

HighFive

2.7.1

Generated by Doxygen 1.9.7

1 HighFive - HDF5 header-only C++ Library	1
2 Version 2.7.1 - 2023-04-04	7
3 Deprecated List	15
4 Namespace Index	17
4.1 Namespace List	17
5 Hierarchical Index	19
5.1 Class Hierarchy	19
6 Class Index	21
6.1 Class List	21
7 File Index	25
7.1 File List	25
8 Namespace Documentation	27
8.1 H5Easy Namespace Reference	27
8.1.1 Detailed Description	28
8.1.2 Enumeration Type Documentation	28
8.1.2.1 DumpMode	28
8.1.2.2 Flush	29
8.1.3 Function Documentation	29
8.1.3.1 dump() [1/6]	29
8.1.3.2 dump() [2/6]	29
8.1.3.3 dump() [3/6]	30
8.1.3.4 dump() [4/6]	30
8.1.3.5 dump() [5/6]	31
8.1.3.6 dump() [6/6]	31
8.1.3.7 dumpAttribute() [1/2]	31
8.1.3.8 dumpAttribute() [2/2]	32
8.1.3.9 getShape()	32
8.1.3.10 getSize()	33
8.1.3.11 load() [1/2]	33
8.1.3.12 load() [2/2]	33
8.1.3.13 loadAttribute()	34
8.2 HighFive Namespace Reference	34
8.2.1 Typedef Documentation	38
8.2.1.1 AttributeCreateProps	38
8.2.1.2 DataSetAccessProps	38
8.2.1.3 DataSetCreateProps	38
8.2.1.4 DataTransferProps	38
8.2.1.5 DataTypeAccessProps	38

8.2.1.6 DataTypeCreateProps	38
8.2.1.7 FileAccessProps	38
8.2.1.8 FileCreateProps	38
8.2.1.9 float16_t	39
8.2.1.10 GroupAccessProps	39
8.2.1.11 GroupCreateProps	39
8.2.1.12 LinkAccessProps	39
8.2.1.13 LinkCreateProps	39
8.2.1.14 ObjectCopyProps	39
8.2.1.15 ObjectCreateProps	39
8.2.1.16 StringCreateProps	39
8.2.1.17 unqualified_t	39
8.2.2 Enumeration Type Documentation	39
8.2.2.1 DataTypeClass	39
8.2.2.2 IndexType	40
8.2.2.3 LinkType	40
8.2.2.4 LogSeverity	40
8.2.2.5 ObjectType	41
8.2.2.6 PropertyType	41
8.2.3 Function Documentation	41
8.2.3.1 compute_total_size()	41
8.2.3.2 create_and_check_datatype()	42
8.2.3.3 create_datatype()	42
8.2.3.4 create_enum_boolean()	42
8.2.3.5 default_logging_callback()	42
8.2.3.6 find_first_atomic_member_size()	42
8.2.3.7 get_global_logger()	42
8.2.3.8 operator&()	42
8.2.3.9 operator" ()	43
8.2.3.10 register_logging_callback()	43
8.2.3.11 to_string()	43
8.2.3.12 toHDF5SizeVector()	43
8.2.3.13 toSTLSizeVector()	43
9 Class Documentation	45
9.1 HighFive::AllocationTime Class Reference	45
9.1.1 Detailed Description	45
9.1.2 Constructor & Destructor Documentation	45
9.1.2.1 AllocationTime() [1/2]	45
9.1.2.2 AllocationTime() [2/2]	45
9.1.3 Member Function Documentation	46
9.1.3.1 getAllocationTime()	46

9.2 HighFive::AnnotateTraits< Derivate > Class Template Reference	46
9.2.1 Member Function Documentation	46
9.2.1.1 createAttribute() [1/3]	46
9.2.1.2 createAttribute() [2/3]	47
9.2.1.3 createAttribute() [3/3]	47
9.2.1.4 deleteAttribute()	48
9.2.1.5 getAttribute()	48
9.2.1.6 getNumberAttributes()	48
9.2.1.7 hasAttribute()	48
9.2.1.8 listAttributeNames()	49
9.3 HighFive::AtomicType< T > Class Template Reference	49
9.3.1 Detailed Description	52
9.3.2 Member Typedef Documentation	52
9.3.2.1 basic_type	52
9.3.3 Constructor & Destructor Documentation	52
9.3.3.1 AtomicType() [1/18]	52
9.3.3.2 AtomicType() [2/18]	52
9.3.3.3 AtomicType() [3/18]	52
9.3.3.4 AtomicType() [4/18]	52
9.3.3.5 AtomicType() [5/18]	52
9.3.3.6 AtomicType() [6/18]	53
9.3.3.7 AtomicType() [7/18]	53
9.3.3.8 AtomicType() [8/18]	53
9.3.3.9 AtomicType() [9/18]	53
9.3.3.10 AtomicType() [10/18]	53
9.3.3.11 AtomicType() [11/18]	53
9.3.3.12 AtomicType() [12/18]	53
9.3.3.13 AtomicType() [13/18]	53
9.3.3.14 AtomicType() [14/18]	53
9.3.3.15 AtomicType() [15/18]	53
9.3.3.16 AtomicType() [16/18]	54
9.3.3.17 AtomicType() [17/18]	54
9.3.3.18 AtomicType() [18/18]	54
9.4 HighFive::AtomicType< char[StrLen]> Class Template Reference	54
9.4.1 Constructor & Destructor Documentation	56
9.4.1.1 AtomicType()	56
9.5 HighFive::AtomicType< FixedLenStringArray< StrLen > > Class Template Reference	57
9.5.1 Constructor & Destructor Documentation	59
9.5.1.1 AtomicType()	59
9.6 HighFive::AtomicType< std::complex< T > > Class Template Reference	59
9.6.1 Constructor & Destructor Documentation	61
9.6.1.1 AtomicType()	61

9.7 HighFive::Attribute Class Reference	62
9.7.1 Detailed Description	64
9.7.2 Constructor & Destructor Documentation	64
9.7.2.1 Attribute()	64
9.7.3 Member Function Documentation	64
9.7.3.1 getCreatePropertyList()	64
9.7.3.2 getDataType()	64
9.7.3.3 getMemSpace()	65
9.7.3.4 getName()	65
9.7.3.5 getSpace()	65
9.7.3.6 getStorageSize()	65
9.7.3.7 Object() [1/4]	65
9.7.3.8 Object() [2/4]	65
9.7.3.9 Object() [3/4]	66
9.7.3.10 Object() [4/4]	66
9.7.3.11 read() [1/3]	66
9.7.3.12 read() [2/3]	66
9.7.3.13 read() [3/3]	66
9.7.3.14 write()	66
9.7.3.15 write_raw()	67
9.7.4 Member Data Documentation	67
9.7.4.1 type	67
9.8 HighFive::AttributeException Class Reference	67
9.8.1 Detailed Description	69
9.8.2 Constructor & Destructor Documentation	69
9.8.2.1 AttributeException()	69
9.9 HighFive::Caching Class Reference	69
9.9.1 Detailed Description	69
9.9.2 Constructor & Destructor Documentation	69
9.9.2.1 Caching() [1/2]	69
9.9.2.2 Caching() [2/2]	70
9.9.3 Member Function Documentation	70
9.9.3.1 getCacheSize()	70
9.9.3.2 getNumSlots()	70
9.9.3.3 getW0()	70
9.10 HighFive::Chunking Class Reference	70
9.10.1 Constructor & Destructor Documentation	70
9.10.1.1 Chunking() [1/4]	70
9.10.1.2 Chunking() [2/4]	71
9.10.1.3 Chunking() [3/4]	71
9.10.1.4 Chunking() [4/4]	71
9.10.2 Member Function Documentation	71

9.10.2.1 getDimensions()	71
9.11 HighFive::CompoundType Class Reference	71
9.11.1 Detailed Description	73
9.11.2 Constructor & Destructor Documentation	74
9.11.2.1 CompoundType() [1/5]	74
9.11.2.2 CompoundType() [2/5]	74
9.11.2.3 CompoundType() [3/5]	74
9.11.2.4 CompoundType() [4/5]	74
9.11.2.5 CompoundType() [5/5]	74
9.11.3 Member Function Documentation	75
9.11.3.1 commit()	75
9.11.3.2 getMembers()	75
9.12 H5Easy::Compression Class Reference	75
9.12.1 Detailed Description	75
9.12.2 Constructor & Destructor Documentation	76
9.12.2.1 Compression() [1/2]	76
9.12.2.2 Compression() [2/2]	77
9.12.3 Member Function Documentation	77
9.12.3.1 get()	77
9.13 HighFive::CreateIntermediateGroup Class Reference	77
9.13.1 Constructor & Destructor Documentation	78
9.13.1.1 CreateIntermediateGroup() [1/3]	78
9.13.1.2 CreateIntermediateGroup() [2/3]	78
9.13.1.3 CreateIntermediateGroup() [3/3]	78
9.13.2 Member Function Documentation	78
9.13.2.1 fromPropertyList()	78
9.13.2.2 isSet()	78
9.14 HighFive::CreationOrder Struct Reference	78
9.14.1 Member Enumeration Documentation	78
9.14.1.1 _CreationOrder	78
9.15 HighFive::DataSet Class Reference	79
9.15.1 Detailed Description	83
9.15.2 Constructor & Destructor Documentation	83
9.15.2.1 DataSet() [1/2]	83
9.15.2.2 DataSet() [2/2]	83
9.15.3 Member Function Documentation	83
9.15.3.1 getAccessPropertyList()	83
9.15.3.2 getCreatePropertyList()	83
9.15.3.3 getDataType()	83
9.15.3.4 getDimensions()	84
9.15.3.5 getElementCount()	84
9.15.3.6 getMemSpace()	84

