create, 138
 extractId, 140
 fileName, 140
 gyroscopeAverage, 141
 gyroscopeMaximum, 142
 gyroscopeX, 142
 gyroscopeY, 143
 gyroscopeZ, 143
 hardwareTimestamp, 145
 rowCount, 145
 size_type, 135
 time, 146
 trigger, 146
 cl::Error, 147
 CL_ERROR_KIND, 148
 Error, 149
 file, 149
 function, 150
 Kind, 148
 kind, 150
 line, 150
 message, 151
 operator<<, 152
 raise, 151
 to_string, 151
 cl::Exception, 152
 base_type, 154
 Exception, 154
 file, 155
 function, 155
 line, 156
 cl::fs, 28
 directoryListing, 29
 DirectoryListingOption, 29
 ExcludeDotAndDotDot, 29
 formatError, 30
 None, 29
 operator<, 31
 operator<<, 31
 operator==, 32
 utf16ToUtf8, 32
 utf8ToUtf16, 33
 cl::fs::File, 157
 copyTo, 158
 create, 158
 exists, 159
 File, 157
 moveTo, 160
 path, 160
 remove, 161
 size, 161
 cl::fs::FileStream, 162
 ~FileStream, 164
 create, 164
 FileStream, 164
 OpenMode, 163
 operator=, 165
 PL_NONCOPYABLE, 166
 Read, 164
 readAll, 166
 ReadWrite, 164
 this_type, 163
 Write, 164
 write, 166
 cl::fs::Path, 186
 exists, 187
 isDirectory, 188
 isFile, 189
 operator<, 190
 operator<<, 191
 operator==, 191
 Path, 187
 str, 189
 cl::Process, 192
 ~Process, 193
 create, 193
 file, 194
 operator=, 194
 PL_NONCOPYABLE, 195
 Process, 193

this_type, 192
CL_CHANNEL
channel.hpp, 304
cl, 14
CL_CHANNEL_X
above_threshold.cpp, 296
channel.cpp, 326
channel.hpp, 304
cl, 14
CL_ERROR_KIND
cl::Error, 148
error.hpp, 311
CL_ERROR_KIND_X
error.cpp, 331
error.hpp, 311
CL_FS_SEPARATOR
separator.hpp, 317
CL_SENSOR
cl, 15
sensor.hpp, 323
CL_SENSOR_X
cl, 15
delete_non_bosch_sensors.cpp, 369
sensor.cpp, 337
sensor.hpp, 323
CL_SPECIALIZE_COL_TRAITS
cl, 15–17
column.hpp, 307
CL_THROW
exception.hpp, 313
CL_THROW_FMT
exception.hpp, 313
CL_UNEXPECTED
error.hpp, 311
closestOne
cm, 37
cm, 34
addTrueSubtractFalseSorter, 54
closestOne, 37
CM_DATA_SET_IDENTIFIER, 36
CM_DATA_SET_IDENTIFIER_X, 36
CM_IMU, 36
CM_IMU_X, 36
CM_SORTER, 38
confusionMatrixBestConfigs, 38
createSegmentationResults, 40
DataSetIdentifier, 36
disregardTrueNegativesSorter, 54
distance, 41
distanceScore, 42
fetch, 43
Imu, 36
imuCount, 55
imus, 55
interpolatedDataSetPaths, 44
operator!=, 45
operator<, 45
operator<<, 46–48
operator==, 49
orderConfigurationsByQuality, 49
pythonOutput, 50
segment, 51
splitString, 53
toDataSetIdentifier, 53
cm::Configuration, 91
Builder, 102
Configuration, 92
createFilePath, 93
deleteTooClose, 93
deleteTooCloseOptions, 93
deleteTooLowVariance, 94
deleteTooLowVarianceOptions, 94
filterKind, 95
filterKindOptions, 95
importSegmentationPoints, 96
imu, 97
imuOptions, 97
isInitialized, 98
operator<, 102
operator<<, 103
operator==, 103
segmentationKind, 98
segmentationKindOptions, 99
serializeSegmentationPoints, 99
skipWindow, 100
skipWindowOptions, 100
std::hash< Configuration >, 103
windowSize, 101
windowSizeOptions, 101
cm::Configuration::Builder, 81
build, 82
Builder, 82
deleteTooClose, 83
deleteTooLowVariance, 84
filterKind, 85
imu, 86
segmentationKind, 87
skipWindow, 88
windowSize, 89
cm::ConfigWithDistanceScore, 104
config, 105
ConfigWithDistanceScore, 104
distScore, 105
cm::ConfigWithTotalConfusionMatrix, 105
config, 107
ConfigWithTotalConfusionMatrix, 106
matrix, 107
operator<<, 107
cm::ConfusionMatrix, 108
ConfusionMatrix, 109
falseNegatives, 109
falsePositives, 109
incrementFalseNegatives, 109
incrementFalsePositives, 110
incrementTrueNegatives, 110
incrementTruePositives, 111

