

`mogasens_csv`

Generated by Doxygen 1.8.17



---

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

---

5.1.3.22 threshold()	21
5.1.3.23 to_string()	22
5.1.3.24 useUnbufferedIo()	22
5.1.4 Variable Documentation	22
5.1.4.1 accelerometerThreshold	23
5.1.4.2 channelCount	23
5.1.4.3 channels	23
5.1.4.4 column_index	23
5.1.4.5 data_set_accessor_v	23
5.1.4.6 gyroscopeThreshold	24
5.1.4.7 sensors	24
5.2 cl::fs Namespace Reference	24
5.2.1 Enumeration Type Documentation	25
5.2.1.1 DirectoryListingOption	25
5.2.2 Function Documentation	25
5.2.2.1 directoryListing()	25
5.2.2.2 formatError()	26
5.2.2.3 operator<()	27
5.2.2.4 operator<<()	27
5.2.2.5 operator==()	28
5.2.2.6 utf16ToUtf8()	28
5.2.2.7 utf8ToUtf16()	29
5.3 cm Namespace Reference	30
5.3.1 Enumeration Type Documentation	31
5.3.1.1 DataSetIdentifier	31
5.3.1.2 Imu	31
5.3.2 Function Documentation	31
5.3.2.1 interpolatedDataSetPaths()	31
5.3.2.2 operator"!="()	32
5.3.2.3 operator<<() [1/3]	33
5.3.2.4 operator<<() [2/3]	33
5.3.2.5 operator<<() [3/3]	34
5.3.2.6 operator==()	34
5.3.2.7 segment()	34
5.3.2.8 splitString()	35
5.3.2.9 toDataSetIdentifier()	36
5.3.3 Variable Documentation	37
5.3.3.1 imuCount	37
5.3.3.2 imus	37
5.4 cs Namespace Reference	37
5.4.1 Enumeration Type Documentation	39
5.4.1.1 FilterKind	39

---

5.4.1.2 SegmentationKind . . . . .	39
5.4.2 Function Documentation . . . . .	39
5.4.2.1 CS_SPECIALIZE_DATA_SET_INFO() [1/20] . . . . .	39
5.4.2.2 CS_SPECIALIZE_DATA_SET_INFO() [2/20] . . . . .	40
5.4.2.3 CS_SPECIALIZE_DATA_SET_INFO() [3/20] . . . . .	40
5.4.2.4 CS_SPECIALIZE_DATA_SET_INFO() [4/20] . . . . .	40
5.4.2.5 CS_SPECIALIZE_DATA_SET_INFO() [5/20] . . . . .	40
5.4.2.6 CS_SPECIALIZE_DATA_SET_INFO() [6/20] . . . . .	40
5.4.2.7 CS_SPECIALIZE_DATA_SET_INFO() [7/20] . . . . .	40
5.4.2.8 CS_SPECIALIZE_DATA_SET_INFO() [8/20] . . . . .	41
5.4.2.9 CS_SPECIALIZE_DATA_SET_INFO() [9/20] . . . . .	41
5.4.2.10 CS_SPECIALIZE_DATA_SET_INFO() [10/20] . . . . .	41
5.4.2.11 CS_SPECIALIZE_DATA_SET_INFO() [11/20] . . . . .	41
5.4.2.12 CS_SPECIALIZE_DATA_SET_INFO() [12/20] . . . . .	41
5.4.2.13 CS_SPECIALIZE_DATA_SET_INFO() [13/20] . . . . .	41
5.4.2.14 CS_SPECIALIZE_DATA_SET_INFO() [14/20] . . . . .	42
5.4.2.15 CS_SPECIALIZE_DATA_SET_INFO() [15/20] . . . . .	42
5.4.2.16 CS_SPECIALIZE_DATA_SET_INFO() [16/20] . . . . .	42
5.4.2.17 CS_SPECIALIZE_DATA_SET_INFO() [17/20] . . . . .	42
5.4.2.18 CS_SPECIALIZE_DATA_SET_INFO() [18/20] . . . . .	42
5.4.2.19 CS_SPECIALIZE_DATA_SET_INFO() [19/20] . . . . .	42
5.4.2.20 CS_SPECIALIZE_DATA_SET_INFO() [20/20] . . . . .	43
5.4.2.21 logFiles() . . . . .	43
5.4.2.22 operator"!="() . . . . .	44
5.4.2.23 operator<<() [1/3] . . . . .	44
5.4.2.24 operator<<() [2/3] . . . . .	44
5.4.2.25 operator<<() [3/3] . . . . .	45
5.4.2.26 operator==() . . . . .	45
5.4.2.27 PL_DEFINE_EXCEPTION_TYPE() . . . . .	46
5.4.2.28 repetitionCount() . . . . .	46
5.4.3 Variable Documentation . . . . .	46
5.4.3.1 logPath . . . . .	47
5.4.3.2 oldLogPath . . . . .	47
5.5 ctg Namespace Reference . . . . .	47
5.5.1 Function Documentation . . . . .	47
5.5.1.1 aboveThreshold() . . . . .	48
5.5.1.2 averageComparisonValueCalculator() . . . . .	48
5.5.1.3 halfMaximumComparisonValueCalculator() . . . . .	49
5.5.1.4 isRelevant() . . . . .	50
5.5.1.5 percentageOf() . . . . .	51
5.5.1.6 runAboveThreshold() . . . . .	52
5.6 fmc Namespace Reference . . . . .	52

---

5.6.1 Function Documentation . . . . .	53
5.6.1.1 adjustHardwareTimestamp() . . . . .	53
5.6.1.2 convertToUnixLineEndings() . . . . .	53
5.6.1.3 createBackupFile() . . . . .	54
5.6.1.4 deleteNonBoschSensors() . . . . .	55
5.6.1.5 deleteOutOfBoundsValues() . . . . .	55
5.6.1.6 removeZerosFromField() . . . . .	55
5.6.1.7 restoreFromBackup() . . . . .	56
5.6.1.8 writeFile() . . . . .	56
<b>6 Class Documentation</b>	<b>59</b>
6.1 cm::Configuration::Builder Class Reference . . . . .	59
6.1.1 Detailed Description . . . . .	59
6.1.2 Constructor & Destructor Documentation . . . . .	60
6.1.2.1 Builder() . . . . .	60
6.1.3 Member Function Documentation . . . . .	60
6.1.3.1 build() . . . . .	60
6.1.3.2 deleteTooClose() . . . . .	61
6.1.3.3 deleteTooLowVariance() . . . . .	62
6.1.3.4 filterKind() . . . . .	62
6.1.3.5 imu() . . . . .	63
6.1.3.6 segmentationKind() . . . . .	64
6.1.3.7 skipWindow() . . . . .	65
6.1.3.8 windowSize() . . . . .	65
6.2 cl::col_traits< Col > Struct Template Reference . . . . .	66
6.2.1 Detailed Description . . . . .	66
6.3 cm::Configuration Class Reference . . . . .	66
6.3.1 Detailed Description . . . . .	67
6.3.2 Member Function Documentation . . . . .	67
6.3.2.1 deleteTooClose() . . . . .	68
6.3.2.2 deleteTooCloseOptions() . . . . .	68
6.3.2.3 deleteTooLowVariance() . . . . .	68
6.3.2.4 deleteTooLowVarianceOptions() . . . . .	69
6.3.2.5 filterKind() . . . . .	69
6.3.2.6 filterKindOptions() . . . . .	70
6.3.2.7 imu() . . . . .	70
6.3.2.8 imuOptions() . . . . .	71
6.3.2.9 segmentationKind() . . . . .	71
6.3.2.10 segmentationKindOptions() . . . . .	72
6.3.2.11 skipWindow() . . . . .	72
6.3.2.12 skipWindowOptions() . . . . .	73
6.3.2.13 windowSize() . . . . .	73

---

6.3.2.14 windowSizeOptions()	74
6.3.3 Friends And Related Function Documentation	74
6.3.3.1 Builder	74
6.4 cs::CsvLineBuilder Class Reference	74
6.4.1 Detailed Description	75
6.4.2 Member Typedef Documentation	75
6.4.2.1 this_type	75
6.4.3 Constructor & Destructor Documentation	76
6.4.3.1 CsvLineBuilder()	76
6.4.4 Member Function Documentation	76
6.4.4.1 build()	76
6.4.4.2 dataSet()	77
6.4.4.3 deleteLowVariance()	77
6.4.4.4 deleteTooClose()	78
6.4.4.5 filter()	79
6.4.4.6 isOld()	80
6.4.4.7 kind()	81
6.4.4.8 repetitions()	82
6.4.4.9 segmentationPoints()	83
6.4.4.10 sensor()	84
6.4.4.11 skipWindow()	85
6.4.4.12 windowSize()	86
6.5 cl::data_set_accessor< Chan > Struct Template Reference	87
6.5.1 Detailed Description	87
6.6 cs::data_set_info< Tag > Struct Template Reference	88
6.6.1 Detailed Description	88
6.7 cl::DataPoint Class Reference	88
6.7.1 Detailed Description	88
6.7.2 Constructor & Destructor Documentation	89
6.7.2.1 DataPoint()	89
6.7.3 Member Function Documentation	89
6.7.3.1 channel()	89
6.7.3.2 fileName()	90
6.7.3.3 sensor()	90
6.7.3.4 time()	91
6.7.3.5 value()	91
6.7.4 Friends And Related Function Documentation	91
6.7.4.1 operator<<	92
6.8 cl::DataSet Class Reference	92
6.8.1 Detailed Description	93
6.8.2 Member Typedef Documentation	93
6.8.2.1 ChannelAccessor	93

6.8.2.2 <code>size_type</code>	93
6.8.3 Member Function Documentation	93
6.8.3.1 <code>accelerometerAverage()</code>	93
6.8.3.2 <code>accelerometerMaximum()</code>	94
6.8.3.3 <code>accelerometerX()</code>	94
6.8.3.4 <code>accelerometerY()</code>	95
6.8.3.5 <code>accelerometerZ()</code>	95
6.8.3.6 <code>create()</code>	96
6.8.3.7 <code>extractId()</code>	97
6.8.3.8 <code>fileName()</code>	97
6.8.3.9 <code>gyroscopeAverage()</code>	98
6.8.3.10 <code>gyroscopeMaximum()</code>	99
6.8.3.11 <code>gyroscopeX()</code>	99
6.8.3.12 <code>gyroscopeY()</code>	100
6.8.3.13 <code>gyroscopeZ()</code>	100
6.8.3.14 <code>hardwareTimestamp()</code>	101
6.8.3.15 <code>rowCount()</code>	101
6.8.3.16 <code>time()</code>	101
6.8.3.17 <code>trigger()</code>	102
6.9 <code>cl::Error</code> Class Reference	102
6.9.1 Detailed Description	103
6.9.2 Member Enumeration Documentation	103
6.9.2.1 <code>Kind</code>	103
6.9.3 Constructor & Destructor Documentation	103
6.9.3.1 <code>Error()</code>	103
6.9.4 Member Function Documentation	104
6.9.4.1 <code>file()</code>	104
6.9.4.2 <code>function()</code>	104
6.9.4.3 <code>kind()</code>	105
6.9.4.4 <code>line()</code>	105
6.9.4.5 <code>message()</code>	105
6.9.4.6 <code>raise()</code>	106
6.9.4.7 <code>to_string()</code>	106
6.9.5 Friends And Related Function Documentation	106
6.9.5.1 <code>operator&lt;&lt;</code>	106
6.10 <code>cl::Exception</code> Class Reference	106
6.10.1 Detailed Description	107
6.10.2 Member Typedef Documentation	107
6.10.2.1 <code>base_type</code>	107
6.10.3 Constructor & Destructor Documentation	108
6.10.3.1 <code>Exception() [1/2]</code>	108
6.10.3.2 <code>Exception() [2/2]</code>	108

---

6.10.4 Member Function Documentation . . . . .	108
6.10.4.1 file() . . . . .	108
6.10.4.2 function() . . . . .	109
6.10.4.3 line() . . . . .	109
6.11 cl::fs::File Class Reference . . . . .	109
6.11.1 Detailed Description . . . . .	110
6.11.2 Constructor & Destructor Documentation . . . . .	110
6.11.2.1 File() . . . . .	110
6.11.3 Member Function Documentation . . . . .	111
6.11.3.1 copyTo() . . . . .	111
6.11.3.2 create() . . . . .	112
6.11.3.3 exists() . . . . .	112
6.11.3.4 moveTo() . . . . .	113
6.11.3.5 path() . . . . .	114
6.11.3.6 remove() . . . . .	114
6.11.3.7 size() . . . . .	115
6.12 cl::fs::FileStream Class Reference . . . . .	115
6.12.1 Detailed Description . . . . .	116
6.12.2 Member Typedef Documentation . . . . .	116
6.12.2.1 this_type . . . . .	116
6.12.3 Member Enumeration Documentation . . . . .	116
6.12.3.1 OpenMode . . . . .	116
6.12.4 Constructor & Destructor Documentation . . . . .	117
6.12.4.1 FileStream() . . . . .	117
6.12.4.2 ~FileStream() . . . . .	117
6.12.5 Member Function Documentation . . . . .	117
6.12.5.1 create() . . . . .	117
6.12.5.2 operator=(()) . . . . .	118
6.12.5.3 PL_NONCOPYABLE() . . . . .	119
6.12.5.4 readAll() . . . . .	119
6.12.5.5 write() . . . . .	119
6.13 std::hash<::cl::fs::Path > Struct Reference . . . . .	120
6.13.1 Detailed Description . . . . .	120
6.13.2 Member Function Documentation . . . . .	120
6.13.2.1 operator()() . . . . .	120
6.14 cs::LogInfo Class Reference . . . . .	120
6.14.1 Detailed Description . . . . .	121
6.14.2 Constructor & Destructor Documentation . . . . .	121
6.14.2.1 LogInfo() . . . . .	121
6.14.3 Member Function Documentation . . . . .	121
6.14.3.1 create() . . . . .	121
6.14.3.2 deleteLowVariance() . . . . .	122

---

6.14.3.3 deleteTooClose()	122
6.14.3.4 filterKind()	123
6.14.3.5 isInitialized()	123
6.14.3.6 logFilePath()	123
6.14.3.7 segmentationKind()	124
6.14.3.8 sensor()	124
6.14.3.9 skipWindow()	124
6.14.3.10 windowSize()	125
6.14.4 Friends And Related Function Documentation	125
6.14.4.1 operator"!="	125
6.14.4.2 operator<<	125
6.14.4.3 operator==	126
6.14.5 Member Data Documentation	126
6.14.5.1 invalidSensor	126
6.15 cs::LogLine Class Reference	126
6.15.1 Detailed Description	127
6.15.2 Member Function Documentation	127
6.15.2.1 fileName()	127
6.15.2.2 filePath()	128
6.15.2.3 parse()	128
6.15.2.4 segmentationPointCount()	129
6.15.2.5 sensor()	129
6.15.3 Member Data Documentation	130
6.15.3.1 invalidSensor	130
6.16 cm::ManualSegmentationPoint Class Reference	130
6.16.1 Detailed Description	131
6.16.2 Constructor & Destructor Documentation	131
6.16.2.1 ManualSegmentationPoint()	131
6.16.3 Member Function Documentation	132
6.16.3.1 asMilliseconds()	132
6.16.3.2 frame()	132
6.16.3.3 hour()	133
6.16.3.4 minute()	133
6.16.3.5 readCsvFile()	134
6.16.3.6 second()	134
6.16.4 Friends And Related Function Documentation	135
6.16.4.1 operator"!="	135
6.16.4.2 operator<<	135
6.16.4.3 operator==	136
6.17 cl::fs::Path Class Reference	136
6.17.1 Detailed Description	137
6.17.2 Constructor & Destructor Documentation	137

---

6.17.2.1 Path() [1/2] . . . . .	137
6.17.2.2 Path() [2/2] . . . . .	137
6.17.3 Member Function Documentation . . . . .	138
6.17.3.1 exists() . . . . .	138
6.17.3.2 isDirectory() . . . . .	138
6.17.3.3 isFile() . . . . .	139
6.17.3.4 str() . . . . .	140
6.17.4 Friends And Related Function Documentation . . . . .	141
6.17.4.1 operator< . . . . .	141
6.17.4.2 operator<< . . . . .	141
6.17.4.3 operator== . . . . .	142
6.18 cl::Process Class Reference . . . . .	142
6.18.1 Detailed Description . . . . .	143
6.18.2 Member Typedef Documentation . . . . .	143
6.18.2.1 this_type . . . . .	143
6.18.3 Constructor & Destructor Documentation . . . . .	143
6.18.3.1 Process() . . . . .	143
6.18.3.2 ~Process() . . . . .	143
6.18.4 Member Function Documentation . . . . .	143
6.18.4.1 create() . . . . .	143
6.18.4.2 file() [1/2] . . . . .	144
6.18.4.3 file() [2/2] . . . . .	144
6.18.4.4 operator=() . . . . .	144
6.18.4.5 PL_NONCOPYABLE() . . . . .	144
<b>7 File Documentation</b> . . . . .	<b>145</b>
7.1 compare_segmentation/CMakeLists.txt File Reference . . . . .	145
7.1.1 Function Documentation . . . . .	145
7.1.1.1 set() . . . . .	145
7.2 compare_segmentation/test/CMakeLists.txt File Reference . . . . .	145
7.2.1 Function Documentation . . . . .	145
7.2.1.1 include() . . . . .	146
7.3 counting/CMakeLists.txt File Reference . . . . .	146
7.3.1 Function Documentation . . . . .	146
7.3.1.1 set() . . . . .	146
7.4 counting/test/CMakeLists.txt File Reference . . . . .	146
7.4.1 Function Documentation . . . . .	146
7.4.1.1 include() . . . . .	146
7.5 csv_lib/CMakeLists.txt File Reference . . . . .	147
7.5.1 Function Documentation . . . . .	147
7.5.1.1 set() . . . . .	147
7.6 csv_lib/test/CMakeLists.txt File Reference . . . . .	147

---

---

7.6.1 Function Documentation . . . . .	147
7.6.1.1 include() . . . . .	147
7.7 fix_csv/CMakeLists.txt File Reference . . . . .	148
7.7.1 Function Documentation . . . . .	148
7.7.1.1 set() . . . . .	148
7.8 fix_csv/test/CMakeLists.txt File Reference . . . . .	148
7.8.1 Function Documentation . . . . .	148
7.8.1.1 include() . . . . .	148
7.9 confusion_matrix/CMakeLists.txt File Reference . . . . .	149
7.9.1 Function Documentation . . . . .	149
7.9.1.1 set() . . . . .	149
7.10 confusion_matrix/test/CMakeLists.txt File Reference . . . . .	149
7.10.1 Function Documentation . . . . .	149
7.10.1.1 include() . . . . .	149
7.11 compare_segmentation/include/csv_line.hpp File Reference . . . . .	150
7.12 compare_segmentation/include/data_set_info.hpp File Reference . . . . .	151
7.12.1 Macro Definition Documentation . . . . .	152
7.12.1.1 CS_SPECIALIZE_DATA_SET_INFO . . . . .	152
7.13 compare_segmentation/include/filter_kind.hpp File Reference . . . . .	153
7.14 compare_segmentation/include/log_files.hpp File Reference . . . . .	154
7.15 compare_segmentation/include/log_info.hpp File Reference . . . . .	154
7.16 compare_segmentation/include/log_line.hpp File Reference . . . . .	155
7.17 compare_segmentation/include/paths.hpp File Reference . . . . .	156
7.18 compare_segmentation/include/segmentation_kind.hpp File Reference . . . . .	157
7.19 compare_segmentation/src/csv_line.cpp File Reference . . . . .	158
7.20 compare_segmentation/src/data_set_info.cpp File Reference . . . . .	159
7.21 compare_segmentation/src/filter_kind.cpp File Reference . . . . .	159
7.22 compare_segmentation/src/log_files.cpp File Reference . . . . .	160
7.23 compare_segmentation/src/log_info.cpp File Reference . . . . .	161
7.24 compare_segmentation/src/log_line.cpp File Reference . . . . .	162
7.25 compare_segmentation/src/main.cpp File Reference . . . . .	162
7.25.1 Function Documentation . . . . .	163
7.25.1.1 main() . . . . .	163
7.26 compare_segmentation/test/main.cpp File Reference . . . . .	164
7.26.1 Function Documentation . . . . .	165
7.26.1.1 main() . . . . .	165
7.27 counting/src/main.cpp File Reference . . . . .	165
7.27.1 Function Documentation . . . . .	166
7.27.1.1 main() . . . . .	166
7.28 counting/test/main.cpp File Reference . . . . .	167
7.28.1 Function Documentation . . . . .	168
7.28.1.1 main() . . . . .	168

---

---

7.29 csv_lib/test/main.cpp File Reference . . . . .	168
7.29.1 Function Documentation . . . . .	169
7.29.1.1 main() . . . . .	169
7.30 fix_csv/src/main.cpp File Reference . . . . .	169
7.30.1 Function Documentation . . . . .	170
7.30.1.1 main() . . . . .	170
7.31 fix_csv/test/main.cpp File Reference . . . . .	170
7.31.1 Function Documentation . . . . .	171
7.31.1.1 main() . . . . .	171
7.32 confusion_matrix/src/main.cpp File Reference . . . . .	171
7.32.1 Function Documentation . . . . .	171
7.32.1.1 main() . . . . .	172
7.33 confusion_matrix/test/main.cpp File Reference . . . . .	172
7.33.1 Function Documentation . . . . .	172
7.33.1.1 main() . . . . .	173
7.34 compare_segmentation/src/segmentation_kind.cpp File Reference . . . . .	173
7.35 compare_segmentation/test/csv_line_test.cpp File Reference . . . . .	174
7.35.1 Function Documentation . . . . .	174
7.35.1.1 TEST() . . . . .	174
7.36 compare_segmentation/test/data_set_info_test.cpp File Reference . . . . .	174
7.36.1 Function Documentation . . . . .	175
7.36.1.1 TEST() . . . . .	175
7.37 compare_segmentation/test/log_files_test.cpp File Reference . . . . .	176
7.37.1 Function Documentation . . . . .	176
7.37.1.1 TEST() [1/3] . . . . .	176
7.37.1.2 TEST() [2/3] . . . . .	177
7.37.1.3 TEST() [3/3] . . . . .	177
7.38 compare_segmentation/test/log_info_test.cpp File Reference . . . . .	177
7.38.1 Function Documentation . . . . .	178
7.38.1.1 TEST() [1/19] . . . . .	178
7.38.1.2 TEST() [2/19] . . . . .	179
7.38.1.3 TEST() [3/19] . . . . .	179
7.38.1.4 TEST() [4/19] . . . . .	180
7.38.1.5 TEST() [5/19] . . . . .	180
7.38.1.6 TEST() [6/19] . . . . .	181
7.38.1.7 TEST() [7/19] . . . . .	181
7.38.1.8 TEST() [8/19] . . . . .	182
7.38.1.9 TEST() [9/19] . . . . .	182
7.38.1.10 TEST() [10/19] . . . . .	183
7.38.1.11 TEST() [11/19] . . . . .	183
7.38.1.12 TEST() [12/19] . . . . .	184
7.38.1.13 TEST() [13/19] . . . . .	184

---

7.38.1.14 TEST() [14/19] . . . . .	185
7.38.1.15 TEST() [15/19] . . . . .	185
7.38.1.16 TEST() [16/19] . . . . .	186
7.38.1.17 TEST() [17/19] . . . . .	186
7.38.1.18 TEST() [18/19] . . . . .	187
7.38.1.19 TEST() [19/19] . . . . .	187
7.39 compare_segmentation/test/log_line_test.cpp File Reference . . . . .	187
7.39.1 Function Documentation . . . . .	188
7.39.1.1 TEST() [1/4] . . . . .	188
7.39.1.2 TEST() [2/4] . . . . .	188
7.39.1.3 TEST() [3/4] . . . . .	189
7.39.1.4 TEST() [4/4] . . . . .	189
7.40 confusion_matrix/include/configuration.hpp File Reference . . . . .	189
7.41 confusion_matrix/include/data_set_identifier.hpp File Reference . . . . .	191
7.41.1 Macro Definition Documentation . . . . .	192
7.41.1.1 CM_DATA_SET_IDENTIFIER . . . . .	192
7.41.1.2 CM_DATA_SET_IDENTIFIER_X . . . . .	192
7.42 confusion_matrix/include imu.hpp File Reference . . . . .	193
7.42.1 Macro Definition Documentation . . . . .	194
7.42.1.1 CM_IMU . . . . .	194
7.42.1.2 CM_IMU_X [1/3] . . . . .	194
7.42.1.3 CM_IMU_X [2/3] . . . . .	194
7.42.1.4 CM_IMU_X [3/3] . . . . .	195
7.43 confusion_matrix/include/interpolated_data_set_paths.hpp File Reference . . . . .	195
7.44 confusion_matrix/include/manual_segmentation_point.hpp File Reference . . . . .	196
7.45 confusion_matrix/include/segment.hpp File Reference . . . . .	197
7.46 confusion_matrix/include/split_string.hpp File Reference . . . . .	198
7.47 confusion_matrix/src/configuration.cpp File Reference . . . . .	199
7.47.1 Macro Definition Documentation . . . . .	199
7.47.1.1 CM_CONTAINS . . . . .	199
7.47.1.2 CM_ENSURE_CONTAINS . . . . .	199
7.47.1.3 CM_ENSURE_HAS_VALUE . . . . .	200
7.48 confusion_matrix/src/data_set_identifier.cpp File Reference . . . . .	200
7.48.1 Macro Definition Documentation . . . . .	201
7.48.1.1 CM_DATA_SET_IDENTIFIER_X . . . . .	201
7.48.1.2 DSI . . . . .	201
7.49 confusion_matrix/src imu.cpp File Reference . . . . .	201
7.49.1 Macro Definition Documentation . . . . .	202
7.49.1.1 CM_IMU_X . . . . .	202
7.50 confusion_matrix/src/interpolated_data_set_paths.cpp File Reference . . . . .	202
7.51 confusion_matrix/src/manual_segmentation_point.cpp File Reference . . . . .	203
7.51.1 Macro Definition Documentation . . . . .	203

---

---

7.51.1.1 DSI . . . . .	203
7.52 confusion_matrix/src/segment.cpp File Reference . . . . .	204
7.53 confusion_matrix/src/split_string.cpp File Reference . . . . .	204
7.54 confusion_matrix/test/data_set_identifier_test.cpp File Reference . . . . .	205
7.54.1 Macro Definition Documentation . . . . .	206
7.54.1.1 DSI . . . . .	206
7.54.2 Function Documentation . . . . .	206
7.54.2.1 TEST() . . . . .	206
7.55 confusion_matrix/test/interpolated_data_set_paths_test.cpp File Reference . . . . .	207
7.55.1 Function Documentation . . . . .	207
7.55.1.1 TEST() . . . . .	207
7.56 confusion_matrix/test/manual_segmentation_point_test.cpp File Reference . . . . .	208
7.56.1 Macro Definition Documentation . . . . .	208
7.56.1.1 DSI . . . . .	208
7.56.2 Function Documentation . . . . .	209
7.56.2.1 TEST() [1/11] . . . . .	209
7.56.2.2 TEST() [2/11] . . . . .	209
7.56.2.3 TEST() [3/11] . . . . .	209
7.56.2.4 TEST() [4/11] . . . . .	210
7.56.2.5 TEST() [5/11] . . . . .	210
7.56.2.6 TEST() [6/11] . . . . .	210
7.56.2.7 TEST() [7/11] . . . . .	210
7.56.2.8 TEST() [8/11] . . . . .	210
7.56.2.9 TEST() [9/11] . . . . .	211
7.56.2.10 TEST() [10/11] . . . . .	211
7.56.2.11 TEST() [11/11] . . . . .	211
7.57 confusion_matrix/test/segment_test.cpp File Reference . . . . .	211
7.57.1 Macro Definition Documentation . . . . .	212
7.57.1.1 EXPECT_SEGMENTATION_POINTS . . . . .	212
7.57.2 Function Documentation . . . . .	212
7.57.2.1 TEST() . . . . .	212
7.58 confusion_matrix/test/split_string_test.cpp File Reference . . . . .	213
7.58.1 Function Documentation . . . . .	213
7.58.1.1 TEST() . . . . .	213
7.59 counting/include/above_threshold.hpp File Reference . . . . .	214
7.60 counting/include/average_comparison_value_calculator.hpp File Reference . . . . .	215
7.61 counting/include/half_maximum_comparison_value_calculator.hpp File Reference . . . . .	215
7.62 counting/include/is_relevant.hpp File Reference . . . . .	216
7.63 counting/include/percentage_of.hpp File Reference . . . . .	217
7.64 counting/include/run_above_threshold.hpp File Reference . . . . .	218
7.65 counting/src/above_threshold.cpp File Reference . . . . .	219
7.65.1 Macro Definition Documentation . . . . .	220

---

---

7.65.1.1 CL_CHANNEL_X . . . . .	220
7.65.2 Variable Documentation . . . . .	220
7.65.2.1 channel . . . . .	221
7.65.2.2 channelAccessor . . . . .	221
7.66 counting/src/average_comparison_value_calculator.cpp File Reference . . . . .	221
7.67 counting/src/half_maximum_comparison_value_calculator.cpp File Reference . . . . .	222
7.68 counting/src/run_above_threshold.cpp File Reference . . . . .	222
7.69 counting/test/above_threshold_test.cpp File Reference . . . . .	223
7.69.1 Macro Definition Documentation . . . . .	223
7.69.1.1 EXPECT_LONG_DOUBLE_EQ . . . . .	224
7.69.2 Function Documentation . . . . .	224
7.69.2.1 TEST() . . . . .	224
7.70 counting/test/percentage_of_test.cpp File Reference . . . . .	225
7.70.1 Macro Definition Documentation . . . . .	225
7.70.1.1 EXPECT_LONG_DOUBLE_EQ . . . . .	225
7.70.2 Function Documentation . . . . .	225
7.70.2.1 TEST() . . . . .	226
7.71 csv_lib/include/cl/channel.hpp File Reference . . . . .	226
7.71.1 Macro Definition Documentation . . . . .	227
7.71.1.1 CL_CHANNEL . . . . .	228
7.71.1.2 CL_CHANNEL_X [1/4] . . . . .	228
7.71.1.3 CL_CHANNEL_X [2/4] . . . . .	228
7.71.1.4 CL_CHANNEL_X [3/4] . . . . .	228
7.71.1.5 CL_CHANNEL_X [4/4] . . . . .	229
7.72 csv_lib/include/cl/column.hpp File Reference . . . . .	229
7.72.1 Macro Definition Documentation . . . . .	230
7.72.1.1 CL_SPECIALIZE_COL_TRAITS . . . . .	231
7.73 csv_lib/include/cl/data_point.hpp File Reference . . . . .	231
7.74 csv_lib/include/cl/data_set.hpp File Reference . . . . .	232
7.75 csv_lib/include/cl/dos2unix.hpp File Reference . . . . .	233
7.76 csv_lib/include/cl/error.hpp File Reference . . . . .	234
7.76.1 Macro Definition Documentation . . . . .	234
7.76.1.1 CL_ERROR_KIND . . . . .	235
7.76.1.2 CL_ERROR_KIND_X . . . . .	235
7.76.1.3 CL_UNEXPECTED . . . . .	235
7.77 csv_lib/include/cl/exception.hpp File Reference . . . . .	235
7.77.1 Macro Definition Documentation . . . . .	236
7.77.1.1 CL_THROW . . . . .	236
7.77.1.2 CL_THROW_FMT . . . . .	236
7.78 csv_lib/include/cl/fs/directory_listing.hpp File Reference . . . . .	237
7.79 csv_lib/include/cl/fs/file.hpp File Reference . . . . .	238
7.80 csv_lib/include/cl/fs/file_stream.hpp File Reference . . . . .	239

---

---

7.81 csv_lib/include/cl/fs/path.hpp File Reference . . . . .	240
7.82 csv_lib/include/cl/fs/separator.hpp File Reference . . . . .	240
7.82.1 Macro Definition Documentation . . . . .	241
7.82.1.1 CL_FS_SEPARATOR . . . . .	241
7.83 csv_lib/include/cl/fs/windows.hpp File Reference . . . . .	242
7.83.1 Detailed Description . . . . .	243
7.84 csv_lib/include/cl/process.hpp File Reference . . . . .	243
7.85 csv_lib/include/cl/read_csv_file.hpp File Reference . . . . .	244
7.86 csv_lib/include/cl/s2n.hpp File Reference . . . . .	245
7.87 csv_lib/include/cl/sensor.hpp File Reference . . . . .	245
7.87.1 Macro Definition Documentation . . . . .	247
7.87.1.1 CL_SENSOR . . . . .	247
7.87.1.2 CL_SENSOR_X [1/2] . . . . .	247
7.87.1.3 CL_SENSOR_X [2/2] . . . . .	247
7.88 csv_lib/include/cl/to_string.hpp File Reference . . . . .	248
7.89 csv_lib/include/cl/use_unbuffered_io.hpp File Reference . . . . .	249
7.90 csv_lib/src/cl/channel.cpp File Reference . . . . .	249
7.90.1 Macro Definition Documentation . . . . .	250
7.90.1.1 CL_CHANNEL_X [1/2] . . . . .	250
7.90.1.2 CL_CHANNEL_X [2/2] . . . . .	250
7.91 csv_lib/src/cl/data_point.cpp File Reference . . . . .	251
7.91.1 Function Documentation . . . . .	251
7.91.1.1 channel() . . . . .	251
7.91.1.2 fileName() . . . . .	252
7.91.1.3 sensor() . . . . .	252
7.91.1.4 time() . . . . .	253
7.91.1.5 value() . . . . .	253
7.92 csv_lib/src/cl/data_set.cpp File Reference . . . . .	254
7.93 csv_lib/src/cl/dos2unix.cpp File Reference . . . . .	254
7.94 csv_lib/src/cl/error.cpp File Reference . . . . .	255
7.94.1 Macro Definition Documentation . . . . .	255
7.94.1.1 CL_ERROR_KIND_X . . . . .	256
7.95 csv_lib/src/cl/exception.cpp File Reference . . . . .	256
7.96 csv_lib/src/cl/fs/directory_listing.cpp File Reference . . . . .	256
7.97 csv_lib/src/cl/fs/file.cpp File Reference . . . . .	257
7.98 csv_lib/src/cl/fs/file_stream.cpp File Reference . . . . .	257
7.99 csv_lib/src/cl/fs/path.cpp File Reference . . . . .	258
7.100 csv_lib/src/cl/fs/windows.cpp File Reference . . . . .	259
7.101 csv_lib/src/cl/process.cpp File Reference . . . . .	259
7.102 csv_lib/src/cl/read_csv_file.cpp File Reference . . . . .	260
7.103 csv_lib/src/cl/sensor.cpp File Reference . . . . .	261
7.103.1 Macro Definition Documentation . . . . .	261

---

---

7.103.1.1 CL_SENSOR_X . . . . .	261
7.104 csv_lib/src/cl/use_unbuffered_io.cpp File Reference . . . . .	262
7.105 csv_lib/test/channel_test.cpp File Reference . . . . .	262
7.105.1 Function Documentation . . . . .	263
7.105.1.1 TEST() [1/4] . . . . .	263
7.105.1.2 TEST() [2/4] . . . . .	263
7.105.1.3 TEST() [3/4] . . . . .	263
7.105.1.4 TEST() [4/4] . . . . .	264
7.106 csv_lib/test/column_test.cpp File Reference . . . . .	264
7.106.1 Function Documentation . . . . .	265
7.106.1.1 TEST() [1/2] . . . . .	265
7.106.1.2 TEST() [2/2] . . . . .	265
7.107 csv_lib/test/data_point_test.cpp File Reference . . . . .	266
7.107.1 Function Documentation . . . . .	266
7.107.1.1 TEST() [1/2] . . . . .	266
7.107.1.2 TEST() [2/2] . . . . .	267
7.107.2 Variable Documentation . . . . .	267
7.107.2.1 dp . . . . .	267
7.108 csv_lib/test/data_set_test.cpp File Reference . . . . .	268
7.108.1 Macro Definition Documentation . . . . .	268
7.108.1.1 EXPECT_LONG_DOUBLE_EQ . . . . .	268
7.108.2 Function Documentation . . . . .	268
7.108.2.1 TEST() [1/4] . . . . .	269
7.108.2.2 TEST() [2/4] . . . . .	269
7.108.2.3 TEST() [3/4] . . . . .	270
7.108.2.4 TEST() [4/4] . . . . .	271
7.109 csv_lib/test/directory_listing_test.cpp File Reference . . . . .	272
7.109.1 Function Documentation . . . . .	273
7.109.1.1 TEST() [1/3] . . . . .	273
7.109.1.2 TEST() [2/3] . . . . .	273
7.109.1.3 TEST() [3/3] . . . . .	274
7.110 csv_lib/test/error_test.cpp File Reference . . . . .	274
7.110.1 Function Documentation . . . . .	275
7.110.1.1 TEST() [1/4] . . . . .	275
7.110.1.2 TEST() [2/4] . . . . .	275
7.110.1.3 TEST() [3/4] . . . . .	275
7.110.1.4 TEST() [4/4] . . . . .	275
7.110.2 Variable Documentation . . . . .	275
7.110.2.1 error . . . . .	276
7.111 csv_lib/test/exception_test.cpp File Reference . . . . .	276
7.111.1 Function Documentation . . . . .	276
7.111.1.1 TEST() . . . . .	276

---

7.112 csv/lib/test/read_csv_file_test.cpp File Reference . . . . .	277
7.112.1 Function Documentation . . . . .	277
7.112.1.1 TEST() [1/2] . . . . .	277
7.112.1.2 TEST() [2/2] . . . . .	278
7.113 csv/lib/test/s2n_test.cpp File Reference . . . . .	278
7.113.1 Function Documentation . . . . .	279
7.113.1.1 TEST() [1/3] . . . . .	279
7.113.1.2 TEST() [2/3] . . . . .	279
7.113.1.3 TEST() [3/3] . . . . .	280
7.114 csv/lib/test/sensor_test.cpp File Reference . . . . .	280
7.114.1 Function Documentation . . . . .	281
7.114.1.1 TEST() [1/2] . . . . .	281
7.114.1.2 TEST() [2/2] . . . . .	281
7.115 csv/lib/test/to_string_test.cpp File Reference . . . . .	281
7.115.1 Function Documentation . . . . .	282
7.115.1.1 TEST() . . . . .	282
7.116 fix_csv/include/adjust_hardware_timestamp.hpp File Reference . . . . .	282
7.117 fix_csv/include/convert_to_unix_line_endings.hpp File Reference . . . . .	283
7.118 fix_csv/include/create_backup_file.hpp File Reference . . . . .	284
7.119 fix_csv/include/delete_non_bosch_sensors.hpp File Reference . . . . .	285
7.120 fix_csv/include/delete_out_of_bounds_values.hpp File Reference . . . . .	286
7.121 fix_csv/include/remove_zeros_from_field.hpp File Reference . . . . .	287
7.122 fix_csv/include/restore_from_backup.hpp File Reference . . . . .	288
7.123 fix_csv/include/write_file.hpp File Reference . . . . .	289
7.124 fix_csv/src/adjust_hardware_timestamp.cpp File Reference . . . . .	290
7.125 fix_csv/src/convert_to_unix_line_endings.cpp File Reference . . . . .	291
7.126 fix_csv/src/create_backup_file.cpp File Reference . . . . .	292
7.127 fix_csv/src/delete_non_bosch_sensors.cpp File Reference . . . . .	292
7.127.1 Macro Definition Documentation . . . . .	293
7.127.1.1 CL_SENSOR_X . . . . .	293
7.128 fix_csv/src/delete_out_of_bounds_values.cpp File Reference . . . . .	293
7.129 fix_csv/src/remove_zeros_from_field.cpp File Reference . . . . .	294
7.130 fix_csv/src/restore_from_backup.cpp File Reference . . . . .	294
7.131 fix_csv/src/write_file.cpp File Reference . . . . .	295
7.132 fix_csv/test/adjust_hardware_timestamp_test.cpp File Reference . . . . .	296
7.132.1 Function Documentation . . . . .	296
7.132.1.1 TEST() [1/5] . . . . .	296
7.132.1.2 TEST() [2/5] . . . . .	297
7.132.1.3 TEST() [3/5] . . . . .	297
7.132.1.4 TEST() [4/5] . . . . .	298
7.132.1.5 TEST() [5/5] . . . . .	298
7.133 fix_csv/test/remove_zeros_from_field_test.cpp File Reference . . . . .	298

---

7.133.1 Function Documentation	299
7.133.1.1 TEST() [1/6]	299
7.133.1.2 TEST() [2/6]	300
7.133.1.3 TEST() [3/6]	300
7.133.1.4 TEST() [4/6]	301
7.133.1.5 TEST() [5/6]	301
7.133.1.6 TEST() [6/6]	302
<b>Index</b>	<b>303</b>

# Chapter 1

## Namespace Index

### 1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

cl . . . . .	11
cl::fs . . . . .	24
cm . . . . .	30
cs . . . . .	37
ctg . . . . .	47
fmc . . . . .	52



# Chapter 2

## Hierarchical Index

### 2.1 Class Hierarchy

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

cm::Configuration::Builder . . . . .	59
cl::col_traits< Col > . . . . .	66
cm::Configuration . . . . .	66
cs::CsvLineBuilder . . . . .	74
cl::data_set_accessor< Chan > . . . . .	87
cs::data_set_info< Tag > . . . . .	88
cl::DataPoint . . . . .	88
cl::DataSet . . . . .	92
cl::Error . . . . .	102
std::exception	
std::runtime_error	
cl::Exception . . . . .	106
cl::fs::File . . . . .	109
cl::fs::FileStream . . . . .	115
std::hash<::cl::fs::Path > . . . . .	120
cs::LogInfo . . . . .	120
cs::LogLine . . . . .	126
cm::ManualSegmentationPoint . . . . .	130
cl::fs::Path . . . . .	136
cl::Process . . . . .	142



# Chapter 3

## Class Index

### 3.1 Class List

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

<code>cm::Configuration::Builder</code>	
Builder type for <code>Configuration</code>	59
<code>cl::col_traits&lt; Col &gt;</code>	66
<code>cm::Configuration</code>	
Represents a possible configuration for the Python segmentor	66
<code>cs::CsvLineBuilder</code>	
Builder for a CSV line	74
<code>cl::data_set_accessor&lt; Chan &gt;</code>	87
<code>cs::data_set_info&lt; Tag &gt;</code>	
Meta function for data set tags	88
<code>cl::DataPoint</code>	88
<code>cl::DataSet</code>	92
<code>cl::Error</code>	102
<code>cl::Exception</code>	106
<code>cl::fs::File</code>	
Represents a file	109
<code>cl::fs::FileStream</code>	
A binary file stream	115
<code>std::hash&lt;::cl::fs::Path&gt;</code>	120
<code>cs::LogInfo</code>	
Information about a log file	120
<code>cs::LogLine</code>	
A line out of a log file	126
<code>cm::ManualSegmentationPoint</code>	
Type used to represent a manual segmentation point	130
<code>cl::fs::Path</code>	
A filesystem path	136
<code>cl::Process</code>	142



# Chapter 4

## File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

compare_segmentation/include/csv_line.hpp . . . . .	150
compare_segmentation/include/data_set_info.hpp . . . . .	151
compare_segmentation/include/filter_kind.hpp . . . . .	153
compare_segmentation/include/log_files.hpp . . . . .	154
compare_segmentation/include/log_info.hpp . . . . .	154
compare_segmentation/include/log_line.hpp . . . . .	155
compare_segmentation/include/paths.hpp . . . . .	156
compare_segmentation/include/segmentation_kind.hpp . . . . .	157
compare_segmentation/src/csv_line.cpp . . . . .	158
compare_segmentation/src/data_set_info.cpp . . . . .	159
compare_segmentation/src/filter_kind.cpp . . . . .	159
compare_segmentation/src/log_files.cpp . . . . .	160
compare_segmentation/src/log_info.cpp . . . . .	161
compare_segmentation/src/log_line.cpp . . . . .	162
compare_segmentation/src/main.cpp . . . . .	162
compare_segmentation/src/segmentation_kind.cpp . . . . .	173
compare_segmentation/test/csv_line_test.cpp . . . . .	174
compare_segmentation/test/data_set_info_test.cpp . . . . .	174
compare_segmentation/test/log_files_test.cpp . . . . .	176
compare_segmentation/test/log_info_test.cpp . . . . .	177
compare_segmentation/test/log_line_test.cpp . . . . .	187
compare_segmentation/test/main.cpp . . . . .	164
confusion_matrix/include/configuration.hpp . . . . .	189
confusion_matrix/include/data_set_identifier.hpp . . . . .	191
confusion_matrix/include imu.hpp . . . . .	193
confusion_matrix/include/interpolated_data_set_paths.hpp . . . . .	195
confusion_matrix/include/manual_segmentation_point.hpp . . . . .	196
confusion_matrix/include/segment.hpp . . . . .	197
confusion_matrix/include/split_string.hpp . . . . .	198
confusion_matrix/src/configuration.cpp . . . . .	199
confusion_matrix/src/data_set_identifier.cpp . . . . .	200
confusion_matrix/src imu.cpp . . . . .	201
confusion_matrix/src/interpolated_data_set_paths.cpp . . . . .	202
confusion_matrix/src/main.cpp . . . . .	171
confusion_matrix/src/manual_segmentation_point.cpp . . . . .	203

confusion_matrix/src/segment.cpp . . . . .	204
confusion_matrix/src/split_string.cpp . . . . .	204
confusion_matrix/test/data_set_identifier_test.cpp . . . . .	205
confusion_matrix/test/interpolated_data_set_paths_test.cpp . . . . .	207
confusion_matrix/test/main.cpp . . . . .	172
confusion_matrix/test/manual_segmentation_point_test.cpp . . . . .	208
confusion_matrix/test/segment_test.cpp . . . . .	211
confusion_matrix/test/split_string_test.cpp . . . . .	213
counting/include/above_threshold.hpp . . . . .	214
counting/include/average_comparison_value_calculator.hpp . . . . .	215
counting/include/half_maximum_comparison_value_calculator.hpp . . . . .	215
counting/include/is_relevant.hpp . . . . .	216
counting/include/percentage_of.hpp . . . . .	217
counting/include/run_above_threshold.hpp . . . . .	218
counting/src/above_threshold.cpp . . . . .	219
counting/src/average_comparison_value_calculator.cpp . . . . .	221
counting/src/half_maximum_comparison_value_calculator.cpp . . . . .	222
counting/src/main.cpp . . . . .	165
counting/src/run_above_threshold.cpp . . . . .	222
counting/test/above_threshold_test.cpp . . . . .	223
counting/test/main.cpp . . . . .	167
counting/test/percentage_of_test.cpp . . . . .	225
csv_lib/include/cl/channel.hpp . . . . .	226
csv_lib/include/cl/column.hpp . . . . .	229
csv_lib/include/cl/data_point.hpp . . . . .	231
csv_lib/include/cl/data_set.hpp . . . . .	232
csv_lib/include/cl/dos2unix.hpp . . . . .	233
csv_lib/include/cl/error.hpp . . . . .	234
csv_lib/include/cl/exception.hpp . . . . .	235
csv_lib/include/cl/process.hpp . . . . .	243
csv_lib/include/cl/read_csv_file.hpp . . . . .	244
csv_lib/include/cl/s2n.hpp . . . . .	245
csv_lib/include/cl/sensor.hpp . . . . .	245
csv_lib/include/cl/to_string.hpp . . . . .	248
csv_lib/include/cl/use_unbuffered_io.hpp . . . . .	249
csv_lib/include/cl/fs/directory_listing.hpp . . . . .	237
csv_lib/include/cl/fs/file.hpp . . . . .	238
csv_lib/include/cl/fs/file_stream.hpp . . . . .	239
csv_lib/include/cl/fs/path.hpp . . . . .	240
csv_lib/include/cl/fs/sePARATOR.hpp . . . . .	240
csv_lib/include/cl/fs/windows.hpp . . . . .	240
Contains Microsoft Windows specific functions . . . . .	242
csv_lib/src/cl/channel.cpp . . . . .	249
csv_lib/src/cl/data_point.cpp . . . . .	251
csv_lib/src/cl/data_set.cpp . . . . .	254
csv_lib/src/cl/dos2unix.cpp . . . . .	254
csv_lib/src/cl/error.cpp . . . . .	255
csv_lib/src/cl/exception.hpp . . . . .	256
csv_lib/src/cl/process.hpp . . . . .	259
csv_lib/src/cl/read_csv_file.hpp . . . . .	260
csv_lib/src/cl/sensor.hpp . . . . .	261
csv_lib/src/cl/use_unbuffered_io.hpp . . . . .	262
csv_lib/src/cl/fs/directory_listing.hpp . . . . .	256
csv_lib/src/cl/fs/file.hpp . . . . .	257
csv_lib/src/cl/fs/file_stream.hpp . . . . .	257
csv_lib/src/cl/fs/path.hpp . . . . .	258
csv_lib/src/cl/fs/windows.hpp . . . . .	259
csv_lib/test/channel_test.cpp . . . . .	262

---

csv_lib/test/column_test.cpp	264
csv_lib/test/data_point_test.cpp	266
csv_lib/test/data_set_test.cpp	268
csv_lib/test/directory_listing_test.cpp	272
csv_lib/test/error_test.cpp	274
csv_lib/test/exception_test.cpp	276
csv_lib/test/main.cpp	168
csv_lib/test/read_csv_file_test.cpp	277
csv_lib/test/s2n_test.cpp	278
csv_lib/test/sensor_test.cpp	280
csv_lib/test/to_string_test.cpp	281
fix_csv/include/adjust_hardware_timestamp.hpp	282
fix_csv/include/convert_to_unix_line_endings.hpp	283
fix_csv/include/create_backup_file.hpp	284
fix_csv/include/delete_non_bosch_sensors.hpp	285
fix_csv/include/delete_out_of_bounds_values.hpp	286
fix_csv/include/remove_zeros_from_field.hpp	287
fix_csv/include/restore_from_backup.hpp	288
fix_csv/include/write_file.hpp	289
fix_csv/src/adjust_hardware_timestamp.cpp	290
fix_csv/src/convert_to_unix_line_endings.cpp	291
fix_csv/src/create_backup_file.cpp	292
fix_csv/src/delete_non_bosch_sensors.cpp	292
fix_csv/src/delete_out_of_bounds_values.cpp	293
fix_csv/src/main.cpp	169
fix_csv/src/remove_zeros_from_field.cpp	294
fix_csv/src/restore_from_backup.cpp	294
fix_csv/src/write_file.cpp	295
fix_csv/test/adjust_hardware_timestamp_test.cpp	296
fix_csv/test/main.cpp	170
fix_csv/test/remove_zeros_from_field_test.cpp	298



# Chapter 5

## Namespace Documentation

### 5.1 cl Namespace Reference

#### Namespaces

- [fs](#)

#### Classes

- struct [col\\_traits](#)
- struct [data\\_set\\_accessor](#)
- class [DataPoint](#)
- class [DataSet](#)
- class [Error](#)
- class [Exception](#)
- class [Process](#)

#### Typedefs

- template<Column Col>  
using [column\\_type](#) = typename [col\\_traits](#)< Col >::type
- template<typename Ty >  
using [Expected](#) = tl::expected< Ty, [Error](#) >

#### Enumerations

- enum [Channel](#) : std::uint64\_t { [Channel::CL\\_CHANNEL\\_X](#), [Channel::CL\\_CHANNEL\\_Y](#) }
- enum [Column](#) : std::size\_t {  
[Column::Time](#), [Column::HardwareTimestamp](#), [Column::ExtractId](#), [Column::Trigger](#),  
[Column::AccelerometerX](#), [Column::AccelerometerY](#), [Column::AccelerometerZ](#), [Column::GyroscopeX](#),  
[Column::GyroscopeY](#), [Column::GyroscopeZ](#), [Column::SamplingRate](#) }
- enum [CsvFileKind](#) { [CsvFileKind::Raw](#), [CsvFileKind::Fixed](#) }
- enum [Sensor](#) : std::uint64\_t { [Sensor::CL\\_SENSOR\\_X](#), [Sensor::CL\\_SENSOR\\_Y](#) }

## Functions

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

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

## Variables

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

### 5.1.1 Typedef Documentation

#### 5.1.1.1 column\_type

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

Definition at line 49 of file column.hpp.