9.15.3.7	getOffset()	84
9.15.3.8	getSpace()	84
9.15.3.9	getStorageSize()	85
9.15.3.10	Object() [1/4]	85
9.15.3.11	Object() [2/4]	85
9.15.3.12	Object() [3/4]	85
9.15.3.13	Object() [4/4]	85
9.15.3.14	resize()	85
9.15.4	Friends And Related Symbol Documentation	86
9.15.4.1	NodeTraits	86
9.15.4.2	Reference	86
9.15.5	Member Data Documentation	86
9.15.5.1	type	86
9.16	HighFive::DataSetException Class Reference	86
9.16.1	Detailed Description	88
9.16.2	Constructor & Destructor Documentation	88
9.16.2.1	DataSetException()	88
9.17	HighFive::DataSpace Class Reference	88
9.17.1	Detailed Description	90
9.17.2	Member Enumeration Documentation	90
9.17.2.1	DataspaceType	90
9.17.3	Constructor & Destructor Documentation	91
9.17.3.1	DataSpace() [1/8]	91
9.17.3.2	DataSpace() [2/8]	91
9.17.3.3	DataSpace() [3/8]	91
9.17.3.4	DataSpace() [4/8]	91
9.17.3.5	DataSpace() [5/8]	91
9.17.3.6	DataSpace() [6/8]	91
9.17.3.7	DataSpace() [7/8]	92
9.17.3.8	DataSpace() [8/8]	92
9.17.4	Member Function Documentation	92
9.17.4.1	clone()	92
9.17.4.2	From()	92
9.17.4.3	FromArrayStrings()	92
9.17.4.4	getDimensions()	92
9.17.4.5	getElementCount()	93
9.17.4.6	getMaxDimensions()	93
9.17.4.7	getNumberDimensions()	93
9.17.5	Friends And Related Symbol Documentation	93
9.17.5.1	Attribute	93
9.17.5.2	DataSet	93
9.17.5.3	File	93

9.17.6 Member Data Documentation	94
9.17.6.1 type	94
9.17.6.2 UNLIMITED	94
9.18 HighFive::DataSpaceException Class Reference	94
9.18.1 Detailed Description	96
9.18.2 Constructor & Destructor Documentation	96
9.18.2.1 DataSpaceException()	96
9.19 HighFive::DataType Class Reference	96
9.19.1 Detailed Description	98
9.19.2 Member Function Documentation	98
9.19.2.1 empty()	98
9.19.2.2 getClass()	98
9.19.2.3 getCreatePropertyList()	99
9.19.2.4 getSize()	99
9.19.2.5 isFixedLenStr()	99
9.19.2.6 isReference()	99
9.19.2.7 isVariableStr()	99
9.19.2.8 Object() [1/4]	99
9.19.2.9 Object() [2/4]	99
9.19.2.10 Object() [3/4]	99
9.19.2.11 Object() [4/4]	100
9.19.2.12 operator"!="()	100
9.19.2.13 operator=="()	100
9.19.2.14 string()	100
9.19.3 Friends And Related Symbol Documentation	100
9.19.3.1 Attribute	100
9.19.3.2 CompoundType	100
9.19.3.3 DataSet	100
9.19.3.4 File	100
9.20 HighFive::DataTypeException Class Reference	101
9.20.1 Detailed Description	102
9.20.2 Constructor & Destructor Documentation	102
9.20.2.1 DataTypeException()	102
9.21 HighFive::Deflate Class Reference	102
9.21.1 Constructor & Destructor Documentation	103
9.21.1.1 Deflate()	103
9.22 H5Easy::DumpOptions Class Reference	103
9.22.1 Detailed Description	104
9.22.2 Constructor & Destructor Documentation	104
9.22.2.1 DumpOptions() [1/2]	104
9.22.2.2 DumpOptions() [2/2]	104
9.22.3 Member Function Documentation	104

9.22.3.1 compress()	104
9.22.3.2 flush()	105
9.22.3.3 getChunkSize()	105
9.22.3.4 getCompressionLevel()	105
9.22.3.5 isChunked()	105
9.22.3.6 overwrite()	105
9.22.3.7 set() [1/4]	105
9.22.3.8 set() [2/4]	106
9.22.3.9 set() [3/4]	106
9.22.3.10 set() [4/4]	106
9.22.3.11 setChunkSize() [1/2]	106
9.22.3.12 setChunkSize() [2/2]	108
9.23 HighFive::ElementSet Class Reference	108
9.23.1 Constructor & Destructor Documentation	108
9.23.1.1 ElementSet() [1/4]	108
9.23.1.2 ElementSet() [2/4]	109
9.23.1.3 ElementSet() [3/4]	109
9.23.1.4 ElementSet() [4/4]	109
9.23.2 Friends And Related Symbol Documentation	109
9.23.2.1 SliceTraits	109
9.24 HighFive::EnumType< T > Class Template Reference	110
9.24.1 Detailed Description	112
9.24.2 Constructor & Destructor Documentation	112
9.24.2.1 EnumType() [1/3]	112
9.24.2.2 EnumType() [2/3]	113
9.24.2.3 EnumType() [3/3]	113
9.24.3 Member Function Documentation	113
9.24.3.1 commit()	113
9.25 HighFive::EstimatedLinkInfo Class Reference	113
9.25.1 Detailed Description	114
9.25.2 Constructor & Destructor Documentation	114
9.25.2.1 EstimatedLinkInfo() [1/2]	114
9.25.2.2 EstimatedLinkInfo() [2/2]	114
9.25.3 Member Function Documentation	114
9.25.3.1 getEntries()	114
9.25.3.2 getNameLength()	114
9.26 HighFive::Exception Class Reference	115
9.26.1 Detailed Description	116
9.26.2 Constructor & Destructor Documentation	116
9.26.2.1 Exception()	116
9.26.2.2 ~Exception()	116
9.26.3 Member Function Documentation	116

9.26.3.1 getErrMajor()	116
9.26.3.2 getErrMinor()	117
9.26.3.3 nextException()	117
9.26.3.4 setErrorMsg()	117
9.26.3.5 what()	117
9.26.4 Friends And Related Symbol Documentation	117
9.26.4.1 HDF5ErrMapper	117
9.26.5 Member Data Documentation	118
9.26.5.1 _err_major	118
9.26.5.2 _err_minor	118
9.26.5.3 _errmsg	118
9.26.5.4 _next	118
9.27 HighFive::File Class Reference	118
9.27.1 Detailed Description	122
9.27.2 Member Enumeration Documentation	122
9.27.2.1 anonymous enum	122
9.27.3 Constructor & Destructor Documentation	122
9.27.3.1 File() [1/3]	122
9.27.3.2 File() [2/3]	122
9.27.3.3 File() [3/3]	123
9.27.4 Member Function Documentation	123
9.27.4.1 flush()	123
9.27.4.2 getAccessPropertyList()	123
9.27.4.3 getCreatePropertyList()	123
9.27.4.4 getMetadataBlockSize()	123
9.27.4.5 getName()	124
9.27.4.6 getPath()	124
9.27.4.7 getVersionBounds()	124
9.27.4.8 Object() [1/4]	124
9.27.4.9 Object() [2/4]	124
9.27.4.10 Object() [3/4]	124
9.27.4.11 Object() [4/4]	124
9.27.5 Friends And Related Symbol Documentation	124
9.27.5.1 PathTraits	124
9.27.6 Member Data Documentation	125
9.27.6.1 type	125
9.28 HighFive::FileDriver Class Reference	125
9.28.1 Detailed Description	127
9.29 HighFive::FileException Class Reference	128
9.29.1 Detailed Description	129
9.29.2 Constructor & Destructor Documentation	129
9.29.2.1 FileException()	129