operator+=, 111
 this_type, 108
 totalCount, 112
 trueNegatives, 112
 truePositives, 112
 cm::CsvFileInfo, 113
 CsvFileInfo, 113
 hardwareTimestamps, 114
 cm::ManualSegmentationPoint, 178
 asMilliseconds, 180
 convertToHardwareTimestamps, 180
 frame, 181
 hour, 182
 ManualSegmentationPoint, 179
 minute, 182
 operator!=, 184
 operator<<, 185
 operator==, 185
 readCsvFile, 183
 second, 184
CM_DATA_SET_IDENTIFIER
 cm, 36
 data_set_identifier.hpp, 256
CM_DATA_SET_IDENTIFIER_X
 cm, 36
 data_set_identifier.cpp, 273
 data_set_identifier.hpp, 256
CM_DEV_NULL
 python_output.cpp, 279
CM_ENSURE_CONTAINS
 configuration.cpp, 269
CM_ENSURE_HAS_VALUE
 configuration.cpp, 269
CM_IMU
 cm, 36
 imu.hpp, 260
CM_IMU_X
 cm, 36
 imu.cpp, 275
 imu.hpp, 261
CM_SEGMENTOR
 python_output.cpp, 279
CM_SORTER
 cm, 38
 confusion_matrix_best_configs.hpp, 252
CMakeLists.txt
 include, 197–201
 set, 197–201
Column
 cl, 14
column.hpp
 CL_SPECIALIZE_COL_TRAITS, 307
column_index
 cl, 27
column_test.cpp
 TEST, 341
column_type
 cl, 13
 compare_segmentation/CMakeLists.txt, 197
 compare_segmentation/include/csv_line.hpp, 202
 compare_segmentation/include/data_set_info.hpp, 203
 compare_segmentation/include/filter_kind.hpp, 205
 compare_segmentation/include/log_files.hpp, 206
 compare_segmentation/include/log_info.hpp, 206
 compare_segmentation/include/log_line.hpp, 207
 compare_segmentation/include/mode.hpp, 208
 compare_segmentation/include/paths.hpp, 210
 compare_segmentation/include/segmentation_kind.hpp,
 211
 compare_segmentation/src/csv_line.cpp, 212
 compare_segmentation/src/data_set_info.cpp, 212
 compare_segmentation/src/filter_kind.cpp, 213
 compare_segmentation/src/log_files.cpp, 213
 compare_segmentation/src/log_info.cpp, 214
 compare_segmentation/src/log_line.cpp, 215
 compare_segmentation/src/main.cpp, 215
 compare_segmentation/src/mode.cpp, 227
 compare_segmentation/src/segmentation_kind.cpp,
 229
 compare_segmentation/test/CMakeLists.txt, 197
 compare_segmentation/test/csv_line_test.cpp, 229
 compare_segmentation/test/data_set_info_test.cpp,
 230
 compare_segmentation/test/log_files_test.cpp, 231
 compare_segmentation/test/log_info_test.cpp, 233
 compare_segmentation/test/log_line_test.cpp, 243
 compare_segmentation/test/main.cpp, 218
 compare_segmentation/test/mode_test.cpp, 245
config
 cm::ConfigWithDistanceScore, 105
 cm::ConfigWithTotalConfusionMatrix, 107
Configuration
 cm::Configuration, 92
configuration.cpp
 CM_ENSURE_CONTAINS, 269
 CM_ENSURE_HAS_VALUE, 269
ConfigWithDistanceScore
 cm::ConfigWithDistanceScore, 104
ConfigWithTotalConfusionMatrix
 cm::ConfigWithTotalConfusionMatrix, 106
confusion_matrix/CMakeLists.txt, 201
confusion_matrix/include/closest_one.hpp, 247
confusion_matrix/include/configuration.hpp, 248
confusion_matrix/include/confusion_matrix.hpp, 249
confusion_matrix/include/confusion_matrix_best_configs.hpp,
 250
confusion_matrix/include/create_segmentation_results.hpp,
 252
confusion_matrix/include/csv_file_info.hpp, 253
confusion_matrix/include/data_set_identifier.hpp, 254
confusion_matrix/include/distance.hpp, 256
confusion_matrix/include/distance_score.hpp, 257
confusion_matrix/include/fetch.hpp, 258
confusion_matrix/include imu.hpp, 259
confusion_matrix/include/interpolated_data_set_paths.hpp,
 262