### 5.1.1.2 Expected

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

Definition at line 64 of file error.hpp.

## 5.1.2 Enumeration Type Documentation

### 5.1.2.1 Channel

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

Enumerator

CL_CHANNEL←_X	
CL_CHANNEL	

Definition at line 20 of file channel.hpp.

### 5.1.2.2 Column

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

Enumerator

Time	
HardwareTimestamp	
ExtractId	
Trigger	
AccelerometerX	
AccelerometerY	
AccelerometerZ	
GyroscopeX	
GyroscopeY	
GyroscopeZ	
SamplingRate	

Definition at line 9 of file column.hpp.

### 5.1.2.3 CsvFileKind

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

Enumerator

Raw	
Fixed	

Definition at line 11 of file read\_csv\_file.hpp.

### 5.1.2.4 Sensor

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

Enumerator

CL_SENSOR_X	
CL_SENSOR_Y	

Definition at line 15 of file sensor.hpp.

## 5.1.3 Function Documentation

### 5.1.3.1 CL\_SPECIALIZE\_COL\_TRAITS() [1/11]

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

### 5.1.3.2 CL\_SPECIALIZE\_COL\_TRAITS() [2/11]

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

**5.1.3.3 CL\_SPECIALIZE\_COL\_TRAITS() [3/11]**

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

**5.1.3.4 CL\_SPECIALIZE\_COL\_TRAITS() [4/11]**

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

**5.1.3.5 CL\_SPECIALIZE\_COL\_TRAITS() [5/11]**

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

**5.1.3.6 CL\_SPECIALIZE\_COL\_TRAITS() [6/11]**

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

**5.1.3.7 CL\_SPECIALIZE\_COL\_TRAITS() [7/11]**

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

**5.1.3.8 CL\_SPECIALIZE\_COL\_TRAITS() [8/11]**

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

### 5.1.3.9 CL\_SPECIALIZE\_COL\_TRAITS() [9/11]

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

### 5.1.3.10 CL\_SPECIALIZE\_COL\_TRAITS() [10/11]

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

### 5.1.3.11 CL\_SPECIALIZE\_COL\_TRAITS() [11/11]

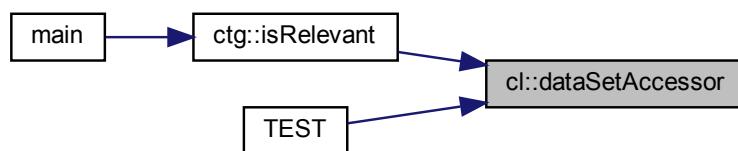
```
cl::CL_SPECIALIZE_COL_TRAITS (
    Column::Trigger ,
    std::uint64_t )
```

### 5.1.3.12 dataSetAccessor()

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

Definition at line 15 of file channel.cpp.

Here is the caller graph for this function:



### 5.1.3.13 dos2unix()

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

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

**Parameters**

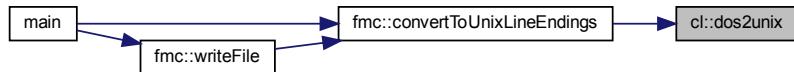
<i>p</i>	The beginning of the data to convert.
<i>size</i>	The size of the data to convert in bytes.

**Returns**

The resulting byte array.

Definition at line 4 of file dos2unix.cpp.

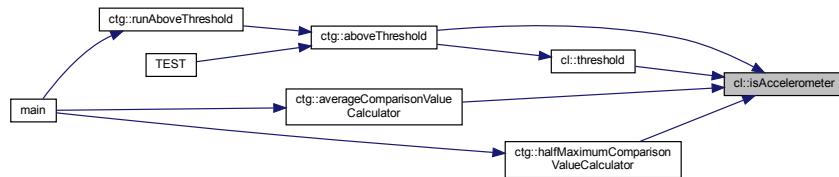
Here is the caller graph for this function:

**5.1.3.14 isAccelerometer()**

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

Definition at line 45 of file channel.cpp.

Here is the caller graph for this function:

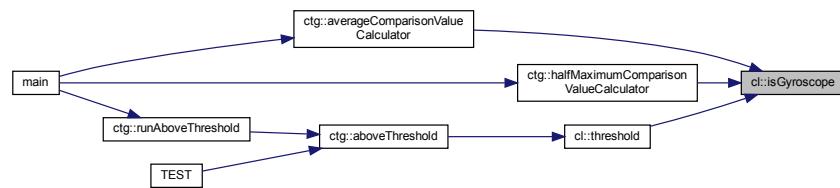


### 5.1.3.15 `isGyroscope()`

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

Definition at line 50 of file channel.cpp.

Here is the caller graph for this function:



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

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

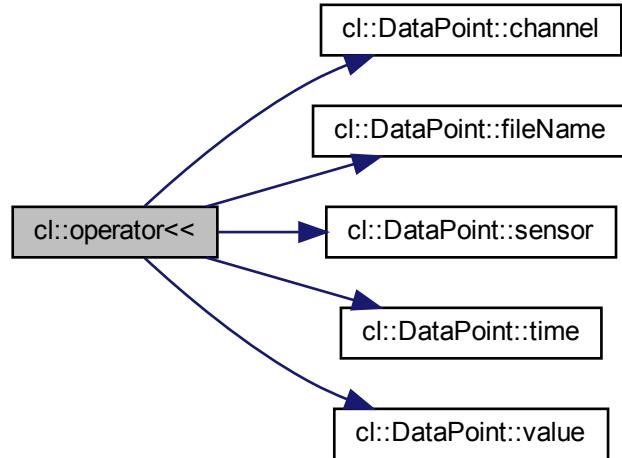
Definition at line 32 of file channel.cpp.

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

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

Definition at line 10 of file data\_point.cpp.

Here is the call graph for this function:

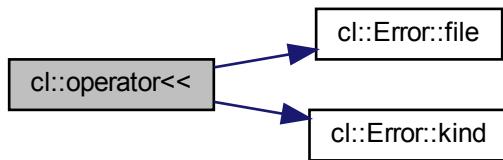


### 5.1.3.18 `operator<<()` [3/4]

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

Definition at line 30 of file error.cpp.

Here is the call graph for this function:



### 5.1.3.19 operator<<() [4/4]

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

Definition at line 8 of file sensor.cpp.

Here is the call graph for this function:

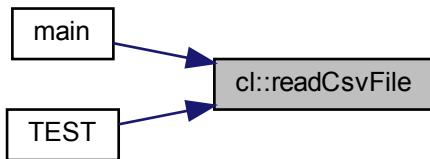


### 5.1.3.20 readCsvFile()

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

Definition at line 50 of file read\_csv\_file.cpp.

Here is the caller graph for this function:



### 5.1.3.21 s2n()

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

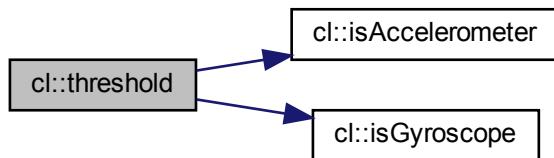
Definition at line 16 of file s2n.hpp.

### 5.1.3.22 threshold()

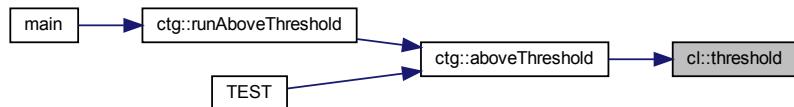
```
long double cl::threshold (
    Channel channel )
```

Definition at line 55 of file channel.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

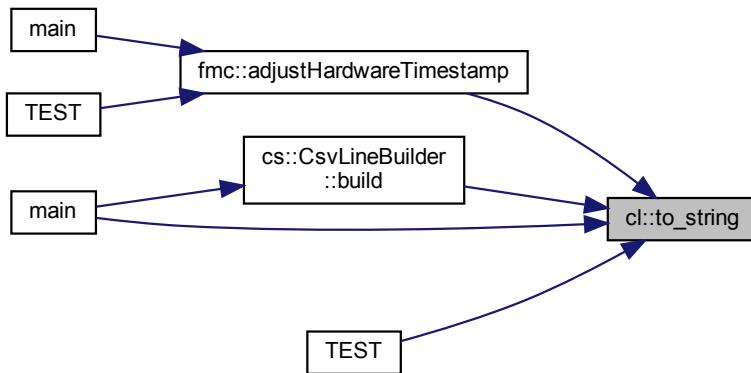


### 5.1.3.23 `to_string()`

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

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

Here is the caller graph for this function:



### 5.1.3.24 `useUnbufferedIo()`

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

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

Here is the caller graph for this function:



## 5.1.4 Variable Documentation

#### 5.1.4.1 accelerometerThreshold

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

Definition at line 61 of file channel.hpp.

#### 5.1.4.2 channelCount

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

##### Initial value:

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

Definition at line 26 of file channel.hpp.

#### 5.1.4.3 channels

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

##### Initial value:

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

Definition at line 32 of file channel.hpp.

#### 5.1.4.4 column\_index

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

Definition at line 46 of file column.hpp.

#### 5.1.4.5 data\_set\_accessor\_v

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

Definition at line 51 of file channel.hpp.

### 5.1.4.6 gyroscopeThreshold

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

Definition at line 62 of file channel.hpp.

### 5.1.4.7 sensors

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

#### Initial value:

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

Definition at line 21 of file sensor.hpp.

## 5.2 cl::fs Namespace Reference

### Classes

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

### Enumerations

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

### Functions

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

## 5.2.1 Enumeration Type Documentation

### 5.2.1.1 DirectoryListingOption

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

Options for directoryListing.

#### Enumerator

None	No option
ExcludeDotAndDotDot	Exclude the . and .. directories

Definition at line 13 of file directory\_listing.hpp.

## 5.2.2 Function Documentation

### 5.2.2.1 directoryListing()

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

Creates a listing of the contents of a directory.

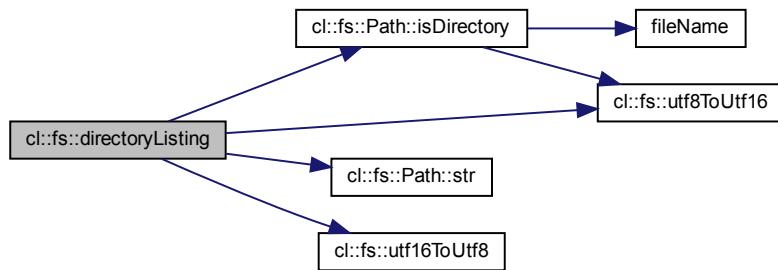
#### Parameters

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

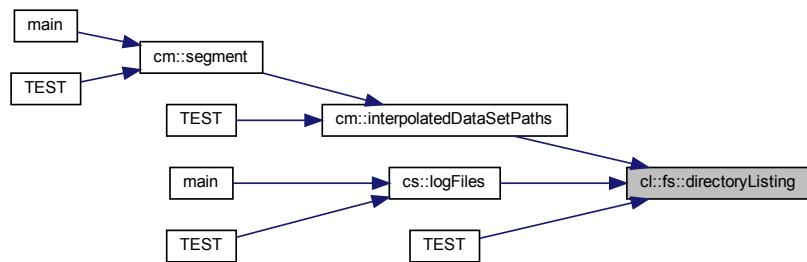
Definition at line 24 of file directory\_listing.cpp.

---

Here is the call graph for this function:



Here is the caller graph for this function:



### 5.2.2.2 formatError()

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

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

#### Parameters

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

#### Returns

The resulting UTF-16 encoded wide character string.

**Note**

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

**Warning**

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

Definition at line 89 of file windows.cpp.

### 5.2.2.3 operator<()

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

**Parameters**

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

**Returns**

true if lhs < rhs; otherwise false.

Definition at line 27 of file path.cpp.

### 5.2.2.4 operator<<()

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

**Parameters**

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

**Returns**

*os*

Definition at line 22 of file path.cpp.

### 5.2.2.5 operator==( )

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

#### Parameters

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

#### Returns

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

Definition at line 32 of file path.cpp.

### 5.2.2.6 utf16ToUtf8()

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

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

#### Parameters

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

#### Returns

The resulting UTF-8 string.

#### Note

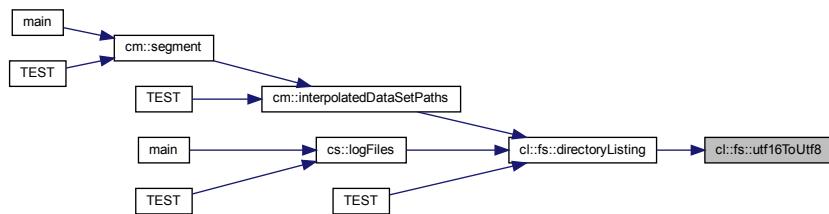
Most WINAPIs expect UTF-16 encoded wide character strings, but we don't want to pollute the code base with UTF-16 strings.

#### Warning

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

Definition at line 61 of file windows.cpp.

Here is the caller graph for this function:



### 5.2.2.7 utf8ToUtf16()

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

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

#### Parameters

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

#### Returns

The resulting UTF-16 string.

#### Note

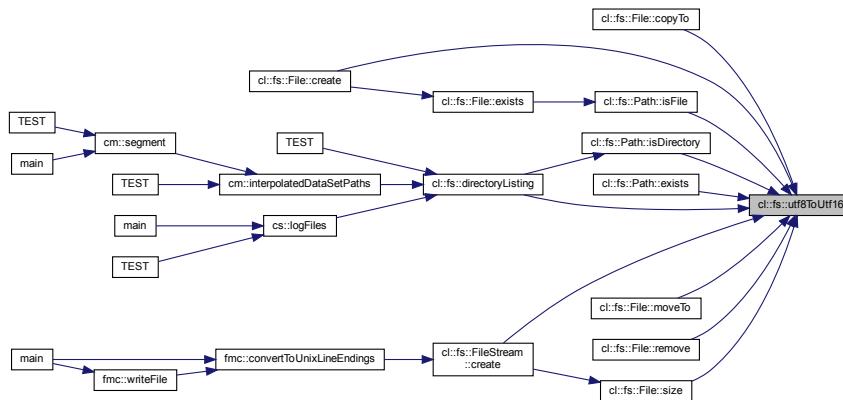
Most WINAPIs expect UTF-16 encoded wide character strings, but we don't want to pollute the code base with UTF-16 strings.

#### Warning

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

Definition at line 35 of file windows.cpp.

Here is the caller graph for this function:



## 5.3 cm Namespace Reference

### Classes

- class [Configuration](#)  
*Represents a possible configuration for the Python segmentor.*
- class [ManualSegmentationPoint](#)  
*Type used to represent a manual segmentation point.*

### Enumerations

- enum [DataSetIdentifier](#) { `DataSetIdentifier::CM_DATA_SET_IDENTIFIER_X`, `DataSetIdentifier::CM_DATA_SET_IDENTIFIER_Y` }
- enum [Imu](#) { `Imu::CM_IMU_X`, `Imu::CM_IMU` }

### Functions

- `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::ostream & operator<< (std::ostream &os, Imu imu)`
- `std::vector< cl::fs::Path > interpolatedDataSetPaths ()`  
*Returns the paths to the interpolated data sets.*
- `std::unordered_map< cl::fs::Path, std::vector< std::uint64_t > > segment ()`
- `std::vector< std::string > splitString (std::string string, std::string_view splitBy)`  
*Splits string by splitBy.*
- `bool operator== (const ManualSegmentationPoint &lhs, const ManualSegmentationPoint &rhs) noexcept`
- `bool operator!= (const ManualSegmentationPoint &lhs, const ManualSegmentationPoint &rhs) noexcept`
- `std::ostream & operator<< (std::ostream &os, const ManualSegmentationPoint &manualSegmentationPoint)`

## Variables

- `constexpr std::size_t imuCount`
- `constexpr std::array<Imu, imuCount> imus`

### 5.3.1 Enumeration Type Documentation

#### 5.3.1.1 DataSetIdentifier

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

Enumerator

CM_DATA_SET_IDENTIFIER	↔	
CM_DATA_SET_IDENTIFIER		

Definition at line 30 of file data\_set\_identifier.hpp.

#### 5.3.1.2 Imu

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

Enumerator

CM_IMU	↔	
CM_IMU		

Definition at line 14 of file imu.hpp.

### 5.3.2 Function Documentation

#### 5.3.2.1 interpolatedDataSetPaths()

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

Returns the paths to the interpolated data sets.

Returns

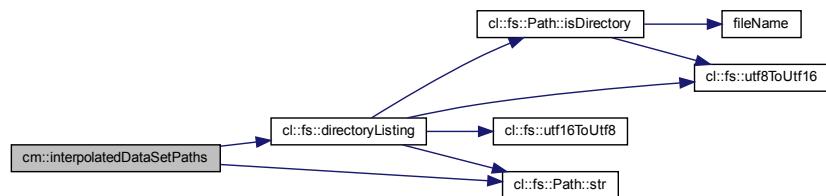
The interpolated data set paths.

## Exceptions

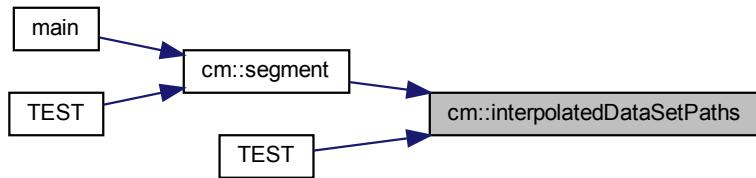
<code>cl::Exception</code>	on error.
----------------------------	-----------

Definition at line 13 of file interpolated\_data\_set\_paths.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



### 5.3.2.2 operator"!=()

```

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

#### Parameters

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

#### Returns

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

Definition at line 139 of file manual\_segmentation\_point.cpp.

### 5.3.2.3 operator<<() [1/3]

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

#### Parameters

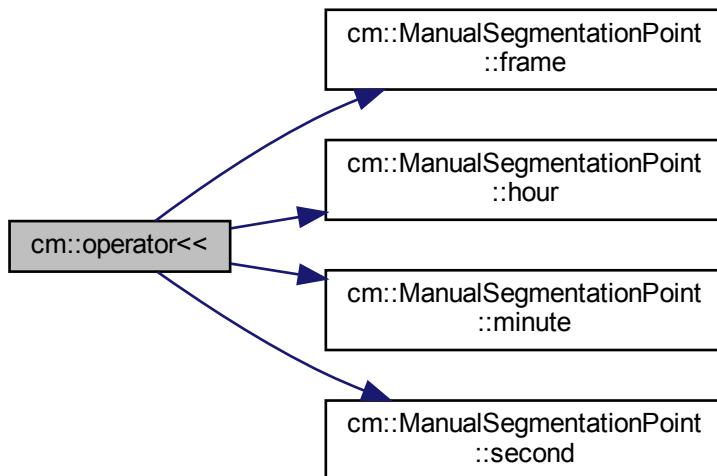
<i>os</i>	The ostream to print to
<i>manualSegmentationPoint</i>	The <code>ManualSegmentationPoint</code> to print.

#### Returns

*os*

Definition at line 146 of file manual\_segmentation\_point.cpp.

Here is the call graph for this function:



### 5.3.2.4 operator<<() [2/3]

```
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 28 of file data\_set\_identifier.cpp.

**5.3.2.5 operator<<() [3/3]**

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

Definition at line 25 of file imu.cpp.

**5.3.2.6 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 equal to *rhs*; false otherwise.

Definition at line 131 of file manual\_segmentation\_point.cpp.

**5.3.2.7 segment()**

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

Invokes Python to segment the interpolated data sets.

**Returns**

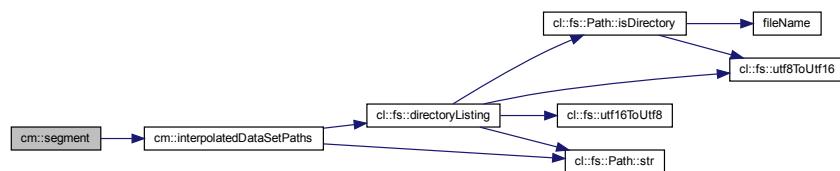
A map that maps the paths to the interpolated data sets to vectors of the hardware timestamps (in milliseconds) that are segmentation points.

### Exceptions

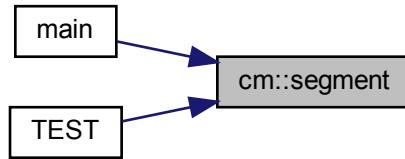
<i>cl::Exception</i>	if an error occurs.
----------------------	---------------------

Definition at line 105 of file segment.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



### 5.3.2.8 splitString()

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

Splits *string* by *splitBy*.

#### Parameters

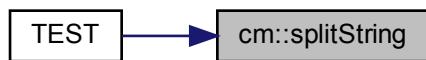
<i>string</i>	The string to split.
<i>splitBy</i>	What to split <i>string</i> by.

**Returns**

The resulting strings.

Definition at line 8 of file split\_string.cpp.

Here is the caller graph for this function:



### 5.3.2.9 toDataSetIdentifier()

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

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

**Parameters**

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

**Returns**

The resulting DataSetIdentifier.

**Exceptions**

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

Definition at line 33 of file data\_set\_identifier.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



### 5.3.3 Variable Documentation

#### 5.3.3.1 imuCount

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

##### Initial value:

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

Definition at line 20 of file imu.hpp.

#### 5.3.3.2 imus

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

##### Initial value:

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

Definition at line 26 of file imu.hpp.

## 5.4 cs Namespace Reference

### Classes

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

## Enumerations

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

## Functions

- `PL_DEFINE_EXCEPTION_TYPE` (`NoSuchDataSetException`, `std::logic_error`)
- `CS_SPECIALIZE_DATA_SET_INFO` (`Felix1`, "11.17.39", 24)
- `CS_SPECIALIZE_DATA_SET_INFO` (`Felix2`, "12.50.00", 20)
- `CS_SPECIALIZE_DATA_SET_INFO` (`Felix3`, "13.00.09", 15)
- `CS_SPECIALIZE_DATA_SET_INFO` (`Marcelle1`, "14.59.59", 10)
- `CS_SPECIALIZE_DATA_SET_INFO` (`Marcelle2`, "15.13.22", 16)
- `CS_SPECIALIZE_DATA_SET_INFO` (`Marcelle3`, "15.31.36", 18)
- `CS_SPECIALIZE_DATA_SET_INFO` (`Mike1`, "14.07.33", 26)
- `CS_SPECIALIZE_DATA_SET_INFO` (`Mike2`, "14.14.32", 22)
- `CS_SPECIALIZE_DATA_SET_INFO` (`Mike3`, "14.20.28", 18)
- `CS_SPECIALIZE_DATA_SET_INFO` (`Andre1`, "Andre\_liegestuetzen1", 27)
- `CS_SPECIALIZE_DATA_SET_INFO` (`Andre2`, "Andre\_liegestuetzen2", 20)
- `CS_SPECIALIZE_DATA_SET_INFO` (`Andre3`, "Andre\_liegestuetzen3", 17)
- `CS_SPECIALIZE_DATA_SET_INFO` (`AndreSquats1`, "Andre\_Squats", 30)
- `CS_SPECIALIZE_DATA_SET_INFO` (`AndreSquats2`, "Andre\_Squats2", 49)
- `CS_SPECIALIZE_DATA_SET_INFO` (`Jan1`, "Jan\_liegestuetzen1", 25)
- `CS_SPECIALIZE_DATA_SET_INFO` (`Jan2`, "Jan\_liegestuetzen2", 19)
- `CS_SPECIALIZE_DATA_SET_INFO` (`Jan3`, "Jan\_liegestuetzen3", 13)
- `CS_SPECIALIZE_DATA_SET_INFO` (`Lucas1`, "Lucas\_liegestuetzen1", 24)
- `CS_SPECIALIZE_DATA_SET_INFO` (`Lucas2`, "Lucas\_liegestuetzen2", 19)
- `CS_SPECIALIZE_DATA_SET_INFO` (`Lucas3`, "Lucas\_liegestuetzen3", 11)
- `std::uint64_t repetitionCount` (`pl::string_view dataSet`)  
*Fetches the repetition count for a given data set identified by its string.*
- `std::ostream & operator<<` (`std::ostream &os`, `FilterKind filterKind`)  
*Prints a FilterKind to an ostream.*
- `cl::Expected< std::vector< cl::fs::Path > > logFiles` (`pl::string_view directoryPath`)  
*Fetches the paths to the log files in the given directory.*
- `std::ostream & operator<<` (`std::ostream &os`, `SegmentationKind segmentationKind`)  
*Prints a SegmentationKind to an ostream.*
- `bool operator==` (`const LogInfo &lhs`, `const LogInfo &rhs`) `noexcept`
- `bool operator!=` (`const LogInfo &lhs`, `const LogInfo &rhs`) `noexcept`
- `std::ostream & operator<<` (`std::ostream &os`, `const LogInfo &logInfo`)

## Variables

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

### 5.4.1 Enumeration Type Documentation

#### 5.4.1.1 FilterKind

```
enum cs::FilterKind [strong]
```

Type for the different kinds of filters.

Enumerator

Butterworth	
MovingAverage	

Definition at line 9 of file filter\_kind.hpp.

#### 5.4.1.2 SegmentationKind

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

The segmentation kind.

Enumerator

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

Definition at line 12 of file segmentation\_kind.hpp.

### 5.4.2 Function Documentation

#### 5.4.2.1 CS\_SPECIALIZE\_DATA\_SET\_INFO() [1/20]

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

#### 5.4.2.2 CS\_SPECIALIZE\_DATA\_SET\_INFO() [2/20]

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

#### 5.4.2.3 CS\_SPECIALIZE\_DATA\_SET\_INFO() [3/20]

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

#### 5.4.2.4 CS\_SPECIALIZE\_DATA\_SET\_INFO() [4/20]

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

#### 5.4.2.5 CS\_SPECIALIZE\_DATA\_SET\_INFO() [5/20]

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

#### 5.4.2.6 CS\_SPECIALIZE\_DATA\_SET\_INFO() [6/20]

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

#### 5.4.2.7 CS\_SPECIALIZE\_DATA\_SET\_INFO() [7/20]

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

**5.4.2.8 CS\_SPECIALIZE\_DATA\_SET\_INFO() [8/20]**

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

**5.4.2.9 CS\_SPECIALIZE\_DATA\_SET\_INFO() [9/20]**

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

**5.4.2.10 CS\_SPECIALIZE\_DATA\_SET\_INFO() [10/20]**

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

**5.4.2.11 CS\_SPECIALIZE\_DATA\_SET\_INFO() [11/20]**

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

**5.4.2.12 CS\_SPECIALIZE\_DATA\_SET\_INFO() [12/20]**

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

**5.4.2.13 CS\_SPECIALIZE\_DATA\_SET\_INFO() [13/20]**

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

#### 5.4.2.14 CS\_SPECIALIZE\_DATA\_SET\_INFO() [14/20]

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

#### 5.4.2.15 CS\_SPECIALIZE\_DATA\_SET\_INFO() [15/20]

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

#### 5.4.2.16 CS\_SPECIALIZE\_DATA\_SET\_INFO() [16/20]

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

#### 5.4.2.17 CS\_SPECIALIZE\_DATA\_SET\_INFO() [17/20]

```
cs::CS_SPECIALIZE_DATA_SET_INFO (
    Marcellle3 ,
    "15.31.36" ,
    18  )
```

#### 5.4.2.18 CS\_SPECIALIZE\_DATA\_SET\_INFO() [18/20]

```
cs::CS_SPECIALIZE_DATA_SET_INFO (
    Mike1 ,
    "14.07.33" ,
    26  )
```

#### 5.4.2.19 CS\_SPECIALIZE\_DATA\_SET\_INFO() [19/20]

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

### 5.4.2.20 CS\_SPECIALIZE\_DATA\_SET\_INFO() [20/20]

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

### 5.4.2.21 logFiles()

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

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

#### Parameters

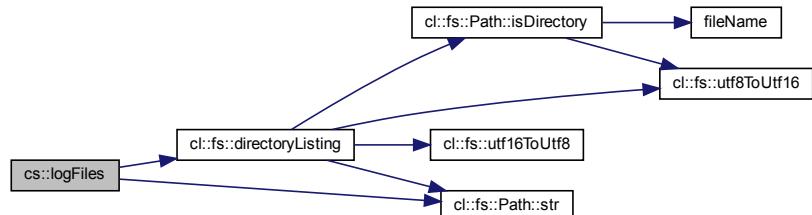
<i>directoryPath</i>	The path to a directory to search for log files.
----------------------	--

#### Returns

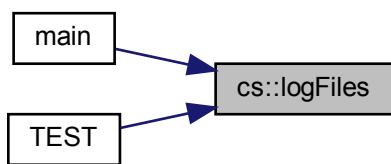
The log files found or an error.

Definition at line 9 of file log\_files.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



### 5.4.2.22 operator"!=()

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

#### Parameters

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

#### Returns

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

Definition at line 287 of file log\_info.cpp.

### 5.4.2.23 operator<<() [1/3]

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

#### Parameters

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

#### Returns

*os*

Definition at line 292 of file log\_info.cpp.

### 5.4.2.24 operator<<() [2/3]

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

Prints a FilterKind to an ostream.

**Parameters**

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

**Returns**

os

Definition at line 6 of file filter\_kind.cpp.

**5.4.2.25 operator<<() [3/3]**

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

Prints a SegmentationKind to an ostream.

**Parameters**

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

**Returns**

os

Definition at line 6 of file segmentation\_kind.cpp.

**5.4.2.26 operator==( )**

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

**Parameters**

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

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

Definition at line 264 of file log\_info.cpp.

#### 5.4.2.27 PL\_DEFINE\_EXCEPTION\_TYPE()

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

#### 5.4.2.28 repetitionCount()

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

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

##### Parameters

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

##### Returns

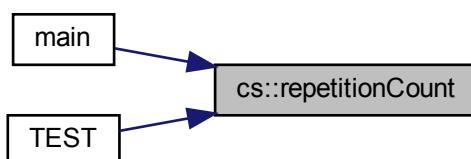
The repetition count of `dataSet`.

##### Warning

`dataSet` may not be invalid!

Definition at line 10 of file data\_set\_info.cpp.

Here is the caller graph for this function:



### 5.4.3 Variable Documentation

#### 5.4.3.1 logPath

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

Relative path to the directory containing the preprocessed log files.

Definition at line 9 of file paths.hpp.

#### 5.4.3.2 oldLogPath

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

Relative path to the directory containing the old log files.

Definition at line 14 of file paths.hpp.

## 5.5 ctg Namespace Reference

### Functions

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

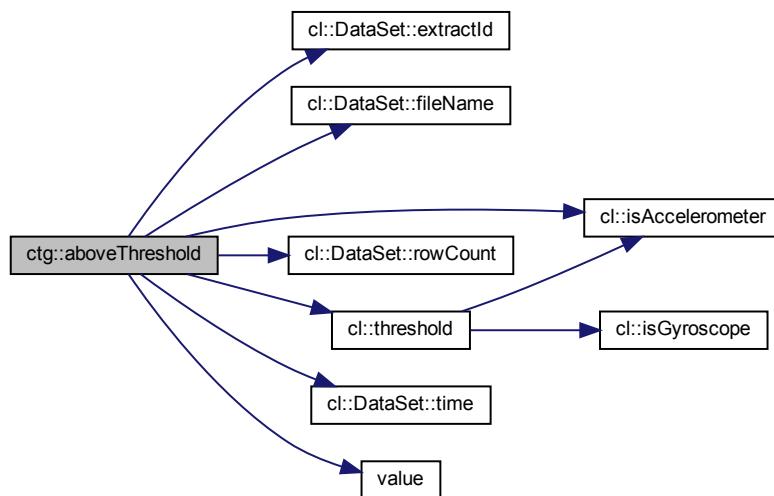
### 5.5.1 Function Documentation

### 5.5.1.1 aboveThreshold()

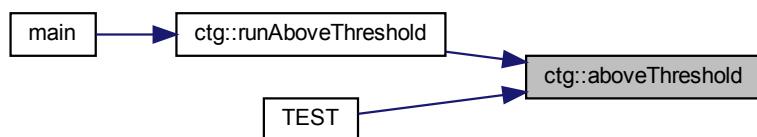
```
std::vector< cl::DataPoint > ctg::aboveThreshold (
    const cl::DataSet & dataSet,
    long double accelerometerThreshold,
    long double gyroscopeThreshold )
```

Definition at line 28 of file above\_threshold.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



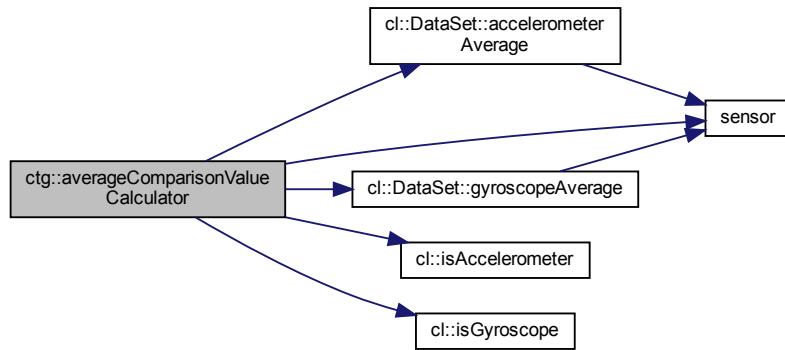
### 5.5.1.2 averageComparisonValueCalculator()

```
long double ctg::averageComparisonValueCalculator (
    cl::Sensor sensor,
```

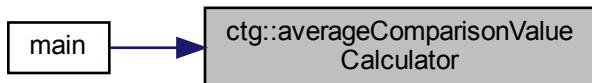
```
    cl::Channel channel,
    const cl::DataSet & dataSet )
```

Definition at line 10 of file average\_comparison\_value\_calculator.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

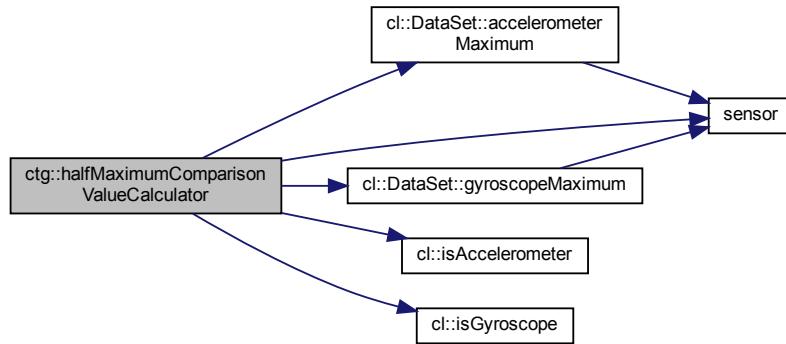


### 5.5.1.3 halfMaximumComparisonValueCalculator()

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

Definition at line 10 of file half\_maximum\_comparison\_value\_calculator.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

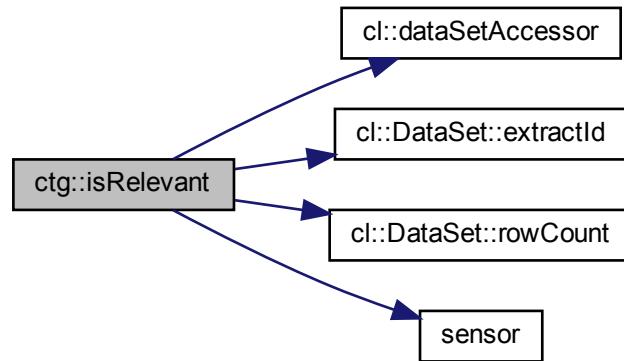


#### 5.5.1.4 `isRelevant()`

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

Definition at line 11 of file `is_relevant.hpp`.

Here is the call graph for this function:



Here is the caller graph for this function:

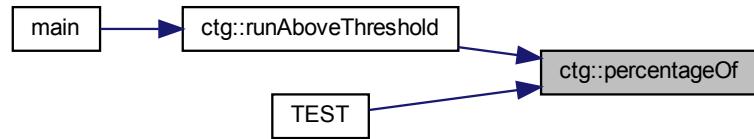


### 5.5.1.5 percentageOf()

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

Definition at line 6 of file percentage\_of.hpp.