9.30 HighFive::FileVersionBounds Class Reference	129
9.30.1 Detailed Description	130
9.30.2 Constructor & Destructor Documentation	130
9.30.2.1 FileVersionBounds() [1/2]	130
9.30.2.2 FileVersionBounds() [2/2]	130
9.30.3 Member Function Documentation	130
9.30.3.1 getVersion()	130
9.31 HighFive::FixedLenStringArray< N > Class Template Reference	131
9.31.1 Detailed Description	132
9.31.2 Member Typedef Documentation	132
9.31.2.1 const_iterator	132
9.31.2.2 const_reverse_iterator	132
9.31.2.3 iterator	132
9.31.2.4 reverse_iterator	132
9.31.2.5 value_type	132
9.31.3 Constructor & Destructor Documentation	132
9.31.3.1 FixedLenStringArray() [1/5]	132
9.31.3.2 FixedLenStringArray() [2/5]	133
9.31.3.3 FixedLenStringArray() [3/5]	133
9.31.3.4 FixedLenStringArray() [4/5]	133
9.31.3.5 FixedLenStringArray() [5/5]	133
9.31.4 Member Function Documentation	133
9.31.4.1 at()	133
9.31.4.2 back()	133
9.31.4.3 begin() [1/2]	134
9.31.4.4 begin() [2/2]	134
9.31.4.5 cbegin()	134
9.31.4.6 cend()	134
9.31.4.7 data() [1/2]	134
9.31.4.8 data() [2/2]	134
9.31.4.9 empty()	134
9.31.4.10 end() [1/2]	134
9.31.4.11 end() [2/2]	134
9.31.4.12 front()	135
9.31.4.13 getString()	135
9.31.4.14 operator[]()	135
9.31.4.15 push_back() [1/2]	135
9.31.4.16 push_back() [2/2]	135
9.31.4.17 rbegin() [1/2]	135
9.31.4.18 rbegin() [2/2]	135
9.31.4.19 rend() [1/2]	135
9.31.4.20 rend() [2/2]	136

9.31.4.21	resize()	136
9.31.4.22	size()	136
9.32	HighFive::Group Class Reference	136
9.32.1	Detailed Description	140
9.32.2	Constructor & Destructor Documentation	140
9.32.2.1	Group() [1/2]	140
9.32.2.2	Group() [2/2]	140
9.32.3	Member Function Documentation	140
9.32.3.1	getCreatePropertyList()	140
9.32.3.2	getEstimatedLinkInfo()	140
9.32.3.3	Object() [1/4]	140
9.32.3.4	Object() [2/4]	141
9.32.3.5	Object() [3/4]	141
9.32.3.6	Object() [4/4]	141
9.32.4	Friends And Related Symbol Documentation	141
9.32.4.1	File	141
9.32.4.2	Reference	141
9.32.5	Member Data Documentation	141
9.32.5.1	type	141
9.33	HighFive::GroupException Class Reference	142
9.33.1	Detailed Description	143
9.33.2	Constructor & Destructor Documentation	143
9.33.2.1	GroupException()	143
9.34	HighFive::HDF5ErrMapper Struct Reference	143
9.34.1	Member Function Documentation	144
9.34.1.1	stackWalk()	144
9.34.1.2	ToException()	144
9.35	HighFive::HyperSlab Class Reference	144
9.35.1	Constructor & Destructor Documentation	145
9.35.1.1	HyperSlab() [1/2]	145
9.35.1.2	HyperSlab() [2/2]	145
9.35.2	Member Function Documentation	145
9.35.2.1	apply()	145
9.35.2.2	notA()	145
9.35.2.3	notB()	145
9.35.2.4	operator&()	145
9.35.2.5	operator&=()	145
9.35.2.6	operator^()	145
9.35.2.7	operator^=()	146
9.35.2.8	operator" ()	146
9.35.2.9	operator" =()	146
9.36	HighFive::LinkCreationOrder Class Reference	146

9.36.1 Detailed Description	146
9.36.2 Constructor & Destructor Documentation	146
9.36.2.1 LinkCreationOrder() [1/3]	146
9.36.2.2 LinkCreationOrder() [2/3]	147
9.36.2.3 LinkCreationOrder() [3/3]	147
9.36.3 Member Function Documentation	147
9.36.3.1 fromPropertyList()	147
9.36.3.2 getFlags()	147
9.37 HighFive::Logger Class Reference	147
9.37.1 Detailed Description	148
9.37.2 Member Typedef Documentation	148
9.37.2.1 callback_type	148
9.37.3 Constructor & Destructor Documentation	148
9.37.3.1 Logger() [1/4]	148
9.37.3.2 Logger() [2/4]	148
9.37.3.3 Logger() [3/4]	149
9.37.3.4 Logger() [4/4]	149
9.37.4 Member Function Documentation	149
9.37.4.1 log()	149
9.37.4.2 operator=() [1/2]	149
9.37.4.3 operator=() [2/2]	149
9.37.4.4 set_logging_callback()	149
9.38 HighFive::CompoundType::member_def Struct Reference	150
9.38.1 Detailed Description	150
9.38.2 Constructor & Destructor Documentation	150
9.38.2.1 member_def()	150
9.38.3 Member Data Documentation	151
9.38.3.1 base_type	151
9.38.3.2 name	151
9.38.3.3 offset	151
9.39 HighFive::EnumType< T >::member_def Struct Reference	151
9.39.1 Detailed Description	151
9.39.2 Constructor & Destructor Documentation	152
9.39.2.1 member_def()	152
9.39.3 Member Data Documentation	152
9.39.3.1 name	152
9.39.3.2 value	152
9.40 HighFive::MetadataBlockSize Class Reference	152
9.40.1 Detailed Description	152
9.40.2 Constructor & Destructor Documentation	153
9.40.2.1 MetadataBlockSize() [1/2]	153
9.40.2.2 MetadataBlockSize() [2/2]	153

9.40.3 Member Function Documentation	153
9.40.3.1 getSize()	153
9.41 HighFive::MPIOCollectiveMetadata Class Reference	153
9.41.1 Detailed Description	153
9.41.2 Constructor & Destructor Documentation	154
9.41.2.1 MPIOCollectiveMetadata() [1/2]	154
9.41.2.2 MPIOCollectiveMetadata() [2/2]	154
9.41.3 Member Function Documentation	154
9.41.3.1 isCollectiveRead()	154
9.41.3.2 isCollectiveWrite()	154
9.42 HighFive::MPIOCollectiveMetadataRead Class Reference	154
9.42.1 Detailed Description	155
9.42.2 Constructor & Destructor Documentation	155
9.42.2.1 MPIOCollectiveMetadataRead() [1/2]	155
9.42.2.2 MPIOCollectiveMetadataRead() [2/2]	155
9.42.3 Member Function Documentation	155
9.42.3.1 isCollective()	155
9.43 HighFive::MPIOCollectiveMetadataWrite Class Reference	155
9.43.1 Detailed Description	156
9.43.2 Constructor & Destructor Documentation	156
9.43.2.1 MPIOCollectiveMetadataWrite() [1/2]	156
9.43.2.2 MPIOCollectiveMetadataWrite() [2/2]	156
9.43.3 Member Function Documentation	156
9.43.3.1 isCollective()	156
9.44 HighFive::MPIOFileAccess Class Reference	156
9.44.1 Detailed Description	157
9.44.2 Constructor & Destructor Documentation	157
9.44.2.1 MPIOFileAccess()	157
9.45 HighFive::MPIOFileDriver Class Reference	157
9.45.1 Detailed Description	159
9.45.2 Constructor & Destructor Documentation	159
9.45.2.1 MPIOFileDriver()	159
9.46 HighFive::MpioNoCollectiveCause Class Reference	160
9.46.1 Detailed Description	160
9.46.2 Constructor & Destructor Documentation	160
9.46.2.1 MpioNoCollectiveCause()	160
9.46.3 Member Function Documentation	160
9.46.3.1 getCause()	160
9.46.3.2 getGlobalCause()	160
9.46.3.3 getLocalCause()	161
9.46.3.4 wasCollective()	161
9.47 HighFive::NodeTraits< Derivate > Class Template Reference	161

9.47.1 Detailed Description	162
9.47.2 Member Function Documentation	163
9.47.2.1 createDataSet() [1/5]	163
9.47.2.2 createDataSet() [2/5]	163
9.47.2.3 createDataSet() [3/5]	163
9.47.2.4 createDataSet() [4/5]	164
9.47.2.5 createDataSet() [5/5]	164
9.47.2.6 createExternalLink()	165
9.47.2.7 createGroup() [1/2]	165
9.47.2.8 createGroup() [2/2]	165
9.47.2.9 createSoftLink() [1/2]	166
9.47.2.10 createSoftLink() [2/2]	166
9.47.2.11 exist()	166
9.47.2.12 getDataSet()	166
9.47.2.13 getGroup()	167
9.47.2.14 getLinkType()	167
9.47.2.15 getNumberObjects()	167
9.47.2.16 getObjectNames()	168
9.47.2.17 getObjectType()	168
9.47.2.18 listObjectNames()	168
9.47.2.19 rename()	169
9.47.2.20 unlink()	169
9.48 HighFive::Object Class Reference	169
9.48.1 Constructor & Destructor Documentation	171
9.48.1.1 Object() [1/4]	171
9.48.1.2 ~Object()	171
9.48.1.3 Object() [2/4]	171
9.48.1.4 Object() [3/4]	171
9.48.1.5 Object() [4/4]	171
9.48.2 Member Function Documentation	171
9.48.2.1 getId()	171
9.48.2.2 getInfo()	171
9.48.2.3 getType()	171
9.48.2.4 isValid()	172
9.48.2.5 operator=()	172
9.48.2.6 operator==(())	172
9.48.3 Friends And Related Symbol Documentation	172
9.48.3.1 CompoundType	172
9.48.3.2 Reference	172
9.48.4 Member Data Documentation	172
9.48.4.1 _hid	172
9.49 HighFive::ObjectException Class Reference	173