confusion_matrix/include/manual_segmentation_point.hpp
 263
confusion_matrix/include/order_configurations_by_quality.hpp
 264
confusion_matrix/include/python_output.hpp, 265
confusion_matrix/include/segment.hpp, 266
confusion_matrix/include/split_string.hpp, 267
confusion_matrix/src/closest_one.cpp, 267
confusion_matrix/src/configuration.cpp, 268
confusion_matrix/src/confusion_matrix.cpp, 270
confusion_matrix/src/confusion_matrix_best_configs.cpp,
 270
confusion_matrix/src/create_segmentation_results.cpp,
 271
confusion_matrix/src/csv_file_info.cpp, 272
confusion_matrix/src/data_set_identifier.cpp, 272
confusion_matrix/src/distance.cpp, 274
confusion_matrix/src/distance_score.cpp, 274
confusion_matrix/src/imu.cpp, 275
confusion_matrix/src/interpolated_data_set_paths.cpp,
 276
confusion_matrix/src/main.cpp, 224
confusion_matrix/src/manual_segmentation_point.cpp,
 276
confusion_matrix/src/order_configurations_by_quality.cpp, cs, 55
 278
confusion_matrix/src/python_output.cpp, 278
confusion_matrix/src/segment.cpp, 279
confusion_matrix/src/split_string.cpp, 280
confusion_matrix/test/CMakeLists.txt, 201
confusion_matrix/test/data_set_identifier_test.cpp, 281
confusion_matrix/test/interpolated_data_set_paths_test.cpp,
 282
confusion_matrix/test/main.cpp, 226
confusion_matrix/test/manual_segmentation_point_test.cpp,
 283
confusion_matrix/test/segment_test.cpp, 287
confusion_matrix/test/split_string_test.cpp, 288
confusion_matrix_best_configs.hpp
 CM_SORTER, 252
ConfusionMatrix
 cm::ConfusionMatrix, 109
confusionMatrixBestConfigs
 cm, 38
convertToHardwareTimestamps
 cm::ManualSegmentationPoint, 180
convertToUnixLineEndings
 fmc, 75
copyTo
 cl::fs::File, 158
counting/CMakeLists.txt, 198
counting/include/above_threshold.hpp, 290
counting/include/average_comparison_value_calculator.hpp,
 291
counting/include/half_maximum_comparison_value_calculator.hpp
 292
counting/include/is_relevant.hpp, 293
counting/include/percentage_of.hpp, 294
 counting/include/run_above_threshold.hpp, 295
 counting/src/above_threshold.cpp, 295
 counting/include/average_comparison_value_calculator.hpp,
 297
 counting/src/half_maximum_comparison_value_calculator.cpp,
 298
 counting/src/main.cpp, 218
 counting/src/run_above_threshold.cpp, 298
 counting/test/above_threshold_test.cpp, 299
 counting/test/CMakeLists.txt, 198
 counting/test/main.cpp, 220
 counting/test/percentage_of_test.cpp, 301
 create
 cl::DataSet, 138
 cl::fs::File, 158
 cl::fs::FileStream, 164
 cl::Process, 193
 cs::LogInfo, 169
 createBackupFile
 fmc, 76
 createFilePath
 cm::Configuration, 93
 createSegmentationResults
 cm, 40
 Both, 58
 Butterworth, 57
 CS_MODE, 57
 CS_MODE_X, 57
 CS_SPECIALIZE_DATA_SET_INFO, 58–61
 FilterKind, 57
 logFiles, 61
 logPath, 66
 Maxima, 58
 Minima, 58
 Mode, 57
 MovingAverage, 57
 oldLogPath, 66
 operator!=, 62
 operator<<, 63, 64
 operator==, 64
 parseMode, 65
 PL_DEFINE_EXCEPTION_TYPE, 65
 repetitionCount, 65
 SegmentationKind, 57
 cs::CsvLineBuilder, 114
 build, 116
 CsvLineBuilder, 116
 dataSet, 116
 deleteLowVariance, 117
 deleteTooClose, 118
 filter, 119
 isOld, 120
 kind, 121
 map, 122
 segmentationPoints, 123
 sensor, 124
 skipWindow, 125