Here is the caller graph for this function:

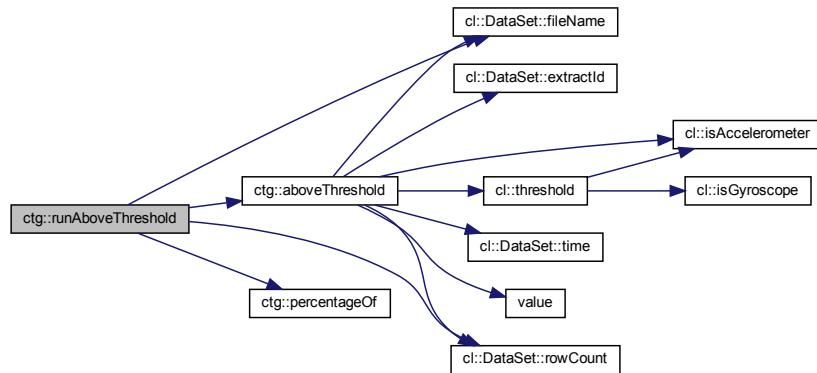


### 5.5.1.6 runAboveThreshold()

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

Definition at line 14 of file run\_above\_threshold.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



## 5.6 fmc Namespace Reference

### Functions

- void [adjustHardwareTimestamp](#) (std::string \*cellContent, const std::string &nextRowHardwareTimestamp, std::uint64\_t \*overflowCount)
- bool [convertToUnixLineEndings](#) (const std::string &csvPath)
- bool [createBackupFile](#) (const std::string &csvFilePath, const std::string &backupFilePath)
- void [deleteNonBoschSensors](#) (std::vector< std::vector< std::string >> \*data)
- [cl::Expected< void >](#) [deleteOutOfBoundsValues](#) (std::vector< std::vector< std::string >> \*data)
- void [removeZerosFromField](#) (std::string \*field)
- bool [restoreFromBackup](#) (const std::string &csvFilePath, const std::string &backupFilePath)
- bool [writeFile](#) (pl::string\_view csvPath, pl::string\_view csvFileExtension, const std::vector< std::string > &columnNames, const std::vector< std::vector< std::string >> &data)

## 5.6.1 Function Documentation

### 5.6.1.1 **adjustHardwareTimestamp()**

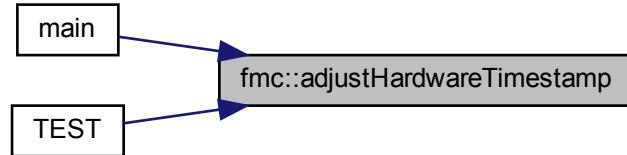
```
void fmc::adjustHardwareTimestamp (
    std::string * cellContent,
    const std::string & nextRowHardwareTimestamp,
    std::uint64_t * overflowCount )
```

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

Here is the call graph for this function:



Here is the caller graph for this function:

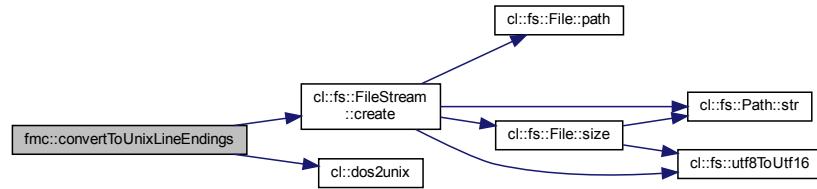


### 5.6.1.2 **convertToUnixLineEndings()**

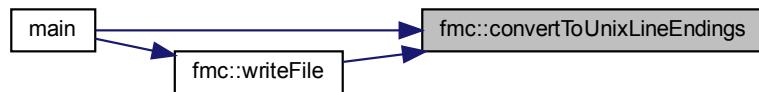
```
bool fmc::convertToUnixLineEndings (
    const std::string & csvPath )
```

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

Here is the call graph for this function:



Here is the caller graph for this function:



### 5.6.1.3 `createBackupFile()`

```

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

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

Here is the caller graph for this function:



### 5.6.1.4 deleteNonBoschSensors()

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

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

Here is the caller graph for this function:



### 5.6.1.5 deleteOutOfBoundsValues()

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

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

Here is the caller graph for this function:

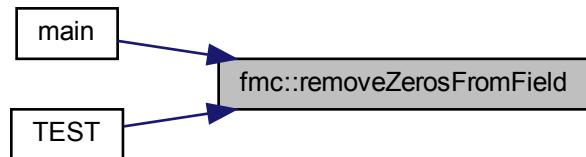


### 5.6.1.6 removeZerosFromField()

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

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

Here is the caller graph for this function:



#### 5.6.1.7 restoreFromBackup()

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

Definition at line 11 of file restore\_from\_backup.cpp.

Here is the caller graph for this function:

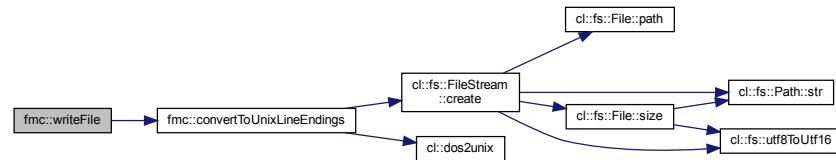


#### 5.6.1.8 writeFile()

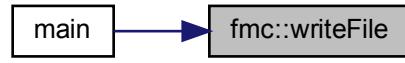
```
bool fmc::writeFile (
    pl::string_view csvPath,
    pl::string_view csvFileExtension,
    const std::vector< std::string > & columnNames,
    const std::vector< std::vector< std::string >> & data )
```

Definition at line 12 of file write\_file.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:





# Chapter 6

## Class Documentation

### 6.1 cm::Configuration::Builder Class Reference

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

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

#### Public Member Functions

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

#### 6.1.1 Detailed Description

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

Definition at line 24 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 33 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

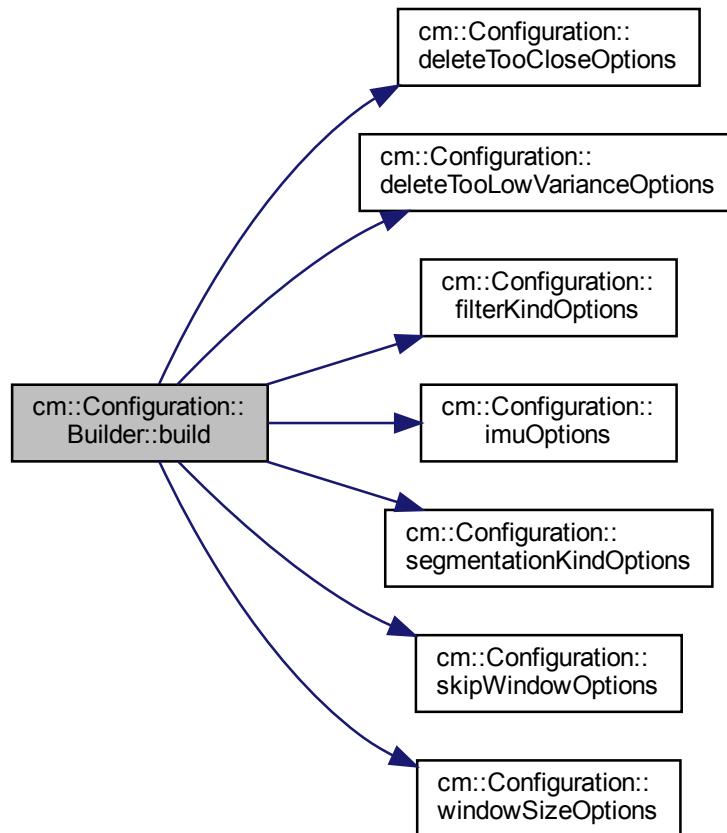
The [Configuration](#) built.

#### Exceptions

<a href="#">cl::Exception</a>	if one of the properties has not been set or is invalid.
-------------------------------	--

Definition at line 87 of file configuration.cpp.

Here is the call graph for this function:



### 6.1.3.2 `deleteTooClose()`

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

Sets the `deleteTooClose` property.

#### Parameters

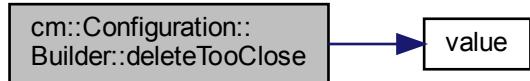
<code>value</code>	The value to use.
--------------------	-------------------

#### Returns

`*this`

Definition at line 50 of file configuration.cpp.

Here is the call graph for this function:



#### 6.1.3.3 deleteTooLowVariance()

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

Sets the deleteTooLowVariance property.

##### Parameters

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

##### Returns

\*this

Definition at line 56 of file configuration.cpp.

Here is the call graph for this function:



#### 6.1.3.4 filterKind()

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

Sets the filterKind property.

**Parameters**

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

**Returns**`*this`

Definition at line 81 of file configuration.cpp.

Here is the call graph for this function:



### 6.1.3.5 imu()

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

Sets the imu property.

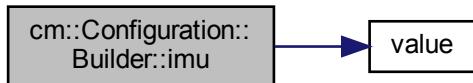
**Parameters**

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

**Returns**`*this`

Definition at line 62 of file configuration.cpp.

Here is the call 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 68 of file configuration.cpp.

Here is the call graph for this function:



### 6.1.3.7 skipWindow()

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

Sets the skipWindow property.

#### Parameters

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

#### Returns

\*this

Definition at line 44 of file configuration.cpp.

Here is the call 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 75 of file configuration.cpp.

Here is the call graph for this function:



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

- confusion\_matrix/include/[configuration.hpp](#)
- confusion\_matrix/src/[configuration.cpp](#)

## 6.2 cl::col\_traits< Col > Struct Template Reference

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

### 6.2.1 Detailed Description

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

Definition at line 24 of file column.hpp.

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

- csv\_lib/include/cl/column.hpp

## 6.3 cm::Configuration Class Reference

Represents a possible configuration for the Python segmentor.

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

### Classes

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

## Public Member Functions

- 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.*

## Static Public Member Functions

- static const std::array< bool, 2 > & `skipWindowOptions () noexcept`  
*Returns the possible skipWindow options.*
- static const std::array< bool, 2 > & `deleteTooCloseOptions () noexcept`  
*Returns the possible deleteTooClose options.*
- static const std::array< bool, 2 > & `deleteTooLowVarianceOptions () noexcept`  
*Returns the possible deleteTooLowVariance options.*
- static const std::array< Imu, 2 > & `imuOptions () noexcept`  
*Returns the possible imu options.*
- static const std::array< std::string, 3 > & `segmentationKindOptions () noexcept`  
*Returns the possible segmentationKind options.*
- static const std::array< std::size\_t, 11 > & `windowSizeOptions () noexcept`  
*Returns the possible windowSize options.*
- static const std::array< std::string, 2 > & `filterKindOptions () noexcept`  
*Returns the possible filterKind options.*

## Friends

- class `Builder`

### 6.3.1 Detailed Description

Represents a possible configuration for the Python segmentor.

Definition at line 17 of file configuration.hpp.

### 6.3.2 Member Function Documentation

### 6.3.2.1 deleteTooClose()

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

Read accessor for the deleteTooClose property.

#### Returns

The deleteTooClose option.

Definition at line 178 of file configuration.cpp.

### 6.3.2.2 deleteTooCloseOptions()

```
const std::array< bool, 2 > & cm::Configuration::deleteTooCloseOptions ( ) [static], [noexcept]
```

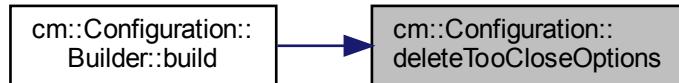
Returns the possible deleteTooClose options.

#### Returns

The deleteTooClose options.

Definition at line 144 of file configuration.cpp.

Here is the caller graph for this function:



### 6.3.2.3 deleteTooLowVariance()

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

Read accessor for the deleteTooLowVariance property.

#### Returns

The deleteTooLowVariance option.

Definition at line 180 of file configuration.cpp.

### 6.3.2.4 deleteTooLowVarianceOptions()

```
const std::array< bool, 2 > & cm::Configuration::deleteTooLowVarianceOptions ( ) [static],  
[noexcept]
```

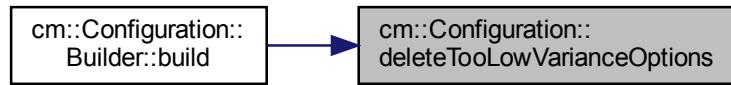
Returns the possible deleteTooLowVariance options.

#### Returns

The deleteTooLowVariance options.

Definition at line 149 of file configuration.cpp.

Here is the caller graph for this function:



### 6.3.2.5 filterKind()

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

Read accessor for the filterKind property.

#### Returns

The filterKind option.

Definition at line 194 of file configuration.cpp.

### 6.3.2.6 filterKindOptions()

```
const std::array< std::string, 2 > & cm::Configuration::filterKindOptions ( ) [static], [noexcept]
```

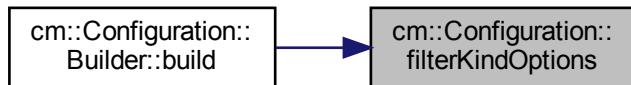
Returns the possible filterKind options.

#### Returns

The filterKind options.

Definition at line 170 of file configuration.cpp.

Here is the caller graph for this function:



### 6.3.2.7 imu()

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

Read accessor for the imu property.

#### Returns

The imu option.

Definition at line 185 of file configuration.cpp.

### 6.3.2.8 imuOptions()

```
const std::array< Imu, 2 > & cm::Configuration::imuOptions ( ) [static], [noexcept]
```

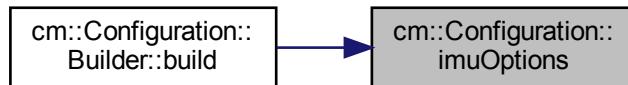
Returns the possible imu options.

#### Returns

The imu options.

Definition at line 154 of file configuration.cpp.

Here is the caller graph for this function:



### 6.3.2.9 segmentationKind()

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

Read accessor for the segmentationKind property.

#### Returns

The segmentationKind option.

Definition at line 187 of file configuration.cpp.

### 6.3.2.10 segmentationKindOptions()

```
const std::array< std::string, 3 > & cm::Configuration::segmentationKindOptions ( ) [static],  
[noexcept]
```

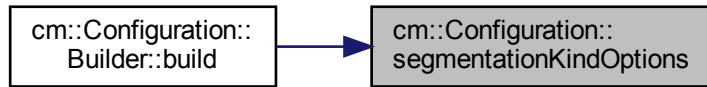
Returns the possible segmentationKind options.

#### Returns

The segmentationKind options.

Definition at line 157 of file configuration.cpp.

Here is the caller graph for this function:



### 6.3.2.11 skipWindow()

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

Read accessor for the skipWindow property.

#### Returns

The skipWindow option.

Definition at line 176 of file configuration.cpp.

### 6.3.2.12 skipWindowOptions()

```
const std::array< bool, 2 > & cm::Configuration::skipWindowOptions ( ) [static], [noexcept]
```

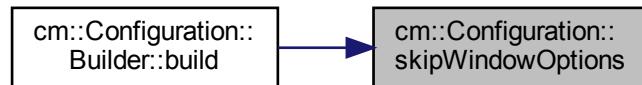
Returns the possible skipWindow options.

#### Returns

The skipWindow options.

Definition at line 139 of file configuration.cpp.

Here is the caller graph for this function:



### 6.3.2.13 windowSize()

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

Read accessor for the windowSize property.

#### Returns

The windowSize option.

Definition at line 192 of file configuration.cpp.

### 6.3.2.14 windowSizeOptions()

```
const std::array< std::size_t, 11 > & cm::Configuration::windowSizeOptions ( ) [static],  
[noexcept]
```

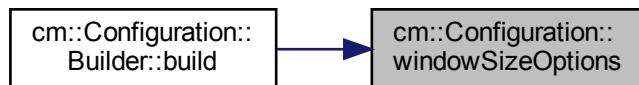
Returns the possible windowSize options.

#### Returns

The windowSize options.

Definition at line 163 of file configuration.cpp.

Here is the caller graph for this function:



## 6.3.3 Friends And Related Function Documentation

### 6.3.3.1 Builder

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

Definition at line 19 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 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.*

### 6.4.1 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.

### 6.4.2 Member Typedef Documentation

#### 6.4.2.1 this\_type

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

Definition at line 23 of file csv\_line.hpp.

## 6.4.3 Constructor & Destructor Documentation

### 6.4.3.1 CsvLineBuilder()

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

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

Definition at line 44 of file csv\_line.cpp.

## 6.4.4 Member Function Documentation

### 6.4.4.1 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:



#### 6.4.4.2 dataSet()

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

Write accessor for the data set property.

##### Parameters

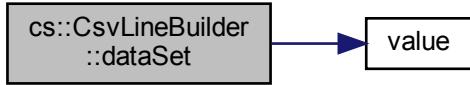
value	The value to use.
-------	-------------------

##### 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:



#### 6.4.4.3 deleteLowVariance()

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

Write accessor for the delete low variance property.

**Parameters**

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

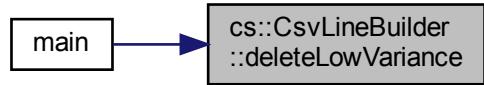
**Returns**`*this`

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:



#### 6.4.4.4 deleteTooClose()

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

Write accessor for the delete too close property.

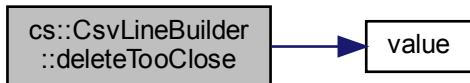
**Parameters**

<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:



#### 6.4.4.5 filter()

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

Write accessor for the filter property.

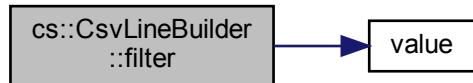
**Parameters**

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

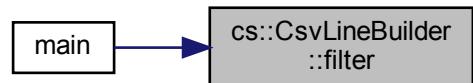
**Returns**`*this`

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:



#### 6.4.4.6 `isOld()`

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

Write accessor for the is old property.

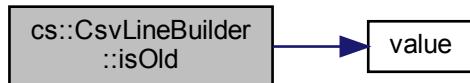
##### Parameters

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

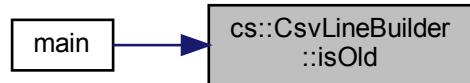
**Returns**`*this`

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:



#### 6.4.4.7 kind()

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

Write accessor for the kind property.

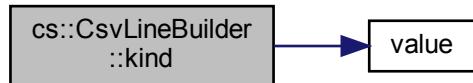
**Parameters**

<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:



#### 6.4.4.8 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:



#### 6.4.4.9 segmentationPoints()

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

Write accessor for the segmentation points property.

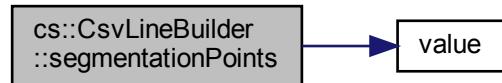
**Parameters**

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

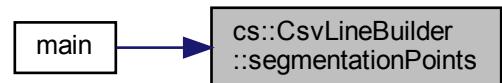
**Returns**`*this`

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:



#### 6.4.4.10 sensor()

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

Write accessor for the sensor property.

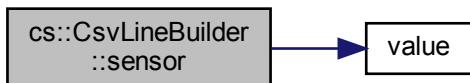
##### Parameters

<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:



#### 6.4.4.11 skipWindow()

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

Write accessor for the skip window property.

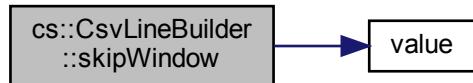
**Parameters**

<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:



#### 6.4.4.12 `windowSize()`

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

Write accessor for the window size property.

##### Parameters

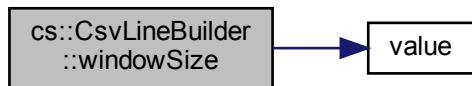
<code>value</code>	The value to use.
--------------------	-------------------

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](#)

## 6.5 cl::data\_set\_accessor< Chan > Struct Template Reference

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

### 6.5.1 Detailed Description

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

Definition at line 39 of file channel.hpp.

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

- [csv\\_lib/include/cl/channel.hpp](#)

## 6.6 cs::data\_set\_info< Tag > Struct Template Reference

Meta function for data set tags.

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

### 6.6.1 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](#)

## 6.7 cl::DataPoint Class Reference

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

### Public Member Functions

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

### Friends

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

### 6.7.1 Detailed Description

Definition at line 10 of file data\_point.hpp.

## 6.7.2 Constructor & Destructor Documentation

### 6.7.2.1 DataPoint()

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

Definition at line 21 of file data\_point.cpp.

Here is the call graph for this function:



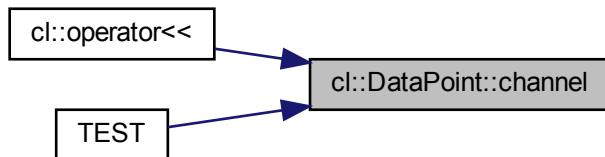
## 6.7.3 Member Function Documentation

### 6.7.3.1 channel()

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

Definition at line 41 of file data\_point.cpp.

Here is the caller graph for this function:

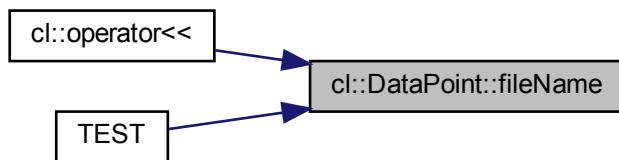


### 6.7.3.2 fileName()

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

Definition at line 35 of file data\_point.cpp.

Here is the caller graph for this function:

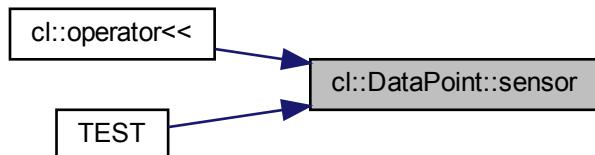


### 6.7.3.3 sensor()

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

Definition at line 39 of file data\_point.cpp.

Here is the caller graph for this function:

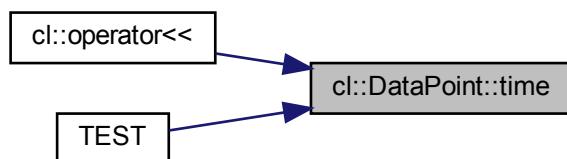


### 6.7.3.4 time()

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

Definition at line 37 of file data\_point.cpp.

Here is the caller graph for this function:

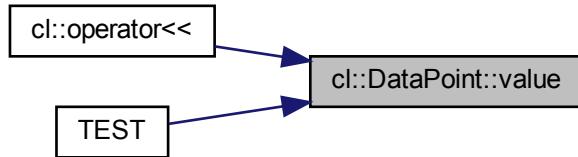


### 6.7.3.5 value()

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

Definition at line 43 of file data\_point.cpp.

Here is the caller graph for this function:



## 6.7.4 Friends And Related Function Documentation

#### 6.7.4.1 operator<<

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

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

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

- [csv\\_lib/include/cl/data\\_point.hpp](#)
- [csv\\_lib/src/cl/data\\_point.cpp](#)

## 6.8 cl::DataSet Class Reference

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

### Public Types

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

### Public Member Functions

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

### Static Public Member Functions

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

### 6.8.1 Detailed Description

Definition at line 14 of file data\_set.hpp.

### 6.8.2 Member Typedef Documentation

#### 6.8.2.1 ChannelAccessor

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

Definition at line 17 of file data\_set.hpp.

#### 6.8.2.2 size\_type

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

Definition at line 16 of file data\_set.hpp.

### 6.8.3 Member Function Documentation

#### 6.8.3.1 accelerometerAverage()

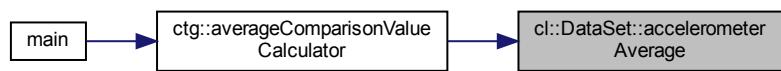
```
long double cl::DataSet::accelerometerAverage ( Sensor sensor ) const
```

Definition at line 255 of file data\_set.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

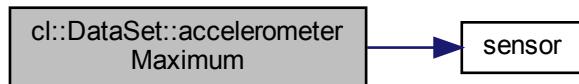


### 6.8.3.2 accelerometerMaximum()

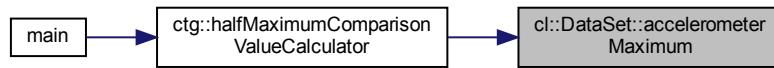
```
long double cl::DataSet::accelerometerMaximum (
    Sensor sensor ) const
```

Definition at line 265 of file data\_set.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

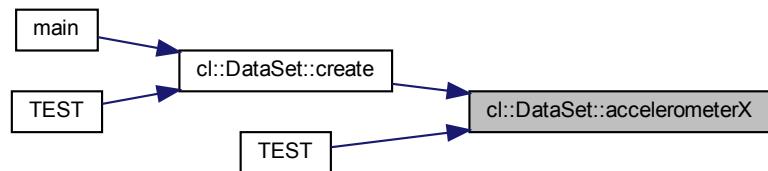


### 6.8.3.3 accelerometerX()

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

Definition at line 200 of file data\_set.cpp.

Here is the caller graph for this function:

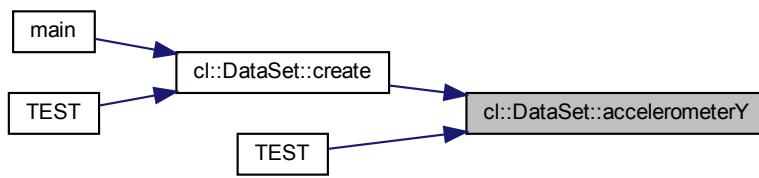


### 6.8.3.4 accelerometerY()

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

Definition at line 208 of file data\_set.cpp.

Here is the caller graph for this function:

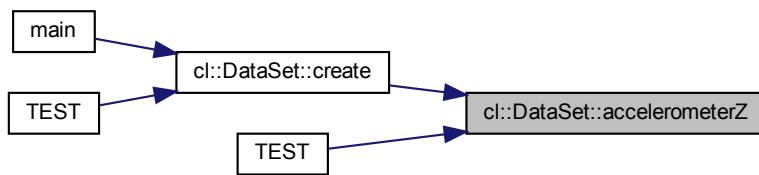


### 6.8.3.5 accelerometerZ()

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

Definition at line 216 of file data\_set.cpp.

Here is the caller graph for this function:

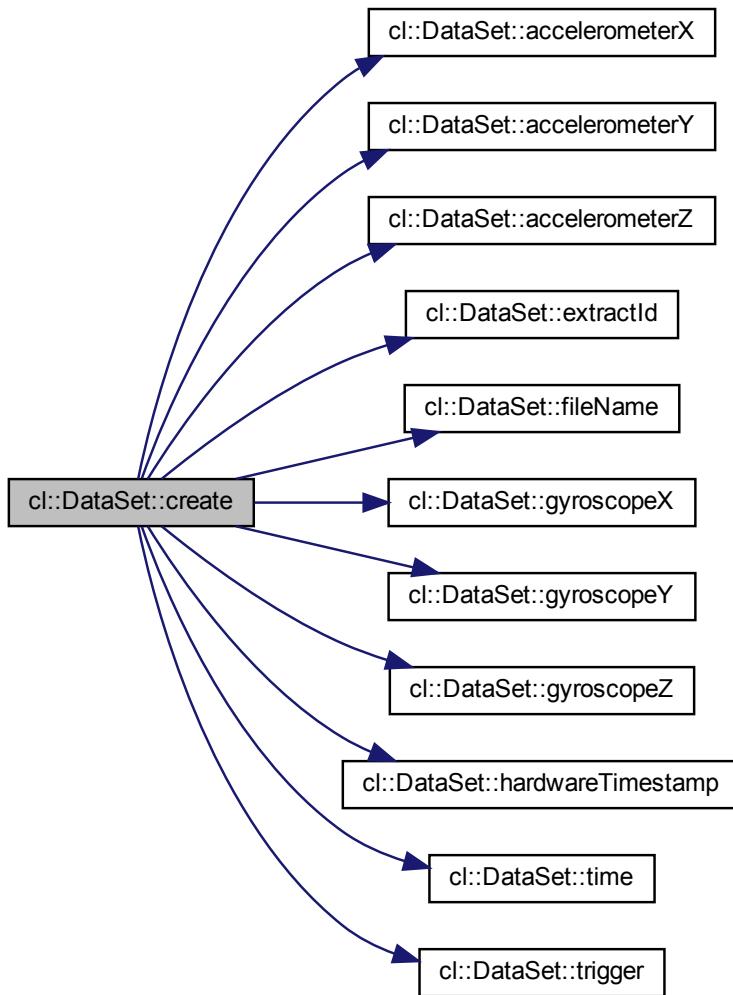


### 6.8.3.6 create()

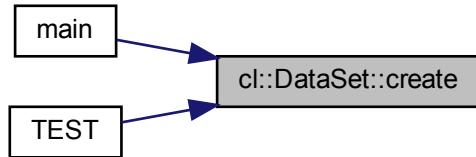
```
Expected< DataSet > cl::DataSet::create (
    std::string fileName,
    const std::vector< std::vector< std::string >> & matrix ) [static]
```

Definition at line 42 of file data\_set.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

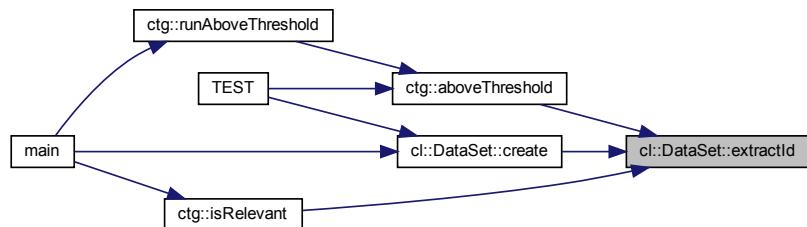


#### 6.8.3.7 extractId()

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

Definition at line 186 of file data\_set.cpp.

Here is the caller graph for this function:

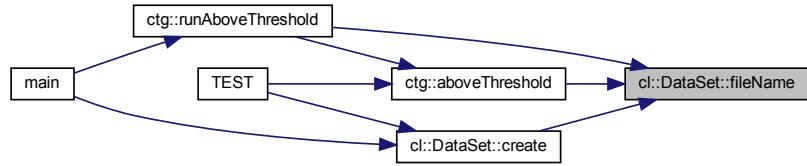


#### 6.8.3.8 fileName()

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

Definition at line 169 of file data\_set.cpp.

Here is the caller graph for this function:



### 6.8.3.9 gyroscopeAverage()

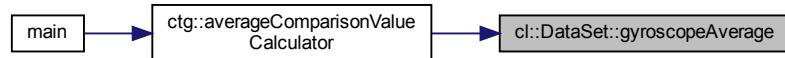
```
long double cl::DataSet::gyroscopeAverage (
    Sensor sensor ) const
```

Definition at line 260 of file data\_set.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



### 6.8.3.10 gyroscopeMaximum()

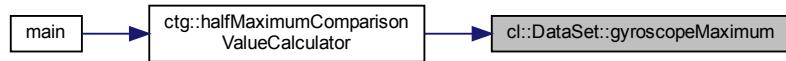
```
long double cl::DataSet::gyroscopeMaximum (
    Sensor sensor ) const
```

Definition at line 270 of file data\_set.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

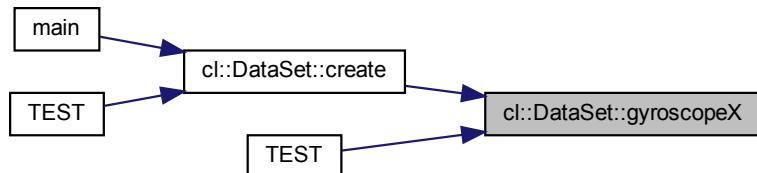


### 6.8.3.11 gyroscopeX()

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

Definition at line 224 of file data\_set.cpp.

Here is the caller graph for this function:

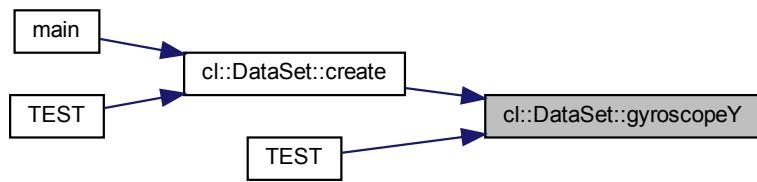


### 6.8.3.12 gyroscopeY()

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

Definition at line 231 of file data\_set.cpp.

Here is the caller graph for this function:

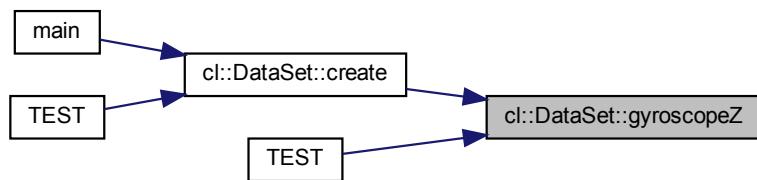


### 6.8.3.13 gyroscopeZ()

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

Definition at line 238 of file data\_set.cpp.

Here is the caller graph for this function:

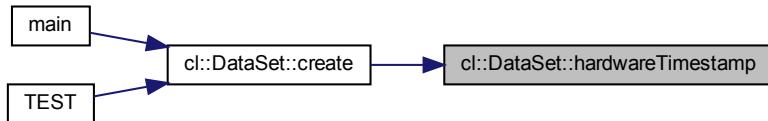


### 6.8.3.14 hardwareTimestamp()

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

Definition at line 178 of file data\_set.cpp.

Here is the caller graph for this function:

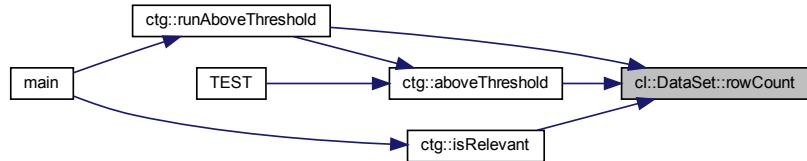


### 6.8.3.15 rowCount()

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

Definition at line 152 of file data\_set.cpp.

Here is the caller graph for this function:

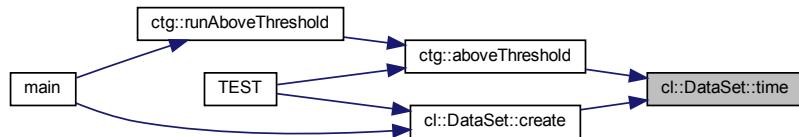


### 6.8.3.16 time()

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

Definition at line 171 of file data\_set.cpp.

Here is the caller graph for this function:

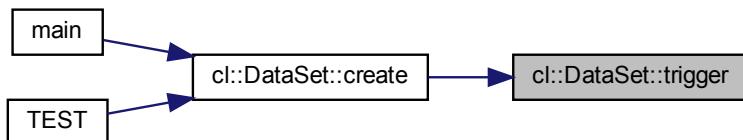


### 6.8.3.17 trigger()

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

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

Here is the caller graph for this function:



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

- [csv\\_lib/include/cl/data\\_set.hpp](#)
- [csv\\_lib/src/cl/data\\_set.cpp](#)

## 6.9 cl::Error Class Reference

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

### Public Types

- enum [Kind { CL\\_ERROR\\_KIND }](#)

## Public Member Functions

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

## Friends

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

### 6.9.1 Detailed Description

Definition at line 23 of file error.hpp.

### 6.9.2 Member Enumeration Documentation

#### 6.9.2.1 Kind

`enum cl::Error::Kind`

Enumerator

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

Definition at line 26 of file error.hpp.

### 6.9.3 Constructor & Destructor Documentation

#### 6.9.3.1 Error()

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

Definition at line 41 of file error.cpp.

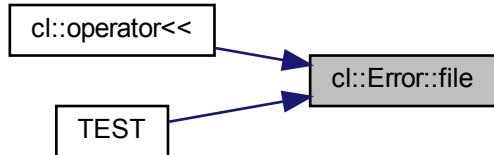
## 6.9.4 Member Function Documentation

### 6.9.4.1 file()

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

Definition at line 57 of file error.cpp.

Here is the caller graph for this function:

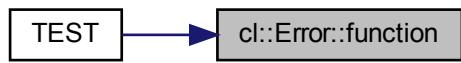


### 6.9.4.2 function()

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

Definition at line 59 of file error.cpp.

Here is the caller graph for this function:

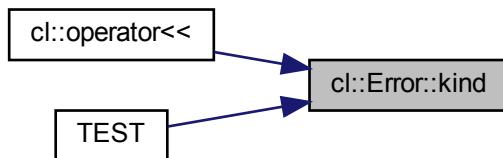


#### 6.9.4.3 kind()

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

Definition at line 55 of file error.cpp.

Here is the caller graph for this function:



#### 6.9.4.4 line()

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

Definition at line 61 of file error.cpp.

#### 6.9.4.5 message()

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

Definition at line 63 of file error.cpp.

Here is the caller graph for this function:



#### 6.9.4.6 `raise()`

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

Definition at line 65 of file error.cpp.

#### 6.9.4.7 `to_string()`

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

Definition at line 74 of file error.cpp.

### 6.9.5 Friends And Related Function Documentation

#### 6.9.5.1 `operator<<`

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

Definition at line 30 of file error.cpp.

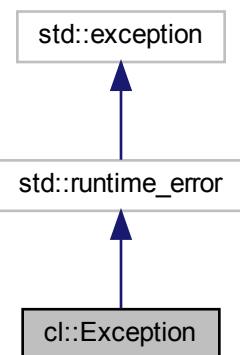
The documentation for this class was generated from the following files:

- csv\_lib/include/cl/error.hpp
- csv\_lib/src/cl/error.cpp

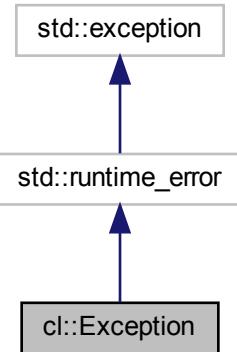
## 6.10 cl::Exception Class Reference

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

Inheritance diagram for cl::Exception:



Collaboration diagram for cl::Exception:



## Public Types

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

## Public Member Functions

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

### 6.10.1 Detailed Description

Definition at line 14 of file exception.hpp.

### 6.10.2 Member Typedef Documentation

#### 6.10.2.1 `base_type`

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

Definition at line 16 of file exception.hpp.

### 6.10.3 Constructor & Destructor Documentation

#### 6.10.3.1 Exception() [1/2]

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

Definition at line 6 of file exception.cpp.

#### 6.10.3.2 Exception() [2/2]

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

Definition at line 18 of file exception.cpp.

### 6.10.4 Member Function Documentation

#### 6.10.4.1 file()

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

Definition at line 30 of file exception.cpp.

Here is the caller graph for this function:

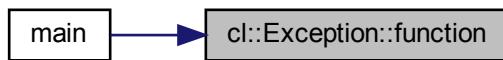


#### 6.10.4.2 function()

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

Definition at line 32 of file exception.cpp.

Here is the caller graph for this function:



#### 6.10.4.3 line()

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

Definition at line 34 of file exception.cpp.

Here is the caller graph for this function:



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

- csv\_lib/include/cl/exception.hpp
- csv\_lib/src/cl/exception.cpp

## 6.11 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 &copyToPath) 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.*

### 6.11.1 Detailed Description

Represents a file.

Definition at line 11 of file file.hpp.

### 6.11.2 Constructor & Destructor Documentation

#### 6.11.2.1 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:



### 6.11.3 Member Function Documentation

#### 6.11.3.1 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:



### 6.11.3.2 `create()`

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

Creates this file.

#### Returns

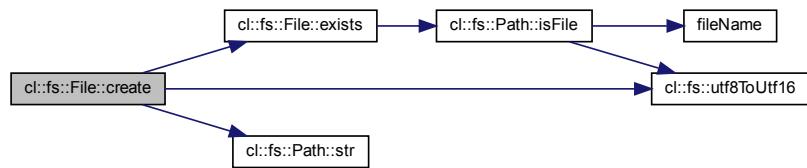
true if the file was successfully created; otherwise false.

#### Note

Will fail if the file already exists.

Definition at line 25 of file file.cpp.

Here is the call graph for this function:



### 6.11.3.3 `exists()`

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

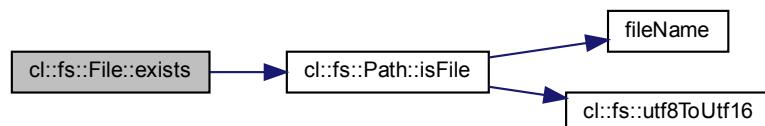
Determines if this file exists.

#### 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:



#### 6.11.3.4 moveTo()

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

Moves this file in the filesystem.

##### Parameters

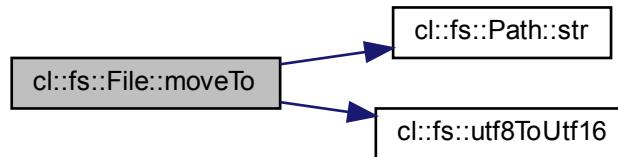
<i>newPath</i>	The path to move this file to.
----------------	--------------------------------

##### 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:



### 6.11.3.5 path()

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

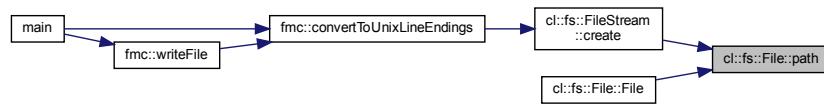
Read accessor for the path of this file.

#### Returns

The path of this file.

Definition at line 169 of file file.cpp.

Here is the caller graph for this function:



### 6.11.3.6 remove()

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

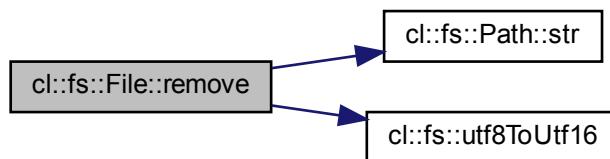
Deletes this file.

#### Returns

true if deleting succeeded; otherwise false.

Definition at line 117 of file file.cpp.

Here is the call graph for this function:



### 6.11.3.7 size()

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

Determines the size of this file in bytes.

#### Returns

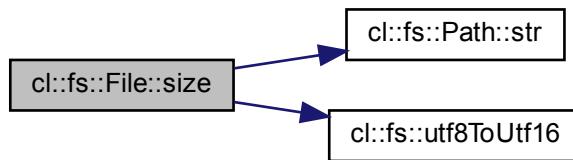
The size of this file in bytes or -1 on error.

#### Warning

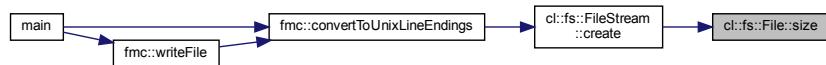
Returns -1 on error.

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

## 6.12 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.*

### 6.12.1 Detailed Description

A binary file stream.

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

### 6.12.2 Member Typedef Documentation

#### 6.12.2.1 `this_type`

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

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

### 6.12.3 Member Enumeration Documentation

#### 6.12.3.1 `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.

## 6.12.4 Constructor & Destructor Documentation

### 6.12.4.1 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.

### 6.12.4.2 ~FileStream()

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

Closes this file stream.

Definition at line 84 of file file\_stream.cpp.

## 6.12.5 Member Function Documentation

### 6.12.5.1 create()

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

Creates a file stream.

**Parameters**

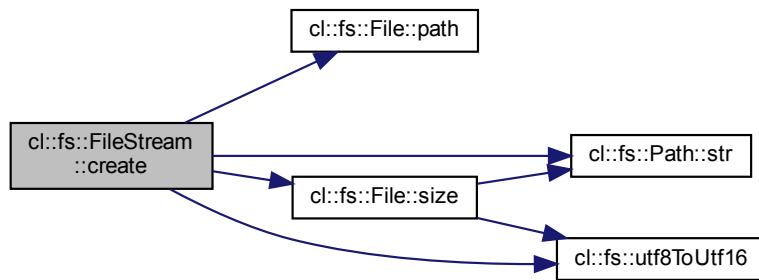
<i>file</i>	The file to open.
<i>openMode</i>	The open mode to use.

**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:

**6.12.5.2 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`.

### 6.12.5.3 PL\_NOCOPYABLE()

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

### 6.12.5.4 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`.

### 6.12.5.5 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](#)

## 6.13 std::hash<::cl::fs::Path> Struct Reference

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

### Public Member Functions

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

#### 6.13.1 Detailed Description

Definition at line 85 of file path.hpp.

#### 6.13.2 Member Function Documentation

##### 6.13.2.1 operator()()

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

Definition at line 86 of file path.hpp.

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

- csv\_lib/include/cl/fs/path.hpp

## 6.14 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.*

### 6.14.1 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`.

### 6.14.2 Constructor & Destructor Documentation

#### 6.14.2.1 `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`.

### 6.14.3 Member Function Documentation

#### 6.14.3.1 `create()`

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

Creates a [LogInfo](#) from the given log file path.

**Parameters**

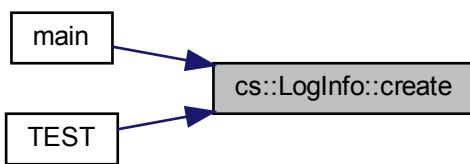
<i>logFilePath</i>	The log file path to create a <a href="#">LogInfo</a> from.
--------------------	---

**Returns**

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

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

Here is the caller graph for this function:

**6.14.3.2 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`.

**6.14.3.3 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`.

#### 6.14.3.4 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.

#### 6.14.3.5 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.

#### 6.14.3.6 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:



### 6.14.3.7 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.

### 6.14.3.8 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.

### 6.14.3.9 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.

### 6.14.3.10 `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.

## 6.14.4 Friends And Related Function Documentation

### 6.14.4.1 `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.

### 6.14.4.2 `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 <a href="#">LogInfo</a> to print.

**Returns**