9.49.1 Detailed Description	174
9.49.2 Constructor & Destructor Documentation	174
9.49.2.1 ObjectException()	174
9.50 HighFive::ObjectInfo Class Reference	174
9.50.1 Detailed Description	175
9.50.2 Member Function Documentation	175
9.50.2.1 getAddress()	175
9.50.2.2 getCreationTime()	175
9.50.2.3 getModificationTime()	175
9.50.2.4 getRefCount()	176
9.50.3 Friends And Related Symbol Documentation	176
9.50.3.1 Object	176
9.50.4 Member Data Documentation	176
9.50.4.1 raw_info	176
9.51 HighFive::PathTraits< Derivate > Class Template Reference	176
9.51.1 Constructor & Destructor Documentation	176
9.51.1.1 PathTraits()	176
9.51.2 Member Function Documentation	177
9.51.2.1 getFile()	177
9.51.2.2 getPath()	177
9.51.3 Member Data Documentation	177
9.51.3.1 _file_obj	177
9.52 HighFive::PropertyException Class Reference	178
9.52.1 Detailed Description	179
9.52.2 Constructor & Destructor Documentation	179
9.52.2.1 PropertyException()	179
9.53 HighFive::PropertyList< T > Class Template Reference	180
9.53.1 Detailed Description	182
9.53.2 Member Function Documentation	182
9.53.2.1 _initializeIfNeeded()	182
9.53.2.2 add()	182
9.53.2.3 Default()	182
9.53.2.4 getType()	182
9.54 HighFive::PropertyListBase Class Reference	183
9.54.1 Detailed Description	184
9.54.2 Constructor & Destructor Documentation	184
9.54.2.1 PropertyListBase()	184
9.54.3 Member Function Documentation	184
9.54.3.1 Default()	184
9.55 HighFive::RawPropertyList< T > Class Template Reference	185
9.55.1 Detailed Description	187
9.55.2 Member Function Documentation	187

9.55.2.1 add()	187
9.56 HighFive::Reference Class Reference	187
9.56.1 Detailed Description	188
9.56.2 Constructor & Destructor Documentation	188
9.56.2.1 Reference() [1/3]	188
9.56.2.2 Reference() [2/3]	188
9.56.2.3 Reference() [3/3]	188
9.56.3 Member Function Documentation	188
9.56.3.1 create_ref()	188
9.56.3.2 dereference()	189
9.56.3.3 getType()	189
9.57 HighFive::ReferenceException Class Reference	190
9.57.1 Detailed Description	191
9.57.2 Constructor & Destructor Documentation	191
9.57.2.1 ReferenceException()	191
9.58 HighFive::RegularHyperSlab Struct Reference	191
9.58.1 Constructor & Destructor Documentation	192
9.58.1.1 RegularHyperSlab() [1/2]	192
9.58.1.2 RegularHyperSlab() [2/2]	192
9.58.2 Member Function Documentation	192
9.58.2.1 fromHDF5Sizes()	192
9.58.2.2 packedDims()	193
9.58.2.3 rank()	193
9.58.3 Member Data Documentation	193
9.58.3.1 block	193
9.58.3.2 count	193
9.58.3.3 offset	193
9.58.3.4 stride	193
9.59 HighFive::Selection Class Reference	194
9.59.1 Detailed Description	195
9.59.2 Constructor & Destructor Documentation	195
9.59.2.1 Selection()	195
9.59.3 Member Function Documentation	195
9.59.3.1 getDataset() [1/2]	195
9.59.3.2 getDataset() [2/2]	196
9.59.3.3 getDataType()	196
9.59.3.4 getMemSpace()	196
9.59.3.5 getSpace()	196
9.60 HighFive::Shuffle Class Reference	197
9.60.1 Constructor & Destructor Documentation	197
9.60.1.1 Shuffle()	197
9.61 HighFive::SilenceHDF5 Class Reference	197

9.61.1 Detailed Description	197
9.61.2 Constructor & Destructor Documentation	197
9.61.2.1 SilenceHDF5()	197
9.61.2.2 ~SilenceHDF5()	198
9.62 HighFive::SliceTraits< Derivate > Class Template Reference	198
9.62.1 Member Function Documentation	198
9.62.1.1 read() [1/3]	198
9.62.1.2 read() [2/3]	199
9.62.1.3 read() [3/3]	199
9.62.1.4 select() [1/4]	199
9.62.1.5 select() [2/4]	199
9.62.1.6 select() [3/4]	200
9.62.1.7 select() [4/4]	200
9.62.1.8 select_impl()	200
9.62.1.9 write()	200
9.62.1.10 write_raw()	200
9.63 HighFive::Szip Class Reference	201
9.63.1 Constructor & Destructor Documentation	201
9.63.1.1 Szip()	201
9.63.2 Member Function Documentation	201
9.63.2.1 getOptionsMask()	201
9.63.2.2 getPixelsPerBlock()	201
9.64 HighFive::UseCollectiveIO Class Reference	202
9.64.1 Constructor & Destructor Documentation	202
9.64.1.1 UseCollectiveIO() [1/2]	202
9.64.1.2 UseCollectiveIO() [2/2]	202
9.64.2 Member Function Documentation	202
9.64.2.1 isCollective()	202
10 File Documentation	203
10.1 /builddir/build/BUILD/HighFive-2.7.1/CHANGELOG.md File Reference	203
10.2 highfive/bits/H5_definitions.hpp File Reference	203
10.2.1 Macro Definition Documentation	204
10.2.1.1 H5_DEPRECATED	204
10.3 H5_definitions.hpp	204
10.4 highfive/bits/H5Annotate_traits.hpp File Reference	205
10.5 H5Annotate_traits.hpp	206
10.6 highfive/bits/H5Annotate_traits_misc.hpp File Reference	206
10.7 H5Annotate_traits_misc.hpp	207
10.8 highfive/bits/H5Attribute_misc.hpp File Reference	209
10.9 H5Attribute_misc.hpp	210
10.10 highfive/bits/H5Converter_misc.hpp File Reference	212

10.11 H5Converter_misc.hpp	214
10.12 highfive/bits/H5DataSet_misc.hpp File Reference	224
10.13 H5DataSet_misc.hpp	225
10.14 highfive/bits/H5Dataspace_misc.hpp File Reference	226
10.15 H5Dataspace_misc.hpp	227
10.16 highfive/bits/H5DataType_misc.hpp File Reference	229
10.16.1 Macro Definition Documentation	231
10.16.1.1 _H5_STRUCT_PADDING	231
10.17 H5DataType_misc.hpp	231
10.18 highfive/bits/H5Exception_misc.hpp File Reference	237
10.19 H5Exception_misc.hpp	238
10.20 highfive/bits/H5File_misc.hpp File Reference	239
10.21 H5File_misc.hpp	240
10.22 highfive/bits/H5FileDriver_misc.hpp File Reference	242
10.23 H5FileDriver_misc.hpp	242
10.24 highfive/bits/H5Friends.hpp File Reference	243
10.25 H5Friends.hpp	243
10.26 highfive/bits/H5Iterables_misc.hpp File Reference	243
10.27 H5Iterables_misc.hpp	244
10.28 highfive/bits/H5Node_traits.hpp File Reference	245
10.29 H5Node_traits.hpp	246
10.30 highfive/bits/H5Node_traits_misc.hpp File Reference	248
10.31 H5Node_traits_misc.hpp	249
10.32 highfive/bits/H5Object_misc.hpp File Reference	254
10.33 H5Object_misc.hpp	255
10.34 highfive/bits/H5Path_traits.hpp File Reference	257
10.35 H5Path_traits.hpp	258
10.36 highfive/bits/H5Path_traits_misc.hpp File Reference	259
10.37 H5Path_traits_misc.hpp	260
10.38 highfive/bits/H5PropertyList_misc.hpp File Reference	261
10.39 H5PropertyList_misc.hpp	262
10.40 highfive/bits/H5ReadWrite_misc.hpp File Reference	269
10.41 H5ReadWrite_misc.hpp	270
10.42 highfive/bits/H5Reference_misc.hpp File Reference	271
10.43 H5Reference_misc.hpp	273
10.44 highfive/bits/H5Selection_misc.hpp File Reference	274
10.45 H5Selection_misc.hpp	274
10.46 highfive/bits/H5Slice_traits.hpp File Reference	275
10.47 H5Slice_traits.hpp	277
10.48 highfive/bits/H5Slice_traits_misc.hpp File Reference	280
10.49 H5Slice_traits_misc.hpp	281
10.50 highfive/bits/H5Utils.hpp File Reference	285