this_type, 115
 windowSize, 126
 cs::data_set_info< Tag >, 128
 cs::LogInfo, 168
 create, 169
 deleteLowVariance, 170
 deleteTooClose, 170
 filterKind, 170
 invalidSensor, 174
 isInitialized, 171
 logFilePath, 171
 LogInfo, 169
 operator!=, 173
 operator<<, 173
 operator==, 174
 segmentationKind, 171
 sensor, 172
 skipWindow, 172
 windowSize, 172
 cs::LogLine, 174
 fileName, 175
 filePath, 176
 invalidSensor, 178
 parse, 176
 segmentationPointCount, 177
 sensor, 177
CS_MODE
 cs, 57
 mode.hpp, 209
CS_MODE_X
 cs, 57
 mode.cpp, 228
 mode.hpp, 209
CS_SPECIALIZE_DATA_SET_INFO
 cs, 58–61
 data_set_info.hpp, 204
 csv_lib/CMakeLists.txt, 199
 csv_lib/include/cl/channel.hpp, 302
 csv_lib/include/cl/column.hpp, 305
 csv_lib/include/cl/data_point.hpp, 307
 csv_lib/include/cl/data_set.hpp, 308
 csv_lib/include/cl/dos2unix.hpp, 309
 csv_lib/include/cl/error.hpp, 310
 csv_lib/include/cl/exception.hpp, 312
 csv_lib/include/cl/fs/directory_listing.hpp, 313
 csv_lib/include/cl/fs/file.hpp, 314
 csv_lib/include/cl/fs/file_stream.hpp, 315
 csv_lib/include/cl/fs/path.hpp, 316
 csv_lib/include/cl/fs/separator.hpp, 317
 csv_lib/include/cl/fs/windows.hpp, 318
 csv_lib/include/cl/process.hpp, 319
 csv_lib/include/cl/read_csv_file.hpp, 320
 csv_lib/include/cl/s2n.hpp, 321
 csv_lib/include/cl/sensor.hpp, 321
 csv_lib/include/cl/to_string.hpp, 324
 csv_lib/include/cl/use_unbuffered_io.hpp, 324
 csv_lib/src/cl/channel.cpp, 325
 csv_lib/src/cl/data_point.cpp, 326
 csv_lib/src/cl/data_set.cpp, 329
 csv_lib/src/cl/dos2unix.cpp, 330
 csv_lib/src/cl/error.cpp, 331
 csv_lib/src/cl/exception.cpp, 332
 csv_lib/src/cl/fs/directory_listing.cpp, 332
 csv_lib/src/cl/fs/file.cpp, 333
 csv_lib/src/cl/fs/file_stream.cpp, 333
 csv_lib/src/cl/fs/path.cpp, 334
 csv_lib/src/cl/fs/windows.cpp, 334
 csv_lib/src/cl/process.cpp, 335
 csv_lib/src/cl/read_csv_file.cpp, 336
 csv_lib/src/cl/sensor.cpp, 336
 csv_lib/src/cl/use_unbuffered_io.cpp, 338
 csv_lib/test/channel_test.cpp, 338
 csv_lib/test/CMakeLists.txt, 199
 csv_lib/test/column_test.cpp, 340
 csv_lib/test/data_point_test.cpp, 342
 csv_lib/test/data_set_test.cpp, 344
 csv_lib/test/directory_listing_test.cpp, 348
 csv_lib/test/error_test.cpp, 350
 csv_lib/test/exception_test.cpp, 352
 csv_lib/test/main.cpp, 221
 csv_lib/test/read_csv_file_test.cpp, 353
 csv_lib/test/s2n_test.cpp, 354
 csv_lib/test/sensor_test.cpp, 356
 csv_lib/test/to_string_test.cpp, 357
 csv_line_test.cpp
 TEST, 230
 CsvFileInfo
 cm::CsvFileInfo, 113
 CsvFileKind
 cl, 14
 CsvLineBuilder
 cs::CsvLineBuilder, 116
 ctg, 67
 aboveThreshold, 67
 averageComparisonValueCalculator, 68
 halfMaximumComparisonValueCalculator, 69
 isRelevant, 70
 percentageOf, 72
 runAboveThreshold, 72
 data_point.cpp
 channel, 327
 fileName, 327
 sensor, 327
 time, 328
 value, 328
 data_point_test.cpp
 dp, 343
 TEST, 342, 343
 data_set_accessor_v
 cl, 28
 data_set_identifier.cpp
 CM_DATA_SET_IDENTIFIER_X, 273
 DSI, 273
 data_set_identifier.hpp
 CM_DATA_SET_IDENTIFIER, 256
 CM_DATA_SET_IDENTIFIER_X, 256