```
os
```

Definition at line 292 of file log\_info.cpp.

**6.14.4.3 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.

**6.14.5 Member Data Documentation****6.14.5.1 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](#)

**6.15 cs::LogLine Class Reference**

A line out of a log file.

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

## Public Member Functions

- std::uint64\_t [segmentationPointCount \(\) const noexcept](#)  
*Read accessor for the segmentation point count.*
- const [cl::fs::Path & filePath \(\) const noexcept](#)  
*Read accessor for the file path.*
- [cl::Expected< std::string > fileName \(\) const](#)  
*Creates the short file name for the file in the log line.*
- std::uint64\_t [sensor \(\) const noexcept](#)  
*Read accessor for the sensor.*

## Static Public Member Functions

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

## Static Public Attributes

- static const std::uint64\_t [invalidSensor = UINT64\\_C\(0xFFFFFFFFFFFFFF\)](#)  
*Indicates an invalid sensor.*

### 6.15.1 Detailed Description

A line out of a log file.

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

### 6.15.2 Member Function Documentation

#### 6.15.2.1 `fileName()`

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

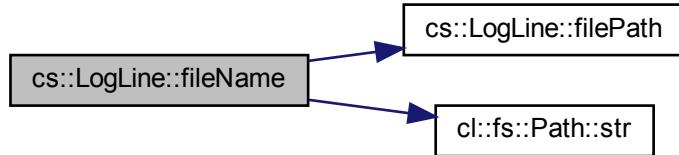
Creates the short file name for the file in the log line.

**Returns**

The resulting short file name or an error.

Definition at line 126 of file log\_line.cpp.

Here is the call graph for this function:

**6.15.2.2 filePath()**

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

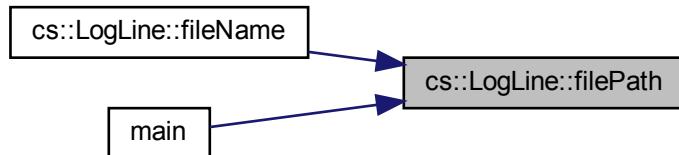
Read accessor for the file path.

**Returns**

The file path of the file in the log line.

Definition at line 124 of file log\_line.cpp.

Here is the caller graph for this function:

**6.15.2.3 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**

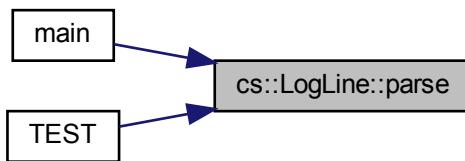
<i>line</i>	The line read.
-------------	----------------

**Returns**

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

Definition at line 31 of file log\_line.cpp.

Here is the caller graph for this function:

**6.15.2.4 segmentationPointCount()**

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

Read accessor for the segmentation point count.

**Returns**

The segmentation point count.

Definition at line 119 of file log\_line.cpp.

**6.15.2.5 sensor()**

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

Read acccessor for the sensor.

**Returns**

The sensor.

**Note**

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

Definition at line 164 of file log\_line.cpp.

### 6.15.3 Member Data Documentation

#### 6.15.3.1 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](#)

## 6.16 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.*

## 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.*

### 6.16.1 Detailed Description

Type used to represent a manual segmentation point.

Definition at line 26 of file `manual_segmentation_point.hpp`.

### 6.16.2 Constructor & Destructor Documentation

#### 6.16.2.1 `ManualSegmentationPoint()`

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

Creates a `ManualSegmentationPoint`.

##### Parameters

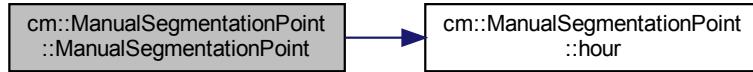
<code>hour</code>	The hour to use. Must be within [0,59].
<code>minute</code>	The minute to use. Must be within [0,59].
<code>second</code>	The second to use. Must be within [0,59].
<code>frame</code>	The frame to use. Must be within [0,29].

##### Exceptions

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

Definition at line 303 of file `manual_segmentation_point.cpp`.

Here is the call graph for this function:



### 6.16.3 Member Function Documentation

#### 6.16.3.1 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 361 of file manual\_segmentation\_point.cpp.

#### 6.16.3.2 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 356 of file manual\_segmentation\_point.cpp.

Here is the caller graph for this function:



### 6.16.3.3 hour()

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

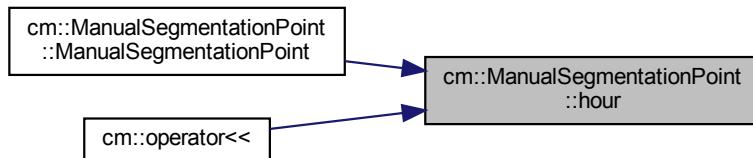
Read accessor for the hour property.

#### Returns

The hour.

Definition at line 344 of file manual\_segmentation\_point.cpp.

Here is the caller graph for this function:



### 6.16.3.4 minute()

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

Read accessor for the minute property.

#### Returns

The minute.

Definition at line 346 of file manual\_segmentation\_point.cpp.

Here is the caller graph for this function:



### 6.16.3.5 `readCsvFile()`

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

Reads the CSV file of the manual segmentation points.

#### Returns

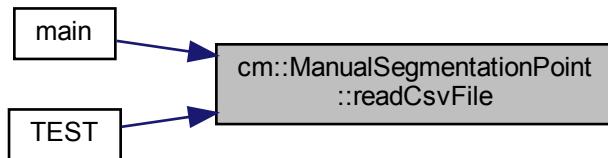
A map that maps the `DataSetIdentifier` enumerators to vectors of the corresponding manual segmentation points extracted from the CSV file.

#### Exceptions

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

Definition at line 161 of file `manual_segmentation_point.cpp`.

Here is the caller graph for this function:



### 6.16.3.6 `second()`

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

Read accessor for the second property.

#### Returns

The second.

Definition at line 351 of file `manual_segmentation_point.cpp`.

Here is the caller graph for this function:



## 6.16.4 Friends And Related Function Documentation

### 6.16.4.1 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 139 of file manual\_segmentation\_point.cpp.

### 6.16.4.2 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 146 of file manual\_segmentation\_point.cpp.

**6.16.4.3 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 131 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

**6.17 cl::fs::Path Class Reference**

A filesystem path.

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

**Public Member Functions**

- 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.*

### 6.17.1 Detailed Description

A filesystem path.

Definition at line 14 of file path.hpp.

### 6.17.2 Constructor & Destructor Documentation

#### 6.17.2.1 Path() [1/2]

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

Creates a path.

##### Parameters

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

Definition at line 37 of file path.cpp.

#### 6.17.2.2 Path() [2/2]

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

Creates a path.

##### Parameters

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

Definition at line 44 of file path.cpp.

### 6.17.3 Member Function Documentation

#### 6.17.3.1 exists()

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

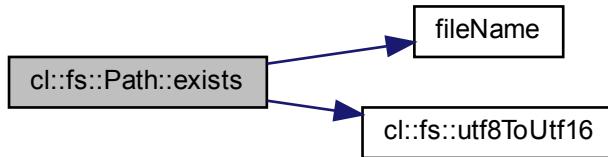
Checks if the path exists.

##### Returns

true if the path exists; otherwise false.

Definition at line 46 of file path.cpp.

Here is the call graph for this function:



#### 6.17.3.2 isDirectory()

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

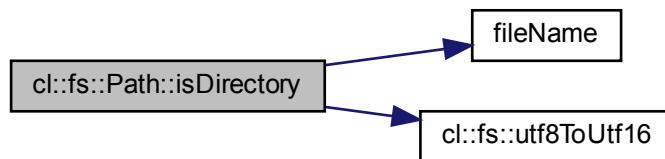
Checks if the path is a directory.

**Returns**

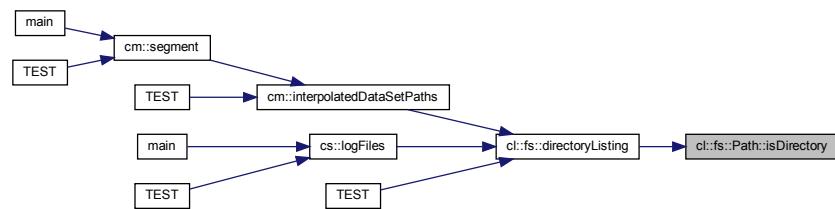
true if the path is a directory; otherwise false.

Definition at line 102 of file path.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

**6.17.3.3 isFile()**

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

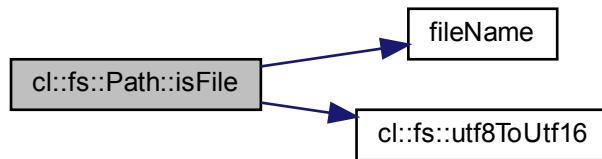
Checks if the path is a file.

**Returns**

true if the path is a file; otherwise false.

Definition at line 75 of file path.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 6.17.3.4 str()

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

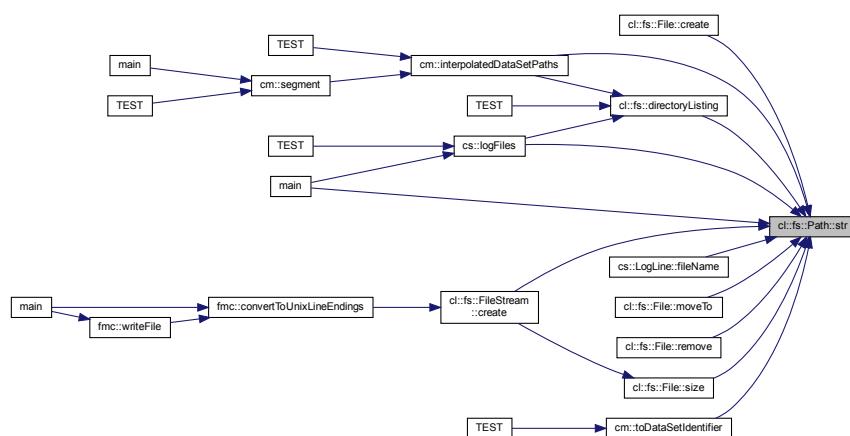
Read accessor for the underlying string.

##### Returns

The underlying string.

Definition at line 125 of file path.cpp.

Here is the caller graph for this function:



## 6.17.4 Friends And Related Function Documentation

### 6.17.4.1 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.

### 6.17.4.2 operator<<

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

Prints a `Path` to an ostream.

#### Parameters

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

#### Returns

`os`

Definition at line 22 of file path.cpp.

### 6.17.4.3 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

## 6.18 cl::Process Class Reference

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

### Public Types

- using `this_type = Process`

### Public Member Functions

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

### Static Public Member Functions

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

### 6.18.1 Detailed Description

Definition at line 11 of file process.hpp.

### 6.18.2 Member Typedef Documentation

#### 6.18.2.1 this\_type

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

Definition at line 15 of file process.hpp.

### 6.18.3 Constructor & Destructor Documentation

#### 6.18.3.1 Process()

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

Definition at line 56 of file process.cpp.

#### 6.18.3.2 ~Process()

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

Definition at line 69 of file process.cpp.

### 6.18.4 Member Function Documentation

#### 6.18.4.1 create()

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

Definition at line 36 of file process.cpp.

#### 6.18.4.2 `file()` [1/2]

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

#### 6.18.4.3 `file()` [2/2]

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

Definition at line 79 of file process.cpp.

#### 6.18.4.4 `operator=()`

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

Definition at line 61 of file process.cpp.

#### 6.18.4.5 `PL_NONCOPYABLE()`

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

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

- csv\_lib/include/cl/process.hpp
- csv\_lib/src/cl/process.cpp

# Chapter 7

## File Documentation

### 7.1 compare\_segmentation/CMakeLists.txt File Reference

#### Functions

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

#### 7.1.1 Function Documentation

##### 7.1.1.1 set()

```
set (
    LIB_NAME compare_segmentation_lib )
```

Definition at line 2 of file CMakeLists.txt.

### 7.2 compare\_segmentation/test/CMakeLists.txt File Reference

#### Functions

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

#### 7.2.1 Function Documentation

### 7.2.1.1 include()

```
include (
    GoogleTest    )
```

Definition at line 1 of file CMakeLists.txt.

## 7.3 counting/CMakeLists.txt File Reference

### Functions

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

### 7.3.1 Function Documentation

#### 7.3.1.1 set()

```
set (
    LIB_NAME counting_lib )
```

Definition at line 2 of file CMakeLists.txt.

## 7.4 counting/test/CMakeLists.txt File Reference

### Functions

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

### 7.4.1 Function Documentation

#### 7.4.1.1 include()

```
include (
    GoogleTest    )
```

Definition at line 1 of file CMakeLists.txt.

## 7.5 csv\_lib/CMakeLists.txt File Reference

### Functions

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

### 7.5.1 Function Documentation

#### 7.5.1.1 set()

```
set (  
    LIB_NAME csv_lib )
```

Definition at line 2 of file CMakeLists.txt.

## 7.6 csv\_lib/test/CMakeLists.txt File Reference

### Functions

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

### 7.6.1 Function Documentation

#### 7.6.1.1 include()

```
include (  
    GoogleTest )
```

Definition at line 1 of file CMakeLists.txt.

## 7.7 fix\_csv/CMakeLists.txt File Reference

### Functions

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

#### 7.7.1 Function Documentation

##### 7.7.1.1 `set()`

```
set (
    LIB_NAME fix_mogasens_csv_lib )
```

Definition at line 2 of file CMakeLists.txt.

## 7.8 fix\_csv/test/CMakeLists.txt File Reference

### Functions

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

#### 7.8.1 Function Documentation

##### 7.8.1.1 `include()`

```
include (
    GoogleTest )
```

Definition at line 1 of file CMakeLists.txt.

## 7.9 confusion\_matrix/CMakeLists.txt File Reference

### Functions

- `set (LIB_NAME confusion_matrix_lib) set(LIB_HEADERS include/configuration.hpp include/data_set_identifier.hpp include/imu.hpp include/interpolated_data_set_paths.hpp include/manual_segmentation_point.hpp include/segment.hpp include/split_string.hpp) set(LIB_SOURCES src/configuration.cpp src/data_set_identifier.cpp src/imu.cpp src/interpolated_data_set_paths.cpp src/manual_segmentation_point.cpp src/segment.cpp src/split_string.cpp) add_library($`

#### 7.9.1 Function Documentation

##### 7.9.1.1 `set()`

```
set (  
    LIB_NAME confusion_matrix_lib )
```

Definition at line 2 of file CMakeLists.txt.

## 7.10 confusion\_matrix/test/CMakeLists.txt File Reference

### Functions

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

#### 7.10.1 Function Documentation

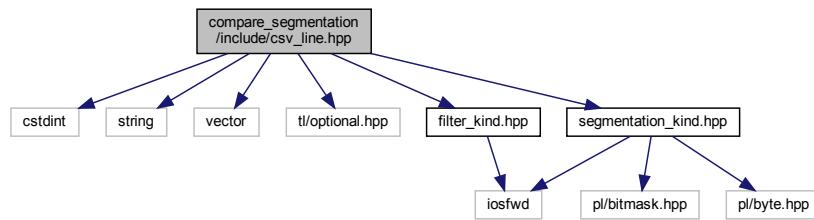
##### 7.10.1.1 `include()`

```
include (  
    GoogleTest )
```

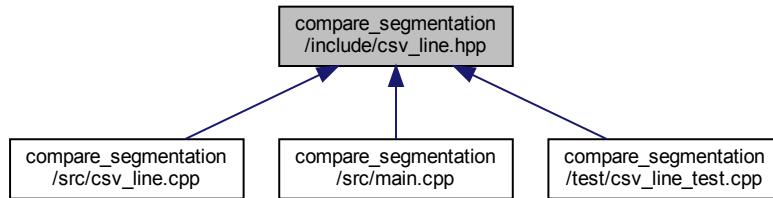
Definition at line 1 of file CMakeLists.txt.

## 7.11 compare\_segmentation/include/csv\_line.hpp File Reference

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



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



## Classes

- class [cs::CsvLineBuilder](#)

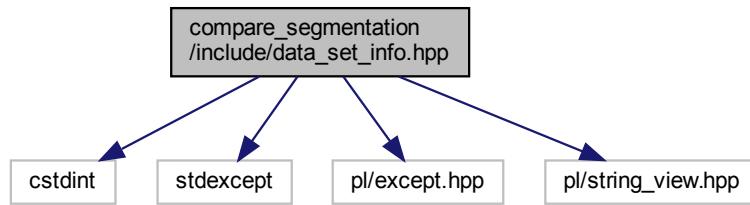
*Builder for a CSV line.*

## Namespaces

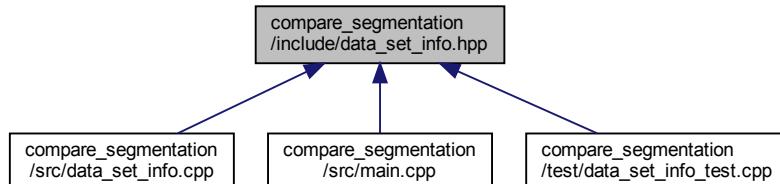
- [CS](#)

## 7.12 compare\_segmentation/include/data\_set\_info.hpp File Reference

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



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



## Classes

- struct [cs::data\\_set\\_info< Tag >](#)

*Meta function for data set tags.*

## Namespaces

- [cs](#)

## Macros

- [#define CS\\_SPECIALIZE\\_DATA\\_SET\\_INFO\(tag, string, repetitionCount\)](#)

## Functions

- `cs::PL_DEFINE_EXCEPTION_TYPE` (NoSuchDataSetException, std::logic\_error)
- `cs::CS_SPECIALIZE_DATA_SET_INFO` (Felix1, "11.17.39", 24)
- `cs::CS_SPECIALIZE_DATA_SET_INFO` (Felix2, "12.50.00", 20)
- `cs::CS_SPECIALIZE_DATA_SET_INFO` (Felix3, "13.00.09", 15)
- `cs::CS_SPECIALIZE_DATA_SET_INFO` (Marcelle1, "14.59.59", 10)
- `cs::CS_SPECIALIZE_DATA_SET_INFO` (Marcelle2, "15.13.22", 16)
- `cs::CS_SPECIALIZE_DATA_SET_INFO` (Marcelle3, "15.31.36", 18)
- `cs::CS_SPECIALIZE_DATA_SET_INFO` (Mike1, "14.07.33", 26)
- `cs::CS_SPECIALIZE_DATA_SET_INFO` (Mike2, "14.14.32", 22)
- `cs::CS_SPECIALIZE_DATA_SET_INFO` (Mike3, "14.20.28", 18)
- `cs::CS_SPECIALIZE_DATA_SET_INFO` (Andre1, "Andre\_liegestuetzen1", 27)
- `cs::CS_SPECIALIZE_DATA_SET_INFO` (Andre2, "Andre\_liegestuetzen2", 20)
- `cs::CS_SPECIALIZE_DATA_SET_INFO` (Andre3, "Andre\_liegestuetzen3", 17)
- `cs::CS_SPECIALIZE_DATA_SET_INFO` (AndreSquats1, "Andre\_Squats", 30)
- `cs::CS_SPECIALIZE_DATA_SET_INFO` (AndreSquats2, "Andre\_Squats2", 49)
- `cs::CS_SPECIALIZE_DATA_SET_INFO` (Jan1, "Jan\_liegestuetzen1", 25)
- `cs::CS_SPECIALIZE_DATA_SET_INFO` (Jan2, "Jan\_liegestuetzen2", 19)
- `cs::CS_SPECIALIZE_DATA_SET_INFO` (Jan3, "Jan\_liegestuetzen3", 13)
- `cs::CS_SPECIALIZE_DATA_SET_INFO` (Lucas1, "Lukas\_liegestuetzen1", 24)
- `cs::CS_SPECIALIZE_DATA_SET_INFO` (Lucas2, "Lukas\_liegestuetzen2", 19)
- `cs::CS_SPECIALIZE_DATA_SET_INFO` (Lucas3, "Lukas\_liegestuetzen3", 11)
- `std::uint64_t cs::repetitionCount` (pl::string\_view dataSet)

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

### 7.12.1 Macro Definition Documentation

#### 7.12.1.1 CS\_SPECIALIZE\_DATA\_SET\_INFO

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

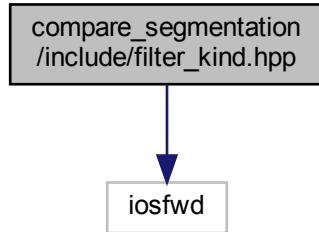
##### Value:

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

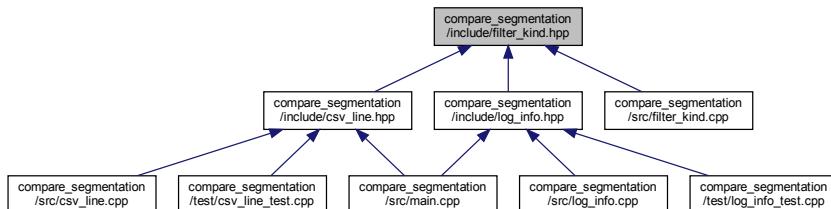
Definition at line 23 of file data\_set\_info.hpp.

## 7.13 compare\_segmentation/include/filter\_kind.hpp File Reference

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



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



## Namespaces

- `cs`

## Enumerations

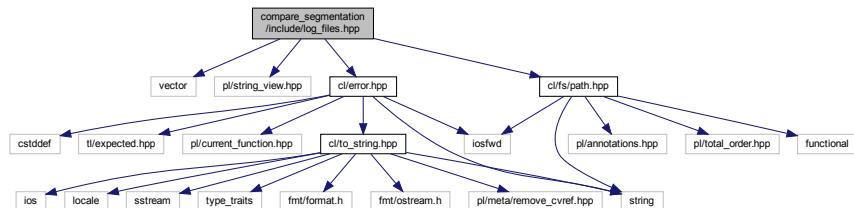
- enum `cs::FilterKind` { `cs::FilterKind::Butterworth`, `cs::FilterKind::MovingAverage` }
- Type for the different kinds of filters.*

## Functions

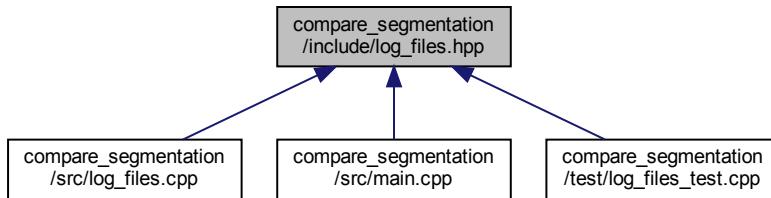
- `std::ostream & cs::operator<<` (`std::ostream &os`, `FilterKind filterKind`)
- Prints a FilterKind to an ostream.*

## 7.14 compare\_segmentation/include/log\_files.hpp File Reference

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



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



### Namespaces

- `cs`

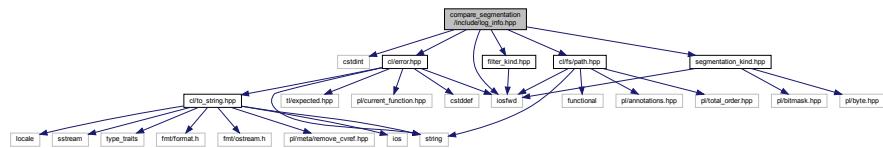
### Functions

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

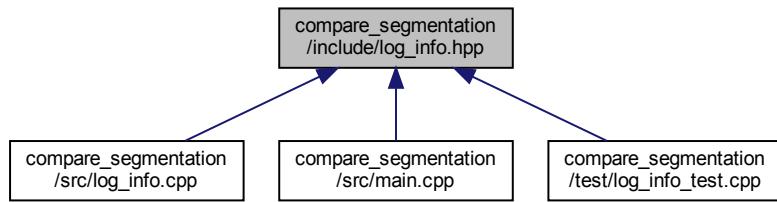
## 7.15 compare\_segmentation/include/log\_info.hpp File Reference

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

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



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



## Classes

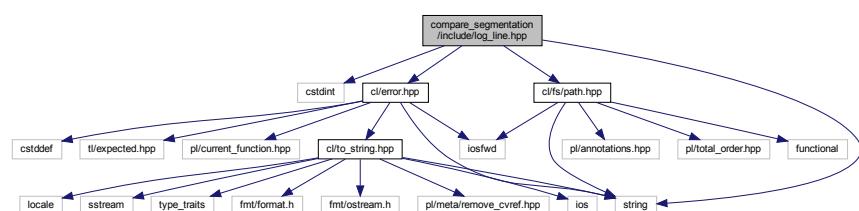
- class [cs::LogInfo](#)  
*Information about a log file.*

## Namespaces

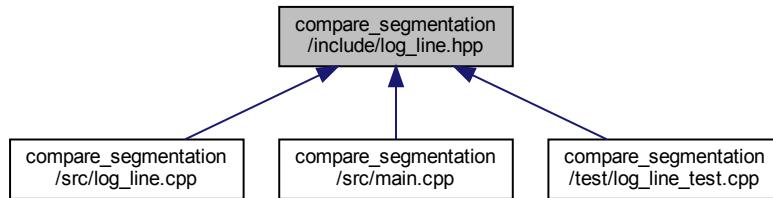
- [cs](#)

## 7.16 compare\_segmentation/include/log\_line.hpp File Reference

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



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



## Classes

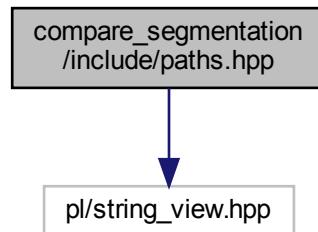
- class [cs::LogLine](#)  
*A line out of a log file.*

## Namespaces

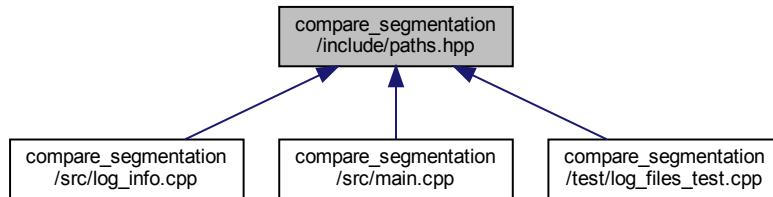
- [cs](#)

## 7.17 compare\_segmentation/include/paths.hpp File Reference

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



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



## Namespaces

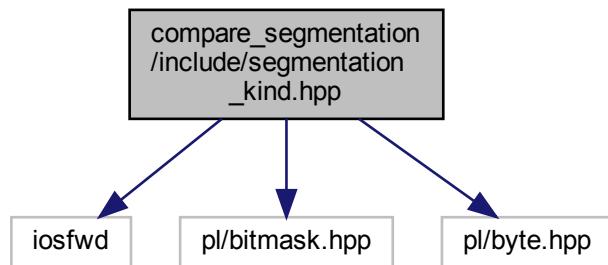
- `cs`

## Variables

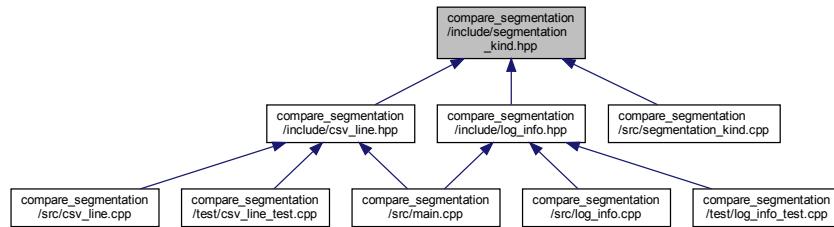
- `constexpr pl::string_view cs::logPath {"segmentation_comparison/logs"}`  
*Relative path to the directory containing the preprocessed log files.*
- `constexpr pl::string_view cs::oldLogPath {"segmentation_comparison/logs/old"}`  
*Relative path to the directory containing the old log files.*

## 7.18 compare\_segmentation/include/segmentation\_kind.hpp File Reference

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



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



## Namespaces

- [cs](#)

## Enumerations

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

*The segmentation kind.*

## Functions

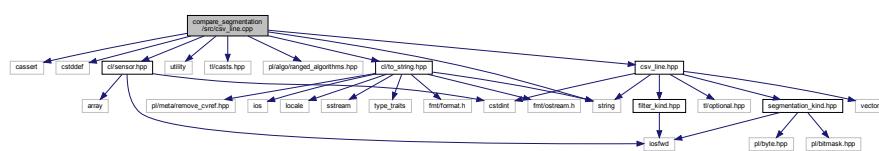
- std::ostream & [cs::operator<<](#) (std::ostream &os, SegmentationKind segmentationKind)

*Prints a SegmentationKind to an ostream.*

## 7.19 compare\_segmentation/src/csv\_line.cpp File Reference

```
#include <cassert>
#include <cstddef>
#include <string>
#include <utility>
#include <tl/casts.hpp>
#include <pl/algo/ranged_algorithms.hpp>
#include "cl/sensor.hpp"
#include "cl/to_string.hpp"
#include "csv_line.hpp"

Include dependency graph for csv_line.cpp:
```

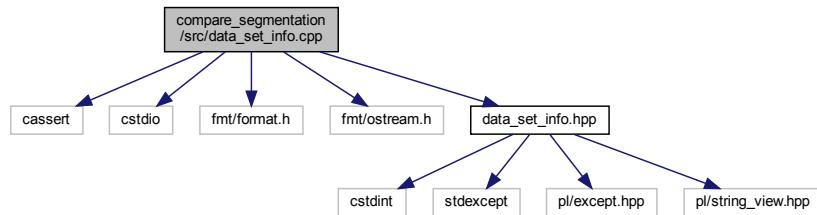


## Namespaces

- [cs](#)

## 7.20 compare\_segmentation/src/data\_set\_info.cpp File Reference

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



## Namespaces

- [cs](#)

## Functions

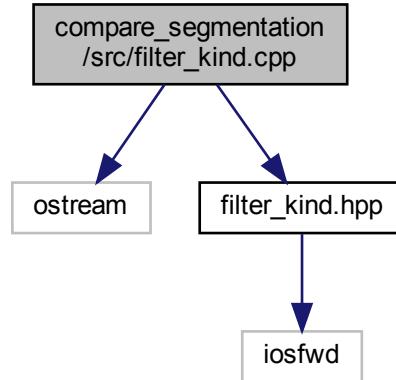
- `std::uint64_t cs::repetitionCount (pl::string_view dataSet)`

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

## 7.21 compare\_segmentation/src/filter\_kind.cpp File Reference

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

Include dependency graph for filter\_kind.cpp:



## Namespaces

- [cs](#)

## Functions

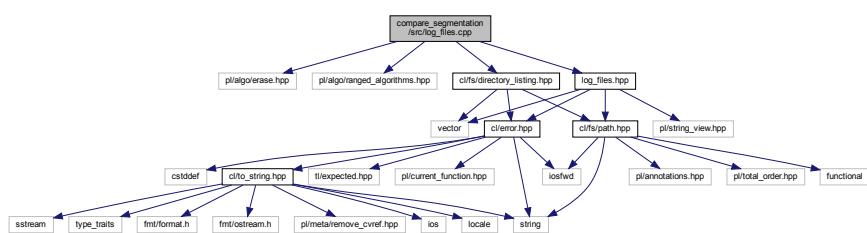
- `std::ostream & cs::operator<< (std::ostream &os, FilterKind filterKind)`

*Prints a FilterKind to an ostream.*

## 7.22 compare\_segmentation/src/log\_files.cpp File Reference

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

Include dependency graph for log\_files.cpp:



# Namespaces

- CS

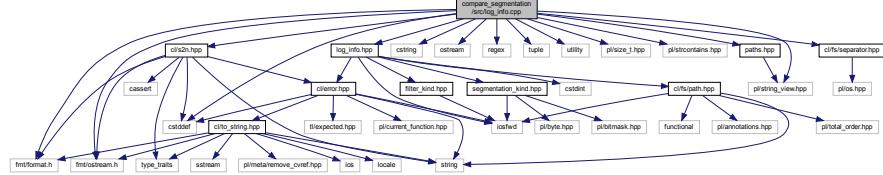
## Functions

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

## 7.23 compare\_segmentation/src/log\_info.cpp File Reference

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

Include dependency graph for log\_info.cpp:



## Namespaces

- CS

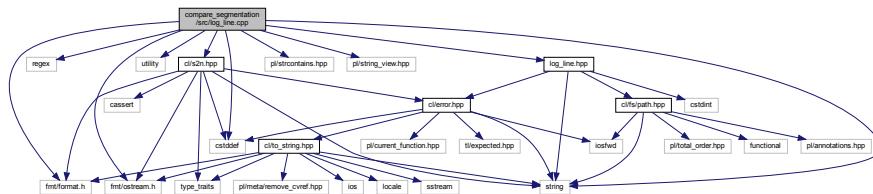
# Functions

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

## 7.24 compare\_segmentation/src/log\_line.cpp File Reference

```
#include <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:



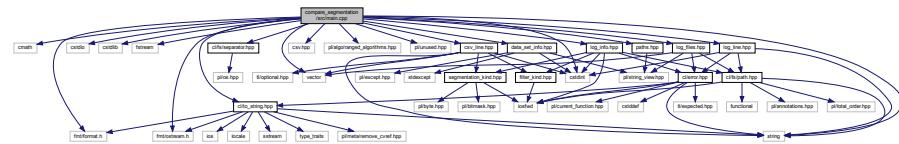
## Namespaces

- CS

## 7.25 compare\_segmentation/src/main.cpp File Reference

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

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



## Functions

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

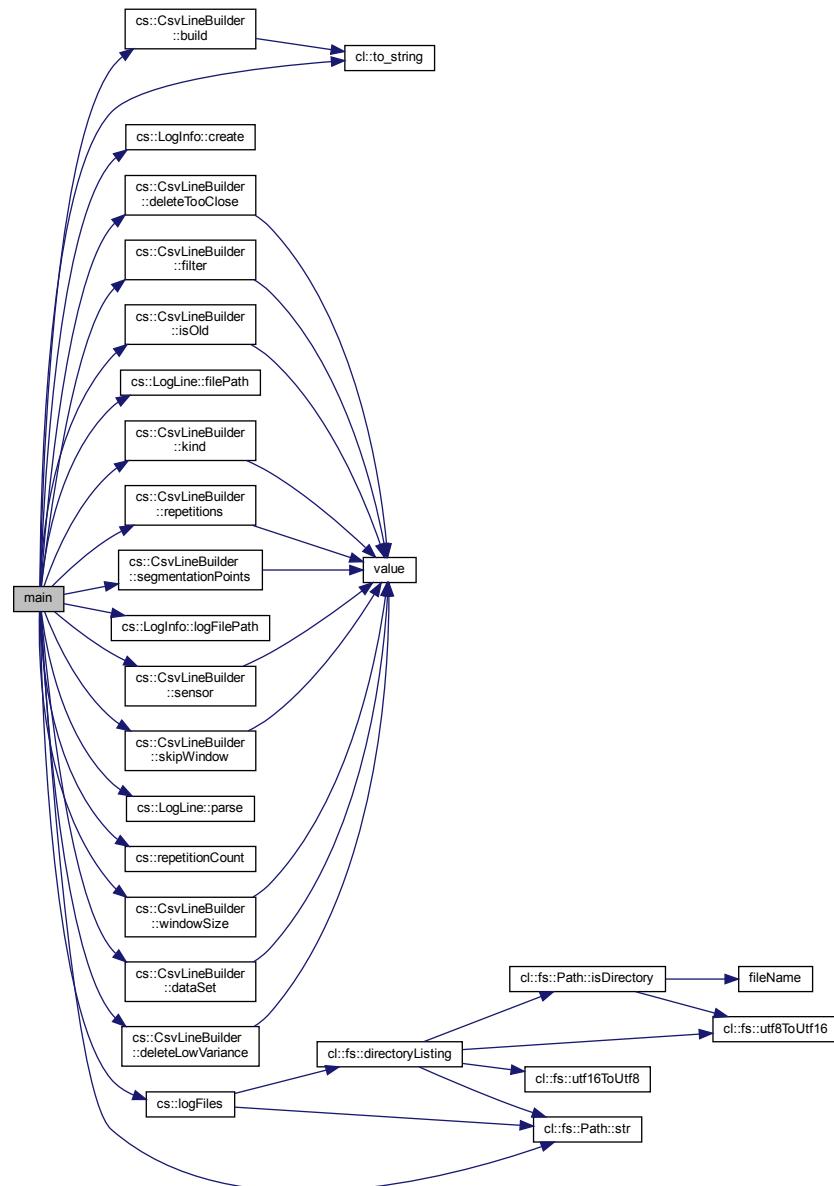
### 7.25.1 Function Documentation

### 7.25.1.1 main()

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

Definition at line 28 of file main.cpp.

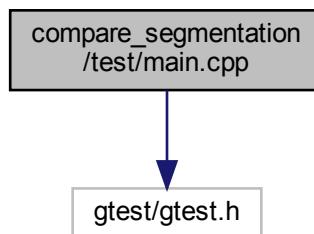
Here is the call graph for this function:



## 7.26 compare\_segmentation/test/main.cpp File Reference

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

Include dependency graph for main.cpp:



## Functions

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

### 7.26.1 Function Documentation

#### 7.26.1.1 `main()`

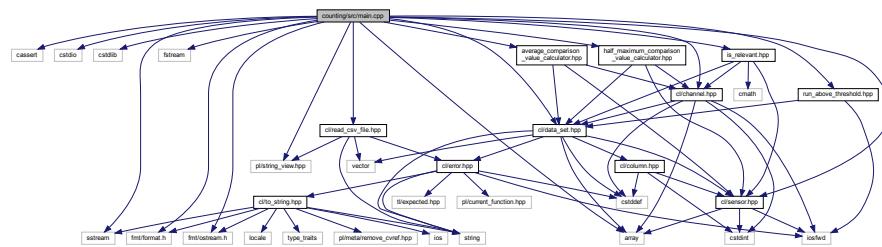
```
int main (
    int argc,
    char * argv[] )
```

Definition at line 3 of file main.cpp.

## 7.27 counting/src/main.cpp File Reference

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

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



# Functions

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

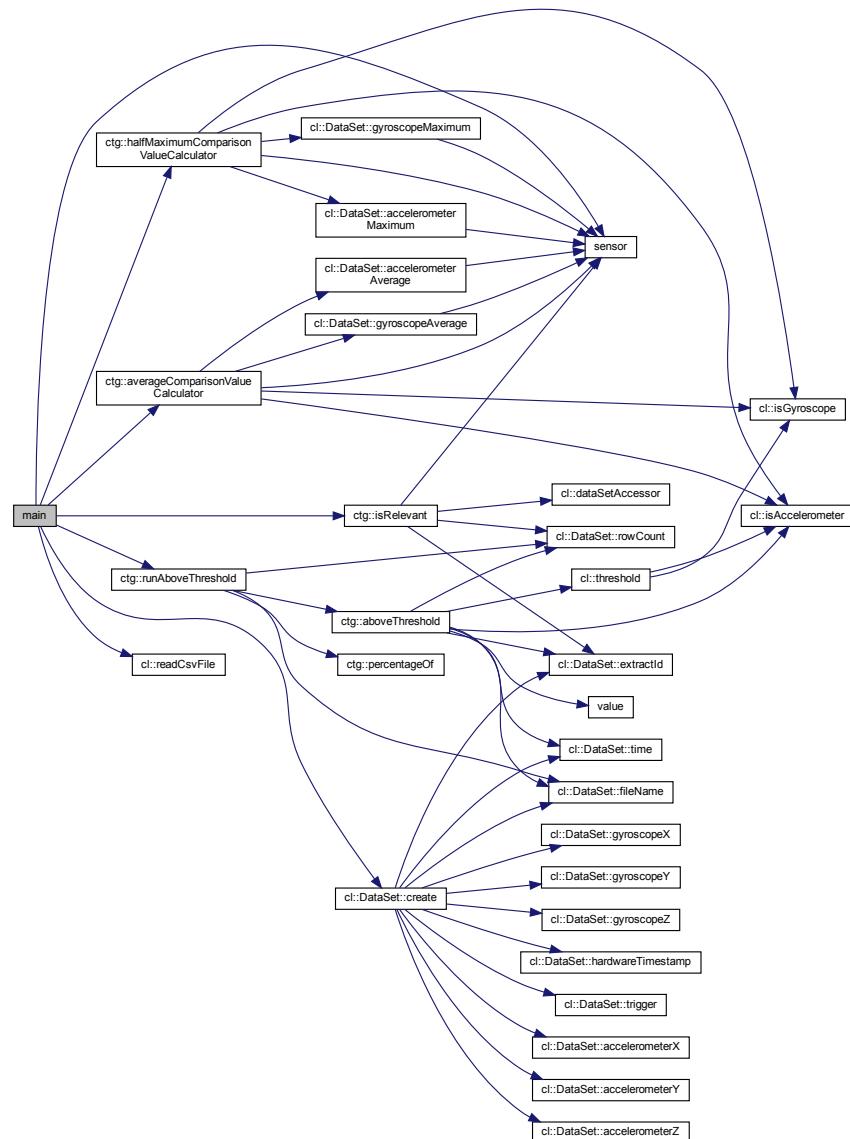
### 7.27.1 Function Documentation

## 7.27.1.1 main()

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

Definition at line 24 of file main.cpp.

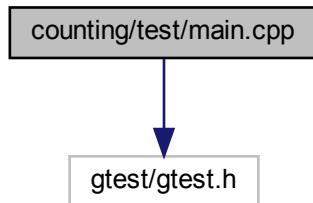
Here is the call graph for this function:



## 7.28 counting/test/main.cpp File Reference

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

Include dependency graph for main.cpp:



## Functions

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

### 7.28.1 Function Documentation

#### 7.28.1.1 `main()`

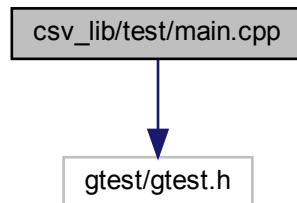
```
int main (
    int argc,
    char * argv[] )
```

Definition at line 3 of file main.cpp.

## 7.29 csv\_lib/test/main.cpp File Reference

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

Include dependency graph for main.cpp:



# Functions

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

### 7.29.1 Function Documentation

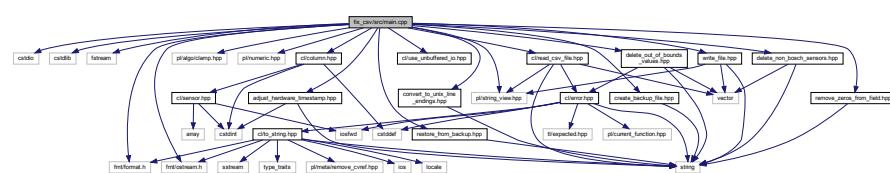
### 7.29.1.1 main()

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

Definition at line 3 of file main.cpp.

## 7.30 fix\_csv/src/main.cpp File Reference

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



## Functions

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

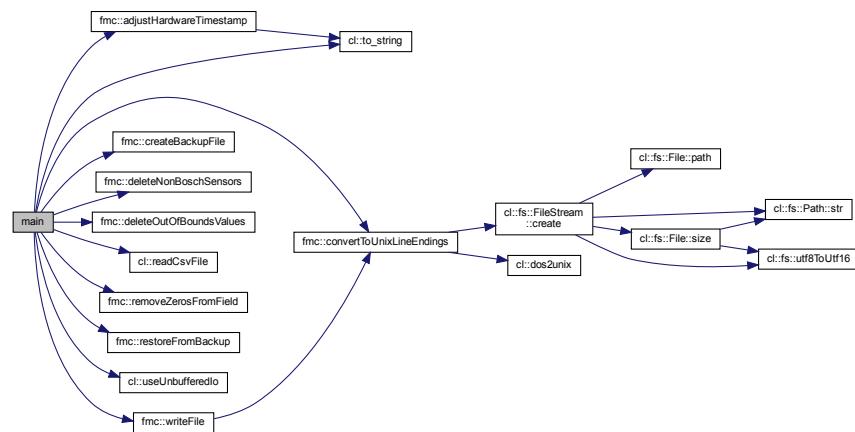
### 7.30.1 Function Documentation

#### 7.30.1.1 main()

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

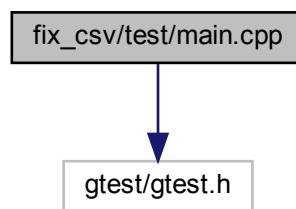
Definition at line 26 of file main.cpp.

Here is the call graph for this function:



### 7.31 fix\_csv/test/main.cpp File Reference

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



## Functions

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

### 7.31.1 Function Documentation

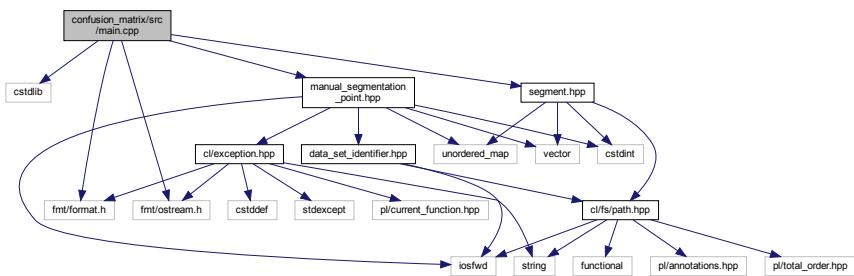
#### 7.31.1.1 main()

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

Definition at line 3 of file main.cpp.

## 7.32 confusion\_matrix/src/main.cpp File Reference

```
#include <cstdlib>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include "manual_segmentation_point.hpp"
#include "segment.hpp"
Include dependency graph for main.cpp:
```



## Functions

- int [main](#) ()

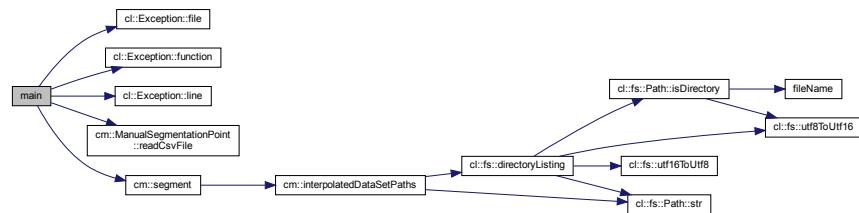
### 7.32.1 Function Documentation

### 7.32.1.1 main()

```
int main ( )
```

Definition at line 9 of file main.cpp.

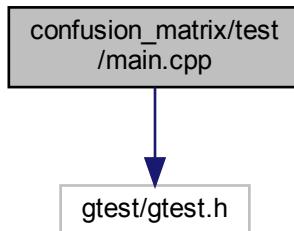
Here is the call graph for this function:



## 7.33 confusion\_matrix/test/main.cpp File Reference

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

Include dependency graph for main.cpp:



## Functions

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

### 7.33.1 Function Documentation

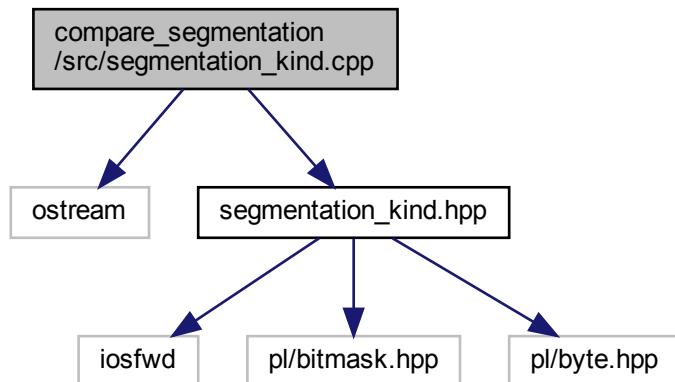
### 7.33.1.1 main()

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

Definition at line 3 of file main.cpp.

## 7.34 compare\_segmentation/src/segmentation\_kind.cpp File Reference

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



## Namespaces

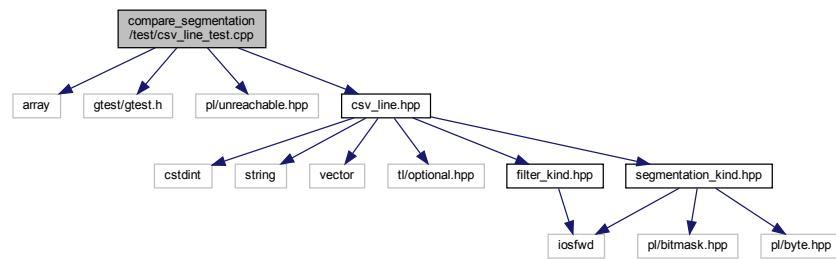
- `cs`

## Functions

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

## 7.35 compare\_segmentation/test/csv\_line\_test.cpp File Reference

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



## Functions

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

### 7.35.1 Function Documentation

#### 7.35.1.1 TEST()

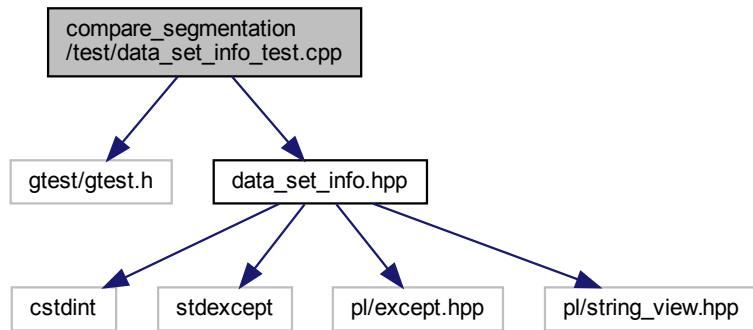
```
TEST (
    CsvLine ,
    shouldWork )
```

Definition at line 30 of file csv\_line\_test.cpp.

## 7.36 compare\_segmentation/test/data\_set\_info\_test.cpp File Reference

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

Include dependency graph for data\_set\_info\_test.cpp:



## Functions

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

### 7.36.1 Function Documentation

#### 7.36.1.1 TEST()

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

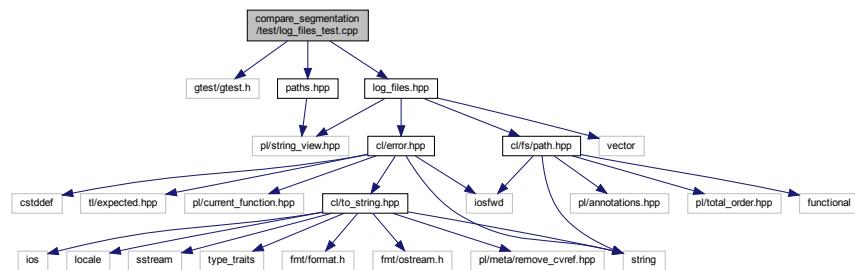
Definition at line 5 of file `data_set_info_test.cpp`.