10.51 H5Utils.hpp	286
10.52 highfive/H5Attribute.hpp File Reference	287
10.53 H5Attribute.hpp	289
10.54 highfive/H5DataSet.hpp File Reference	290
10.55 H5DataSet.hpp	291
10.56 highfive/H5DataSpace.hpp File Reference	292
10.57 H5DataSpace.hpp	293
10.58 highfive/H5DataType.hpp File Reference	294
10.58.1 Macro Definition Documentation	297
10.58.1.1 HIGHFIVE_REGISTER_TYPE	297
10.59 H5DataType.hpp	297
10.60 highfive/H5Easy.hpp File Reference	300
10.61 H5Easy.hpp	303
10.62 highfive/h5easy_bits/H5Easy_Eigen.hpp File Reference	305
10.63 H5Easy_Eigen.hpp	306
10.64 highfive/h5easy_bits/H5Easy_misc.hpp File Reference	308
10.65 H5Easy_misc.hpp	309
10.66 highfive/h5easy_bits/H5Easy_opencv.hpp File Reference	310
10.67 H5Easy_opencv.hpp	311
10.68 highfive/h5easy_bits/H5Easy_public.hpp File Reference	313
10.69 H5Easy_public.hpp	314
10.70 highfive/h5easy_bits/H5Easy_scalar.hpp File Reference	316
10.71 H5Easy_scalar.hpp	317
10.72 highfive/h5easy_bits/H5Easy_vector.hpp File Reference	319
10.73 H5Easy_vector.hpp	320
10.74 highfive/h5easy_bits/H5Easy_xtensor.hpp File Reference	321
10.75 H5Easy_xtensor.hpp	322
10.76 highfive/H5Exception.hpp File Reference	323
10.77 H5Exception.hpp	324
10.78 highfive/H5File.hpp File Reference	326
10.79 H5File.hpp	327
10.80 highfive/H5FileDriver.hpp File Reference	328
10.81 H5FileDriver.hpp	329
10.82 highfive/H5Group.hpp File Reference	329
10.83 H5Group.hpp	330
10.84 highfive/H5Object.hpp File Reference	331
10.85 H5Object.hpp	333
10.86 highfive/H5PropertyList.hpp File Reference	334
10.87 H5PropertyList.hpp	337
10.88 highfive/H5Reference.hpp File Reference	342
10.89 H5Reference.hpp	343
10.90 highfive/H5Selection.hpp File Reference	344

10.91 H5Selection.hpp	345
10.92 highfive/H5Utility.hpp File Reference	346
10.92.1 Macro Definition Documentation	348
10.92.1.1 HIGHFIVE_LOG_DEBUG	348
10.92.1.2 HIGHFIVE_LOG_DEBUG_IF	348
10.92.1.3 HIGHFIVE_LOG_ERROR	348
10.92.1.4 HIGHFIVE_LOG_ERROR_IF	349
10.92.1.5 HIGHFIVE_LOG_INFO	349
10.92.1.6 HIGHFIVE_LOG_INFO_IF	349
10.92.1.7 HIGHFIVE_LOG_LEVEL	349
10.92.1.8 HIGHFIVE_LOG_LEVEL_DEBUG	349
10.92.1.9 HIGHFIVE_LOG_LEVEL_ERROR	349
10.92.1.10 HIGHFIVE_LOG_LEVEL_INFO	349
10.92.1.11 HIGHFIVE_LOG_LEVEL_WARN	349
10.92.1.12 HIGHFIVE_LOG_WARN	350
10.92.1.13 HIGHFIVE_LOG_WARN_IF	350
10.93 H5Utility.hpp	350
10.94 /builddir/build/BUILD/HighFive-2.7.1/README.md File Reference	352
Index	353

Chapter 1

HighFive - HDF5 header-only C++ Library

Documentation: <https://bluebrain.github.io/HighFive/>

Brief

[HighFive](#) is a modern header-only C++11 friendly interface for libhdf5.

[HighFive](#) supports STL vector/string, Boost::UBLAS, Boost::Multi-array, Eigen and Xtensor. It handles C++ from/to HDF5 with automatic type mapping. [HighFive](#) does not require additional libraries (see dependencies) and supports both HDF5 thread safety and Parallel HDF5 (contrary to the official hdf5 cpp)

It integrates nicely with other CMake projects by defining (and exporting) a [HighFive](#) target.

Design

- Simple C++-ish minimalist interface
- No other dependency than libhdf5
- Zero overhead
- Support C++11

Feature support

- create/read/write files, datasets, attributes, groups, dataspace.
- automatic memory management / ref counting
- automatic conversion of `std::vector` and nested `std::vector` from/to any dataset with basic types
- automatic conversion of `std::string` to/from variable length string dataset
- selection() / slice support
- parallel Read/Write operations from several nodes with Parallel HDF5
- Advanced types: Compound, Enum, Arrays of Fixed-length strings, References
- half-precision (16-bit) floating-point datasets
- `std::byte` in C++17 mode (with `-DCMAKE_CXX_STANDARD=17` or higher)
- etc... (see [ChangeLog](#))

Dependencies

- hdf5 (dev)
- hdf5-mpi (optional, opt-in with `-D*HIGHFIVE_PARALLEL_HDF5*=ON`)
- boost $\geq$ 1.41 (recommended, opt-out with `-D*HIGHFIVE_USE_BOOST*=OFF`)
- eigen3 (optional, opt-in with `-D*HIGHFIVE_USE_EIGEN*=ON`)
- xtensor (optional, opt-in with `-D*HIGHFIVE_USE_XTENSOR*=ON`)
- half (optional, opt-in with `-D*HIGHFIVE_USE_HALF_FLOAT*=ON`)

Examples

Write a `std::vector<int>` to 1D HDF5 dataset and read it back

```
++
#include <highfive/H5File.hpp>

using namespace HighFive;

std::string filename = "/tmp/new_file.h5";

{
    // We create an empty HDF5 file, by truncating an existing
    // file if required:
    File file(filename, File::Truncate);

    std::vector<int> data(50, 1);
    file.createDataSet("grp/data", data);
}

{
    // We open the file as read-only:
    File file(filename, File::ReadOnly);
    auto dataset = file.getDataSet("grp/data");

    // Read back, with allocating:
    auto data = dataset.read<std::vector<int>>();

    // Because `data` has the correct size, this will
    // not cause `data` to be reallocated:
    dataset.read(data);
}
```

Note: `H5File.hpp` is the top-level header of [HighFive](#) core which should be always included.

Note: For advanced usecases the dataset can be created without immediately writing to it. This is common in MPI-IO related patterns, or when growing a dataset over the course of a simulation.

Write a 2 dimensional C double float array to a 2D HDF5 dataset

See [create_dataset_double.cpp](#)

Write and read a matrix of double float (boost::ublas) to a 2D HDF5 dataset

See [boost_ublas_double.cpp](#)

Write and read a subset of a 2D double dataset

See [select_partial_dataset_cpp11.cpp](#)

Create, write and list HDF5 attributes

See [create_attribute_string_integer.cpp](#)

And others

See [src/examples/](#) subdirectory for more info.

Compiling with HighFive

```
c++ -o program -I/path/to/highfive/include source.cpp -lhdf5
```

H5Easy

For several 'standard' use cases the [highfive/H5Easy.hpp](#) interface is available. It allows:

- Reading/writing in a single line of:
 - scalars (to/from an extendible DataSet),
 - strings,
 - vectors (of standard types),
 - [Eigen::Matrix](#) (optional, enable CMake option HIGHFIVE_USE_EIGEN),
 - [xt::xarray](#) and [xt::xtensor](#) (optional, enable CMake option HIGHFIVE_USE_XTENSOR).
 - [cv::Mat_](#) (optional, enable CMake option HIGHFIVE_USE_OPENCV).
- Getting in a single line:
 - the size of a DataSet,
 - the shape of a DataSet.

Example

```
#include <highfive/H5Easy.hpp>

int main() {
    H5Easy::File file("example.h5", H5Easy::File::Overwrite);

    int A = ...;
    H5Easy::dump(file, "/path/to/A", A);

    A = H5Easy::load<int>(file, "/path/to/A");
}
```

whereby the `int` type of this example can be replaced by any of the above types. See [easy_load_dump.cpp](#) for more details.

Note: Classes such as `H5Easy::File` are just short for the regular [HighFive](#) classes (in this case [HighFive::File](#)). They can thus be used interchangeably.

CMake integration

HighFive can easily be used by other C++ CMake projects.

You may use HighFive from a folder in your project (typically a git submodule).

```
cmake_minimum_required(VERSION 3.1 FATAL_ERROR)
project(foo)
set(CMAKE_CXX_STANDARD 11)

add_subdirectory(highfive_folder)
add_executable(bar bar.cpp)
target_link_libraries(bar HighFive)
```

Alternatively you can install HighFive once and use it in several projects via `find_package()`.