data_set_identifier_test.cpp
DSI, 281
TEST, 281
data_set_info.hpp
CS_SPECIALIZE_DATA_SET_INFO, 204
data_set_info_test.cpp
TEST, 230
data_set_test.cpp
EXPECT_LONG_DOUBLE_EQ, 344
TEST, 344–347
DataPoint
cl::DataPoint, 129
dataSet
cs::CsvLineBuilder, 116
dataSetAccessor
cl, 17
DataSetIdentifier
cm, 36
delete_non_bosch_sensors.cpp
CL_SENSOR_X, 369
deleteLowVariance
cs::CsvLineBuilder, 117
cs::LogInfo, 170
deleteNonBoschSensors
fmc, 76
deleteOutOfBoundsValues
fmc, 77
deleteTooClose
cm::Configuration, 93
cm::Configuration::Builder, 83
cs::CsvLineBuilder, 118
cs::LogInfo, 170
deleteTooCloseOptions
cm::Configuration, 93
deleteTooLowVariance
cm::Configuration, 94
cm::Configuration::Builder, 84
deleteTooLowVarianceOptions
cm::Configuration, 94
directory_listing_test.cpp
TEST, 349
directoryListing
cl::fs, 29
DirectoryListingOption
cl::fs, 29
disregardTrueNegativesSorter
cm, 54
distance
cm, 41
distanceScore
cm, 42
distScore
cm::ConfigWithDistanceScore, 105
dos2unix
cl, 18
dp
data_point_test.cpp, 343
DSI
data_set_identifier.cpp, 273
data_set_identifier_test.cpp, 281
manual_segmentation_point.cpp, 277
manual_segmentation_point_test.cpp, 284
Error
cl::Error, 149
error
error_test.cpp, 351
error.hpp
CL_ERROR_KIND_X, 331
error.hpp
CL_ERROR_KIND, 311
CL_ERROR_KIND_X, 311
CL_UNEXPECTED, 311
error_test.cpp
error, 351
TEST, 351
Exception
cl::Exception, 154
exception.hpp
CL_THROW, 313
CL_THROW_FMT, 313
exception_test.cpp
TEST, 352
ExcludeDotAndDotDot
cl::fs, 29
exists
cl::fs::File, 159
cl::fs::Path, 187
EXPECT_LONG_DOUBLE_EQ
above_threshold_test.cpp, 299
data_set_test.cpp, 344
percentage_of_test.cpp, 301
EXPECT_SEGMENTATION_POINTS
segment_test.cpp, 287
Expected
cl, 13
ExtractId
cl, 14
extractId
cl::DataSet, 140
falseNegatives
cm::ConfusionMatrix, 109
falsePositives
cm::ConfusionMatrix, 109
fetch
cm, 43
File
cl::fs::File, 157
file
cl::Error, 149
cl::Exception, 155
cl::Process, 194
fileName
cl::DataPoint, 130
cl::DataSet, 140
cs::LogLine, 175