Here is the call graph for this function:



## 7.37 compare\_segmentation/test/log\_files\_test.cpp File Reference

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



## Functions

- `TEST` (`logFiles`, `shouldFindLogFiles`)
- `TEST` (`logFiles`, `shouldFindOldLogFiles`)
- `TEST` (`logFiles`, `shouldNotFindGarbage`)

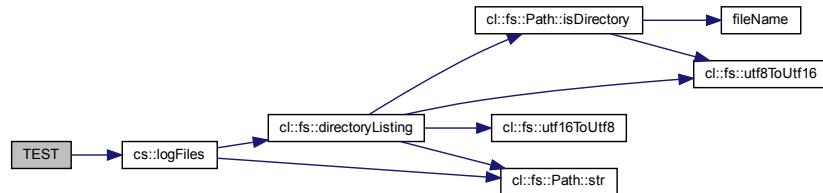
### 7.37.1 Function Documentation

#### 7.37.1.1 TEST() [1/3]

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

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

Here is the call graph for this function:

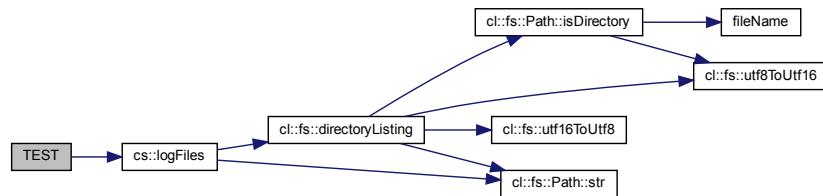


### 7.37.1.2 TEST() [2/3]

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

Definition at line 23 of file log\_files\_test.cpp.

Here is the call graph for this function:

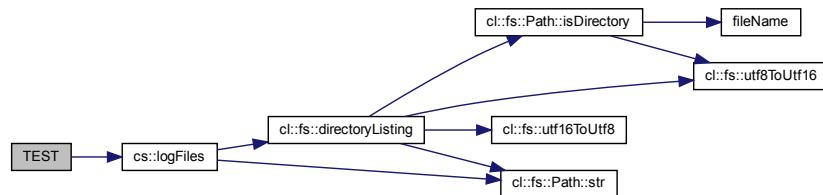


### 7.37.1.3 TEST() [3/3]

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

Definition at line 40 of file log\_files\_test.cpp.

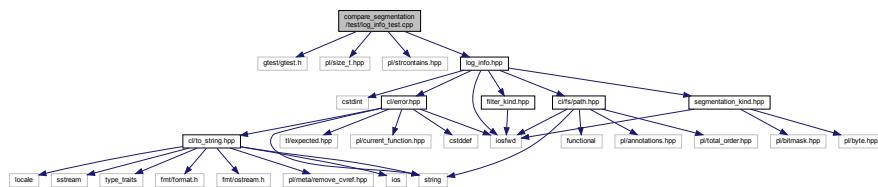
Here is the call graph for this function:



## 7.38 compare\_segmentation/test/log\_info\_test.cpp File Reference

```
#include "gtest/gtest.h"
#include <pl/size_t.hpp>
#include <pl/strcontains.hpp>
```

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



# Functions

- **TEST** (LogInfo, shouldWork)
  - **TEST** (LogInfo, shouldWork2)
  - **TEST** (LogInfo, shouldWork3)
  - **TEST** (LogInfo, shouldWork4)
  - **TEST** (LogInfo, shouldWork5)
  - **TEST** (LogInfo, shouldWork6)
  - **TEST** (LogInfo, shouldWork7)
  - **TEST** (LogInfo, shouldWork8)
  - **TEST** (LogInfo, shouldWork9)
  - **TEST** (LogInfo, shouldWorkWithOldPath)
  - **TEST** (LogInfo, shouldWorkWithOldPath2)
  - **TEST** (LogInfo, shouldResultInErrorIfFilePathIsTooShort)
  - **TEST** (LogInfo, shouldFailIfSkipWindowIsInvalid)
  - **TEST** (LogInfo, shouldFailIfDeleteTooCloseIsInvalid)
  - **TEST** (LogInfo, shouldFailIfDeleteTooLowVarianceIsInvalid)
  - **TEST** (LogInfo, shouldFailIfSegmentationKindIsInvalid)
  - **TEST** (LogInfo, shouldFailIfWindowSizeIsInvalid)
  - **TEST** (LogInfo, shouldFailIfFilterIsInvalid)
  - **TEST** (LogInfo, shouldCreateUninitializedObjectWhenDefaultConstructorIsCalled)

### 7.38.1 Function Documentation

### 7.38.1.1 TEST() [1/19]

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

Definition at line 388 of file log\_info\_test.cpp.

### 7.38.1.2 TEST() [2/19]

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

Definition at line 341 of file log\_info\_test.cpp.

Here is the call graph for this function:



### 7.38.1.3 TEST() [3/19]

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

Definition at line 350 of file log\_info\_test.cpp.

Here is the call graph for this function:



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

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

Definition at line 379 of file log\_info\_test.cpp.

Here is the call graph for this function:



#### 7.38.1.5 TEST() [5/19]

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

Definition at line 359 of file log\_info\_test.cpp.

Here is the call graph for this function:



**7.38.1.6 TEST() [6/19]**

```
TEST (
    LogInfo ,
    shouldFailIfSkipWindowIsValid )
```

Definition at line 332 of file log\_info\_test.cpp.

Here is the call graph for this function:

**7.38.1.7 TEST() [7/19]**

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

Definition at line 368 of file log\_info\_test.cpp.

Here is the call graph for this function:



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

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

Definition at line 325 of file log\_info\_test.cpp.

Here is the call graph for this function:



### 7.38.1.9 TEST() [9/19]

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

Definition at line 8 of file log\_info\_test.cpp.

Here is the call graph for this function:



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

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

Definition at line 37 of file log\_info\_test.cpp.

Here is the call graph for this function:

**7.38.1.11 TEST() [11/19]**

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

Definition at line 66 of file log\_info\_test.cpp.

Here is the call graph for this function:



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

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

Definition at line 95 of file log\_info\_test.cpp.

Here is the call graph for this function:

**7.38.1.13 TEST() [13/19]**

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

Definition at line 124 of file log\_info\_test.cpp.

Here is the call graph for this function:



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

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

Definition at line 153 of file log\_info\_test.cpp.

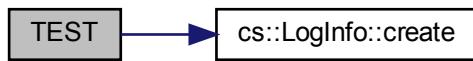
Here is the call graph for this function:

**7.38.1.15 TEST() [15/19]**

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

Definition at line 182 of file log\_info\_test.cpp.

Here is the call graph for this function:



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

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

Definition at line 211 of file log\_info\_test.cpp.

Here is the call graph for this function:

**7.38.1.17 TEST() [17/19]**

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

Definition at line 240 of file log\_info\_test.cpp.

Here is the call graph for this function:



### 7.38.1.18 TEST() [18/19]

```
TEST (
    LogInfo ,
    shouldWorkWithOldPath )
```

Definition at line 269 of file log\_info\_test.cpp.

Here is the call graph for this function:



### 7.38.1.19 TEST() [19/19]

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

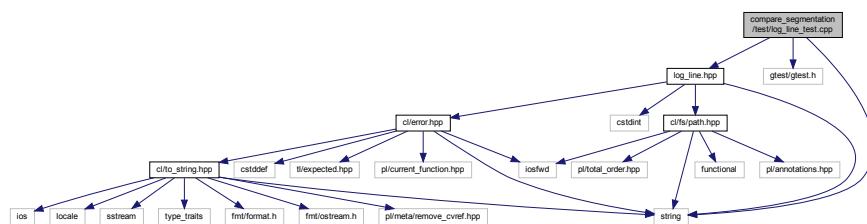
Definition at line 297 of file log\_info\_test.cpp.

Here is the call graph for this function:



## 7.39 compare\_segmentation/test/log\_line\_test.cpp File Reference

```
#include <string>
#include "gtest/gtest.h"
#include "log_line.hpp"
Include dependency graph for log_line_test.cpp:
```



## Functions

- [TEST](#) (LogLine, shouldWorkWithPreprocessedLine)
- [TEST](#) (LogLine, shouldWorkWithOldLine)
- [TEST](#) (LogLine, shouldNotMatchGarbage)
- [TEST](#) (LogLine, shouldNotParseGarbageSensor)

### 7.39.1 Function Documentation

#### 7.39.1.1 TEST() [1/4]

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

Definition at line 41 of file log\_line\_test.cpp.

Here is the call graph for this function:



#### 7.39.1.2 TEST() [2/4]

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

Definition at line 48 of file log\_line\_test.cpp.

Here is the call graph for this function:



### 7.39.1.3 TEST() [3/4]

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

Definition at line 25 of file log\_line\_test.cpp.

Here is the call graph for this function:



### 7.39.1.4 TEST() [4/4]

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

Definition at line 9 of file log\_line\_test.cpp.

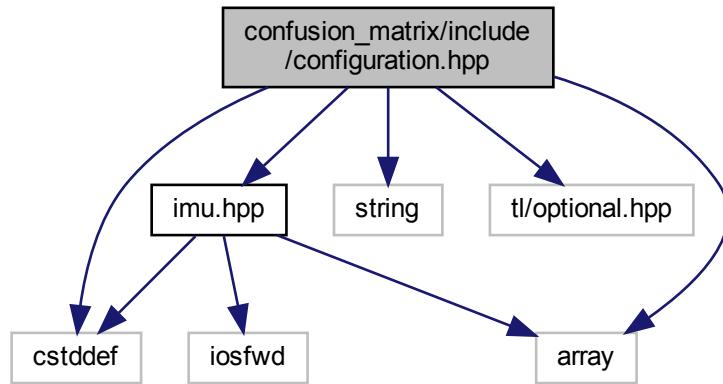
Here is the call graph for this function:



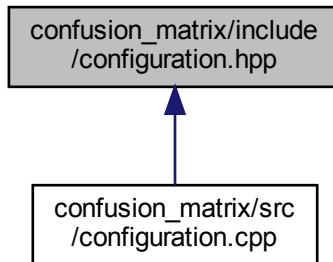
## 7.40 confusion\_matrix/include/configuration.hpp File Reference

```
#include <cstddef>  
#include <array>  
#include <string>  
#include <t1/optional.hpp>
```

```
#include "imu.hpp"
Include dependency graph for configuration.hpp:
```



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



## Classes

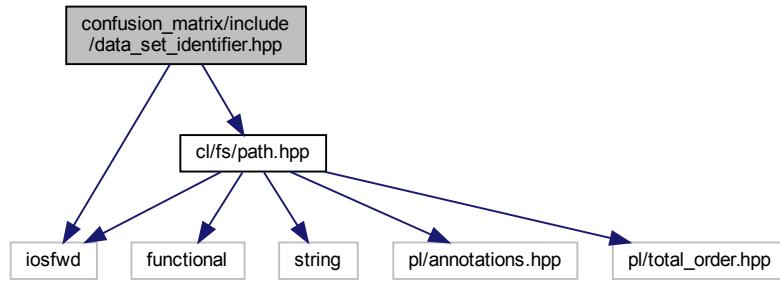
- class [cm::Configuration](#)  
*Represents a possible configuration for the Python segmentor.*
- class [cm::Configuration::Builder](#)  
*Builder type for Configuration.*

## Namespaces

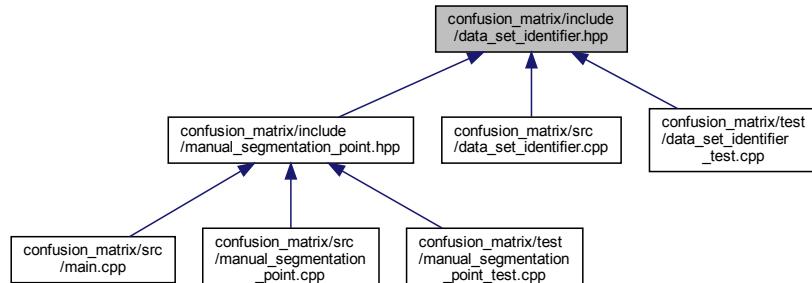
- [cm](#)

## 7.41 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_Y }`

## Functions

- std::ostream & `cm::operator<<` (std::ostream &os, DataSetIdentifier dsi)  
*Prints a DataSetIdentifier to an ostream.*
- DataSetIdentifier `cm::toDataSetIdentifier` (const cl::fs::Path &path)  
*Converts a path to a CSV file to the corresponding DataSetIdentifier.*

### 7.41.1 Macro Definition Documentation

#### 7.41.1.1 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.

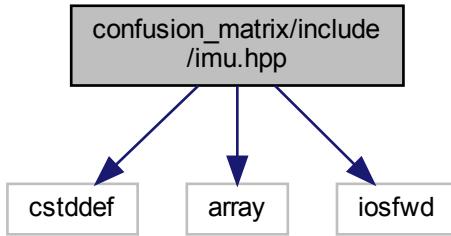
#### 7.41.1.2 CM\_DATA\_SET\_IDENTIFIER\_X

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

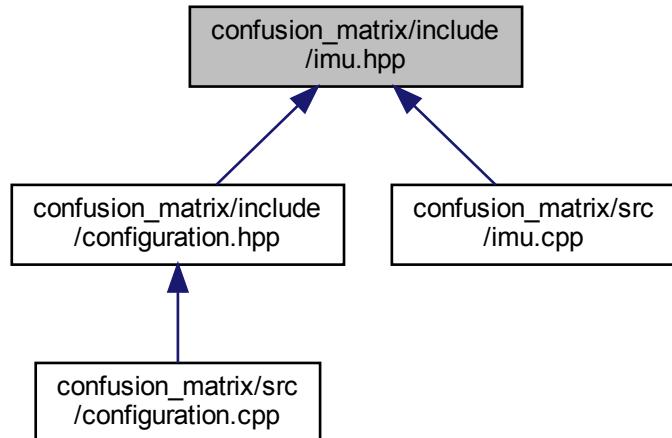
Definition at line 31 of file data\_set\_identifier.hpp.

## 7.42 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 }`

## Functions

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

## Variables

- `constexpr std::size_t cm::imuCount`
- `constexpr std::array<Imu, imuCount> cm::imus`

### 7.42.1 Macro Definition Documentation

#### 7.42.1.1 CM\_IMU

```
#define CM_IMU
```

##### Value:

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

Definition at line 10 of file imu.hpp.

#### 7.42.1.2 CM\_IMU\_X [1/3]

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

Definition at line 15 of file imu.hpp.

#### 7.42.1.3 CM\_IMU\_X [2/3]

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

Definition at line 15 of file imu.hpp.

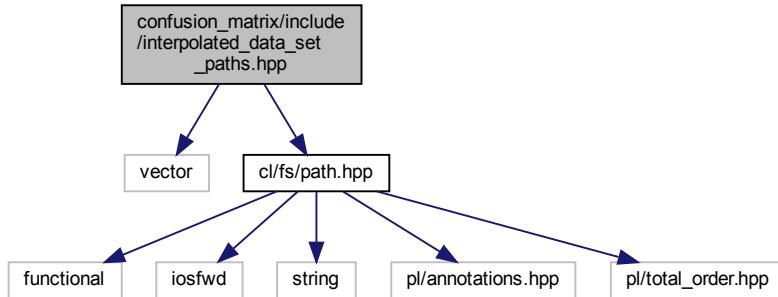
#### 7.42.1.4 CM\_IMU\_X [3/3]

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

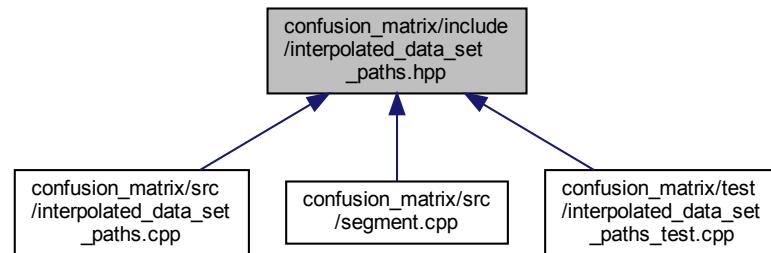
Definition at line 15 of file imu.hpp.

## 7.43 confusion\_matrix/include/interpolated\_data\_set\_paths.hpp File Reference

```
#include <vector>
#include <ccl/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`

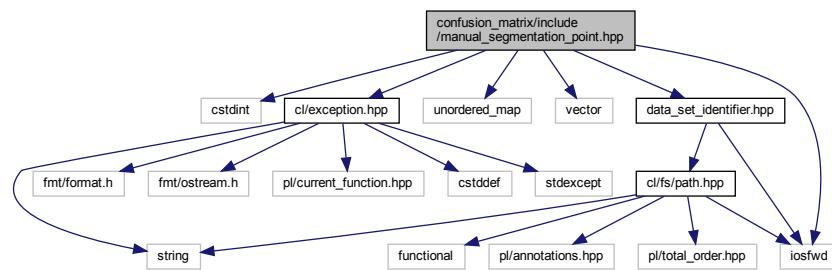
## Functions

- std::vector< [cl::fs::Path](#) > cm::interpolatedDataSetPaths ()

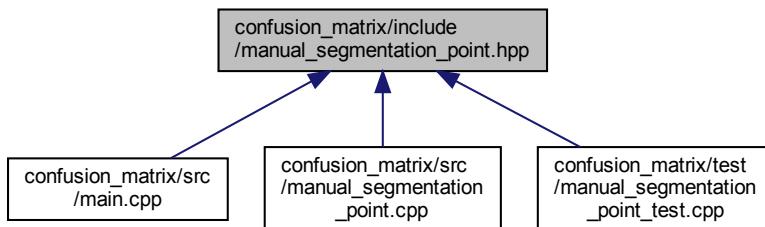
*Returns the paths to the interpolated data sets.*

## 7.44 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

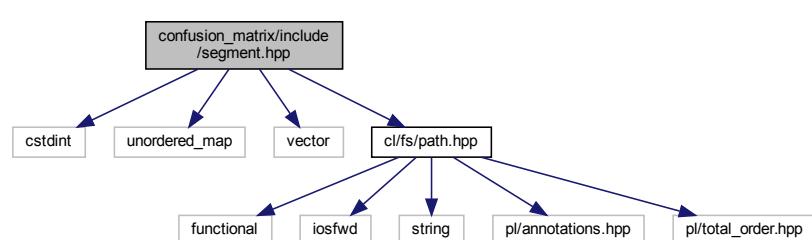
- [cm](#)

## 7.45 confusion\_matrix/include/segment.hpp File Reference

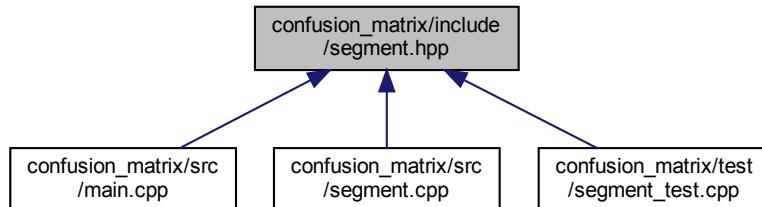
```
#include <cstdint>
#include <unordered_map>
#include <vector>
```

```
#include <cl/fs/path.hpp>
```

Include dependency graph for segment.hpp:



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



## Namespaces

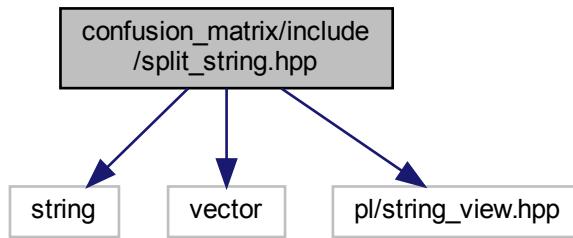
- [cm](#)

## Functions

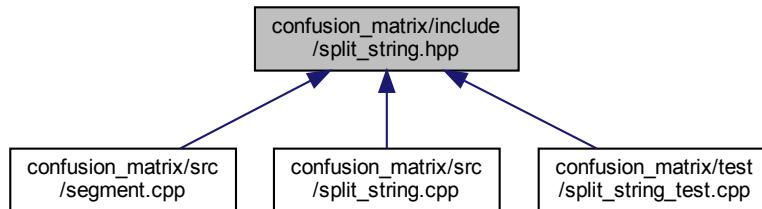
- `std::unordered_map< cl::fs::Path, std::vector< std::uint64_t > > cm::segment ()`

## 7.46 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

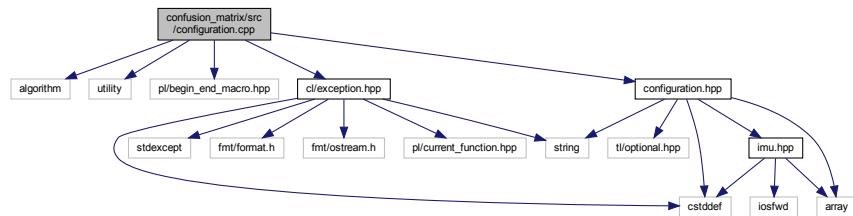
- [cm](#)

### Functions

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

## 7.47 confusion\_matrix/src/configuration.cpp File Reference

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



## Namespaces

- cm

## Macros

- #define CM\_ENSURE\_HAS\_VALUE(dataMember)
- #define CM\_CONTAINS(container, value) contains(container.begin(), container.end(), value)
- #define CM\_ENSURE\_CONTAINS(container, dataMember)

### 7.47.1 Macro Definition Documentation

#### 7.47.1.1 CM\_CONTAINS

```
#define CM_CONTAINS(
    container,
    value ) contains(container.begin(), container.end(), value)
```

#### 7.47.1.2 CM\_ENSURE\_CONTAINS

```
#define CM_ENSURE_CONTAINS (
    container,
    dataMember )
```

##### Value:

```
PL_BEGIN_MACRO
if (!CM_CONTAINS(container, dataMember)) {
    CL_THROW_FMT(
        "\\"{}\" is not a valid option for \"{}\"", *dataMember, #dataMember);
}
PL_END_MACRO
```

### 7.47.1.3 CM\_ENSURE\_HAS\_VALUE

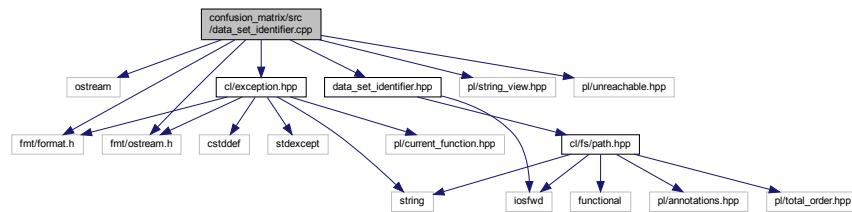
```
#define CM_ENSURE_HAS_VALUE(
    dataMember )
```

#### Value:

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

## 7.48 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.hpp:
```



## Namespaces

- [cm](#)

## Macros

- `#define CM_DATA_SET_IDENTIFIER_X(enm) case DataSetIdentifier::enm: return #enm;`
- `#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.*

### 7.48.1 Macro Definition Documentation

#### 7.48.1.1 CM\_DATA\_SET\_IDENTIFIER\_X

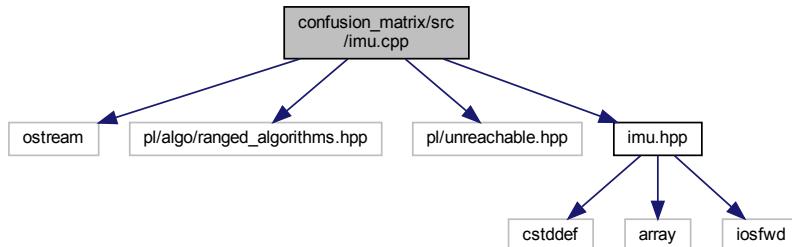
```
#define CM_DATA_SET_IDENTIFIER_X( enm ) case DataSetIdentifier::enm: return #enm;
```

#### 7.48.1.2 DSI

```
#define DSI DataSetIdentifier
```

## 7.49 confusion\_matrix/src/imu.cpp File Reference

```
#include <iostream>
#include <pl/algo/ranged_algorithms.hpp>
#include <pl/unreachable.hpp>
#include "imu.hpp"
Include dependency graph for imu.cpp:
```



## Namespaces

- `cm`

## Macros

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

## Functions

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

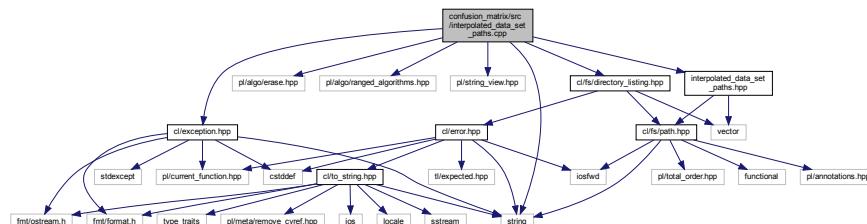
## 7.49.1 Macro Definition Documentation

### 7.49.1.1 CM\_IMU\_X

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

## 7.50 confusion\_matrix/src/interpolated\_data\_set\_paths.cpp File Reference

```
#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

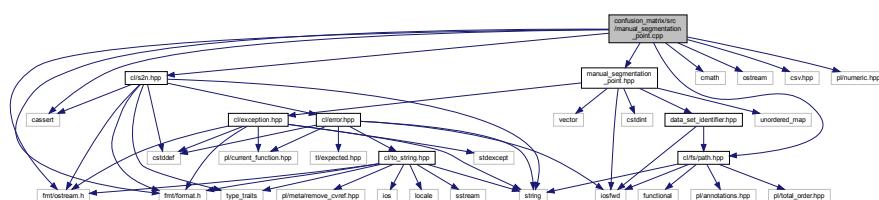
- `std::vector< cl::fs::Path > cm::interpolatedDataSetPaths ()`

*Returns the paths to the interpolated data sets.*

## 7.51 confusion\_matrix/src/manual\_segmentation\_point.cpp File Reference

```
#include <cassert>
#include <cmath>
#include <iostream>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <csv.hpp>
#include <pl/numeric.hpp>
#include "cl/fs/path.hpp"
#include "cl/s2n.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 &manual)`

### 7.51.1 Macro Definition Documentation

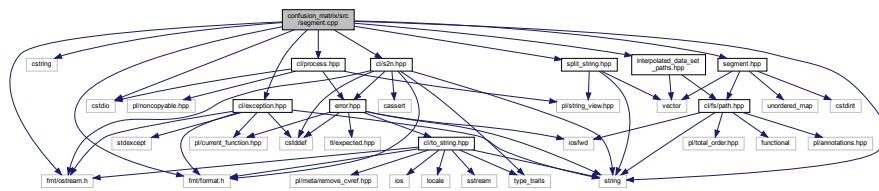
#### 7.51.1.1 DSI

```
#define DSI DataSetIdentifier
```

## 7.52 confusion\_matrix/src/segment.cpp File Reference

```
#include <cstdio>
#include <cstring>
#include <string>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <cl/exception.hpp>
#include <cl/process.hpp>
#include <cl/s2n.hpp>
#include "interpolated_data_set_paths.hpp"
#include "segment.hpp"
#include "split_string.hpp"

Include dependency graph for segment.cpp:
```



## Namespaces

- [cm](#)

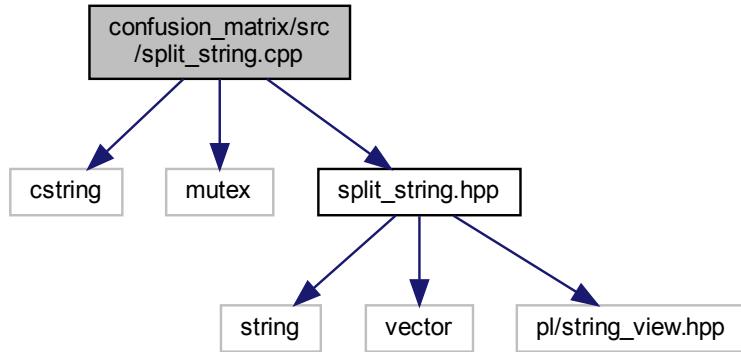
## Functions

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

## 7.53 confusion\_matrix/src/split\_string.cpp File Reference

```
#include <cstring>
#include <mutex>
#include "split_string.hpp"
```

Include dependency graph for split\_string.cpp:



## Namespaces

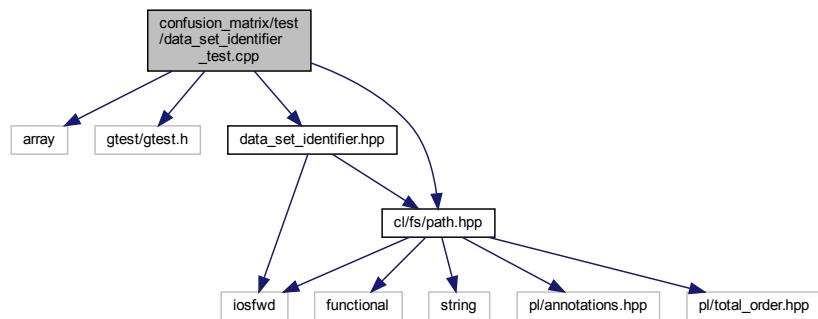
- cm

## Functions

- std::vector< std::string > cm::splitString (std::string string, pl::string\_view splitBy)  
*Splits string by splitBy.*

## 7.54 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_test.cpp:
```



## Macros

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

## Functions

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

### 7.54.1 Macro Definition Documentation

#### 7.54.1.1 DSI

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

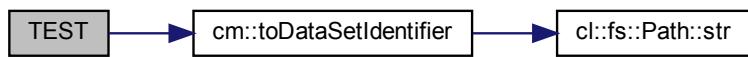
### 7.54.2 Function Documentation

#### 7.54.2.1 TEST()

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

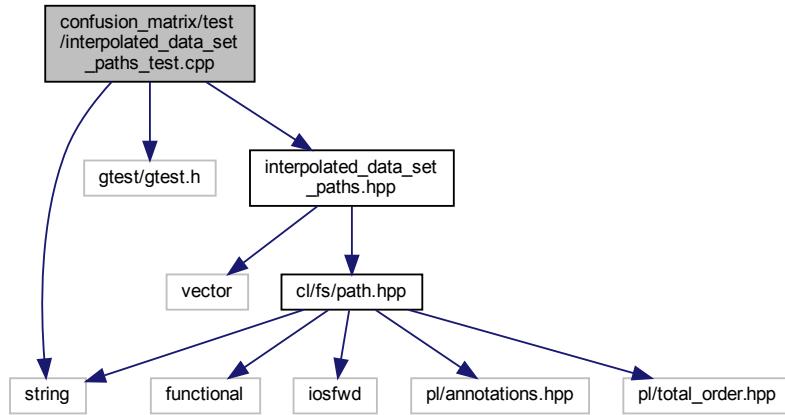
Definition at line 9 of file `data_set_identifier_test.cpp`.

Here is the call graph for this function:



## 7.55 confusion\_matrix/test/interpolated\_data\_set\_paths\_test.cpp File Reference

```
#include <string>
#include "gtest/gtest.h"
#include "interpolated_data_set_paths.hpp"
Include dependency graph for interpolated_data_set_paths_test.cpp:
```



## Functions

- [TEST](#) (`interpolatedDataSetPaths`, `shouldFetchPaths`)

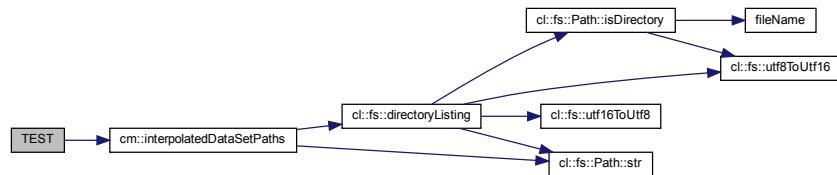
### 7.55.1 Function Documentation

#### 7.55.1.1 TEST()

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

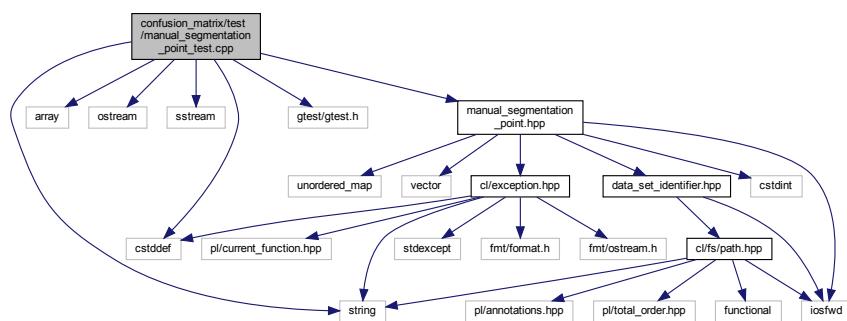
Definition at line 7 of file `interpolated_data_set_paths_test.cpp`.

Here is the call graph for this function:



## 7.56 confusion\_matrix/test/manual\_segmentation\_point\_test.cpp File Reference

```
#include <cstddef>
#include <array>
#include <iostream>
#include <iostream>
#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)`

#### 7.56.1 Macro Definition Documentation

##### 7.56.1.1 DSI

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

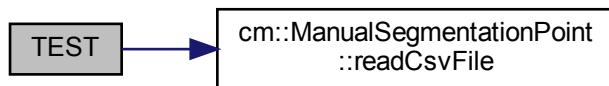
## 7.56.2 Function Documentation

### 7.56.2.1 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:



### 7.56.2.2 TEST() [2/11]

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

Definition at line 12 of file manual\_segmentation\_point\_test.cpp.

### 7.56.2.3 TEST() [3/11]

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

Definition at line 82 of file manual\_segmentation\_point\_test.cpp.

**7.56.2.4 TEST() [4/11]**

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

Definition at line 64 of file manual\_segmentation\_point\_test.cpp.

**7.56.2.5 TEST() [5/11]**

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

Definition at line 70 of file manual\_segmentation\_point\_test.cpp.

**7.56.2.6 TEST() [6/11]**

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

Definition at line 76 of file manual\_segmentation\_point\_test.cpp.

**7.56.2.7 TEST() [7/11]**

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

Definition at line 58 of file manual\_segmentation\_point\_test.cpp.

**7.56.2.8 TEST() [8/11]**

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

Definition at line 370 of file manual\_segmentation\_point\_test.cpp.

**7.56.2.9 TEST() [9/11]**

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

Definition at line 46 of file manual\_segmentation\_point\_test.cpp.

**7.56.2.10 TEST() [10/11]**

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

Definition at line 22 of file manual\_segmentation\_point\_test.cpp.

**7.56.2.11 TEST() [11/11]**

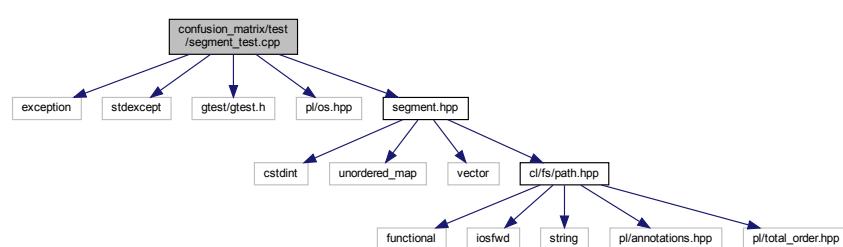
```
TEST (
    ManualSegmentationPoint ,
    shouldThrowWhenConstructingWithInvalidSecond )
```

Definition at line 34 of file manual\_segmentation\_point\_test.cpp.

## 7.57 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)

### 7.57.1 Macro Definition Documentation

#### 7.57.1.1 EXPECT\_SEGMENTATION\_POINTS

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

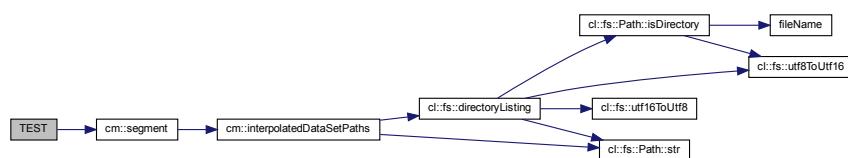
### 7.57.2 Function Documentation

#### 7.57.2.1 TEST()

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

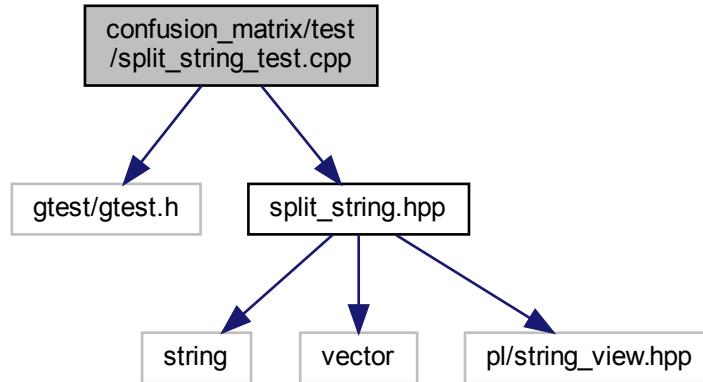
Definition at line 11 of file segment\_test.cpp.

Here is the call graph for this function:



## 7.58 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`)

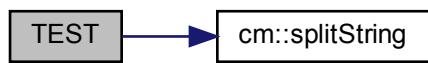
#### 7.58.1 Function Documentation

##### 7.58.1.1 TEST()

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

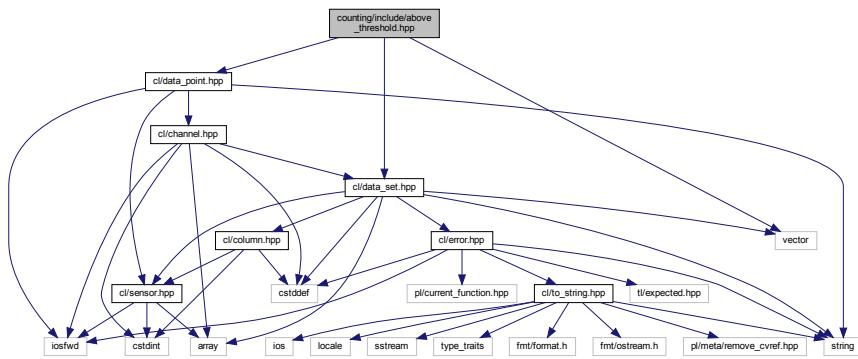
Definition at line 5 of file `split_string_test.cpp`.

Here is the call graph for this function:

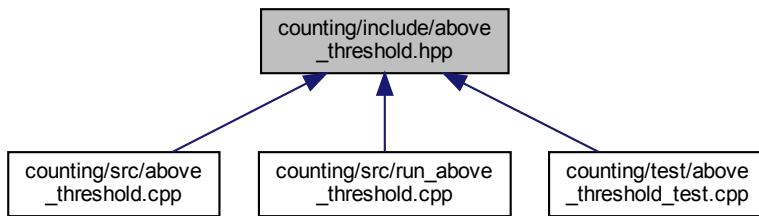


## 7.59 counting/include/above\_threshold.hpp File Reference

```
#include <vector>
#include "cl/data_point.hpp"
#include "cl/data_set.hpp"
Include dependency graph for above_threshold.hpp:
```



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



### Namespaces

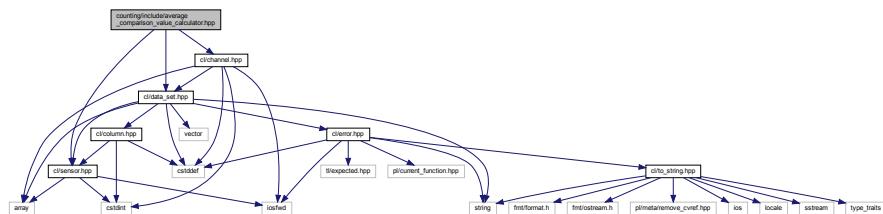
- [ctg](#)

### Functions

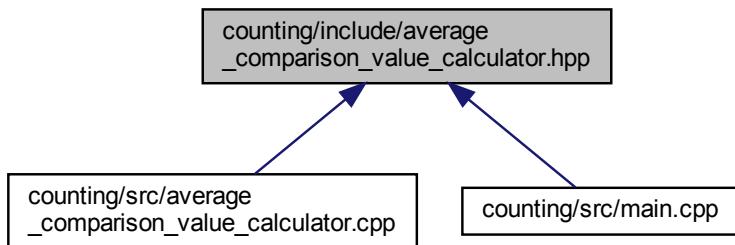
- std::vector< [cl::DataPoint](#) > [ctg::aboveThreshold](#) (const [cl::DataSet](#) &dataSet, long double accelerometerThreshold, long double gyroscopeThreshold)

## 7.60 counting/include/average\_comparison\_value\_calculator.hpp File Reference

```
#include "cl/channel.hpp"
#include "cl/data_set.hpp"
#include "cl/sensor.hpp"
Include dependency graph for average_comparison_value_calculator.hpp:
```



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



### Namespaces

- `ctg`

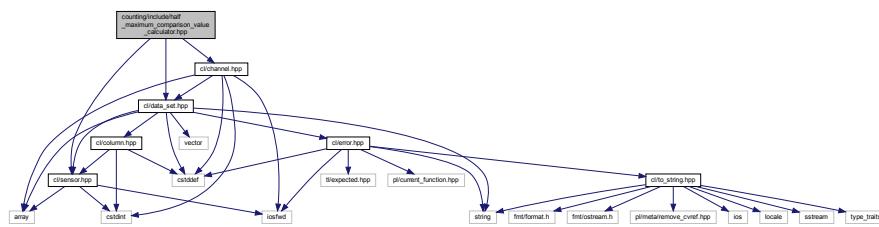
### Functions

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

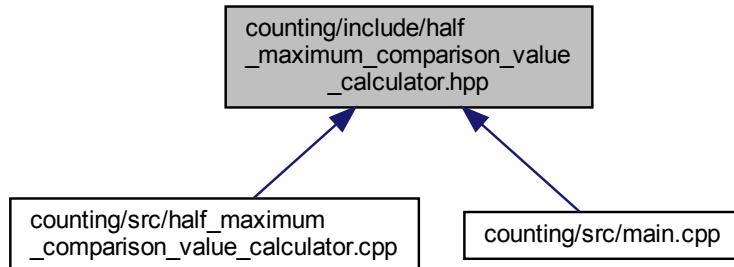
## 7.61 counting/include/half\_maximum\_comparison\_value\_calculator.hpp File Reference

```
#include "cl/channel.hpp"
#include "cl/data_set.hpp"
```

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



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



## Namespaces

- `ctg`

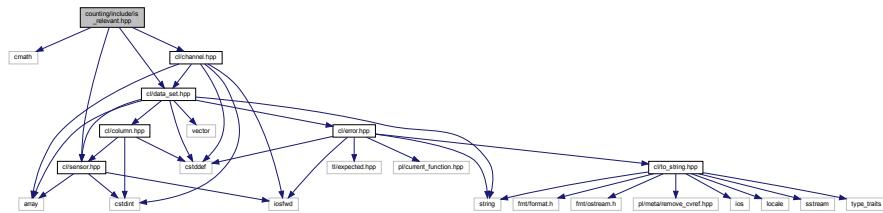
## Functions

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

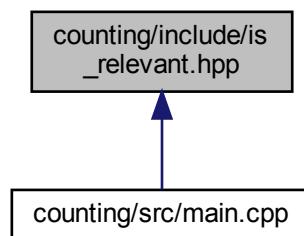
## 7.62 counting/include/is\_relevant.hpp File Reference

```
#include <cmath>
#include "cl/channel.hpp"
#include "cl/data_set.hpp"
```

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



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



# Namespaces

- ctg

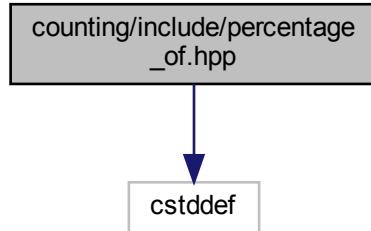
## Functions

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

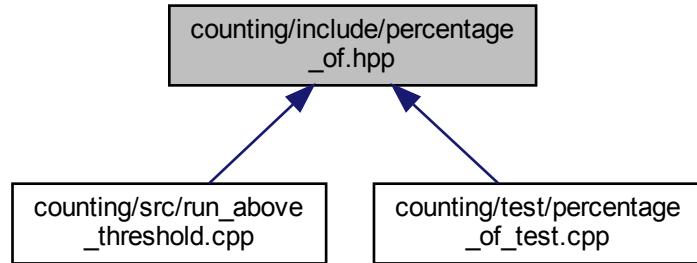
## 7.63 counting/include/percentage\_of.hpp File Reference

```
#include <cstddef>
```

Include dependency graph for percentage\_of.hpp:



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



## Namespaces

- [ctg](#)

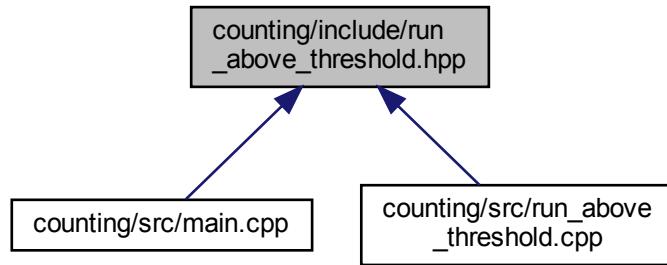
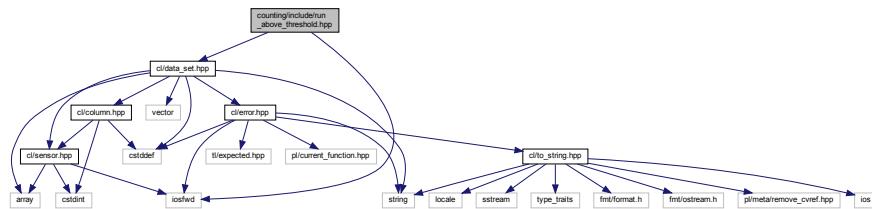
## Functions

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

## 7.64 counting/include/run\_above\_threshold.hpp File Reference

```
#include <iostream>
#include "cl/data_set.hpp"
```

Include dependency graph for run\_above\_threshold.hpp:



## Namespaces

- `ctg`

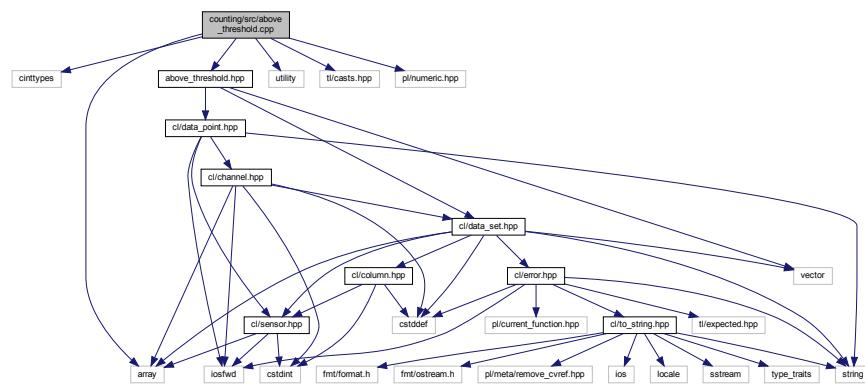
## Functions

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

## 7.65 counting/src/above\_threshold.cpp File Reference

```
#include <cinttypes>
#include <array>
#include <utility>
#include <tl/casts.hpp>
#include <pl/numeric.hpp>
```

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



## Namespaces

- `ctg`

## Macros

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

## Functions

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

### 7.65.1 Macro Definition Documentation

#### 7.65.1.1 CL\_CHANNEL\_X

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

### 7.65.2 Variable Documentation

### 7.65.2.1 channel

```
cl::Channel channel
```

Definition at line 18 of file above\_threshold.cpp.