A HighFive target will bring the compilation settings to find HighFive headers and all chosen dependencies.

```
# ...
find_package(HighFive REQUIRED)
add_executable(bar bar.cpp)
target_link_libraries(bar HighFive)
```

Note: Like with other libraries you may need to provide CMake the location to find highfive: `CMAKE_PREFIX_PATH=<highfive_install_dir>`

Note: `find_package(HighFive)` will search dependencies as well (e.g. Boost if requested). In order to use the same dependencies found at HighFive install time (e.g. for system deployments) you may set `HIGHFIVE_USE_INSTALL_DEPS=YES`

Installing

When installing via CMake, besides the headers, a `HighFiveConfig.cmake` is generated which provides the HighFive target, as seen before. Note: You may need to set `CMAKE_INSTALL_PREFIX`:

```
mkdir build && cd build
# Look up HighFive CMake options, consider inspecting with `cmake`
cmake .. -DHIGHFIVE_EXAMPLES=OFF -DCMAKE_INSTALL_PREFIX=<highfive_install_dir>
make install
```

Test Compilation

As a header-only library, HighFive doesn't require compilation. You may however build tests and examples.

```
mkdir build && cd build
cmake ../
make # build tests and examples
make test # build and run unit tests
```

Note: Unit tests require Boost. In case it's unavailable you may use `-DHIGHFIVE_USE_BOOST=OFF`. HighFive with disabled support for Boost types as well as unit tests (though most examples will build).

Code formatting

If you want to propose pull requests to this project, do not forget to format code with clang-format version 12. The `.clang-format` is at the root of the git repository.

Questions?

Do you have questions on how to use HighFive? Would you like to share an interesting example or discuss HighFive features? Head over to the [Discussions](#) forum and join the community.

Funding & Acknowledgment

The development of this software was supported by funding to the Blue Brain Project, a research center of the École polytechnique fédérale de Lausanne (EPFL), from the Swiss government's ETH Board of the Swiss Federal Institutes of Technology.

Copyright © 2015-2022 Blue Brain Project/EPFL

License

Boost Software License 1.0

Chapter 2

Version 2.7.1 - 2023-04-04

Bug Fix

- Revert removing `#include "H5FileDriver.hpp"` from [H5File.hpp](#) (#711).
- Change relative import to `../H5Utility.hpp` (#726).
- Fix nameclash with macros on Windows (#717 #722 #723).
- Add workaround for MSVC bug (#728).
- Don't downgrade the requested C++ standard (#729).

Version 2.7.0 - 2023-03-31

New Features

- Properties can now be read (#684).
- Adding a property for LinkCreationOrder (#683).
- Adding a logging infrastructure (#690).
- Support of bool in the way of h5py (#654).
- Support `std::bool` in C++17 mode (#698).

Improvements

- Catch2 move to v3 (#655).

Bug Fix

- To avoid build failure in certain circumstances, user can not set `Boost_NO_BOOST_CMAKE` (#687).
- Fix leak when reading variable length strings (#660).
- Use `H5free_memory` instead of `free` in error handler (#665). Thanks to Moritz Koenemann.
- Fix a bug with old GCC due to templated friend classes (#688).
- Fix regression in broadcasting support (#697).
- Fix bug related to zero-length datasets (#702).

Version 2.6.2 - 2022-11-10

Bug Fix

- Allow CMake to use Config mode to find HDF5.

Version 2.6.1 - 2022-11-08

Bug Fix

- Version bump in `CMakeLists.txt`.

Version 2.6.0 - 2022-11-08

New Features

- Enable page buffered reading (#639).

Improvements

- Warn when detecting lossy reads or write of floating point data (#636).

Version 2.5.1 - 2022-11-07

Bug Fix

- Fix missing `inline` for collective metadata properties.

Version 2.5.0 - 2022-11-03

New Features

- Enable collective MPI IO using the Data Transfer Property (#623). Thanks to Rob Latham.
- Add a support for half-precision (16-bit) floating-point based on the Half library (<http://half.sourceforge.net>) (#587). Thanks to Sergio Botelh.
- Enable choosing the allocation time of datasets (#627).
- Add possibility to get and set file space strategy. For page allocated files wrap the API to set/retrieve the page size (#618).
- Add API for getting Access and Create property lists of [HighFive](#) objects (#629).
- Let users configure metadata reads and writes at file level (#624). Thanks to Rob Latham.

Improvements

- `MPIOFileDriver` is now deprecated. Use `FileAccessProps` (#622).
- Support of block argument in API (#584).
- Serialization of types is now automagic and so recursive (#586).
- Add an argument to specific File Create Properties in File class constructor (#626).

Bug Fixes

- Padding of Compound Types (#581).
- Compilation with Visual Studio with C++17 or later (#578). Thanks to Mark Bicknell.
- Avoid leaking when printing stack for error (#583).

Version 2.4.1 - 2022-05-11

New Features

- Support `std::complex`. Thanks to Philipp.

Improvements

- Improve `EnumType/CompoundType`
- Revert quirky behaviour of `select(const HyperSlab&)`.
- All `get_name` functions takes `size_t` and not `hsize_t`.
- Remove nix recipes.

Bug Fixes

- Computation of padding.
- Related to 0 being an invalid hid but not equal to `H5I_INVALID_HID`.

Version 2.4.0 - 2022-04-05

New Features

- Construct a compound type from an already existing hid (#469). Thanks to Maximilian Nöthe.
- Add support for long double (#494)
- Add support for `H5Pset_libver_bounds` and `H5Pset_meta_block_size` support (#500)
- New interface to select complex hyperslabs, irregular hyperslabs are limited to/from 1D array (#538 and #545)

Improvements

- Use inline where it is needed, otherwise some code can lead to "multiple definition" (#516). Thanks to Chris Byrohl.
- Use Catch2 instead of boost for tests, reduces dependencies (#521)
- CI reworked to test external libraries more thoroughly (boost, eigen, xtensor) (#536)

Bug Fixes

- Better support of const types (#460). Thanks to Philip Deegan.
- Vector of size zero was previously lead to UB (#502). Thanks to Haoran Ni.
- Use H5T_NATIVE_SCHAR instead of H5T_NATIVE_CHAR for "signed char" (#518)

Version 2.3.1 - 2021-08-04

Improvements

- Clean cmake files from old code (#465)
- Adding path to type warning message (#471)
- Adding compound types example, w dataset and attr (#467)

Bug Fixes

- Resolve an issue where padding of nested compound types were being calculated incorrectly (#461) (#468)
- GHA: drop previous runs (#462)

Version 2.3 - 2021-05-07

New Features:

- Add SZIP support (#435)
- Add option *parents* to createDataSet (#425)
- Implementing getting the filename dynamically (#424)
- Ability to create soft and external links (#421)
- Generalizing getPath and adding getFile as PathTraits (#417)

Improvements:

- Unified reading/writing attributes and datasets (#450)
- Old compilers have been removed from docker image (#430)
- Cleaning up and improving property lists (#429)
- An example using hdf5 references (#396) (#397)
- Add all property lists alias for completeness (#427)
- Add property CreateIntermediateGroup (#423)
- Add code coverage through codecov.io (#420)
- Introducing GitHub Actions CI (#416)
- Create issue and PR templates (#412)
- Initialize SilenceHDF5 to true in _exist (#411)
- Generalizing xtensor API (#407)
- Minor doc updates (#409)
- Fixing minor error in GH Action (#408)
- Uploading docs to gh-pages using GitHub Actions (#403)
- Various minor documentation updates (#405)
- optional documentation building in cmake (#377)
- From can be automatic now (#384)
- get_dim_vector in inspector (#383)
- Put type_of_array in inspector (#382)
- Move array_dims in the future manipulator (#381)
- Unify interface of H5Attribute with H5Slice_traits (#378)
- Use std::move in NRVO depending of version of GCC (#375)
- Fixed typo '-DD' to '-D' in 'Dependencies'. (#371)
- Changing date format (#364)

Bug fixes:

- Fix use before initialization (#414)
- Adding CMake include guard (#389)

Version 2.2.2 - 2020-07-30

New Features:

- [\[H5Easy\]](#) Adding OpenCV support (#343)
- [\[H5Easy\]](#) Enabling compression & Adding attributes (#337)
- Adding missing function to H5Attribute (#337)
- Add methods to retrieve Node paths or Dataset names and rename objects (#346)
- Add a file with the current version number of [HighFive](#) (#349)

Improvements

- [\[H5Easy\]](#) Updating error message dump (#335)
- [\[H5Easy\]](#) Switching implementation to partial specialization based on static dispatch (#327)
- Simplifying imports, new policy (#324)

Version 2.2.1 - 2020-04-28

Improvements

- Add a mechanism to not include target [HighFive](#) several times (#336)
- Fix SilenceHDF5 initialization for NodeTraits (#333)

Version 2.2 - 2020-03-23

New Features:

- Compound Types: API to register and read/write structs (#78). Thanks to Richard Shaw.
- Fixed-length strings. API via `char[]` and `FixedLenStringArray` (#277)
- Enum data types (#297)
- Datasets of HDF5 References. Support to dereference groups and datasets (#306)
- Objects (hard/soft link) can now be deleted with `unlink` (#284). Thanks to Tom Vander Aa.
- Attributes can be deleted with `deleteAttribute` (#239)

Improvements:

- `Attributes` (metadata) now support additional types (#298)
- [H5Easy](#): Reworked for compatibility with `Eigen::ref` and `Eigen::Map` (#291, #293)
- Hdf5 1.12 compatibility: working `Object::getInfo` and marking `getAddress` deprecated (#311)
- Strict compatibility with CMake 3.1 and C++11 (#304)
- CMake: Dependencies may be re-detected on `FindPackage`, fixed export targets and added integration tests (#255, #304, #312, #317)
- Support for array of `Eigen::Matrix` (#258)
- Selection: `ElementSet` working for N-dimensions (#247)