data_point.cpp, 327
 filePath
 cs::LogLine, 176
 FileStream
 cl::fs::FileStream, 164
 filter
 cs::CsvLineBuilder, 119
 FilterKind
 cs, 57
 filterKind
 cm::Configuration, 95
 cm::Configuration::Builder, 85
 cs::LogInfo, 170
 filterKindOptions
 cm::Configuration, 95
 fix_csv/CMakeLists.txt, 200
 fix_csv/include/adjust_hardware_timestamp.hpp, 358
 fix_csv/include/convert_to_unix_line_endings.hpp, 359
 fix_csv/include/create_backup_file.hpp, 360
 fix_csv/include/delete_non_bosch_sensors.hpp, 361
 fix_csv/include/delete_out_of_bounds_values.hpp, 362
 fix_csv/include/remove_zeros_from_field.hpp, 363
 fix_csv/include/restore_from_backup.hpp, 364
 fix_csv/include/write_file.hpp, 365
 fix_csv/src/adjust_hardware_timestamp.cpp, 366
 fix_csv/src/convert_to_unix_line_endings.cpp, 367
 fix_csv/src/create_backup_file.cpp, 368
 fix_csv/src/delete_non_bosch_sensors.cpp, 368
 fix_csv/src/delete_out_of_bounds_values.cpp, 369
 fix_csv/src/main.cpp, 222
 fix_csv/src/remove_zeros_from_field.cpp, 370
 fix_csv/src/restore_from_backup.cpp, 371
 fix_csv/src/write_file.cpp, 371
 fix_csv/test/adjust_hardware_timestamp_test.cpp, 372
 fix_csv/test/CMakeLists.txt, 200
 fix_csv/test/main.cpp, 223
 fix_csv/test/remove_zeros_from_field_test.cpp, 375
 Fixed
 cl, 15
 fmc, 73
 adjustHardwareTimestamp, 74
 convertToUnixLineEndings, 75
 createBackupFile, 76
 deleteNonBoschSensors, 76
 deleteOutOfBoundsValues, 77
 removeZerosFromField, 77
 restoreFromBackup, 78
 writeFile, 79
 formatError
 cl::fs, 30
 frame
 cm::ManualSegmentationPoint, 181
 function
 cl::Error, 150
 cl::Exception, 155
 gyroscopeAverage
 cl::DataSet, 141
 gyroscopeMaximum
 cl::DataSet, 142
 gyroscopeThreshold
 cl, 28
 GyroscopeX
 cl, 14
 gyroscopeX
 cl::DataSet, 142
 GyroscopeY
 cl, 14
 gyroscopeY
 cl::DataSet, 143
 GyroscopeZ
 cl, 14
 gyroscopeZ
 cl::DataSet, 143
 halfMaximumComparisonValueCalculator
 ctg, 69
 HardwareTimestamp
 cl, 14
 hardwareTimestamp
 cl::DataSet, 145
 hardwareTimestamps
 cm::CsvFileInfo, 114
 hour
 cm::ManualSegmentationPoint, 182
 importSegmentationPoints
 cm::Configuration, 96
 Imu
 cm, 36
 imu
 cm::Configuration, 97
 cm::Configuration::Builder, 86
 imu.cpp
 CM_IMU_X, 275
 imu.hpp
 CM_IMU, 260
 CM_IMU_X, 261
 imuCount
 cm, 55
 imuOptions
 cm::Configuration, 97
 imus
 cm, 55
 include
 CMakeLists.txt, 197–201
 incrementFalseNegatives
 cm::ConfusionMatrix, 109
 incrementFalsePositives
 cm::ConfusionMatrix, 110
 incrementTrueNegatives
 cm::ConfusionMatrix, 110
 incrementTruePositives
 cm::ConfusionMatrix, 111
 interpolated_data_set_paths_test.cpp
 TEST, 282
 interpolatedDataSetPaths
 cm, 44

invalidSensor
 cs::LogInfo, 174
 cs::LogLine, 178
isAccelerometer
 cl, 18
isDirectory
 cl::fs::Path, 188
isFile
 cl::fs::Path, 189
isGyroscope
 cl, 19
isInitialized
 cm::Configuration, 98
 cs::LogInfo, 171
isOld
 cs::CsvLineBuilder, 120
isRelevant
 ctg, 70