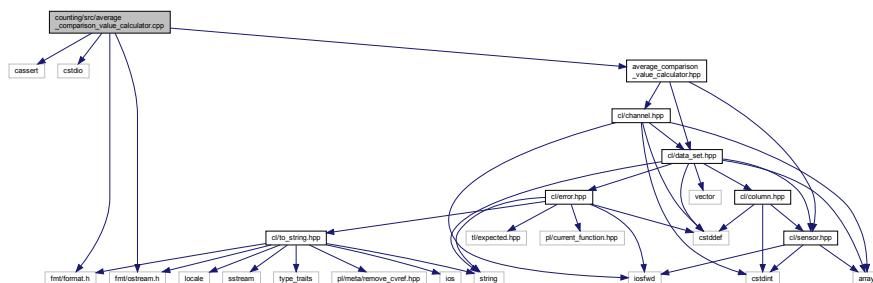
### 7.65.2.2 channelAccessor

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

Definition at line 17 of file above\_threshold.cpp.

## 7.66 counting/src/average\_comparison\_value\_calculator.cpp File Reference

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



## Namespaces

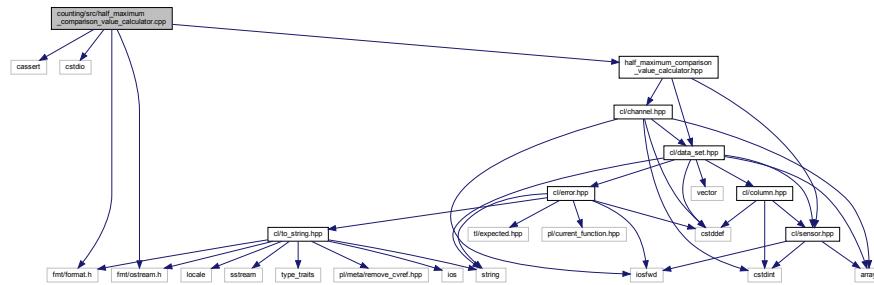
- `ctg`

## Functions

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

## 7.67 counting/src/half\_maximum\_comparison\_value\_calculator.cpp File Reference

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



## Namespaces

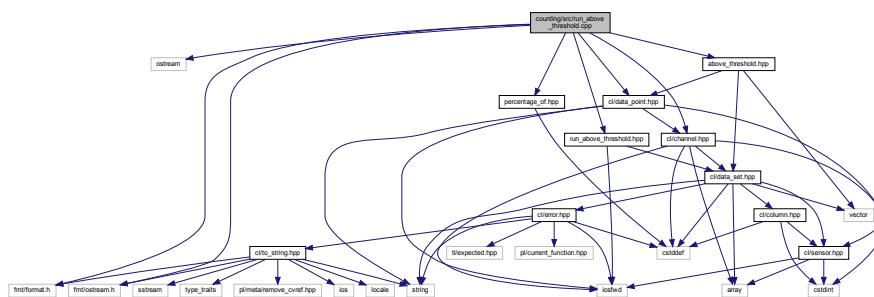
- `ctg`

## Functions

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

## 7.68 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

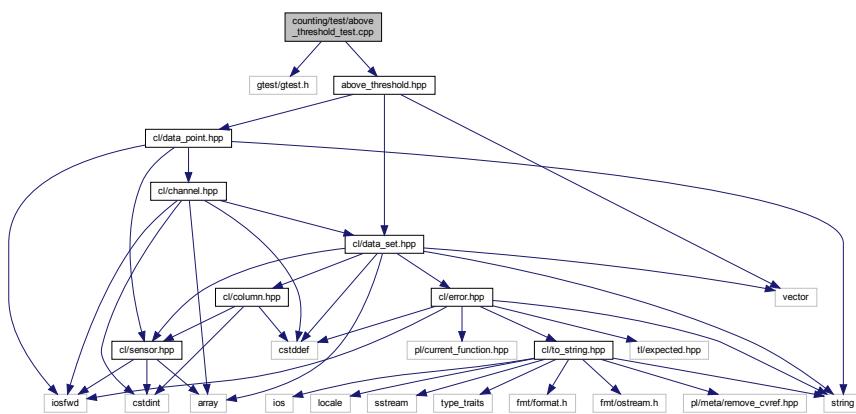
- `ctg`

## Functions

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

## 7.69 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`)

### 7.69.1 Macro Definition Documentation

### 7.69.1.1 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.

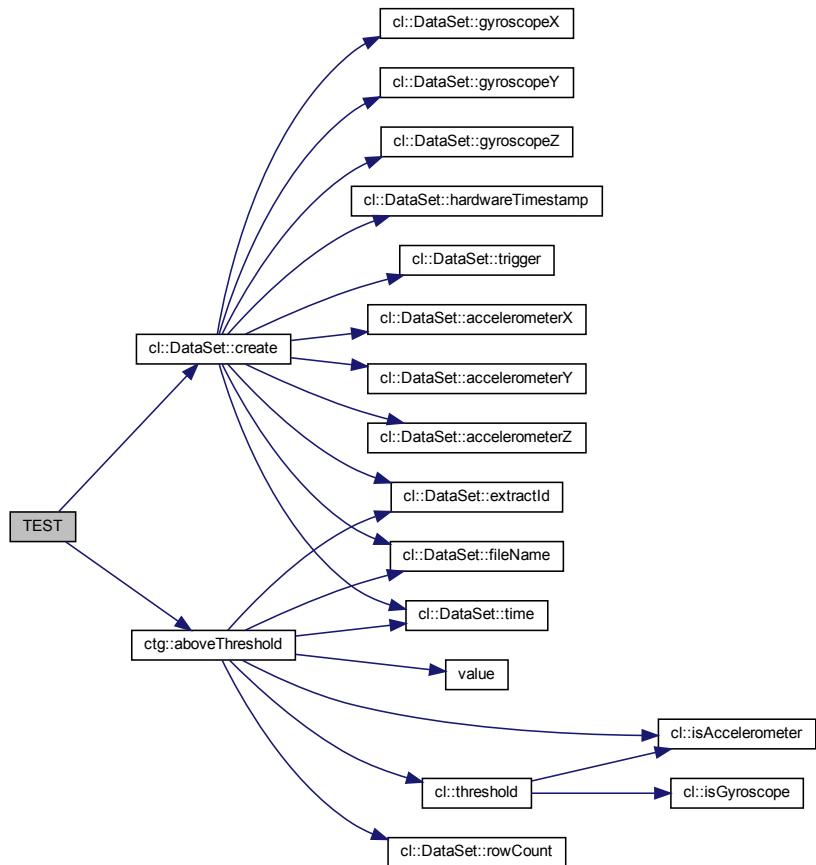
## 7.69.2 Function Documentation

### 7.69.2.1 TEST()

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

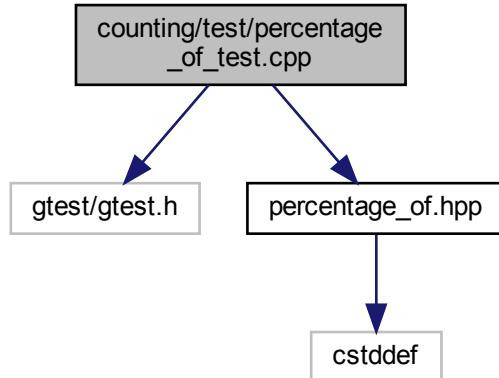
Definition at line 10 of file above\_threshold\_test.cpp.

Here is the call graph for this function:



## 7.70 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)`

#### 7.70.1 Macro Definition Documentation

##### 7.70.1.1 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.

#### 7.70.2 Function Documentation

### 7.70.2.1 TEST()

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

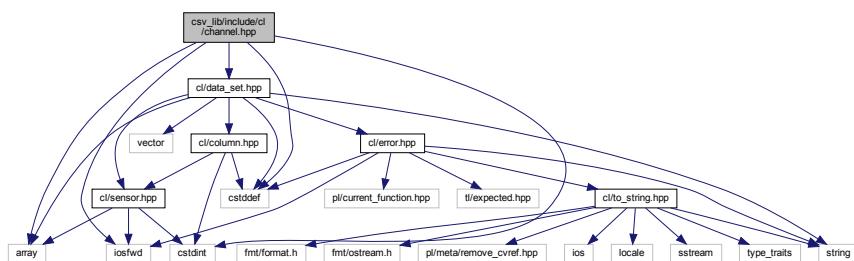
Definition at line 10 of file percentage\_of\_test.cpp.

Here is the call graph for this function:

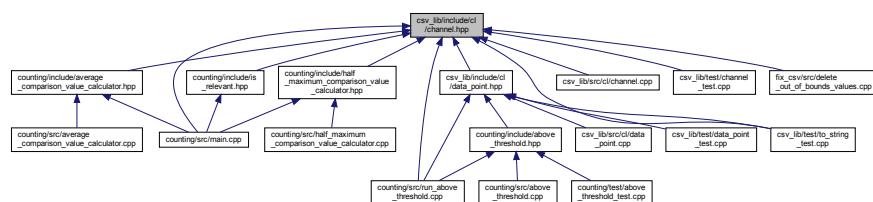


## 7.71 csv\_lib/include/cl/channel.hpp File Reference

```
#include <cstddef>
#include <cstdint>
#include <array>
#include <iostream>
#include "cl/data_set.hpp"
Include dependency graph for channel.hpp:
```



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



## Classes

- struct `cl::data_set_accessor< Chan >`

## Namespaces

- `cl`

## Macros

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

## Enumerations

- enum `cl::Channel : std::uint64_t { CL_CHANNEL, CL_CHANNEL }`

## Functions

- `DataSet::ChannelAccessor cl::dataSetAccessor (Channel channel)`
- `std::ostream & cl::operator<< (std::ostream &os, Channel channel)`
- `bool cl::isAccelerometer (Channel channel)`
- `bool cl::isGyroscope (Channel channel)`
- `long double cl::threshold (Channel channel)`

## Variables

- `constexpr std::size_t cl::channelCount`
- `constexpr std::array< Channel, channelCount > cl::channels`
- `template<Channel Chan>`  
`constexpr CL_CHANNEL DataSet::ChannelAccessor cl::data_set_accessor_v = data_set_accessor<Chan>::f`
- `constexpr long double cl::accelerometerThreshold {1.99L}`
- `constexpr long double cl::gyroscopeThreshold {1999.99L}`

### 7.71.1 Macro Definition Documentation

### 7.71.1.1 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.

### 7.71.1.2 CL\_CHANNEL\_X [1/4]

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

Definition at line 41 of file channel.hpp.

### 7.71.1.3 CL\_CHANNEL\_X [2/4]

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

Definition at line 41 of file channel.hpp.

### 7.71.1.4 CL\_CHANNEL\_X [3/4]

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

Definition at line 41 of file channel.hpp.

### 7.71.1.5 CL\_CHANNEL\_X [4/4]

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

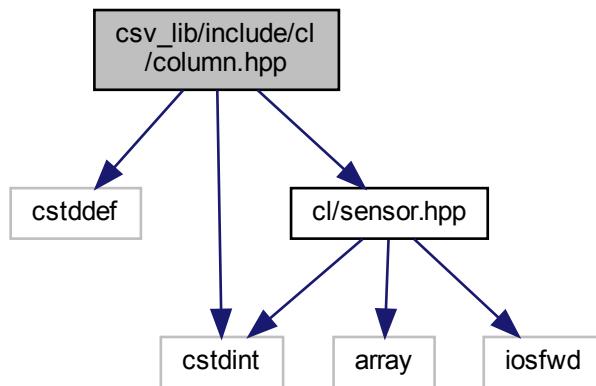
## Value:

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

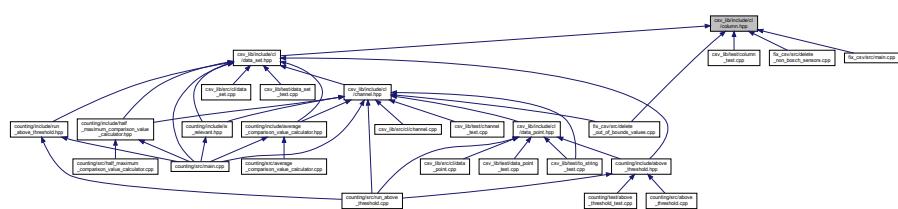
Definition at line 41 of file channel.hpp.

## 7.72 csv\_lib/include/cl/column.hpp File Reference

```
#include <cstdint>
#include "cl/sensor.hpp"
Include dependency graph for column.hpp:
```



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



## Classes

- struct `cl::col_traits< Col >`

## Namespaces

- `cl`

## Macros

- `#define CL_SPECIALIZE_COL_TRAITS(column, columnType)`

## Typedefs

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

## Enumerations

- enum `cl::Column` : `std::size_t` {  
`cl::Column::Time, cl::Column::HardwareTimestamp, cl::Column::ExtractId, cl::Column::Trigger,`  
`cl::Column::AccelerometerX, cl::Column::AccelerometerY, cl::Column::AccelerometerZ, cl::Column::GyroscopeX,`  
`cl::Column::GyroscopeY, cl::Column::GyroscopeZ, cl::Column::SamplingRate }`

## Functions

- `cl::CL_SPECIALIZE_COL_TRAITS (Column::Time, long double)`
- `cl::CL_SPECIALIZE_COL_TRAITS (Column::HardwareTimestamp, std::uint64_t)`
- `cl::CL_SPECIALIZE_COL_TRAITS (Column::ExtractId, Sensor)`
- `cl::CL_SPECIALIZE_COL_TRAITS (Column::Trigger, std::uint64_t)`
- `cl::CL_SPECIALIZE_COL_TRAITS (Column::AccelerometerX, long double)`
- `cl::CL_SPECIALIZE_COL_TRAITS (Column::AccelerometerY, long double)`
- `cl::CL_SPECIALIZE_COL_TRAITS (Column::AccelerometerZ, long double)`
- `cl::CL_SPECIALIZE_COL_TRAITS (Column::GyroscopeX, long double)`
- `cl::CL_SPECIALIZE_COL_TRAITS (Column::GyroscopeY, long double)`
- `cl::CL_SPECIALIZE_COL_TRAITS (Column::GyroscopeZ, long double)`
- `cl::CL_SPECIALIZE_COL_TRAITS (Column::SamplingRate, std::uint64_t)`

## Variables

- template<Column Col>  
constexpr `std::size_t cl::column_index` = `col_traits<Col>::index`

### 7.72.1 Macro Definition Documentation

### 7.72.1.1 CL\_SPECIALIZE\_COL\_TRAITS

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

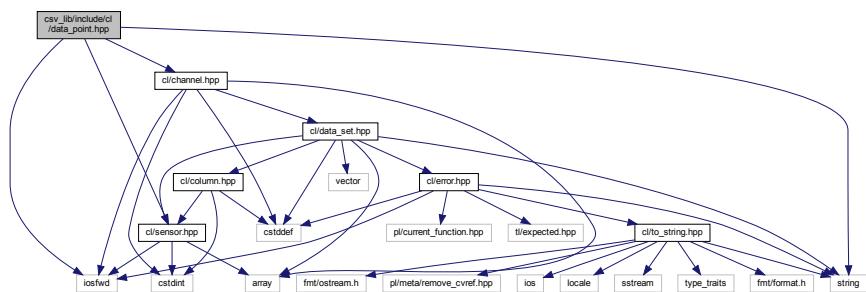
## Value:

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

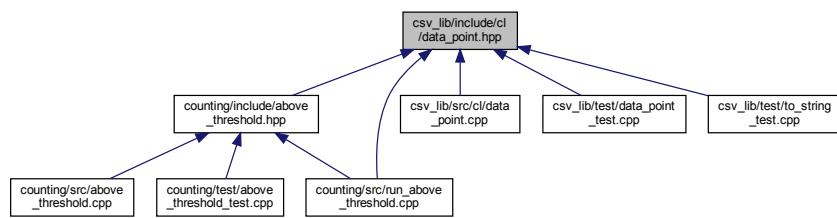
Definition at line 26 of file column.hpp.

## 7.73 csv\_lib/include/cl/data\_point.hpp File Reference

```
#include <iostream>
#include <string>
#include "cl/channel.hpp"
#include "cl/sensor.hpp"
Include dependency graph for data_point.hpp:
```



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



## Classes

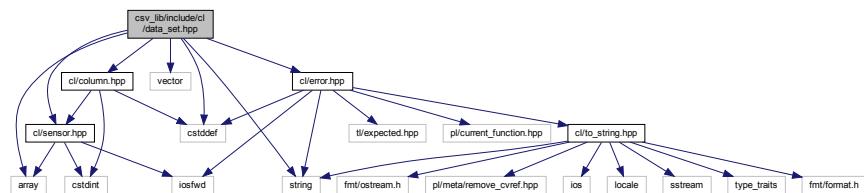
- class cl::DataPoint

# Namespaces

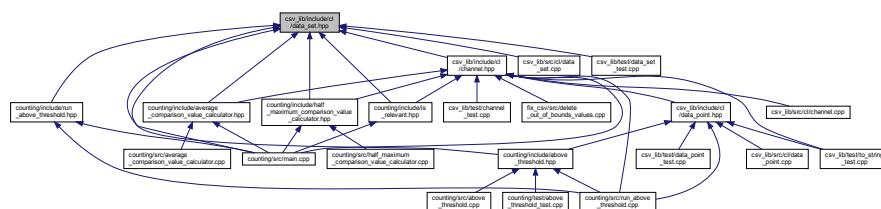
- cl

## 7.74 csv\_lib/include/cl/data\_set.hpp File Reference

```
#include <cstddef>
#include <array>
#include <string>
#include <vector>
#include "cl/column.hpp"
#include "cl/error.hpp"
#include "cl/sensor.hpp"
Include dependency graph for data_set.hpp
```



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



## Classes

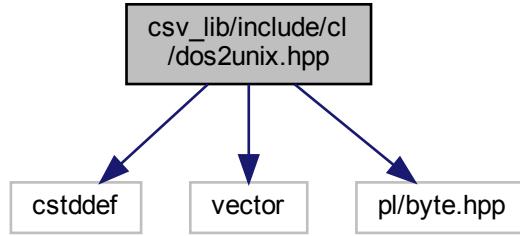
- class cl::DataSet

## Namespaces

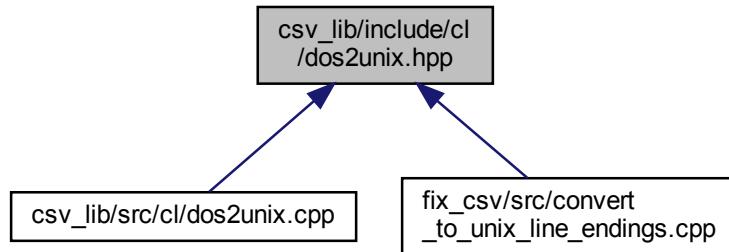
- C

## 7.75 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

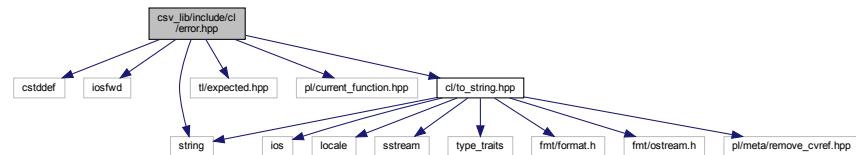
- `cl`

### Functions

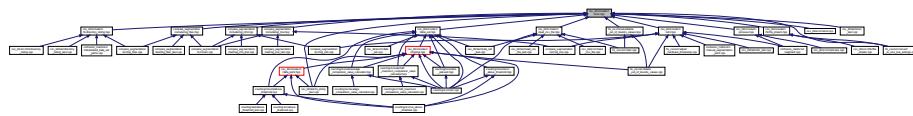
- `std::vector< pl::byte > cl::dos2unix (const void *p, std::size_t size)`  
*Converts DOS / Microsoft Windows line endings to UNIX line endings.*

## 7.76 csv\_lib/include/cl/error.hpp File Reference

```
#include <cstddef>
#include <iostream>
#include <string>
#include <tl/expected.hpp>
#include <pl/current_function.hpp>
#include "cl/to_string.hpp"
Include dependency graph for error.hpp:
```



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



### Classes

- class `cl::Error`

### Namespaces

- `cl`

### Macros

- `#define CL_ERROR_KIND`
- `#define CL_ERROR_KIND_X(kind) kind,`
- `#define CL_UNEXPECTED(kind, message)`

### Typedefs

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

#### 7.76.1 Macro Definition Documentation

### 7.76.1.1 CL\_ERROR\_KIND

```
#define CL_ERROR_KIND
```

**Value:**

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

Definition at line 14 of file error.hpp.

### 7.76.1.2 CL\_ERROR\_KIND\_X

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

Definition at line 27 of file error.hpp.

### 7.76.1.3 CL\_UNEXPECTED

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

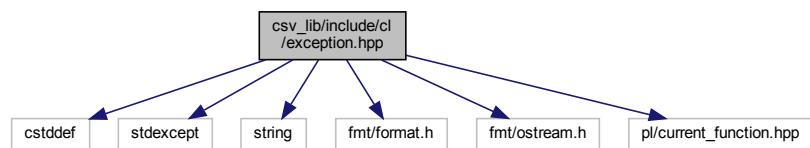
**Value:**

```
::tl::make_unexpected(
    ::cl::Error{kind, __FILE__, PL_CURRENT_FUNCTION, __LINE__, message})
```

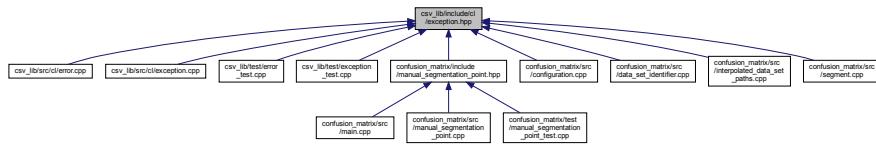
Definition at line 67 of file error.hpp.

## 7.77 csv\_lib/include/cl/exception.hpp File Reference

```
#include <cstddef>
#include <stdexcept>
#include <string>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <pl/current_function.hpp>
Include dependency graph for exception.hpp:
```



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



## Classes

- class `cl::Exception`

## Namespaces

- cl

## Macros

- `#define CL_THROW(what_arg) throw ::cl::Exception { __FILE__, PL_CURRENT_FUNCTION, __LINE__, what_arg }`
  - `#define CL_THROW_FMT(fmt_str, ...) CL_THROW(::fmt::format(fmt_str, __VA_ARGS__))`

### **7.77.1 Macro Definition Documentation**

### 7.77.1.1 CL THROW

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

Definition at line 42 of file exception.hpp.

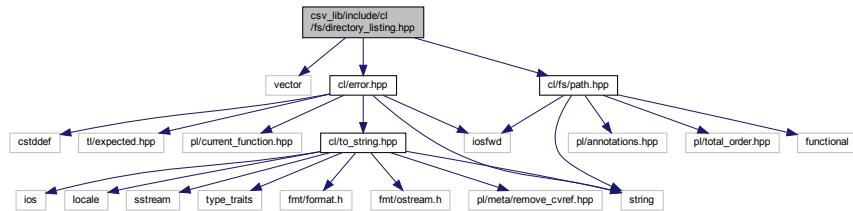
### **7.77.1.2 CL THROW FMT**

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

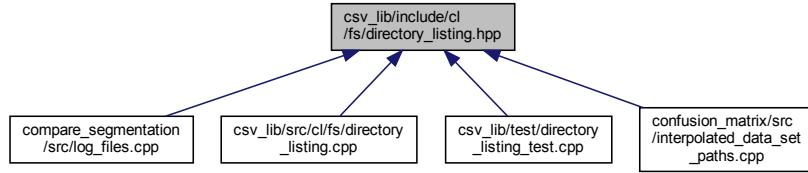
Definition at line 45 of file exception.hpp.

## 7.78 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` }

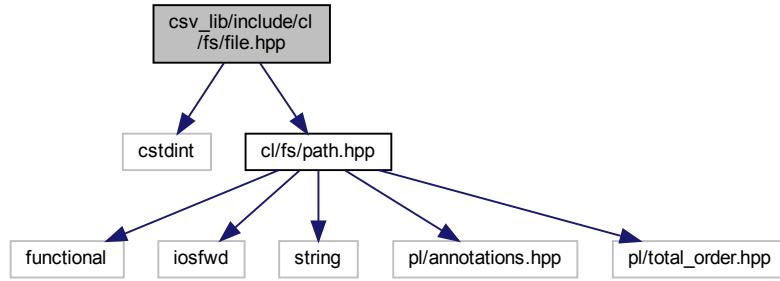
*Options for directoryListing.*

## Functions

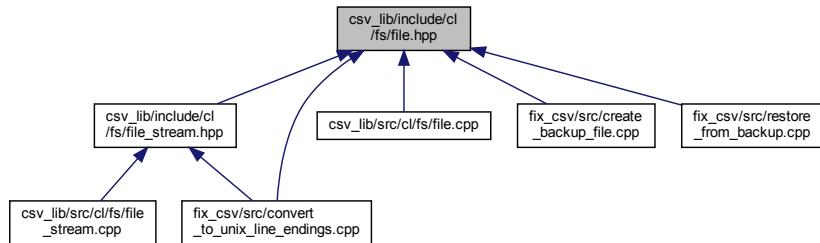
- `Expected< std::vector< Path > > cl::fs::directoryListing (const Path &directoryPath, DirectoryListingOption directoryListingOption=DirectoryListingOption::ExcludeDotAndDotDot)`
- Creates a listing of the contents of a directory.*

## 7.79 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](#)

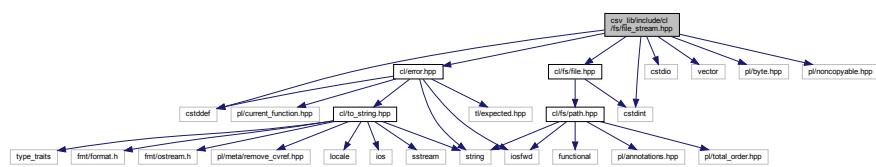
*Represents a file.*

## Namespaces

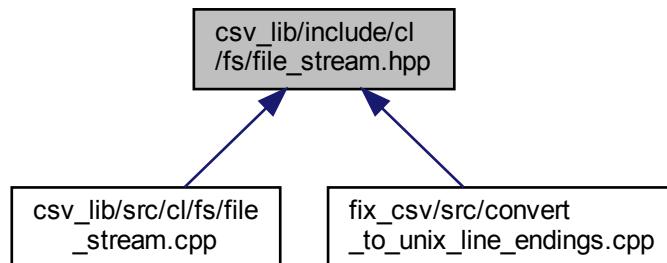
- [cl](#)
- [cl::fs](#)

## 7.80 csv\_lib/include/cl/fs/file\_stream.hpp File Reference

```
#include <cstddef>
#include <cstdint>
#include <cstdio>
#include <vector>
#include <pl/byte.hpp>
#include <pl/noncopyable.hpp>
#include "cl/error.hpp"
#include "cl/fs/file.hpp"
Include dependency graph for file_stream.hpp:
```



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



### Classes

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

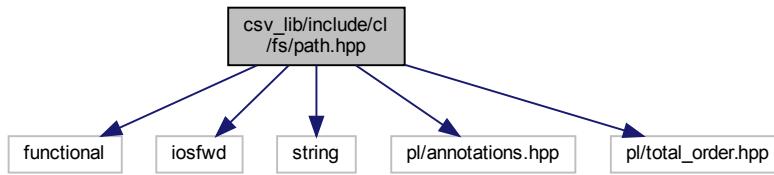
*A binary file stream.*

### Namespaces

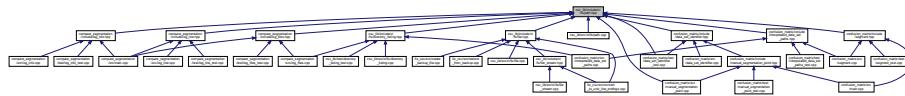
- [cl](#)
- [cl::fs](#)

## 7.81 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 >](#)

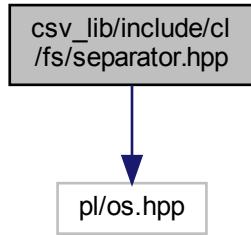
### Namespaces

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

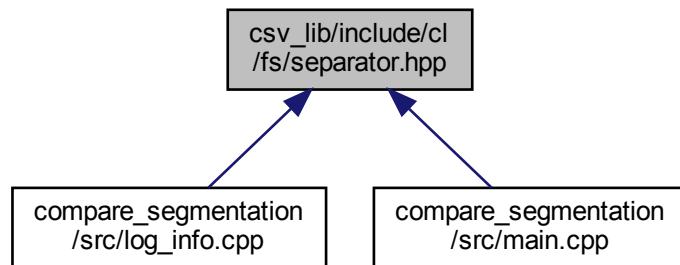
## 7.82 csv\_lib/include/cl/fs/sePARATOR.hpp File Reference

```
#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.*

### 7.82.1 Macro Definition Documentation

#### 7.82.1.1 CL\_FS\_SEPARATOR

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

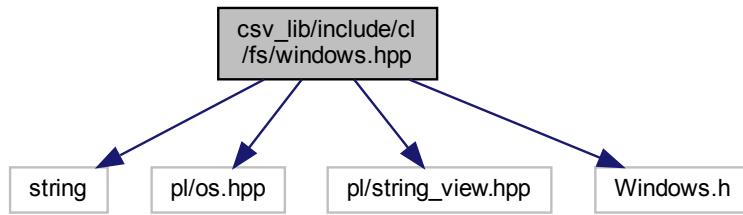
The filesystem separator of the operating system.

Definition at line 11 of file separator.hpp.

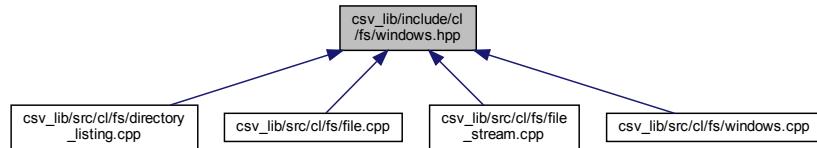
## 7.83 csv\_lib/include/cl/fs/windows.hpp File Reference

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

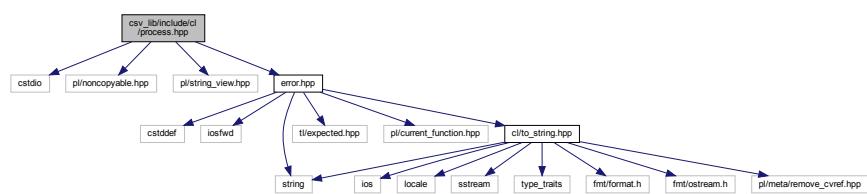
- `std::wstring cl::fs::utf8ToUtf16 (pl::string_view utf8)`  
*Converts a UTF-8 encoded string to a UTF-16 encoded wstring.*
- `std::string cl::fs::utf16ToUtf8 (pl::wstring_view utf16)`  
*Converts a UTF-16 encoded wide character string to UTF-8 string.*
- `std::wstring cl::fs::formatError (DWORD errorCode)`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

### 7.83.1 Detailed Description

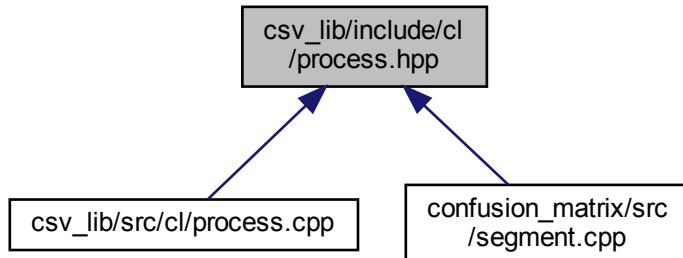
Contains Microsoft Windows specific functions.

## 7.84 csv\_lib/include/cl/process.hpp File Reference

```
#include <cstdio>
#include <pl/noncopyable.hpp>
#include <pl/string_view.hpp>
#include "error.hpp"
Include dependency graph for process.hpp:
```



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



## Classes

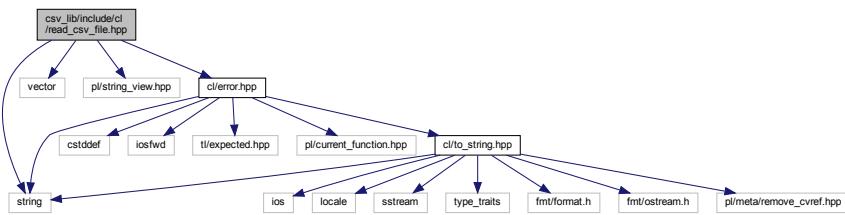
- class [cl::Process](#)

## Namespaces

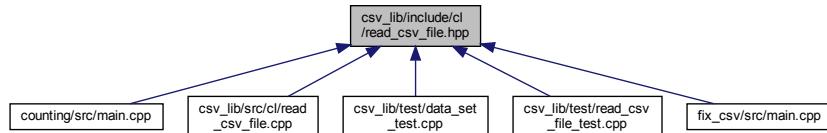
- [cl](#)

## 7.85 csv\_lib/include/cl/read\_csv\_file.hpp File Reference

```
#include <string>
#include <vector>
#include <pl/string_view.hpp>
#include "cl/error.hpp"
Include dependency graph for read_csv_file.hpp:
```



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



## Namespaces

- `cl`

## Enumerations

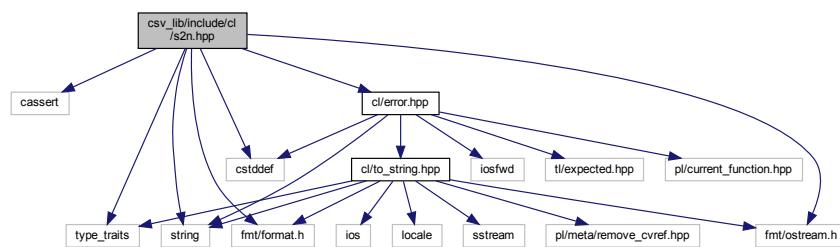
- enum `cl::CsvFileKind` { `cl::CsvFileKind::Raw`, `cl::CsvFileKind::Fixed` }

## Functions

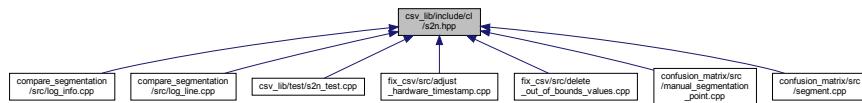
- `Expected< std::vector< std::vector< std::string > >> cl::readCsvFile (ppl::string_view csvFilePath, std::vector< std::string > *columnNames=nullptr, CsvFileKind csvFileKind=CsvFileKind::Fixed) noexcept`

## 7.86 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`

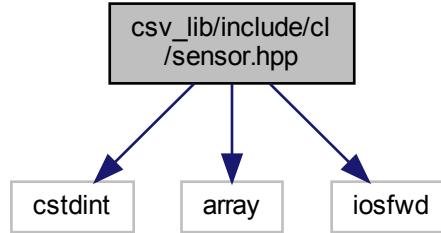
## Functions

- template<typename Integer >  
Expected< Integer > `cl::s2n` (const std::string &str, std::size\_t \*pos=nullptr, [[maybe\_unused]] int base=10)

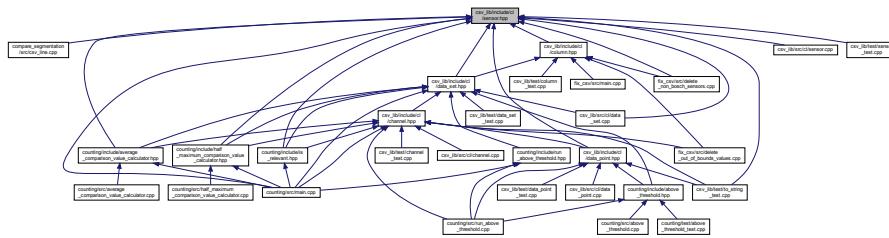
## 7.87 csv\_lib/include/cl/sensor.hpp File Reference

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

```
#include <iostfwd>
Include dependency graph for sensor.hpp:
```



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



## Namespaces

- `cl`

## Macros

- `#define CL_SENSOR`
- `#define CL_SENSOR_X(enum, value) enum = value,`
- `#define CL_SENSOR_X(enm, v) ::cl::Sensor::enm,`

## Enumerations

- enum `cl::Sensor` : std::uint64\_t { `cl::Sensor::CL_SENSOR_X, cl::Sensor::CL_SENSOR` }

## Functions

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

## Variables

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

### 7.87.1 Macro Definition Documentation

#### 7.87.1.1 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.

#### 7.87.1.2 CL\_SENSOR\_X [1/2]

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

Definition at line 16 of file sensor.hpp.

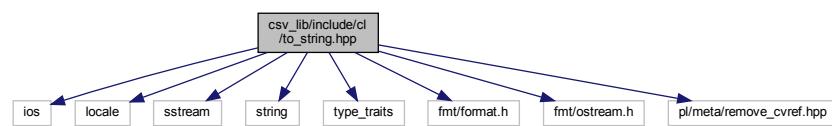
#### 7.87.1.3 CL\_SENSOR\_X [2/2]

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

Definition at line 16 of file sensor.hpp.

## 7.88 csv\_lib/include/cl/to\_string.hpp File Reference

```
#include <iostream>
#include <locale>
#include <iostream>
#include <string>
#include <type_traits>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <ppl/meta/remove_cvref.hpp>
Include dependency graph for to_string.hpp:
```



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



### Namespaces

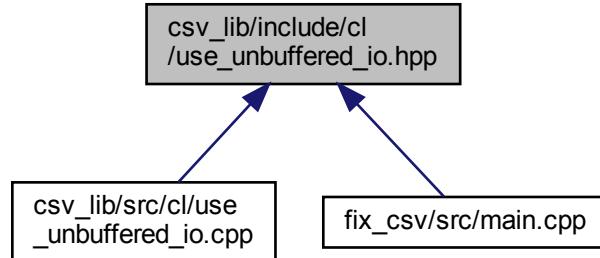
- [cl](#)

### Functions

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

## 7.89 csv\_lib/include/cl/use\_unbuffered\_io.hpp File Reference

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



### Namespaces

- `cl`

### Functions

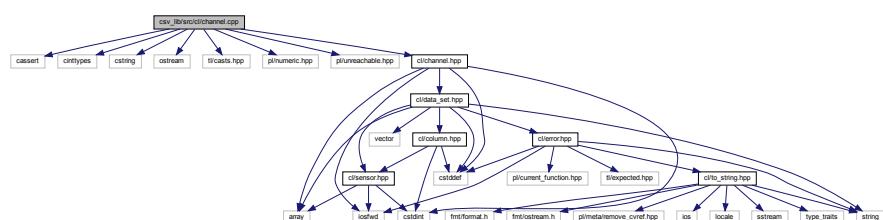
- void `cl::useUnbufferedIo ()`

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

## Functions

- `DataSet::ChannelAccessor cl::dataSetAccessor (Channel channel)`
- `std::ostream & cl::operator<< (std::ostream &os, Channel channel)`
- `bool cl::isAccelerometer (Channel channel)`
- `bool cl::isGyroscope (Channel channel)`
- `long double cl::threshold (Channel channel)`

### 7.90.1 Macro Definition Documentation

#### 7.90.1.1 CL\_CHANNEL\_X [1/2]

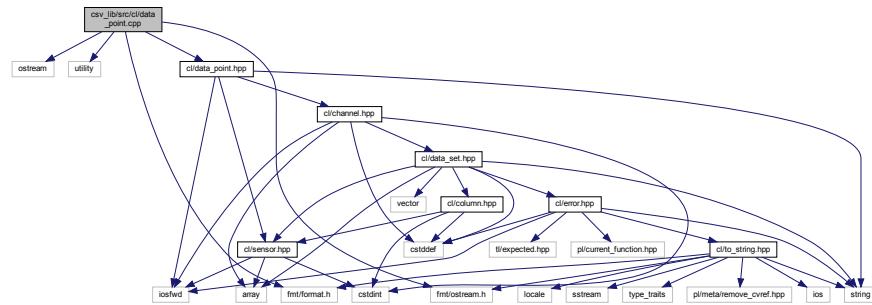
```
#define CL_CHANNEL_X(
    enm,
    v,
    acc ) case Channel::enm: return data_set_accessor_v<Channel::enm>;
```

#### 7.90.1.2 CL\_CHANNEL\_X [2/2]

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

## 7.91 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 ()`

### 7.91.1 Function Documentation

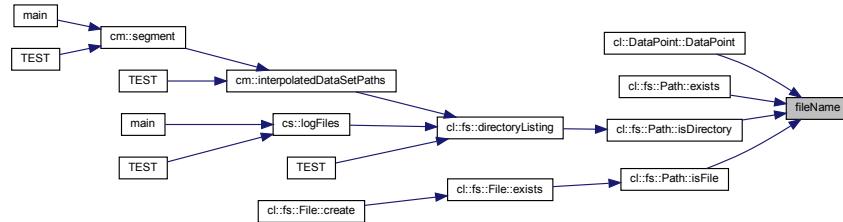
#### 7.91.1.1 channel()

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

### 7.91.1.2 fileName()

```
dataPoint fileName ( )
```

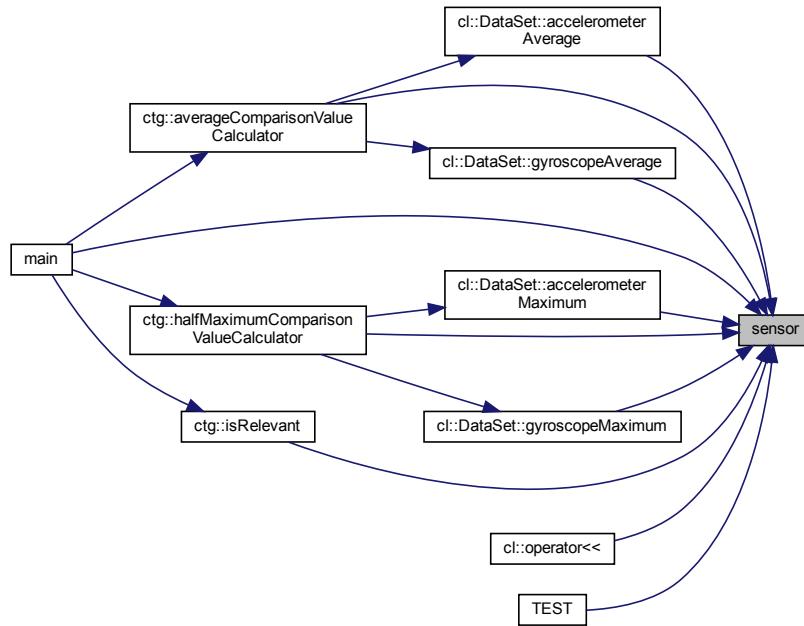
Here is the caller graph for this function:



### 7.91.1.3 sensor()

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

Here is the caller graph for this function:



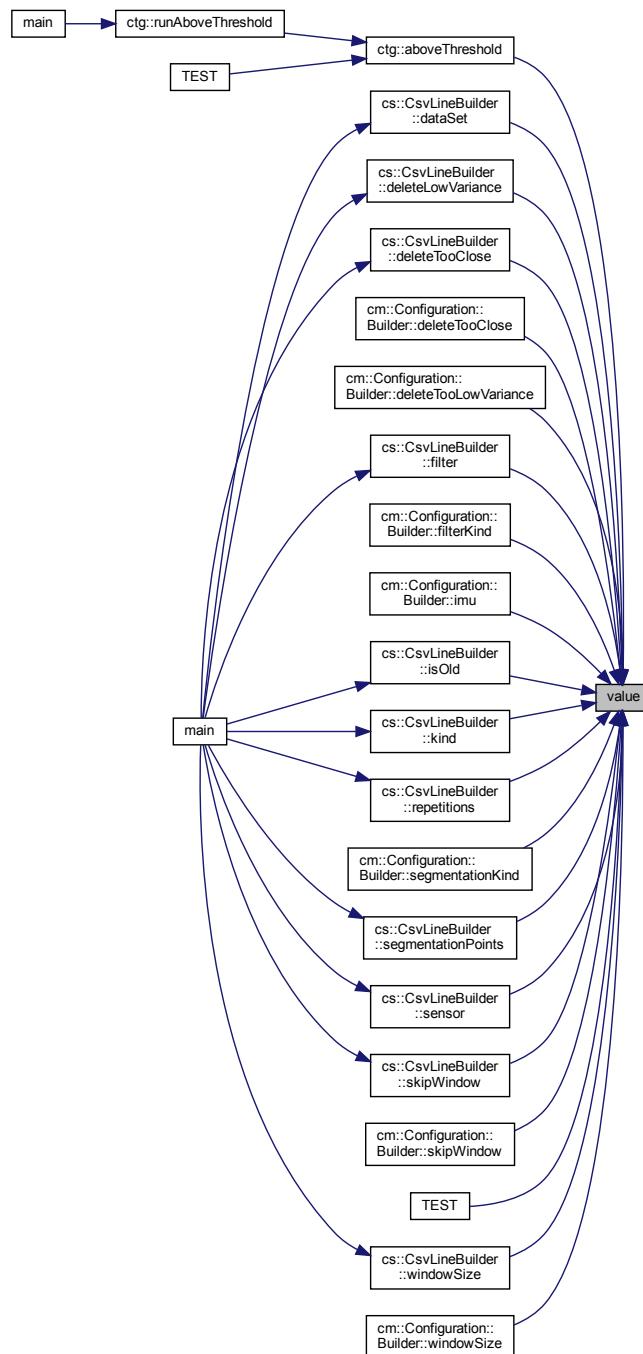
#### 7.91.1.4 time()