Bug Fixes:

- Shortcut syntax with c arrays (#273)
- Compatibility with in MSVC (Exception messages #263 and avoid throwing in `exist` check #308)

Version 2.1 - 2019-10-30

New Features:

- Inspection: API to get the type of links/objects and datasets data-types (#221)
- [H5Easy](#): API for simple import/export to Eigen and xtensor (#141)
- Support for chunk and deflate configuration at dataset creation/open (#125). Added generic RawProperty↔ Lists. (#157)
- Recursive `createGroup` and `exist` (#152)
- Shortcut syntax: ability to create a filled dataset in a single line (#130)
- DataSet now accepts `std::complex` and `std::array`'s (#128, #129)

Improvements:

- Improved compat with MSVC and ICC compilers
- CMake build system: modernized, create exported targets, better messages, etc.
- Building and publishing documentation: <https://bluebrain.github.io/HighFive/>
- Several other. See #231

Bug Fixes:

- Fixed header dependencies. They are now all include-able (#225)
- Fixed read/write of N-Dimensional data as nested vectors (#191)
- Fixed data broadcasting for reading (#136)

Version 2.0 - 2018-07-19

- First version with C++11 enforcement
- Support for property list
- Support for Chunking
- Support for Compression / Deflate
- Fix: missing move constructor for properties
- Fix: typo in MPI IO driver
- Fix: several typo fixes
- Fix: Add missing include

Version 1.5 - 2018-01-06

- SliceTraits::read split in two overloads, the first one for plain C arrays and the second one for other types.
- Add support for complex number
- Add exist() method to the API
- Will be last release before 2.0 and enforcement of C++11

Version 1.4 - 2017-08-25

- Support id selection for the `select` function
- Support STL containers of const elements
- Support scalar values and strings management
- Fix attribute assignment issue #40
- Fix Object assignment operator missing unref (possible memory leak)
- Introduce SilenceHDF5 for HDF5 error report
- Fix a unit test issue with SilenceHDF5

Version 1.3 - 2017-06-21

- Minor fixes

Version 1.2 - 2017-04-03

- Add Attribute support for Dataset
- Extend testing of Attribute support
- Fix issue related to multiple definitions in default driver
- Add more examples about attribute support

Version 1.1 - 2017-03-23

- Add support and examples for Parallel HDF5
- Initial implementation for H5 Properties
- Support for Attributes
- Improve documentation
- Add example for boost.Ublas matrix support

Version 1.0 - Init

- Initial release

Chapter 3

Deprecated List

Member [HighFive::DataSet::DataSet](#) ()=default

Default constructor creates unsafe uninitialized objects

Class [HighFive::FileDriver](#)

Use FileAccessProps directly

Member [HighFive::Group::Group](#) ()=default

Default constructor creates unsafe uninitialized objects

Class [HighFive::MPIOFileDriver](#)

Add [MPIOFileAccess](#) directly to FileAccessProps

Member [HighFive::ObjectInfo::getAddress](#) () const noexcept

Deprecated since [HighFive](#) 2.2. Soon supporting VOL tokens

Chapter 4

Namespace Index

4.1 Namespace List

Here is a list of all namespaces with brief descriptions:

H5Easy	Read/dump DataSets or Attribute using a minimalistic syntax. To this end, the functions are templated, and accept:	27
HighFive		34

Chapter 5

Hierarchical Index

5.1 Class Hierarchy

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

HighFive::AllocationTime	45
HighFive::AnnotateTraits< Derivate >	46
HighFive::AnnotateTraits< DataSet >	46
HighFive::DataSet	79
HighFive::AnnotateTraits< File >	46
HighFive::File	118
HighFive::AnnotateTraits< Group >	46
HighFive::Group	136
HighFive::Caching	69
HighFive::Chunking	70
H5Easy::Compression	75
HighFive::CreateIntermediateGroup	77
HighFive::CreationOrder	78
HighFive::Deflate	102
H5Easy::DumpOptions	103
HighFive::ElementSet	108
HighFive::EstimatedLinkInfo	113
std::exception	
HighFive::Exception	115
HighFive::AttributeException	67
HighFive::DataSetException	86
HighFive::DataSpaceException	94
HighFive::DataTypeException	101
HighFive::FileException	128
HighFive::GroupException	142
HighFive::ObjectException	173
HighFive::PropertyException	178
HighFive::ReferenceException	190
HighFive::FileVersionBounds	129
HighFive::FixedLenStringArray< N >	131
HighFive::HDF5ErrMapper	143
HighFive::HyperSlab	144
HighFive::LinkCreationOrder	146
HighFive::Logger	147

HighFive::CompoundType::member_def	150
HighFive::EnumType< T >::member_def	151
HighFive::MetadataBlockSize	152
HighFive::MPIOCollectiveMetadata	153
HighFive::MPIOCollectiveMetadataRead	154
HighFive::MPIOCollectiveMetadataWrite	155
HighFive::MPIOFileAccess	156
HighFive::MpioNoCollectiveCause	160
HighFive::NodeTraits< Derivate >	161
HighFive::NodeTraits< File >	161
HighFive::File	118
HighFive::NodeTraits< Group >	161
HighFive::Group	136
HighFive::Object	169
HighFive::Attribute	62
HighFive::DataSet	79
HighFive::DataSpace	88
HighFive::DataType	96
HighFive::AtomicType< T >	49
HighFive::AtomicType< FixedLenStringArray< StrLen > >	57
HighFive::AtomicType< char[StrLen]>	54
HighFive::AtomicType< std::complex< T > >	59
HighFive::CompoundType	71
HighFive::EnumType< T >	110
HighFive::File	118
HighFive::Group	136
HighFive::PropertyListBase	183
HighFive::PropertyList< T >	180
HighFive::FileDriver	125
HighFive::MPIOFileDriver	157
HighFive::RawPropertyList< T >	185
HighFive::ObjectInfo	174
HighFive::PathTraits< Derivate >	176
HighFive::PathTraits< Attribute >	176
HighFive::Attribute	62
HighFive::PathTraits< DataSet >	176
HighFive::DataSet	79
HighFive::PathTraits< Group >	176
HighFive::Group	136
HighFive::Reference	187
HighFive::RegularHyperSlab	191
HighFive::Shuffle	197
HighFive::SilenceHDF5	197
HighFive::SliceTraits< Derivate >	198
HighFive::SliceTraits< DataSet >	198
HighFive::DataSet	79
HighFive::SliceTraits< Selection >	198
HighFive::Selection	194
HighFive::Szip	201
HighFive::UseCollectiveIO	202

Chapter 6

Class Index

6.1 Class List

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

HighFive::AllocationTime	
When are datasets allocated?	45
HighFive::AnnotateTraits< Derivate >	46
HighFive::AtomicType< T >	
Create an HDF5 DataType from a C++ type	49
HighFive::AtomicType< char[StrLen]>	54
HighFive::AtomicType< FixedLenStringArray< StrLen > >	57
HighFive::AtomicType< std::complex< T > >	59
HighFive::Attribute	
Class representing an attribute of a dataset or group	62
HighFive::AttributeException	
Exception specific to HighFive Attribute interface	67
HighFive::Caching	69
HighFive::Chunking	70
HighFive::CompoundType	
Create a compound HDF5 datatype	71
H5Easy::Compression	
Signal to set compression level for written DataSets	75
HighFive::CreateIntermediateGroup	77
HighFive::CreationOrder	78
HighFive::DataSet	
Class representing a dataset	79
HighFive::DataSetException	
Exception specific to HighFive DataSet interface	86
HighFive::DataSpace	
Class representing the space (dimensions) of a dataset	88
HighFive::DataSpaceException	
Exception specific to HighFive DataSpace interface	94
HighFive::DataType	
HDF5 Data Type	96
HighFive::DataTypeException	
Exception specific to HighFive DataType interface	101
HighFive::Deflate	102
H5Easy::DumpOptions	
Define options for dumping data	103