Kind
 cl::Error, 148
kind
 cl::Error, 150
 cs::CsvLineBuilder, 121

line
 cl::Error, 150
 cl::Exception, 156
log_files_test.cpp
 TEST, 231, 232
log_info_test.cpp
 TEST, 234–242
log_line_test.cpp
 TEST, 243–245
logFilePath
 cs::LogInfo, 171
logFiles
 cs, 61
LogInfo
 cs::LogInfo, 169
logPath
 cs, 66

main
 main.cpp, 216, 218, 219, 221–225, 227
main.cpp
 main, 216, 218, 219, 221–225, 227
 SORT_PRINT, 225
manual_segmentation_point.cpp
 DSI, 277
manual_segmentation_point_test.cpp
 DSI, 284
 TEST, 284–286
ManualSegmentationPoint
 cm::ManualSegmentationPoint, 179
matrix
 cm::ConfigWithTotalConfusionMatrix, 107
Maxima
 cs, 58
message
 cl::Error, 151
Minima
 cs, 58
minute
 cm::ManualSegmentationPoint, 182
Mode
 cs, 57
mode.cpp
 CS_MODE_X, 228
mode.hpp
 CS_MODE, 209
 CS_MODE_X, 209
mode_test.cpp
 TEST, 246
moveTo
 cl::fs::File, 160
MovingAverage
 cs, 57

None
 cl::fs, 29

oldLogPath
 cs, 66
OpenMode
 cl::fs::FileStream, 163
operator!=
 cm, 45
 cm::ManualSegmentationPoint, 184
 cs, 62
 cs::LogInfo, 173
operator<
 cl::fs, 31
 cl::fs::Path, 190
 cm, 45
 cm::Configuration, 102
operator<<
 cl, 20–22
 cl::DataPoint, 133
 cl::Error, 152
 cl::fs, 31
 cl::fs::Path, 191
 cm, 46–48
 cm::Configuration, 103
 cm::ConfigWithTotalConfusionMatrix, 107
 cm::ManualSegmentationPoint, 185
 cs, 63, 64
 cs::LogInfo, 173
operator()
 std::hash<::cl::fs::Path>, 167
 std::hash<::cm::Configuration>, 167
operator+=
 cm::ConfusionMatrix, 111
operator=
 cl::fs::FileStream, 165
 cl::Process, 194
operator==
 cl::fs, 32

cl::fs::Path, 191
 cm, 49
 cm::Configuration, 103
 cm::ManualSegmentationPoint, 185
 cs, 64
 cs::LogInfo, 174
 orderConfigurationsByQuality
 cm, 49

parse
 cs::LogLine, 176

parseMode
 cs, 65

Path
 cl::fs::Path, 187

path
 cl::fs::File, 160

percentage_of_test.cpp
 EXPECT_LONG_DOUBLE_EQ, 301
 TEST, 301

percentageOf
 ctg, 72

PL_DEFINE_EXCEPTION_TYPE
 cs, 65

PL_NOCOPYABLE
 cl::fs::FileStream, 166
 cl::Process, 195

Process
 cl::Process, 193

python_output.cpp
 CM_DEV_NULL, 279
 CM_SEGMENTOR, 279

pythonOutput
 cm, 50

raise
 cl::Error, 151

Raw
 cl, 15

Read
 cl::fs::FileStream, 164

read_csv_file_test.cpp
 TEST, 353, 354

readAll
 cl::fs::FileStream, 166

readCsvFile
 cl, 22
 cm::ManualSegmentationPoint, 183

ReadWrite
 cl::fs::FileStream, 164

remove
 cl::fs::File, 161

remove_zeros_from_field_test.cpp
 TEST, 375–377

removeZerosFromField
 fmc, 77

repetitionCount
 cs, 65

repetitions

cs::CsvLineBuilder, 122
 restoreFromBackup
 fmc, 78

rowCount
 cl::DataSet, 145

runAboveThreshold
 ctg, 72

s2n
 cl, 23

s2n_test.cpp
 TEST, 355

SamplingRate
 cl, 14

second
 cm::ManualSegmentationPoint, 184

segment
 cm, 51

segment_test.cpp
 EXPECT_SEGMENTATION_POINTS, 287
 TEST, 287

SegmentationKind
 cs, 57

segmentationKind
 cm::Configuration, 98
 cm::Configuration::Builder, 87
 cs::LogInfo, 171