```
dataPoint dataPoint time ( )
```

#### 7.91.1.5 value()

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

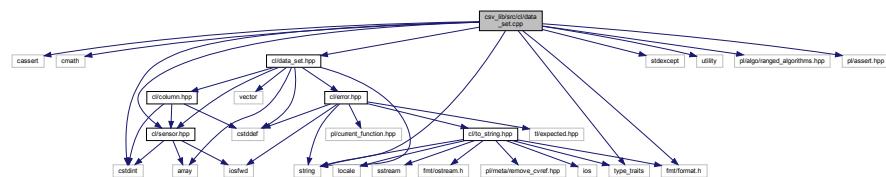
Here is the caller graph for this function:



## 7.92 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:
```



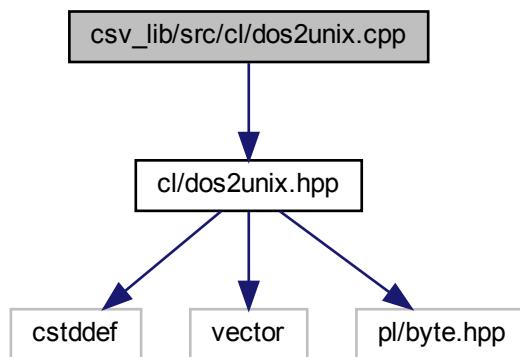
## Namespaces

- `cl`

## 7.93 csv\_lib/src/cl/dos2unix.cpp File Reference

```
#include "cl/dos2unix.hpp"

Include dependency graph for dos2unix.cpp:
```



## Namespaces

- `cl`

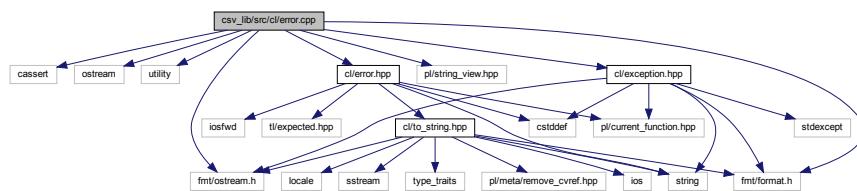
## Functions

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

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

## 7.94 csv\_lib/src/cl/error.cpp File Reference

```
#include <cassert>
#include <iostream>
#include <utility>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <pl/string_view.hpp>
#include "cl/error.hpp"
#include "cl/exception.hpp"
Include dependency graph for error.cpp:
```



## Namespaces

- `cl`

## Macros

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

## Functions

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

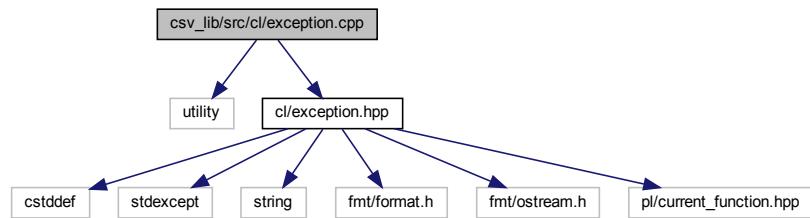
### 7.94.1 Macro Definition Documentation

### 7.94.1.1 CL\_ERROR\_KIND\_X

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

## 7.95 csv\_lib/src/cl/exception.cpp File Reference

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

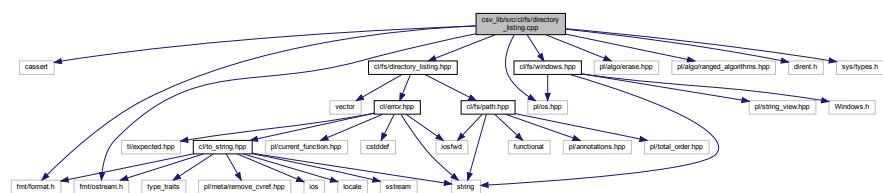


## Namespaces

- `cl`

## 7.96 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`

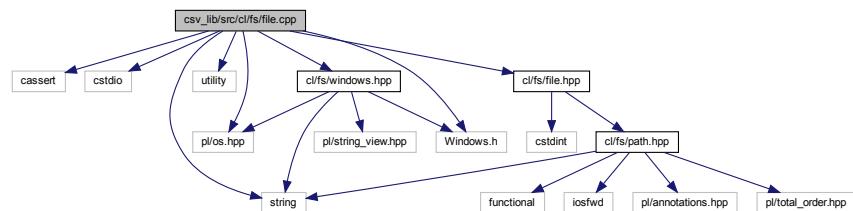
## Functions

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

*Creates a listing of the contents of a directory.*

## 7.97 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:
```



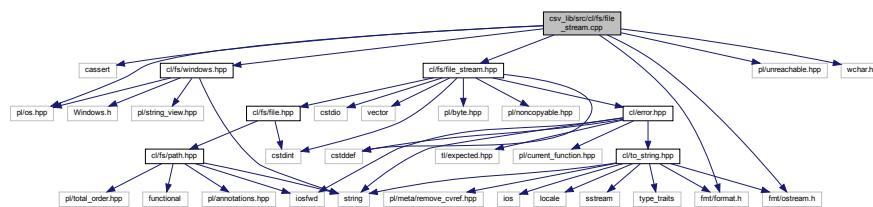
## Namespaces

- `cl`
- `cl::fs`

## 7.98 csv\_lib/src/cl/fs/file\_stream.cpp File Reference

```
#include <cassert>
#include <pl/os.hpp>
#include <pl/unreachable.hpp>
#include "cl/fs/windows.hpp"
#include <wchar.h>
#include <fmt/format.hpp>
#include <fmt/ostream.hpp>
```

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

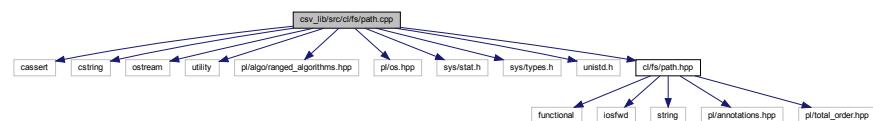


## Namespaces

- `cl`
- `cl::fs`

## 7.99 csv\_lib/src/cl/fs/path.cpp File Reference

```
#include <cassert>
#include <cstring>
#include <ostream>
#include <utility>
#include <p1/algo/ranged_algorithms.hpp>
#include <p1/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

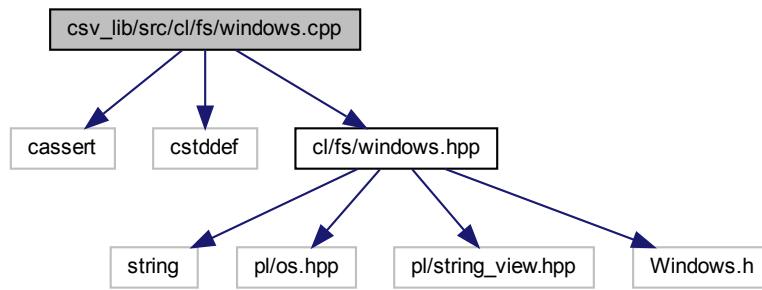
- `cl`
- `cl::fs`

## 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`

## 7.100 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`

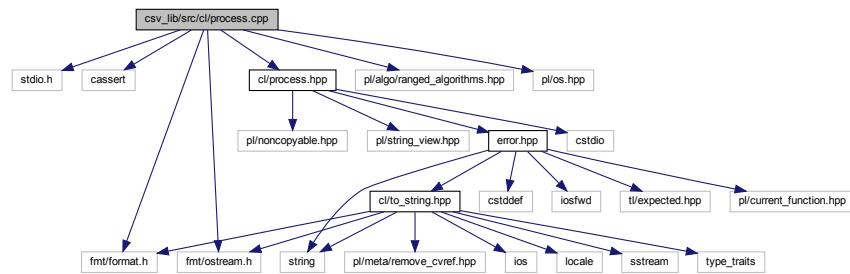
### Functions

- `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.*

## 7.101 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:
```

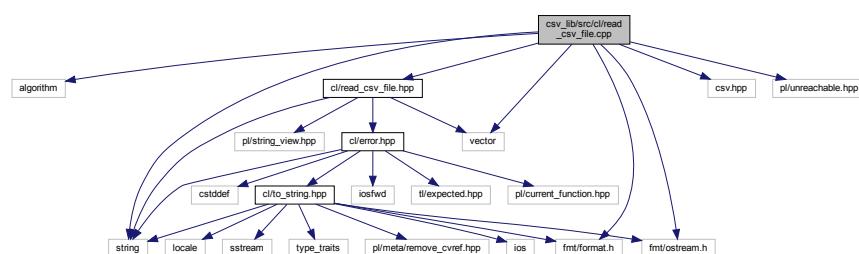


## Namespaces

- `cl`

## 7.102 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 <p/unreachable.hpp>
#include "cl/read_csv_file.hpp"
Include dependency graph for read_csv_file.cpp:
```



## Namespaces

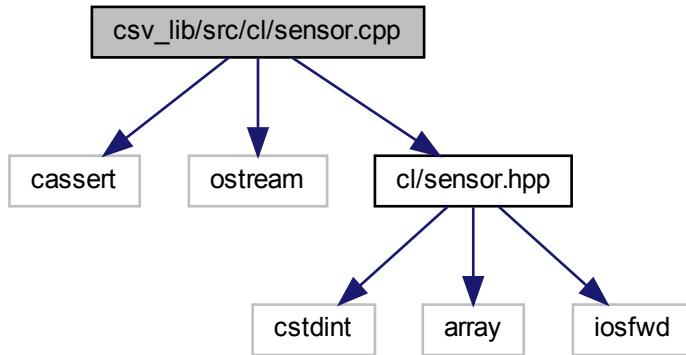
- `cl`

## Functions

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

## 7.103 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(enumerator, value) case Sensor::enumerator: return os << #enumerator;`

### Functions

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

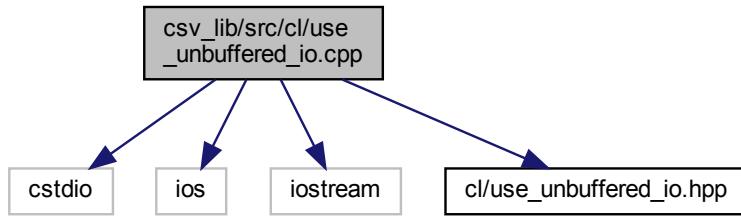
#### 7.103.1 Macro Definition Documentation

##### 7.103.1.1 CL\_SENSOR\_X

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

## 7.104 csv\_lib/src/cl/use\_unbuffered\_io.cpp File Reference

```
#include <cstdio>
#include <iostream>
#include <iostream>
#include "cl/use_unbuffered_io.hpp"
Include dependency graph for use_unbuffered_io.cpp:
```



## Namespaces

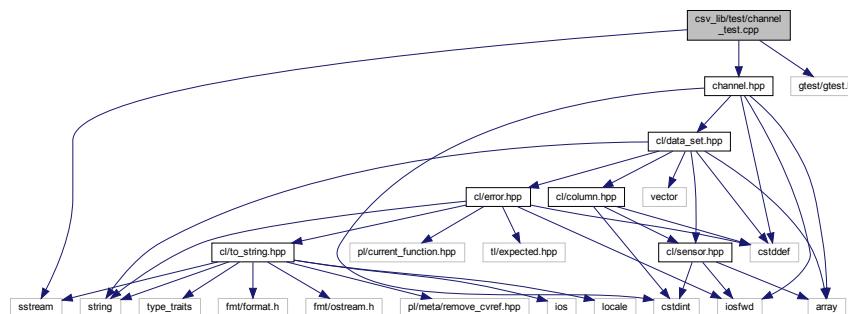
- `cl`

## Functions

- `void cl::useUnbufferedIo ()`

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

### 7.105.1 Function Documentation

#### 7.105.1.1 TEST() [1/4]

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

Definition at line 7 of file channel\_test.cpp.

#### 7.105.1.2 TEST() [2/4]

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

Definition at line 9 of file channel\_test.cpp.

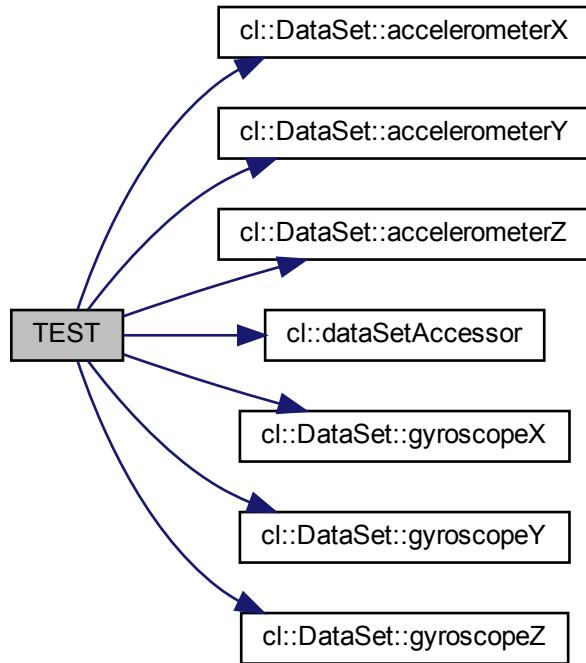
#### 7.105.1.3 TEST() [3/4]

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

Definition at line 35 of file channel\_test.cpp.

---

Here is the call graph for this function:



#### 7.105.1.4 TEST() [4/4]

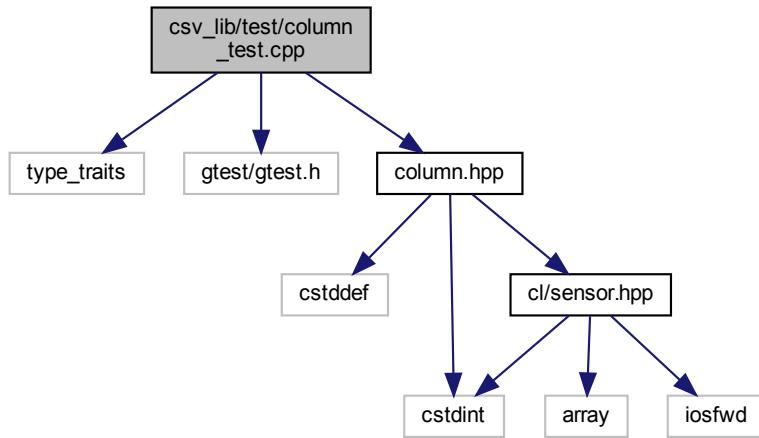
```
TEST (
    channel ,
    shouldPrintCorrectly )
```

Definition at line 19 of file channel\_test.cpp.

## 7.106 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`)

### 7.106.1 Function Documentation

#### 7.106.1.1 TEST() [1/2]

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

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

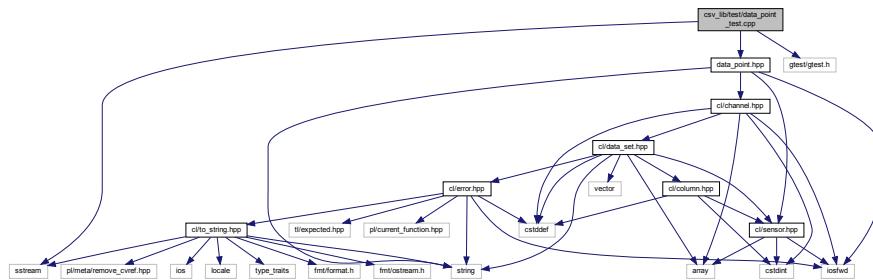
#### 7.106.1.2 TEST() [2/2]

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

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

## 7.107 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

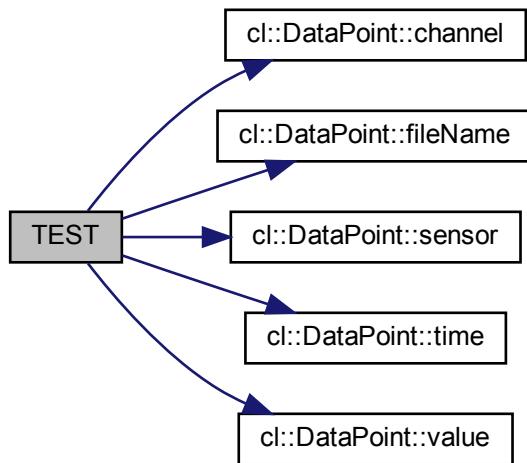
### 7.107.1 Function Documentation

#### 7.107.1.1 TEST() [1/2]

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

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

Here is the call graph for this function:



### 7.107.1.2 TEST() [2/2]

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

Definition at line 14 of file data\_point\_test.cpp.

## 7.107.2 Variable Documentation

### 7.107.2.1 dp

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

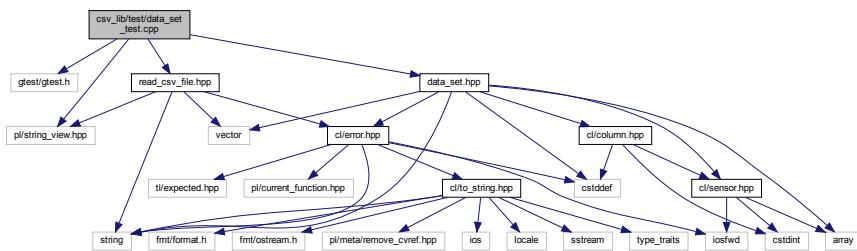
#### Initial value:

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

Definition at line 7 of file data\_point\_test.cpp.

## 7.108 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)`

#### 7.108.1 Macro Definition Documentation

##### 7.108.1.1 EXPECT\_LONG\_DOUBLE\_EQ

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

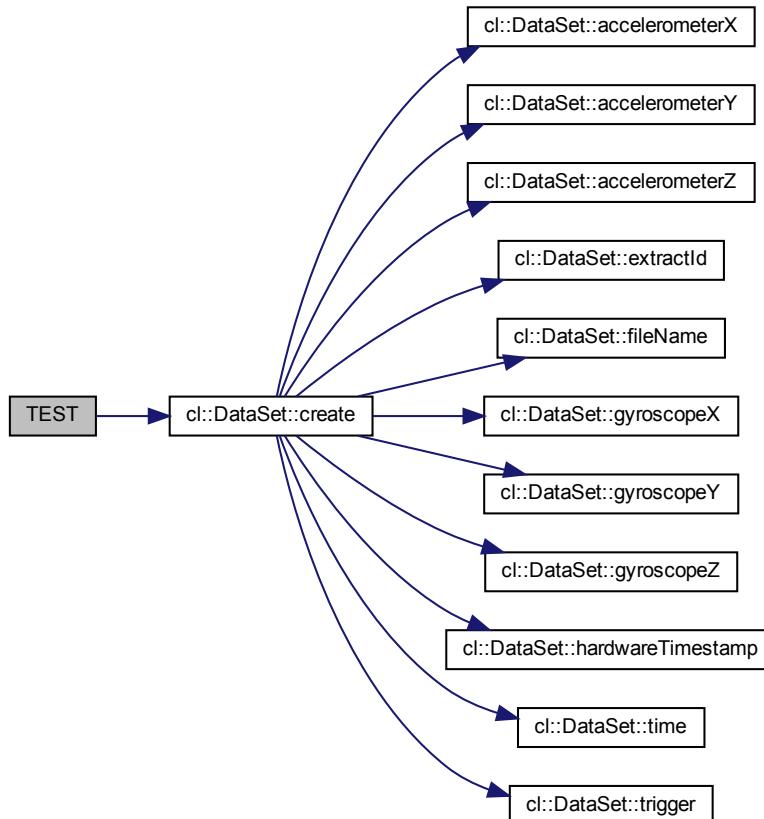
#### 7.108.2 Function Documentation

### 7.108.2.1 TEST() [1/4]

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

Definition at line 17 of file data\_set\_test.cpp.

Here is the call graph for this function:

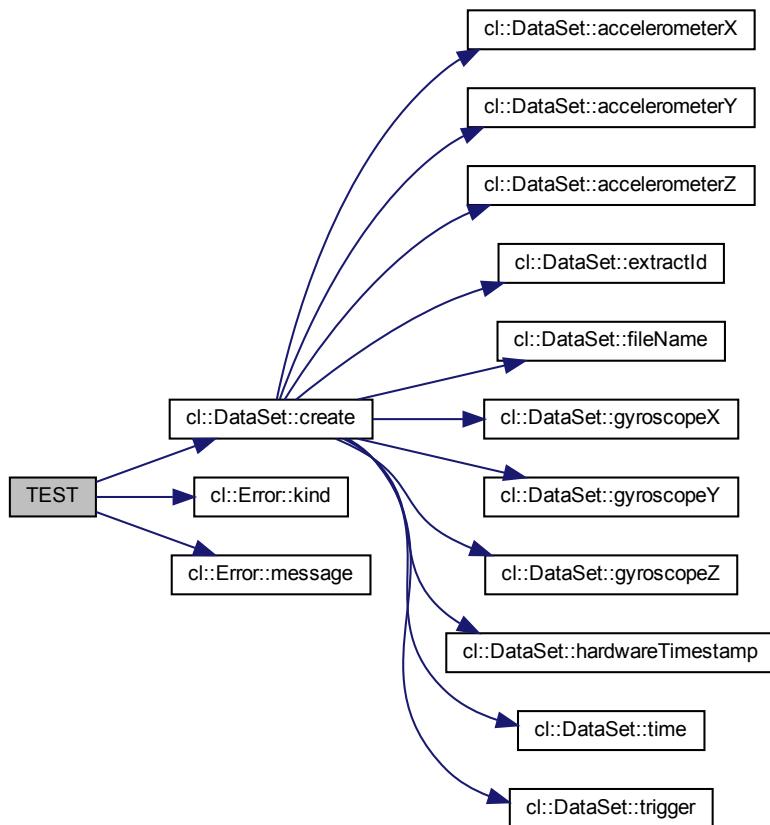


### 7.108.2.2 TEST() [2/4]

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

Definition at line 68 of file data\_set\_test.cpp.

Here is the call graph for this function:

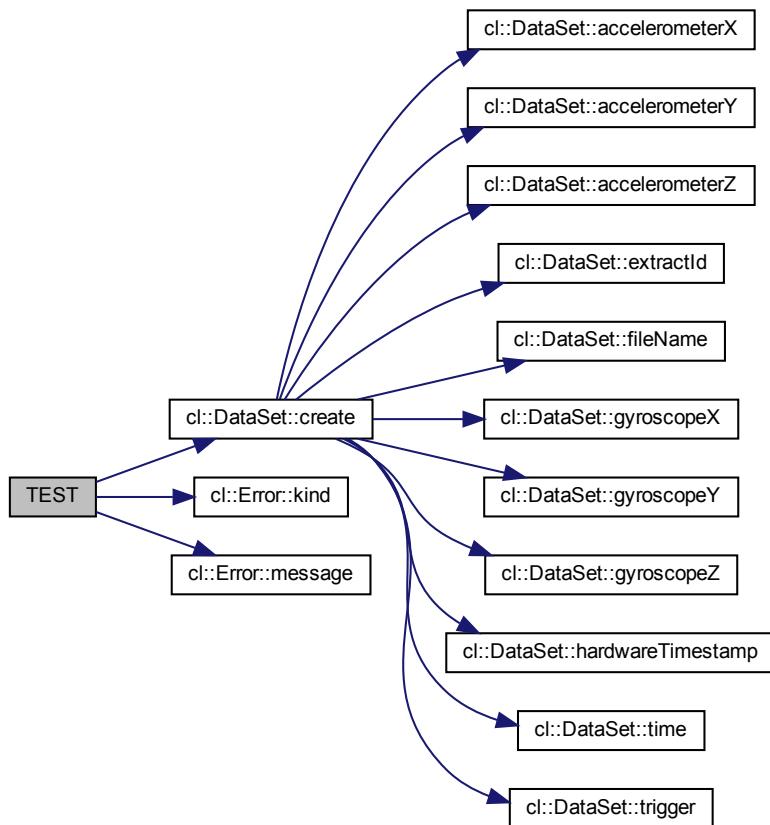


### 7.108.2.3 TEST() [3/4]

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

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

Here is the call graph for this function:

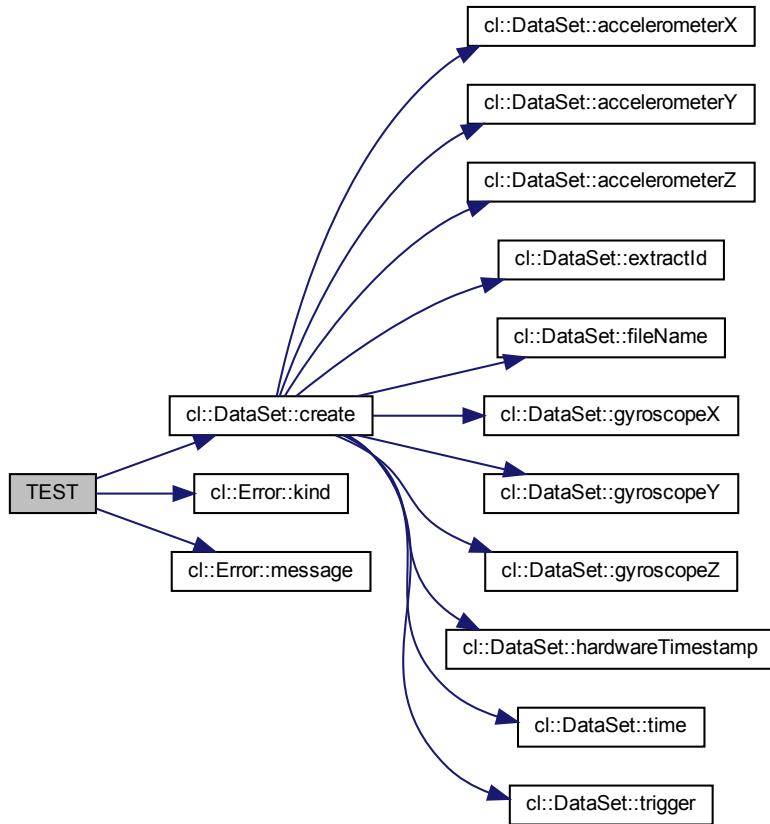


#### 7.108.2.4 TEST() [4/4]

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

Definition at line 80 of file data\_set\_test.cpp.

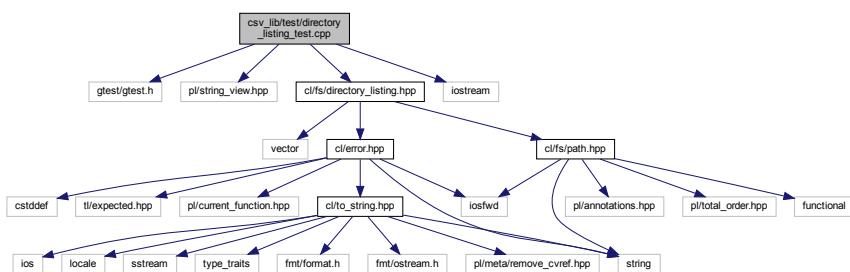
Here is the call graph for this function:



## 7.109 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`)

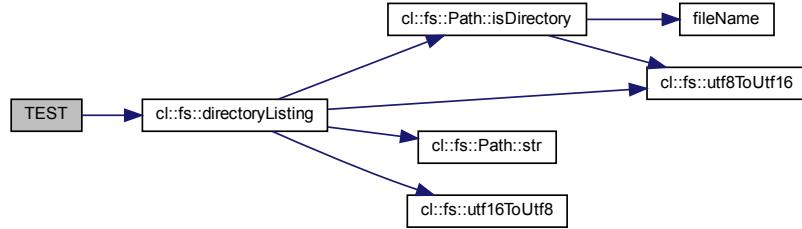
### 7.109.1 Function Documentation

#### 7.109.1.1 TEST() [1/3]

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

Definition at line 13 of file `directory_listing_test.cpp`.

Here is the call graph for this function:

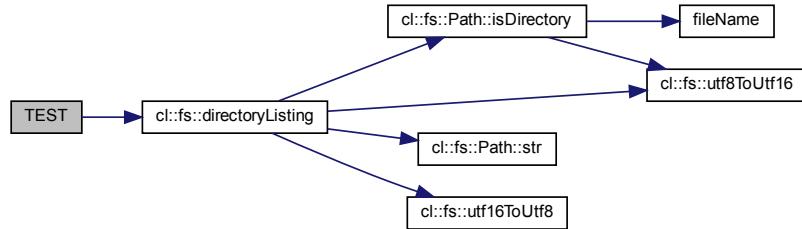


#### 7.109.1.2 TEST() [2/3]

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

Definition at line 28 of file `directory_listing_test.cpp`.

Here is the call graph for this function:

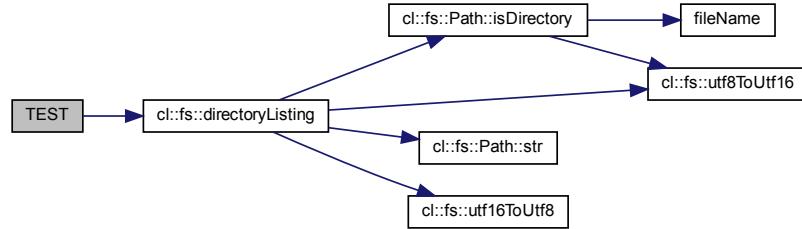


### 7.109.1.3 TEST() [3/3]

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

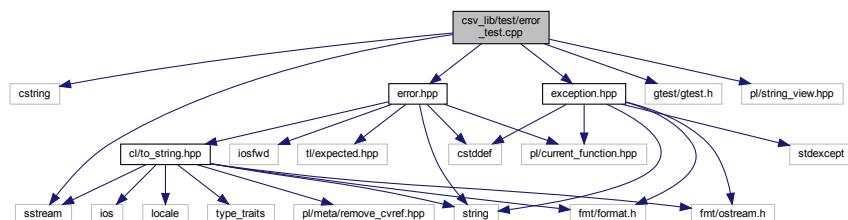
Definition at line 46 of file directory\_listing\_test.cpp.

Here is the call graph for this function:



## 7.110 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`

### 7.110.1 Function Documentation

#### 7.110.1.1 TEST() [1/4]

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

Definition at line 59 of file error\_test.cpp.

#### 7.110.1.2 TEST() [2/4]

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

Definition at line 19 of file error\_test.cpp.

#### 7.110.1.3 TEST() [3/4]

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

Definition at line 29 of file error\_test.cpp.

#### 7.110.1.4 TEST() [4/4]

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

Definition at line 37 of file error\_test.cpp.

### 7.110.2 Variable Documentation

### 7.110.2.1 error

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

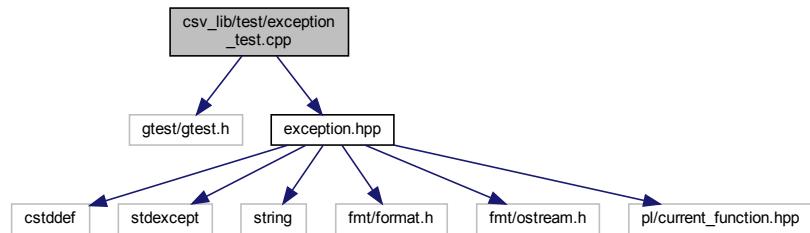
#### Initial value:

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

Definition at line 12 of file error\_test.cpp.

## 7.111 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)

### 7.111.1 Function Documentation

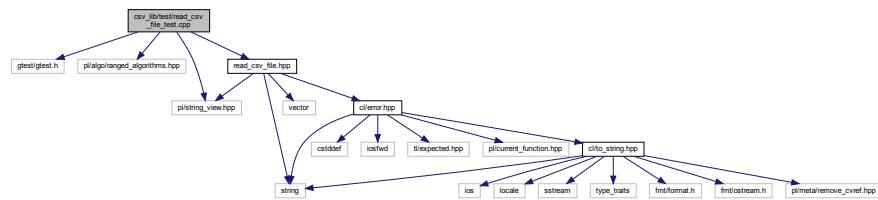
#### 7.111.1.1 TEST()

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

Definition at line 5 of file exception\_test.cpp.

## 7.112 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`)

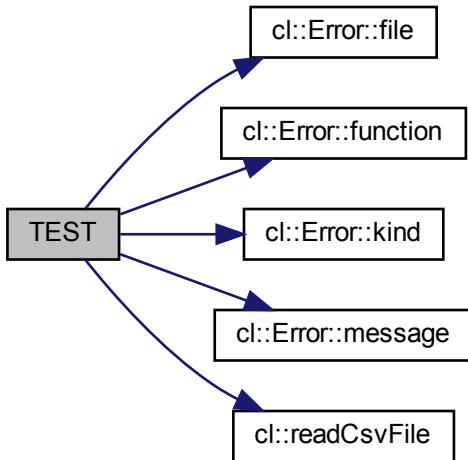
### 7.112.1 Function Documentation

#### 7.112.1.1 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:



### 7.112.1.2 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:

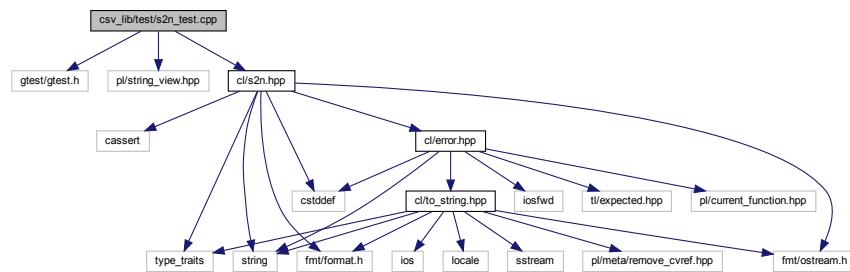


## 7.113 csv\_lib/test/s2n\_test.cpp File Reference

```
#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)

### 7.113.1 Function Documentation

#### 7.113.1.1 TEST() [1/3]

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

Definition at line 21 of file s2n\_test.cpp.

#### 7.113.1.2 TEST() [2/3]

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

Definition at line 29 of file s2n\_test.cpp.

### 7.113.1.3 TEST() [3/3]

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

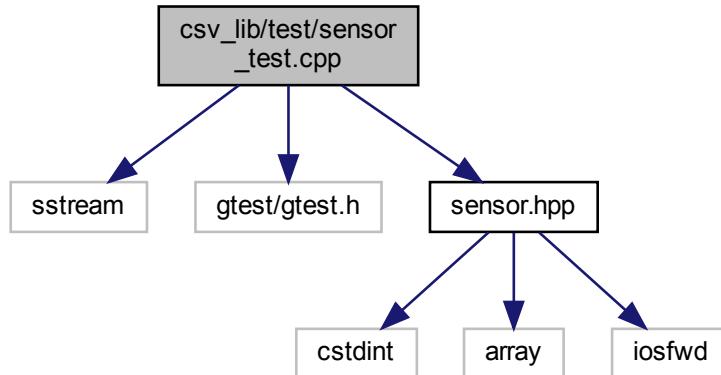
Definition at line 7 of file s2n\_test.cpp.

Here is the call graph for this function:



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

## 7.114.1 Function Documentation

### 7.114.1.1 TEST() [1/2]

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

Definition at line 7 of file sensor\_test.cpp.

### 7.114.1.2 TEST() [2/2]

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

Definition at line 15 of file sensor\_test.cpp.

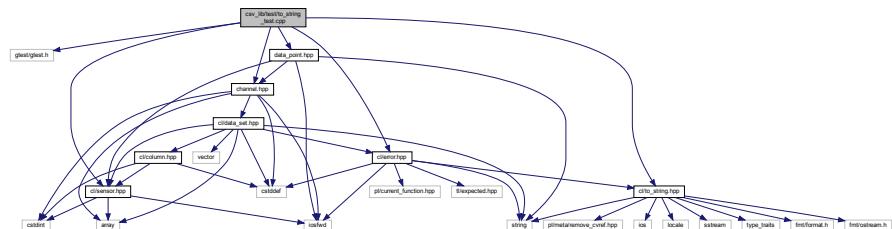
Here is the call graph for this function:



## 7.115 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)

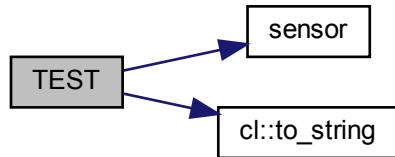
### 7.115.1 Function Documentation

#### 7.115.1.1 TEST()

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

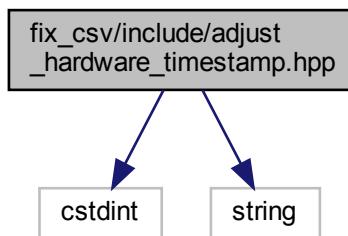
Definition at line 9 of file to\_string\_test.cpp.

Here is the call graph for this function:

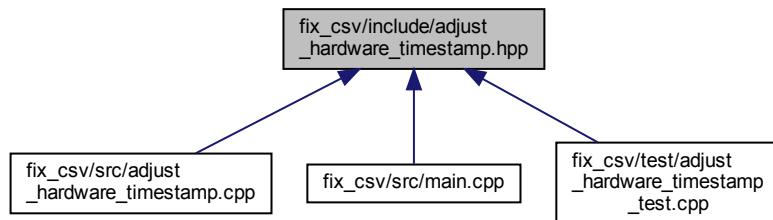


### 7.116 fix\_csv/include/adjust\_hw\_timestamp.hpp File Reference

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



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



## Namespaces

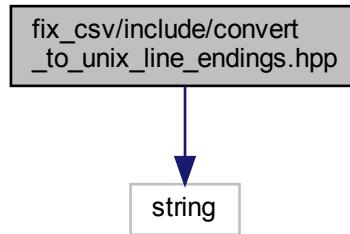
- `fmc`

## Functions

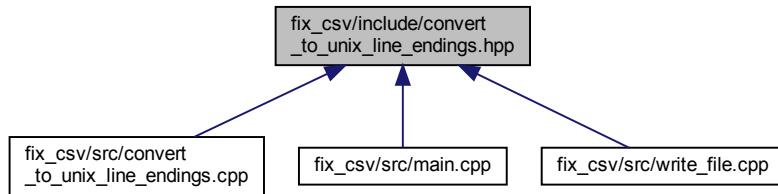
- void `fmc::adjustHardwareTimestamp` (`std::string *cellContent, const std::string &nextRowHardwareTimestamp, std::uint64_t *overflowCount)`

## 7.117 fix\_csv/include/convert\_to\_unix\_line\_endings.hpp File Reference

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



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



## Namespaces

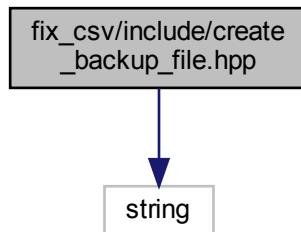
- [fmc](#)

## Functions

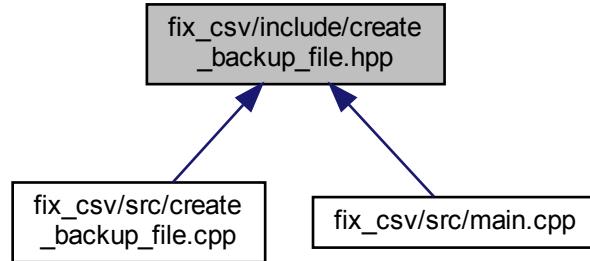
- bool [fmc::convertToUnixLineEndings](#) (const std::string &csvPath)

## 7.118 fix\_csv/include/create\_backup\_file.hpp File Reference

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



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



## Namespaces

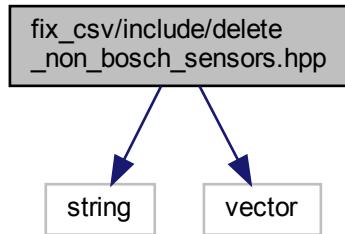
- [fmc](#)

## Functions

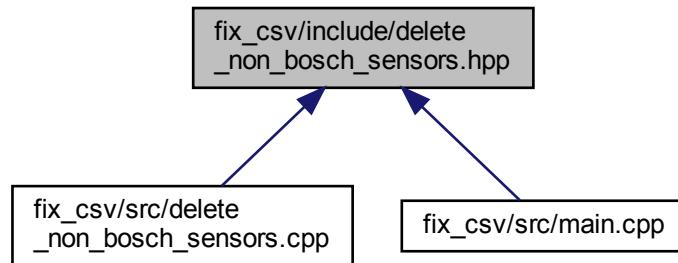
- `bool fmc::createBackupFile (const std::string &csvFilePath, const std::string &backupFilePath)`

## 7.119 fix\_csv/include/delete\_non\_bosch\_sensors.hpp File Reference

```
#include <string>
#include <vector>
Include dependency graph for delete_non_bosch_sensors.hpp:
```



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



## Namespaces

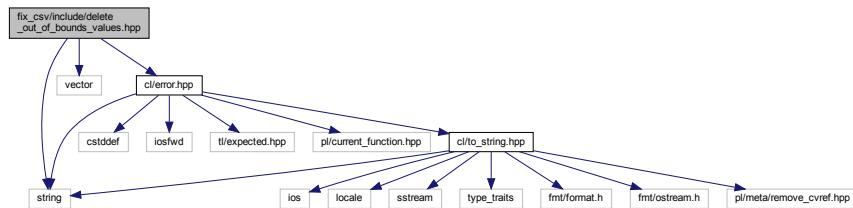
- [fmc](#)

## Functions

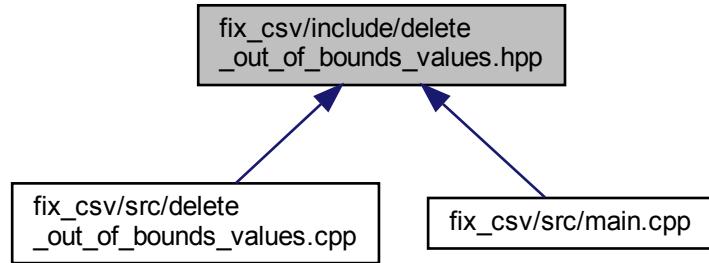
- void [fmc::deleteNonBoschSensors](#) (std::vector< std::vector< std::string >> \*data)

## 7.120 fix\_csv/include/delete\_out\_of\_bounds\_values.hpp File Reference

```
#include <string>
#include <vector>
#include "cl/error.hpp"
Include dependency graph for delete_out_of_bounds_values.hpp:
```



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



## Namespaces

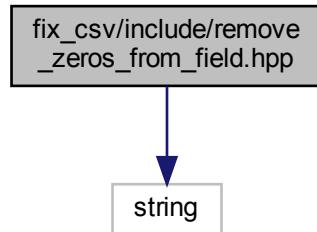
- fmc

## Functions

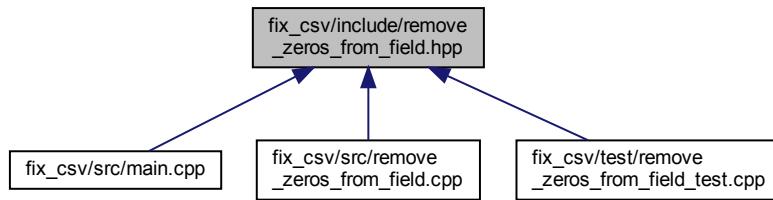
- `cl::Expected< void > fmc::deleteOutOfBoundsValues (std::vector< std::vector< std::string >> *data)`

## 7.121 fix\_csv/include/remove\_zeros\_from\_field.hpp File Reference

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



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



## Namespaces

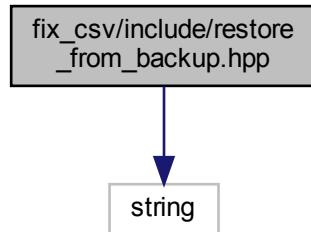
- [fmc](#)

## Functions

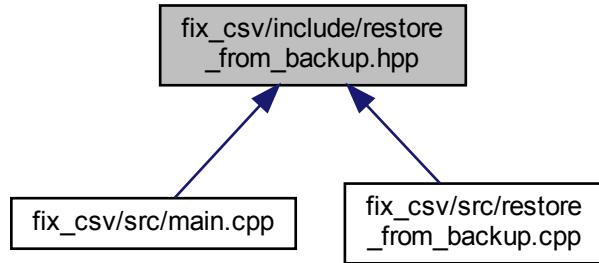
- void [fmc::removeZerosFromField](#) (std::string \*field)

## 7.122 fix\_csv/include/restore\_from\_backup.hpp File Reference

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



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



## Namespaces

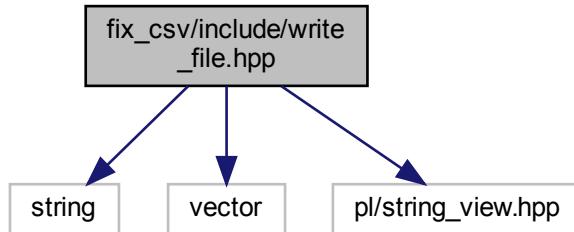
- `fmc`

## Functions

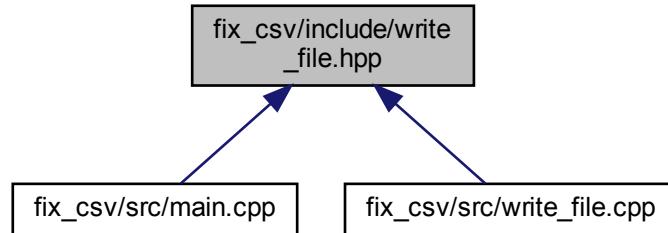
- bool `fmc::restoreFromBackup` (const std::string &csvFilePath, const std::string &backupFilePath)

## 7.123 fix\_csv/include/write\_file.hpp File Reference