segmentationKindOptions
 cm::Configuration, 99

segmentationPointCount
 cs::LogLine, 177

segmentationPoints
 cs::CsvLineBuilder, 123

Sensor
 cl, 15

sensor
 cl::DataPoint, 131
 cs::CsvLineBuilder, 124
 cs::LogInfo, 172
 cs::LogLine, 177
 data_point.cpp, 327

sensor.cpp
 CL_SENSOR_X, 337

sensor.hpp
 CL_SENSOR, 323
 CL_SENSOR_X, 323

sensor_test.cpp
 TEST, 357

sensors
 cl, 28

separator.hpp
 CL_FS_SEPARATOR, 317

serializeSegmentationPoints
 cm::Configuration, 99

set
 CMakeLists.txt, 197–201

size
 cl::fs::File, 161

size_type

cl::DataSet, 135
skipWindow
 cm::Configuration, 100
 cm::Configuration::Builder, 88
 cs::CsvLineBuilder, 125
 cs::LogInfo, 172
skipWindowOptions
 cm::Configuration, 100
SORT_PRINT
 main.cpp, 225
split_string_test.cpp
 TEST, 289
splitString
 cm, 53
std::hash< Configuration >
 cm::Configuration, 103
std::hash<::cl::fs::Path >, 167
 operator(), 167
std::hash<::cm::Configuration >, 167
 operator(), 167
str
 cl::fs::Path, 189

TEST
 above_threshold_test.cpp, 300
 adjust_hardware_timestamp_test.cpp, 372–374
 channel_test.cpp, 339, 340
 column_test.cpp, 341
 csv_line_test.cpp, 230
 data_point_test.cpp, 342, 343
 data_set_identifier_test.cpp, 281
 data_set_info_test.cpp, 230
 data_set_test.cpp, 344–347
 directory_listing_test.cpp, 349
 error_test.cpp, 351
 exception_test.cpp, 352
 interpolated_data_set_paths_test.cpp, 282
 log_files_test.cpp, 231, 232
 log_info_test.cpp, 234–242
 log_line_test.cpp, 243–245
 manual_segmentation_point_test.cpp, 284–286
 mode_test.cpp, 246
 percentage_of_test.cpp, 301
 read_csv_file_test.cpp, 353, 354
 remove_zeros_from_field_test.cpp, 375–377
 s2n_test.cpp, 355
 segment_test.cpp, 287
 sensor_test.cpp, 357
 split_string_test.cpp, 289
 to_string_test.cpp, 358
this_type
 cl::fs::FileStream, 163
 cl::Process, 192
 cm::ConfusionMatrix, 108
 cs::CsvLineBuilder, 115
threshold
 cl, 24
Time
 cl, 14
time
 cl::DataPoint, 131
 cl::DataSet, 146
 data_point.cpp, 328
to_string
 cl, 25
 cl::Error, 151
to_string_test.cpp
 TEST, 358
toDataSetIdentifier
 cm, 53
totalCount
 cm::ConfusionMatrix, 112
Trigger
 cl, 14
trigger
 cl::DataSet, 146
trueNegatives
 cm::ConfusionMatrix, 112
truePositives
 cm::ConfusionMatrix, 112

useUnbufferedIo
 cl, 26
utf16ToUtf8
 cl::fs, 32
utf8ToUtf16
 cl::fs, 33

value
 cl::DataPoint, 132
 data_point.cpp, 328

windowSize
 cm::Configuration, 101
 cm::Configuration::Builder, 89
 cs::CsvLineBuilder, 126
 cs::LogInfo, 172
windowSizeOptions
 cm::Configuration, 101
Write
 cl::fs::FileStream, 164
write
 cl::fs::FileStream, 166
writeFile
 fmc, 79