```
#include <string>
#include <vector>
#include <pl/string_view.hpp>
Include dependency graph for write_file.hpp:
```



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



## Namespaces

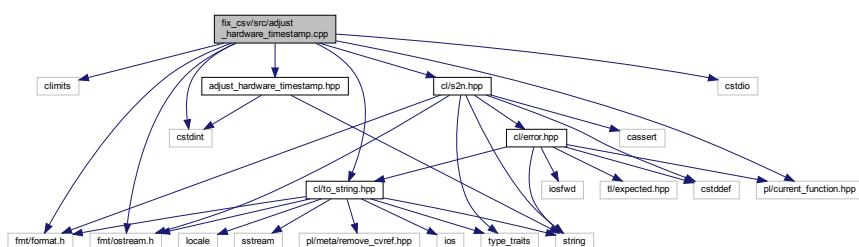
- [fmc](#)

## Functions

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

## 7.124 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

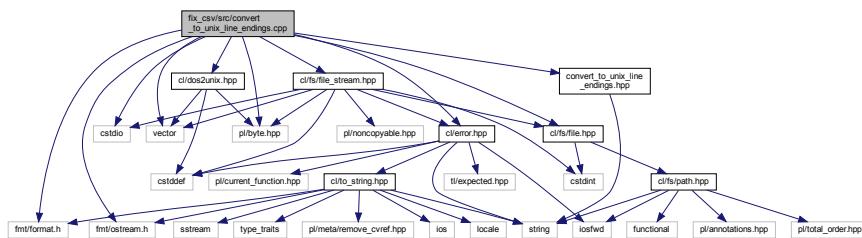
- [fmc](#)

## Functions

- void [fmc::adjustHardwareTimestamp](#) (std::string \*cellContent, const std::string &nextRowHardwareTimestamp, std::uint64\_t \*overflowCount)

## 7.125 fix\_csv/src/convert\_to\_unix\_line\_endings.cpp File Reference

```
#include <cstdio>
#include <vector>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <pl/byte.hpp>
#include "cl/dos2unix.hpp"
#include "cl/error.hpp"
#include "cl/fs/file.hpp"
#include "cl/fs/file_stream.hpp"
#include "convert_to_unix_line_endings.hpp"
Include dependency graph for convert_to_unix_line_endings.cpp:
```



## Namespaces

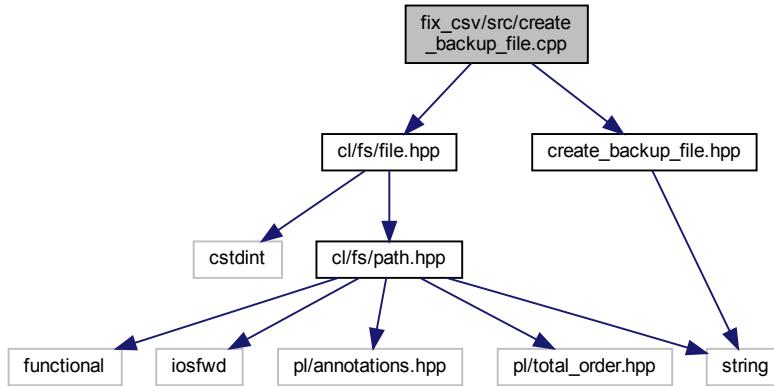
- [fmc](#)

## Functions

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

## 7.126 fix\_csv/src/create\_backup\_file.cpp File Reference

```
#include "cl/fs/file.hpp"
#include "create_backup_file.hpp"
Include dependency graph for create_backup_file.cpp:
```



## Namespaces

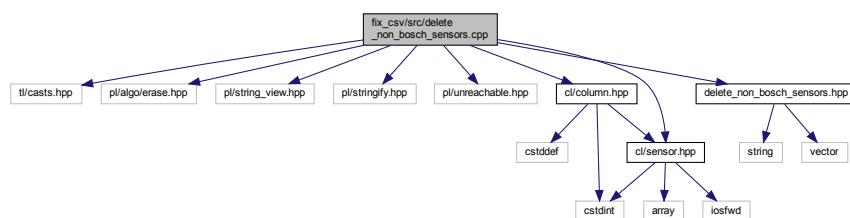
- `fmc`

## Functions

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

## 7.127 fix\_csv/src/delete\_non\_bosch\_sensors.cpp File Reference

```
#include <tl/casts.hpp>
#include <pl/algo/erase.hpp>
#include <pl/string_view.hpp>
#include <pl/stringify.hpp>
#include <pl/unreachable.hpp>
#include "cl/column.hpp"
#include "cl/sensor.hpp"
#include "delete_non_bosch_sensors.hpp"
Include dependency graph for delete_non_bosch_sensors.cpp:
```



## Namespaces

- fmc

## Macros

- #define CL\_SENSOR\_X(enm, value) case cl::Sensor::enm: return PL\_STRINGIFY(value);

## Functions

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

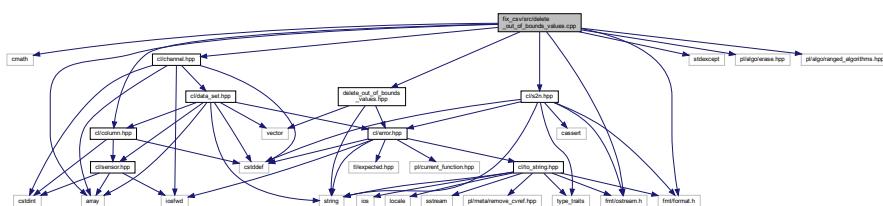
### 7.127.1 Macro Definition Documentation

#### 7.127.1.1 CL\_SENSOR\_X

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

## 7.128 fix\_csv/src/delete\_out\_of\_bounds\_values.cpp File Reference

```
#include <cmath>
#include <array>
#include <stdexcept>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <pl/algo/erase.hpp>
#include <pl/algo/ranged_algorithms.hpp>
#include "cl/channel.hpp"
#include "cl/column.hpp"
#include "cl/s2n.hpp"
#include "delete_out_of_bounds_values.hpp"
Include dependency graph for delete_out_of_bounds_values.cpp:
```



## Namespaces

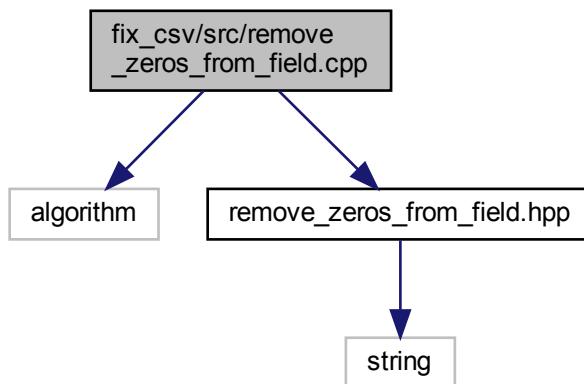
- fmc

## Functions

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

## 7.129 fix\_csv/src/remove\_zeros\_from\_field.cpp File Reference

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



## Namespaces

- `fmc`

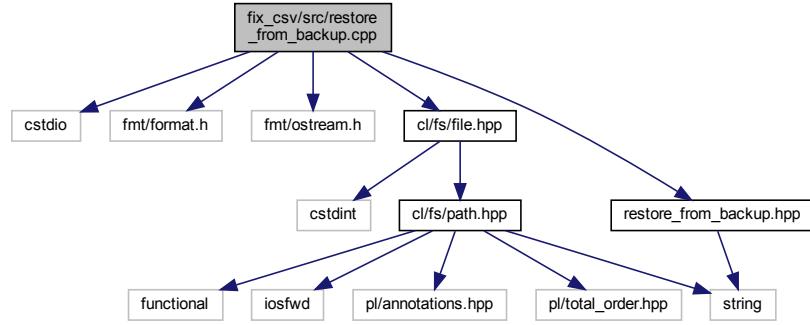
## Functions

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

## 7.130 fix\_csv/src/restore\_from\_backup.cpp File Reference

```
#include <cstdio>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include "cl/fs/file.hpp"
```

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



## Namespaces

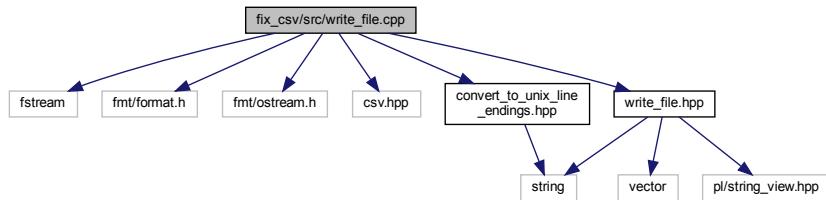
- `fmc`

## Functions

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

## 7.131 fix\_csv/src/write\_file.cpp File Reference

```
#include <fstream>
#include <fmt/format.h>
#include <fmt/ostream.h>
#include <csv.hpp>
#include "convert_to_unix_line_endings.hpp"
#include "write_file.hpp"
Include dependency graph for write_file.cpp:
```



## Namespaces

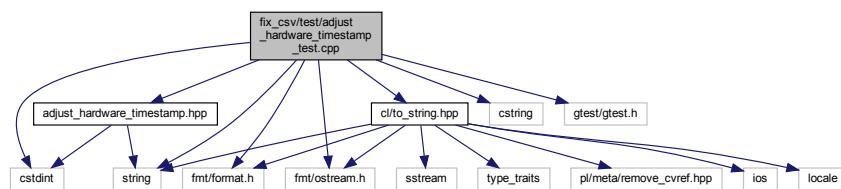
- `fmc`

## Functions

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

## 7.132 fix\_csv/test/adjust\_hwre\_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_hwre_timestamp.hpp"
Include dependency graph for adjust_hwre_timestamp_test.cpp:
```



## Functions

- `TEST` (`adjustHardwareTimestamp`, `shouldDoNothingForNonOverflowedValue`)
- `TEST` (`adjustHardwareTimestamp`, `shouldIncrementOverflowCount`)
- `TEST` (`adjustHardwareTimestamp`, `shouldWorkForOneRoundOfOverflow`)
- `TEST` (`adjustHardwareTimestamp`, `shouldWorkForTwoRoundsOfOverflow`)
- `TEST` (`adjustHardwareTimestamp`, `shouldWork`)

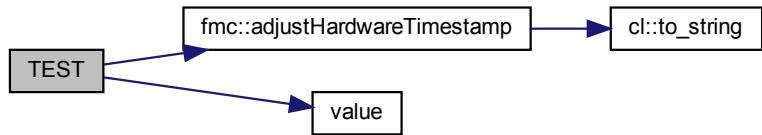
### 7.132.1 Function Documentation

#### 7.132.1.1 TEST() [1/5]

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

Definition at line 15 of file `adjust_hwre_timestamp_test.cpp`.

Here is the call graph for this function:



### 7.132.1.2 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:

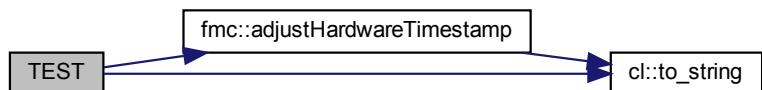


### 7.132.1.3 TEST() [3/5]

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

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

Here is the call graph for this function:



#### 7.132.1.4 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:

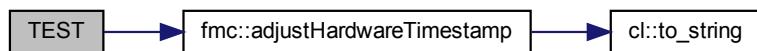


#### 7.132.1.5 TEST() [5/5]

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

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

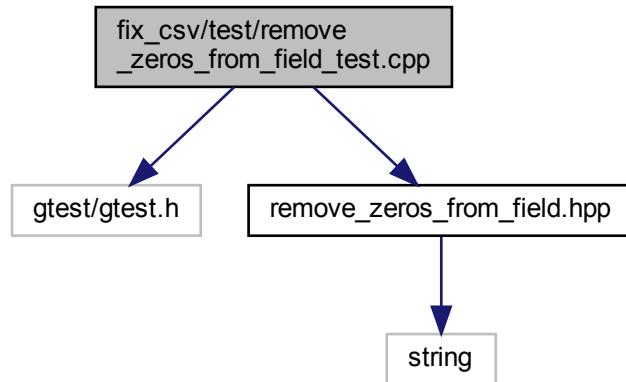
Here is the call graph for this function:



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

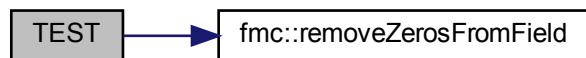
### 7.133.1 Function Documentation

#### 7.133.1.1 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:

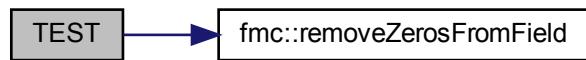


### 7.133.1.2 TEST() [2/6]

```
TEST (
    removeZerosFromField ,
    shouldDeleteStringIfStringIsSingleDot )
```

Definition at line 44 of file remove\_zeros\_from\_field\_test.cpp.

Here is the call graph for this function:



### 7.133.1.3 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:

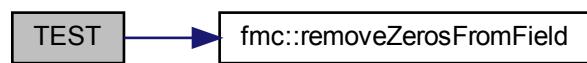


**7.133.1.4 TEST() [4/6]**

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

Definition at line 25 of file remove\_zeros\_from\_field\_test.cpp.

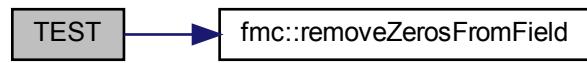
Here is the call graph for this function:

**7.133.1.5 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:



### 7.133.1.6 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, 117

~Process  
    cl::Process, 143

above\_threshold.cpp  
    channel, 220  
    channelAccessor, 221  
    CL\_CHANNEL\_X, 220

above\_threshold\_test.cpp  
    EXPECT\_LONG\_DOUBLE\_EQ, 223  
    TEST, 224

aboveThreshold  
    ctg, 47

accelerometerAverage  
    cl::DataSet, 93

accelerometerMaximum  
    cl::DataSet, 93

accelerometerThreshold  
    cl, 22

AccelerometerX  
    cl, 13

accelerometerX  
    cl::DataSet, 94

AccelerometerY  
    cl, 13

accelerometerY  
    cl::DataSet, 94

AccelerometerZ  
    cl, 13

accelerometerZ  
    cl::DataSet, 95

adjust\_hw\_timestamp\_test.cpp  
    TEST, 296–298

adjustHardwareTimestamp  
    fmc, 53

asMilliseconds  
    cm::ManualSegmentationPoint, 132

averageComparisonValueCalculator  
    ctg, 48

base\_type  
    cl::Exception, 107

Both  
    cs, 39

build  
    cm::Configuration::Builder, 60  
    cs::CsvLineBuilder, 76

Builder  
    cm::Configuration, 74

    cm::Configuration::Builder, 60

Butterworth  
    cs, 39

Channel  
    cl, 13

channel  
    above\_threshold.cpp, 220  
    cl::DataPoint, 89  
    data\_point.cpp, 251

channel.cpp  
    CL\_CHANNEL\_X, 250

channel.hpp  
    CL\_CHANNEL, 227  
    CL\_CHANNEL\_X, 228

channel\_test.cpp  
    TEST, 263, 264

ChannelAccessor  
    cl::DataSet, 93

channelAccessor  
    above\_threshold.cpp, 221

channelCount  
    cl, 23

channels  
    cl, 23

cl, 11  
    accelerometerThreshold, 22  
    AccelerometerX, 13  
    AccelerometerY, 13  
    AccelerometerZ, 13  
    Channel, 13  
    channelCount, 23  
    channels, 23  
    CL\_CHANNEL, 13  
    CL\_CHANNEL\_X, 13  
    CL\_SENSOR, 14  
    CL\_SENSOR\_X, 14  
    CL\_SPECIALIZE\_COL\_TRAITS, 14–16  
    Column, 13  
    column\_index, 23  
    column\_type, 12  
    CsvFileKind, 13  
    data\_set\_accessor\_v, 23  
    dataSetAccessor, 16  
    dos2unix, 16  
    Expected, 12  
    ExtractId, 13  
    Fixed, 14  
    gyroscopeThreshold, 23  
    GyroscopeX, 13

GyroscopeY, 13  
 GyroscopeZ, 13  
 HardwareTimestamp, 13  
 isAccelerometer, 17  
 isGyroscope, 17  
 operator<<, 18, 19  
 Raw, 14  
 readCsvFile, 20  
 s2n, 20  
 SamplingRate, 13  
 Sensor, 14  
 sensors, 24  
 threshold, 21  
 Time, 13  
 to\_string, 21  
 Trigger, 13  
 useUnbufferedIo, 22  
 cl::col\_traits< Col >, 66  
 cl::data\_set\_accessor< Chan >, 87  
 cl::DataPoint, 88  
     channel, 89  
     DataPoint, 89  
     fileName, 89  
     operator<<, 91  
     sensor, 90  
     time, 90  
     value, 91  
 cl::DataSet, 92  
     accelerometerAverage, 93  
     accelerometerMaximum, 93  
     accelerometerX, 94  
     accelerometerY, 94  
     accelerometerZ, 95  
     ChannelAccessor, 93  
     create, 95  
     extractId, 97  
     fileName, 97  
     gyroscopeAverage, 98  
     gyroscopeMaximum, 98  
     gyroscopeX, 99  
     gyroscopeY, 99  
     gyroscopeZ, 100  
     hardwareTimestamp, 100  
     rowCount, 101  
     size\_type, 93  
     time, 101  
     trigger, 102  
 cl::Error, 102  
     CL\_ERROR\_KIND, 103  
     Error, 103  
     file, 104  
     function, 104  
     Kind, 103  
     Kind, 104  
     line, 105  
     message, 105  
     operator<<, 106  
     raise, 105  
     to\_string, 106  
 cl::Exception, 106  
     base\_type, 107  
     Exception, 108  
     file, 108  
     function, 108  
     line, 109  
 cl::fs, 24  
     directoryListing, 25  
     DirectoryListingOption, 25  
     ExcludeDotAndDotDot, 25  
     formatError, 26  
     None, 25  
     operator<, 27  
     operator<<, 27  
     operator==, 27  
     utf16ToUtf8, 28  
     utf8ToUtf16, 29  
 cl::fs::File, 109  
     copyTo, 111  
     create, 111  
     exists, 112  
     File, 110  
     moveTo, 113  
     path, 113  
     remove, 114  
     size, 114  
 cl::fs::FileStream, 115  
     ~FileStream, 117  
     create, 117  
     FileStream, 117  
     OpenMode, 116  
     operator=, 118  
     PL\_NONCOPYABLE, 119  
     Read, 117  
     readAll, 119  
     ReadWrite, 117  
     this\_type, 116  
     Write, 117  
     write, 119  
 cl::fs::Path, 136  
     exists, 138  
     isDirectory, 138  
     isFile, 139  
     operator<, 141  
     operator<<, 141  
     operator==, 141  
     Path, 137  
     str, 140  
 cl::Process, 142  
     ~Process, 143  
     create, 143  
     file, 143, 144  
     operator=, 144  
     PL\_NONCOPYABLE, 144  
     Process, 143  
     this\_type, 143  
     CL\_CHANNEL

channel.hpp, 227  
cl, 13  
**CL\_CHANNEL\_X**  
above\_threshold.cpp, 220  
channel.cpp, 250  
channel.hpp, 228  
cl, 13  
**CL\_ERROR\_KIND**  
cl::Error, 103  
error.hpp, 234  
**CL\_ERROR\_KIND\_X**  
error.cpp, 255  
error.hpp, 235  
**CL\_FS\_SEPARATOR**  
separator.hpp, 241  
**CL\_SENSOR**  
cl, 14  
sensor.hpp, 247  
**CL\_SENSOR\_X**  
cl, 14  
delete\_non\_bosch\_sensors.cpp, 293  
sensor.cpp, 261  
sensor.hpp, 247  
**CL\_SPECIALIZE\_COL\_TRAITS**  
cl, 14–16  
column.hpp, 230  
**CL\_THROW**  
exception.hpp, 236  
**CL\_THROW\_FMT**  
exception.hpp, 236  
**CL\_UNEXPECTED**  
error.hpp, 235  
**cm**, 30  
CM\_DATA\_SET\_IDENTIFIER, 31  
CM\_DATA\_SET\_IDENTIFIER\_X, 31  
CM\_IMU, 31  
CM\_IMU\_X, 31  
DataSetIdentifier, 31  
Imu, 31  
imuCount, 37  
imus, 37  
interpolatedDataSetPaths, 31  
operator!=, 32  
operator<<, 33, 34  
operator==, 34  
segment, 34  
splitString, 35  
toDataSetIdentifier, 36  
**cm::Configuration**, 66  
Builder, 74  
deleteTooClose, 67  
deleteTooCloseOptions, 68  
deleteTooLowVariance, 68  
deleteTooLowVarianceOptions, 68  
filterKind, 69  
filterKindOptions, 69  
imu, 70  
imuOptions, 70  
segmentationKind, 71  
segmentationKindOptions, 71  
skipWindow, 72  
skipWindowOptions, 72  
windowSize, 73  
windowSizeOptions, 73  
**cm::Configuration::Builder**, 59  
build, 60  
Builder, 60  
deleteTooClose, 61  
deleteTooLowVariance, 62  
filterKind, 62  
imu, 63  
segmentationKind, 64  
skipWindow, 64  
windowSize, 65  
**cm::ManualSegmentationPoint**, 130  
asMilliseconds, 132  
frame, 132  
hour, 132  
ManualSegmentationPoint, 131  
minute, 133  
operator!=, 135  
operator<<, 135  
operator==, 136  
readCsvFile, 133  
second, 134  
**CM\_CONTAINS**  
configuration.cpp, 199  
**CM\_DATA\_SET\_IDENTIFIER**  
cm, 31  
data\_set\_identifier.hpp, 192  
**CM\_DATA\_SET\_IDENTIFIER\_X**  
cm, 31  
data\_set\_identifier.cpp, 201  
data\_set\_identifier.hpp, 192  
**CM\_ENSURE\_CONTAINS**  
configuration.cpp, 199  
**CM\_ENSURE\_HAS\_VALUE**  
configuration.cpp, 199  
**CM\_IMU**  
cm, 31  
imu.hpp, 194  
**CM\_IMU\_X**  
cm, 31  
imu.cpp, 202  
imu.hpp, 194  
**CMakeLists.txt**  
include, 145–149  
set, 145–149  
**Column**  
cl, 13  
**column.hpp**  
CL\_SPECIALIZE\_COL\_TRAITS, 230  
**column\_index**  
cl, 23  
**column\_test.cpp**  
TEST, 265

column\_type  
     cl, 12  
 compare\_segmentation/CMakeLists.txt, 145  
 compare\_segmentation/include/csv\_line.hpp, 150  
 compare\_segmentation/include/data\_set\_info.hpp, 151  
 compare\_segmentation/include/filter\_kind.hpp, 153  
 compare\_segmentation/include/log\_files.hpp, 154  
 compare\_segmentation/include/log\_info.hpp, 154  
 compare\_segmentation/include/log\_line.hpp, 155  
 compare\_segmentation/include/paths.hpp, 156  
 compare\_segmentation/include/segmentation\_kind.hpp,  
     157  
 compare\_segmentation/src/csv\_line.cpp, 158  
 compare\_segmentation/src/data\_set\_info.cpp, 159  
 compare\_segmentation/src/filter\_kind.cpp, 159  
 compare\_segmentation/src/log\_files.cpp, 160  
 compare\_segmentation/src/log\_info.cpp, 161  
 compare\_segmentation/src/log\_line.cpp, 162  
 compare\_segmentation/src/main.cpp, 162  
 compare\_segmentation/src/segmentation\_kind.cpp,  
     173  
 compare\_segmentation/test/CMakeLists.txt, 145  
 compare\_segmentation/test/csv\_line\_test.cpp, 174  
 compare\_segmentation/test/data\_set\_info\_test.cpp,  
     174  
 compare\_segmentation/test/log\_files\_test.cpp, 176  
 compare\_segmentation/test/log\_info\_test.cpp, 177  
 compare\_segmentation/test/log\_line\_test.cpp, 187  
 compare\_segmentation/test/main.cpp, 164  
 configuration.cpp  
     CM\_CONTAINS, 199  
     CM\_ENSURE\_CONTAINS, 199  
     CM\_ENSURE\_HAS\_VALUE, 199  
 confusion\_matrix/CMakeLists.txt, 149  
 confusion\_matrix/include/configuration.hpp, 189  
 confusion\_matrix/include/data\_set\_identifier.hpp, 191  
 confusion\_matrix/include/imu.hpp, 193  
 confusion\_matrix/include/interpolated\_data\_set\_paths.hpp,  
     195  
 confusion\_matrix/include/manual\_segmentation\_point.hpp,  
     196  
 confusion\_matrix/include/segment.hpp, 197  
 confusion\_matrix/include/split\_string.hpp, 198  
 confusion\_matrix/src/configuration.cpp, 199  
 confusion\_matrix/src/data\_set\_identifier.cpp, 200  
 confusion\_matrix/src/imu.cpp, 201  
 confusion\_matrix/src/interpolated\_data\_set\_paths.cpp,  
     202  
 confusion\_matrix/src/main.cpp, 171  
 confusion\_matrix/src/manual\_segmentation\_point.cpp,  
     203  
 confusion\_matrix/src/segment.cpp, 204  
 confusion\_matrix/src/split\_string.cpp, 204  
 confusion\_matrix/test/CMakeLists.txt, 149  
 confusion\_matrix/test/data\_set\_identifier\_test.cpp, 205  
 confusion\_matrix/test/interpolated\_data\_set\_paths\_test.cpp,  
     207  
 confusion\_matrix/test/main.cpp, 172  
 confusion\_matrix/test/manual\_segmentation\_point\_test.cpp,  
     208  
 confusion\_matrix/test/segment\_test.cpp, 211  
 confusion\_matrix/test/split\_string\_test.cpp, 213  
 convertToUnixLineEndings  
     fmc, 53  
 copyTo  
     cl::fs::File, 111  
 counting/CMakeLists.txt, 146  
 counting/include/above\_threshold.hpp, 214  
 counting/include/average\_comparison\_value\_calculator.hpp,  
     215  
 counting/include/half\_maximum\_comparison\_value\_calculator.hpp,  
     215  
 counting/include/is\_relevant.hpp, 216  
 counting/include/percentage\_of.hpp, 217  
 counting/include/run\_above\_threshold.hpp, 218  
 counting/src/above\_threshold.cpp, 219  
 counting/src/average\_comparison\_value\_calculator.cpp,  
     221  
 counting/src/half\_maximum\_comparison\_value\_calculator.cpp,  
     222  
 counting/src/main.cpp, 165  
 counting/src/run\_above\_threshold.cpp, 222  
 counting/test/above\_threshold\_test.cpp, 223  
 counting/test/CMakeLists.txt, 146  
 counting/test/main.cpp, 167  
 counting/test/percentage\_of\_test.cpp, 225  
 create  
     cl::DataSet, 95  
     cl::fs::File, 111  
     cl::fs::FileStream, 117  
     cl::Process, 143  
     cs::LogInfo, 121  
 createBackupFile  
     fmc, 54  
 cs, 37  
     Both, 39  
     Butterworth, 39  
     CS\_SPECIALIZE\_DATA\_SET\_INFO, 39–42  
     FilterKind, 39  
     logFiles, 43  
     logPath, 46  
     Maxima, 39  
     Minima, 39  
     MovingAverage, 39  
     oldLogPath, 47  
     operator!=, 44  
     operator<<, 44, 45  
     operator==, 45  
     PL\_DEFINE\_EXCEPTION\_TYPE, 46  
     repetitionCount, 46  
     SegmentationKind, 39  
 cs::CsvLineBuilder, 74  
     build, 76  
     CsvLineBuilder, 76  
     dataSet, 76  
     deleteLowVariance, 77

deleteTooClose, 78  
filter, 79  
isOld, 80  
kind, 81  
repetitions, 82  
segmentationPoints, 83  
sensor, 84  
skipWindow, 85  
this\_type, 75  
windowSize, 86  
cs::data\_set\_info< Tag >, 88  
cs::LogInfo, 120  
    create, 121  
    deleteLowVariance, 122  
    deleteTooClose, 122  
    filterKind, 122  
    invalidSensor, 126  
    isInitialized, 123  
    logFilePath, 123  
    LogInfo, 121  
    operator!=, 125  
    operator<<, 125  
    operator==, 126  
    segmentationKind, 123  
    sensor, 124  
    skipWindow, 124  
    windowSize, 124  
cs::LogLine, 126  
    fileName, 127  
    filePath, 128  
    invalidSensor, 130  
    parse, 128  
    segmentationPointCount, 129  
    sensor, 129  
CS\_SPECIALIZE\_DATA\_SET\_INFO  
    cs, 39–42  
        data\_set\_info.hpp, 152  
    csv\_lib/CMakeLists.txt, 147  
    csv\_lib/include/cl/channel.hpp, 226  
    csv\_lib/include/cl/column.hpp, 229  
    csv\_lib/include/cl/data\_point.hpp, 231  
    csv\_lib/include/cl/data\_set.hpp, 232  
    csv\_lib/include/cl/dos2unix.hpp, 233  
    csv\_lib/include/cl/error.hpp, 234  
    csv\_lib/include/cl/exception.hpp, 235  
    csv\_lib/include/cl/fs/directory\_listing.hpp, 237  
    csv\_lib/include/cl/fs/file.hpp, 238  
    csv\_lib/include/cl/fs/file\_stream.hpp, 239  
    csv\_lib/include/cl/fs/path.hpp, 240  
    csv\_lib/include/cl/fs/sePARATOR.hpp, 240  
    csv\_lib/include/cl/fs/windows.hpp, 242  
    csv\_lib/include/cl/process.hpp, 243  
    csv\_lib/include/cl/read\_csv\_file.hpp, 244  
    csv\_lib/include/cl/s2n.hpp, 245  
    csv\_lib/include/cl/sensor.hpp, 245  
    csv\_lib/include/cl/to\_string.hpp, 248  
    csv\_lib/include/cl/use\_unbuffered\_io.hpp, 249  
    csv\_lib/src/cl/channel.cpp, 249  
          csv/lib/src/cl/data\_point.cpp, 251  
          csv/lib/src/cl/data\_set.cpp, 254  
          csv/lib/src/cl/dos2unix.cpp, 254  
          csv/lib/src/cl/error.cpp, 255  
          csv/lib/src/cl/exception.cpp, 256  
          csv/lib/src/cl/fs/directory\_listing.cpp, 256  
          csv/lib/src/cl/fs/file.cpp, 257  
          csv/lib/src/cl/fs/file\_stream.cpp, 257  
          csv/lib/src/cl/fs/path.cpp, 258  
          csv/lib/src/cl/fs/windows.cpp, 259  
          csv/lib/src/cl/process.cpp, 259  
          csv/lib/src/cl/read\_csv\_file.cpp, 260  
          csv/lib/src/cl/sensor.cpp, 261  
          csv/lib/src/cl/use\_unbuffered\_io.cpp, 262  
          csv/lib/test/channel\_test.cpp, 262  
          csv/lib/test/CMakeLists.txt, 147  
          csv/lib/test/column\_test.cpp, 264  
          csv/lib/test/data\_point\_test.cpp, 266  
          csv/lib/test/data\_set\_test.cpp, 268  
          csv/lib/test/directory\_listing\_test.cpp, 272  
          csv/lib/test/error\_test.cpp, 274  
          csv/lib/test/exception\_test.cpp, 276  
          csv/lib/test/main.cpp, 168  
          csv/lib/test/read\_csv\_file\_test.cpp, 277  
          csv/lib/test/s2n\_test.cpp, 278  
          csv/lib/test/sensor\_test.cpp, 280  
          csv/lib/test/to\_string\_test.cpp, 281  
          csv\_line\_test.cpp  
              TEST, 174  
CsvFileKind  
    cl, 13  
CsvLineBuilder  
    cs::CsvLineBuilder, 76  
ctg, 47  
    aboveThreshold, 47  
    averageComparisonValueCalculator, 48  
    halfMaximumComparisonValueCalculator, 49  
    isRelevant, 50  
    percentageOf, 51  
    runAboveThreshold, 51  
data\_point.cpp  
    channel, 251  
    fileName, 251  
    sensor, 252  
    time, 252  
    value, 253  
data\_point\_test.cpp  
    dp, 267  
    TEST, 266, 267  
data\_set\_accessor\_v  
    cl, 23  
data\_set\_identifier.cpp  
    CM\_DATA\_SET\_IDENTIFIER\_X, 201  
    DSI, 201  
data\_set\_identifier.hpp  
    CM\_DATA\_SET\_IDENTIFIER, 192  
    CM\_DATA\_SET\_IDENTIFIER\_X, 192  
data\_set\_identifier\_test.cpp

DSI, 206  
TEST, 206  
data\_set\_info.hpp  
CS\_SPECIALIZE\_DATA\_SET\_INFO, 152  
data\_set\_info\_test.cpp  
TEST, 175  
data\_set\_test.cpp  
EXPECT\_LONG\_DOUBLE\_EQ, 268  
TEST, 268–271  
DataPoint  
cl::DataPoint, 89  
dataSet  
cs::CsvLineBuilder, 76  
dataSetAccessor  
cl, 16  
DataSetIdentifier  
cm, 31  
delete\_non\_bosch\_sensors.cpp  
CL\_SENSOR\_X, 293  
deleteLowVariance  
cs::CsvLineBuilder, 77  
cs::LogInfo, 122  
deleteNonBoschSensors  
fmc, 54  
deleteOutOfBoundsValues  
fmc, 55  
deleteTooClose  
cm::Configuration, 67  
cm::Configuration::Builder, 61  
cs::CsvLineBuilder, 78  
cs::LogInfo, 122  
deleteTooCloseOptions  
cm::Configuration, 68  
deleteTooLowVariance  
cm::Configuration, 68  
cm::Configuration::Builder, 62  
deleteTooLowVarianceOptions  
cm::Configuration, 68  
directory\_listing\_test.cpp  
TEST, 273  
directoryListing  
cl::fs, 25  
DirectoryListingOption  
cl::fs, 25  
dos2unix  
cl, 16  
dp  
data\_point\_test.cpp, 267  
DSI  
data\_set\_identifier.cpp, 201  
data\_set\_identifier\_test.cpp, 206  
manual\_segmentation\_point.cpp, 203  
manual\_segmentation\_point\_test.cpp, 208  
Error  
cl::Error, 103  
error  
error\_test.cpp, 275  
error.cpp  
CL\_ERROR\_KIND\_X, 255  
error.hpp  
CL\_ERROR\_KIND, 234  
CL\_ERROR\_KIND\_X, 235  
CL\_UNEXPECTED, 235  
error\_test.cpp  
error, 275  
TEST, 275  
Exception  
cl::Exception, 108  
exception.hpp  
CL\_THROW, 236  
CL\_THROW\_FMT, 236  
exception\_test.cpp  
TEST, 276  
ExcludeDotAndDotDot  
cl::fs, 25  
exists  
cl::fs::File, 112  
cl::fs::Path, 138  
EXPECT\_LONG\_DOUBLE\_EQ  
above\_threshold\_test.cpp, 223  
data\_set\_test.cpp, 268  
percentage\_of\_test.cpp, 225  
EXPECT\_SEGMENTATION\_POINTS  
segment\_test.cpp, 212  
Expected  
cl, 12  
ExtractId  
cl, 13  
extractId  
cl::DataSet, 97  
File  
cl::fs::File, 110  
file  
cl::Error, 104  
cl::Exception, 108  
cl::Process, 143, 144  
fileName  
cl::DataPoint, 89  
cl::DataSet, 97  
cs::LogLine, 127  
data\_point.cpp, 251  
filePath  
cs::LogLine, 128  
FileStream  
cl::fs::FileStream, 117  
filter  
cs::CsvLineBuilder, 79  
FilterKind  
cs, 39  
filterKind  
cm::Configuration, 69  
cm::Configuration::Builder, 62  
cs::LogInfo, 122  
filterKindOptions  
cm::Configuration, 69  
fix\_csv/CMakeLists.txt, 148

fix\_csv/include/adjust\_hardware\_timestamp.hpp, 282  
fix\_csv/include/convert\_to\_unix\_line\_endings.hpp, 283  
fix\_csv/include/create\_backup\_file.hpp, 284  
fix\_csv/include/delete\_non\_bosch\_sensors.hpp, 285  
fix\_csv/include/delete\_out\_of\_bounds\_values.hpp, 286  
fix\_csv/include/remove\_zeros\_from\_field.hpp, 287  
fix\_csv/include/restore\_from\_backup.hpp, 288  
fix\_csv/include/write\_file.hpp, 289  
fix\_csv/src/adjust\_hardware\_timestamp.cpp, 290  
fix\_csv/src/convert\_to\_unix\_line\_endings.cpp, 291  
fix\_csv/src/create\_backup\_file.cpp, 292  
fix\_csv/src/delete\_non\_bosch\_sensors.cpp, 292  
fix\_csv/src/delete\_out\_of\_bounds\_values.cpp, 293  
fix\_csv/src/main.cpp, 169  
fix\_csv/src/remove\_zeros\_from\_field.cpp, 294  
fix\_csv/src/restore\_from\_backup.cpp, 294  
fix\_csv/src/write\_file.cpp, 295  
fix\_csv/test/adjust\_hardware\_timestamp\_test.cpp, 296  
fix\_csv/test/CMakeLists.txt, 148  
fix\_csv/test/main.cpp, 170  
fix\_csv/test/remove\_zeros\_from\_field\_test.cpp, 298  
Fixed  
    cl, 14  
fmc, 52  
    adjustHardwareTimestamp, 53  
    convertToUnixLineEndings, 53  
    createBackupFile, 54  
    deleteNonBoschSensors, 54  
    deleteOutOfBoundsValues, 55  
    removeZerosFromField, 55  
    restoreFromBackup, 56  
    writeFile, 56  
formatError  
    cl::fs, 26  
frame  
    cm::ManualSegmentationPoint, 132  
function  
    cl::Error, 104  
    cl::Exception, 108  
  
gyroscopeAverage  
    cl::DataSet, 98  
gyroscopeMaximum  
    cl::DataSet, 98  
gyroscopeThreshold  
    cl, 23  
GyroscopeX  
    cl, 13  
gyroscopeX  
    cl::DataSet, 99  
GyroscopeY  
    cl, 13  
gyroscopeY  
    cl::DataSet, 99  
GyroscopeZ  
    cl, 13  
gyroscopeZ  
    cl::DataSet, 100  
  
halfMaximumComparisonValueCalculator  
    ctg, 49  
HardwareTimestamp  
    cl, 13  
hardwareTimestamp  
    cl::DataSet, 100  
hour  
    cm::ManualSegmentationPoint, 132  
  
Imu  
    cm, 31  
imu  
    cm::Configuration, 70  
    cm::Configuration::Builder, 63  
imu.cpp  
    CM\_IMU\_X, 202  
imu.hpp  
    CM\_IMU, 194  
    CM\_IMU\_X, 194  
imuCount  
    cm, 37  
imuOptions  
    cm::Configuration, 70  
imus  
    cm, 37  
include  
    CMakeLists.txt, 145–149  
interpolated\_data\_set\_paths\_test.cpp  
    TEST, 207  
interpolatedDataSetPaths  
    cm, 31  
invalidSensor  
    cs::LogInfo, 126  
    cs::LogLine, 130  
isAccelerometer  
    cl, 17  
isDirectory  
    cl::fs::Path, 138  
isFile  
    cl::fs::Path, 139  
isGyroscope  
    cl, 17  
isInitialized  
    cs::LogInfo, 123  
isOld  
    cs::CsvLineBuilder, 80  
isRelevant  
    ctg, 50  
  
Kind  
    cl::Error, 103  
kind  
    cl::Error, 104  
    cs::CsvLineBuilder, 81  
  
line  
    cl::Error, 105  
    cl::Exception, 109  
log\_files\_test.cpp

TEST, 176, 177  
`log_info_test.cpp`  
 TEST, 178–187  
`log_line_test.cpp`  
 TEST, 188, 189  
`logFilePath`  
`cs::LogInfo`, 123  
`logFiles`  
`cs`, 43  
`LogInfo`  
`cs::LogInfo`, 121  
`logPath`  
`cs`, 46  
  
`main`  
`main.cpp`, 163, 165, 166, 168–172  
`main.cpp`  
 main, 163, 165, 166, 168–172  
`manual_segmentation_point.cpp`  
`DSI`, 203  
`manual_segmentation_point_test.cpp`  
`DSI`, 208  
 TEST, 209–211  
`ManualSegmentationPoint`  
`cm::ManualSegmentationPoint`, 131  
`Maxima`  
`cs`, 39  
`message`  
`cl::Error`, 105  
`Minima`  
`cs`, 39  
`minute`  
`cm::ManualSegmentationPoint`, 133  
`moveTo`  
`cl::fs::File`, 113  
`MovingAverage`  
`cs`, 39  
  
`None`  
`cl::fs`, 25  
  
`oldLogPath`  
`cs`, 47  
`OpenMode`  
`cl::fs::FileStream`, 116  
`operator!=`  
`cm`, 32  
`cm::ManualSegmentationPoint`, 135  
`cs`, 44  
`cs::LogInfo`, 125  
`operator<`  
`cl::fs`, 27  
`cl::fs::Path`, 141  
`operator<<`  
`cl`, 18, 19  
`cl::DataPoint`, 91  
`cl::Error`, 106  
`cl::fs`, 27  
`cl::fs::Path`, 141  
  
`cm`, 33, 34  
`cm::ManualSegmentationPoint`, 135  
`cs`, 44, 45  
`cs::LogInfo`, 125  
`operator()`  
`std::hash<::cl::fs::Path >`, 120  
`operator=`  
`cl::fs::FileStream`, 118  
`cl::Process`, 144  
`operator==`  
`cl::fs`, 27  
`cl::fs::Path`, 141  
`cm`, 34  
`cm::ManualSegmentationPoint`, 136  
`cs`, 45  
`cs::LogInfo`, 126  
  
`parse`  
`cs::LogLine`, 128  
`Path`  
`cl::fs::Path`, 137  
`path`  
`cl::fs::File`, 113  
`percentage_of_test.cpp`  
`EXPECT_LONG_DOUBLE_EQ`, 225  
 TEST, 225  
`percentageOf`  
`ctg`, 51  
`PL_DEFINE_EXCEPTION_TYPE`  
`cs`, 46  
`PL_NONCOPYABLE`  
`cl::fs::FileStream`, 119  
`cl::Process`, 144  
`Process`  
`cl::Process`, 143  
  
`raise`  
`cl::Error`, 105  
`Raw`  
`cl`, 14  
`Read`  
`cl::fs::FileStream`, 117  
`read_csv_file_test.cpp`  
 TEST, 277, 278  
`readAll`  
`cl::fs::FileStream`, 119  
`readCsvFile`  
`cl`, 20  
`cm::ManualSegmentationPoint`, 133  
`ReadWrite`  
`cl::fs::FileStream`, 117  
`remove`  
`cl::fs::File`, 114  
`remove_zeros_from_field_test.cpp`  
 TEST, 299–301  
`removeZerosFromField`  
`fmc`, 55  
`repetitionCount`  
`cs`, 46

repetitions  
    cs::CsvLineBuilder, 82

restoreFromBackup  
    fmc, 56

rowCount  
    cl::DataSet, 101

runAboveThreshold  
    ctg, 51

s2n  
    cl, 20

s2n\_test.cpp  
    TEST, 279

SamplingRate  
    cl, 13

second  
    cm::ManualSegmentationPoint, 134

segment  
    cm, 34

segment\_test.cpp  
    EXPECT\_SEGMENTATION\_POINTS, 212  
    TEST, 212

SegmentationKind  
    cs, 39

segmentationKind  
    cm::Configuration, 71  
    cm::Configuration::Builder, 64  
    cs::LogInfo, 123

segmentationKindOptions  
    cm::Configuration, 71

segmentationPointCount  
    cs::LogLine, 129

segmentationPoints  
    cs::CsvLineBuilder, 83

Sensor  
    cl, 14

sensor  
    cl::DataPoint, 90  
    cs::CsvLineBuilder, 84  
    cs::LogInfo, 124  
    cs::LogLine, 129  
    data\_point.cpp, 252

sensor.cpp  
    CL\_SENSOR\_X, 261

sensor.hpp  
    CL\_SENSOR, 247  
    CL\_SENSOR\_X, 247

sensor\_test.cpp  
    TEST, 281

sensors  
    cl, 24

separator.hpp  
    CL\_FS\_SEPARATOR, 241

set  
    CMakeLists.txt, 145–149

size  
    cl::fs::File, 114

size\_type  
    cl::DataSet, 93

skipWindow  
    cm::Configuration, 72  
    cm::Configuration::Builder, 64  
    cs::CsvLineBuilder, 85  
    cs::LogInfo, 124

skipWindowOptions  
    cm::Configuration, 72

split\_string\_test.cpp  
    TEST, 213

splitString  
    cm, 35

std::hash<::cl::fs::Path >, 120  
    operator(), 120

str  
    cl::fs::Path, 140

TEST  
    above\_threshold\_test.cpp, 224  
    adjust\_hardware\_timestamp\_test.cpp, 296–298  
    channel\_test.cpp, 263, 264  
    column\_test.cpp, 265  
    csv\_line\_test.cpp, 174  
    data\_point\_test.cpp, 266, 267  
    data\_set\_identifier\_test.cpp, 206  
    data\_set\_info\_test.cpp, 175  
    data\_set\_test.cpp, 268–271  
    directory\_listing\_test.cpp, 273  
    error\_test.cpp, 275  
    exception\_test.cpp, 276  
    interpolated\_data\_set\_paths\_test.cpp, 207  
    log\_files\_test.cpp, 176, 177  
    log\_info\_test.cpp, 178–187  
    log\_line\_test.cpp, 188, 189  
    manual\_segmentation\_point\_test.cpp, 209–211  
    percentage\_of\_test.cpp, 225  
    read\_csv\_file\_test.cpp, 277, 278  
    remove\_zeros\_from\_field\_test.cpp, 299–301  
    s2n\_test.cpp, 279  
    segment\_test.cpp, 212  
    sensor\_test.cpp, 281  
    split\_string\_test.cpp, 213  
    to\_string\_test.cpp, 282

this\_type  
    cl::fs::FileStream, 116  
    cl::Process, 143  
    cs::CsvLineBuilder, 75

threshold  
    cl, 21

Time  
    cl, 13

time  
    cl::DataPoint, 90  
    cl::DataSet, 101  
    data\_point.cpp, 252

to\_string  
    cl, 21  
    cl::Error, 106

to\_string\_test.cpp  
    TEST, 282

toDataSetIdentifier  
    cm, 36  
Trigger  
    cl, 13  
trigger  
    cl::DataSet, 102

useUnbufferedIo  
    cl, 22  
utf16ToUtf8  
    cl::fs, 28  
utf8ToUtf16  
    cl::fs, 29

value  
    cl::DataPoint, 91  
    data\_point.cpp, 253

windowSize  
    cm::Configuration, 73  
    cm::Configuration::Builder, 65  
    cs::CsvLineBuilder, 86  
    cs::LoginInfo, 124  
windowSizeOptions  
    cm::Configuration, 73  
Write  
    cl::fs::FileStream, 117  
write  
    cl::fs::FileStream, 119  
writeFile  
    fmc, 56