HighFive::ElementSet	108
HighFive::EnumType< T >	
Create a enum HDF5 datatype	110
HighFive::EstimatedLinkInfo	
Set hints as to how many links to expect and their average length	113
HighFive::Exception	
Basic HighFive Exception class	115
HighFive::File	
File class	118
HighFive::FileDriver	
File driver base concept	125
HighFive::FileException	
Exception specific to HighFive File interface	128
HighFive::FileVersionBounds	
Configure the version bounds for the file	129
HighFive::FixedLenStringArray< N >	
A structure representing a set of fixed-length strings	131
HighFive::Group	
Represents an hdf5 group	136
HighFive::GroupException	
Exception specific to HighFive Group interface	142
HighFive::HDF5ErrMapper	143
HighFive::HyperSlab	144
HighFive::LinkCreationOrder	
Track and index creation order time	146
HighFive::Logger	
A logger with supporting basic functionality	147
HighFive::CompoundType::member_def	
Use for defining a sub-type of compound type	150
HighFive::EnumType< T >::member_def	
Use for defining a member of enum type	151
HighFive::MetadataBlockSize	
Configure the metadata block size to use writing to files	152
HighFive::MPIOCollectiveMetadata	
Use collective MPI-IO for metadata read and write	153
HighFive::MPIOCollectiveMetadataRead	
Use collective MPI-IO for metadata read?	154
HighFive::MPIOCollectiveMetadataWrite	
Use collective MPI-IO for metadata write?	155
HighFive::MPIOFileAccess	
Configure MPI access for the file	156
HighFive::MPIOFileDriver	
MPIIO Driver for Parallel HDF5	157
HighFive::MpioNoCollectiveCause	
The cause for non-collective I/O	160
HighFive::NodeTraits< Derivate >	
NodeTraits: Base class for Group and File	161
HighFive::Object	169
HighFive::ObjectException	
Exception specific to HighFive Object interface	173
HighFive::ObjectInfo	
A class for accessing hdf5 objects info	174
HighFive::PathTraits< Derivate >	176
HighFive::PropertyException	
Exception specific to HighFive Property interface	178
HighFive::PropertyList< T >	
HDF5 property Lists	180

HighFive::PropertyListBase	
Base Class for Property lists, providing global default	183
HighFive::RawPropertyList< T >	185
HighFive::Reference	
An HDF5 (object) reference type	187
HighFive::ReferenceException	
Exception specific to HighFive Reference interface	190
HighFive::RegularHyperSlab	191
HighFive::Selection	
Selection: represent a view on a slice/part of a dataset	194
HighFive::Shuffle	197
HighFive::SilenceHDF5	
Utility class to disable HDF5 stack printing inside a scope	197
HighFive::SliceTraits< Derivate >	198
HighFive::Szip	201
HighFive::UseCollectiveIO	202

Chapter 7

File Index

7.1 File List

Here is a list of all files with brief descriptions:

highfive/H5Attribute.hpp	287
highfive/H5DataSet.hpp	290
highfive/H5DataSpace.hpp	292
highfive/H5DataType.hpp	294
highfive/H5Easy.hpp	300
highfive/H5Exception.hpp	323
highfive/H5File.hpp	326
highfive/H5FileDriver.hpp	328
highfive/H5Group.hpp	329
highfive/H5Object.hpp	331
highfive/H5PropertyList.hpp	334
highfive/H5Reference.hpp	342
highfive/H5Selection.hpp	344
highfive/H5Utility.hpp	346
highfive/bits/H5_definitions.hpp	203
highfive/bits/H5Annotate_traits.hpp	205
highfive/bits/H5Annotate_traits_misc.hpp	206
highfive/bits/H5Attribute_misc.hpp	209
highfive/bits/H5Converter_misc.hpp	212
highfive/bits/H5DataSet_misc.hpp	224
highfive/bits/H5Dataspace_misc.hpp	226
highfive/bits/H5DataType_misc.hpp	229
highfive/bits/H5Exception_misc.hpp	237
highfive/bits/H5File_misc.hpp	239
highfive/bits/H5FileDriver_misc.hpp	242
highfive/bits/H5Friends.hpp	243
highfive/bits/H5Iterables_misc.hpp	243
highfive/bits/H5Node_traits.hpp	245
highfive/bits/H5Node_traits_misc.hpp	248
highfive/bits/H5Object_misc.hpp	254
highfive/bits/H5Path_traits.hpp	257
highfive/bits/H5Path_traits_misc.hpp	259
highfive/bits/H5PropertyList_misc.hpp	261
highfive/bits/H5ReadWrite_misc.hpp	269
highfive/bits/H5Reference_misc.hpp	271

highfive/bits/H5Selection_misc.hpp	274
highfive/bits/H5Slice_traits.hpp	275
highfive/bits/H5Slice_traits_misc.hpp	280
highfive/bits/H5Utils.hpp	285
highfive/h5easy_bits/H5Easy_Eigen.hpp	305
highfive/h5easy_bits/H5Easy_misc.hpp	308
highfive/h5easy_bits/H5Easy_opencv.hpp	310
highfive/h5easy_bits/H5Easy_public.hpp	313
highfive/h5easy_bits/H5Easy_scalar.hpp	316
highfive/h5easy_bits/H5Easy_vector.hpp	319
highfive/h5easy_bits/H5Easy_xtensor.hpp	321

Chapter 8

Namespace Documentation

8.1 H5Easy Namespace Reference

Read/dump DataSets or Attribute using a minimalistic syntax. To this end, the functions are templated, and accept:

Classes

- class [Compression](#)
Signal to set compression level for written DataSets.
- class [DumpOptions](#)
Define options for dumping data.

Enumerations

- enum class [DumpMode](#) { [Create](#) = 0 , [Overwrite](#) = 1 }
Write mode for DataSets.
- enum class [Flush](#) { [False](#) = 0 , [True](#) = 1 }
Signal to enable/disable automatic flushing after write operations.

Functions

- `size_t getSize (const File &file, const std::string &path)`
Get the size of an existing DataSet in an open HDF5 file.
- `std::vector< size_t > getShape (const File &file, const std::string &path)`
Get the shape of an existing DataSet in an readable file.
- `template<class T >`
`DataSet dump (File &file, const std::string &path, const T &data, DumpMode mode=DumpMode::Create)`
Write object (templated) to a (new) DataSet in an open HDF5 file.
- `template<class T >`
`DataSet dump (File &file, const std::string &path, const T &data, const DumpOptions &options)`
Write object (templated) to a (new) DataSet in an open HDF5 file.
- `template<class T >`
`DataSet dump (File &file, const std::string &path, const T &data, const std::vector< size_t > &idx)`
Write a scalar to a (new, extendible) DataSet in an open HDF5 file.

- `template<class T >`
[DataSet dump](#) ([File](#) &file, const std::string &path, const T &data, const std::initializer_list< size_t > &idx)
Write a scalar to a (new, extendable) DataSet in an open HDF5 file.
- `template<class T >`
[DataSet dump](#) ([File](#) &file, const std::string &path, const T &data, const std::vector< size_t > &idx, const [DumpOptions](#) &options)
Write a scalar to a (new, extendable) DataSet in an open HDF5 file.
- `template<class T >`
[DataSet dump](#) ([File](#) &file, const std::string &path, const T &data, const std::initializer_list< size_t > &idx, const [DumpOptions](#) &options)
Write a scalar to a (new, extendable) DataSet in an open HDF5 file.
- `template<class T >`
[T load](#) (const [File](#) &file, const std::string &path, const std::vector< size_t > &idx)
Load entry {i, j, ...} from a DataSet in an open HDF5 file to a scalar.
- `template<class T >`
[T load](#) (const [File](#) &file, const std::string &path)
Load a DataSet in an open HDF5 file to an object (templated).
- `template<class T >`
[Attribute dumpAttribute](#) ([File](#) &file, const std::string &path, const std::string &key, const T &data, [DumpMode](#) mode=[DumpMode::Create](#))
Write object (templated) to a (new) Attribute in an open HDF5 file.
- `template<class T >`
[Attribute dumpAttribute](#) ([File](#) &file, const std::string &path, const std::string &key, const T &data, const [DumpOptions](#) &options)
Write object (templated) to a (new) Attribute in an open HDF5 file.
- `template<class T >`
[T loadAttribute](#) (const [File](#) &file, const std::string &path, const std::string &key)
Load a Attribute in an open HDF5 file to an object (templated).

8.1.1 Detailed Description

Read/dump DataSets or Attribute using a minimalistic syntax. To this end, the functions are templated, and accept:

- Any type accepted by [HighFive](#)
- Eigen objects
- xtensor objects
- OpenCV objects

8.1.2 Enumeration Type Documentation

8.1.2.1 DumpMode

```
enum class H5Easy::DumpMode [strong]
```

Write mode for DataSets.

Enumerator

Create	Dump only if DataSet does not exist, otherwise throw.
Overwrite	Create or overwrite if DataSet of correct shape exists, otherwise throw.

8.1.2.2 Flush

```
enum class H5Easy::Flush [strong]
```

Signal to enable/disable automatic flushing after write operations.

Enumerator

False	No automatic flushing.
True	Automatic flushing.

8.1.3 Function Documentation

8.1.3.1 dump() [1/6]

```
template<class T >
DataSet H5Easy::dump (
    File & file,
    const std::string & path,
    const T & data,
    const DumpOptions & options ) [inline]
```

Write object (templated) to a (new) DataSet in an open HDF5 file.

Parameters

<i>file</i>	opened file (has to be writeable)
<i>path</i>	path of the DataSet
<i>data</i>	the data to write (any supported type)
<i>options</i>	dump options

Returns

The newly created DataSet

8.1.3.2 dump() [2/6]

```
template<class T >
DataSet H5Easy::dump (
    File & file,
    const std::string & path,
    const T & data,
    const std::initializer_list< size_t > & idx ) [inline]
```

Write a scalar to a (new, extendable) DataSet in an open HDF5 file.

Parameters

<i>file</i>	open File (has to be writeable)
<i>path</i>	path of the DataSet
<i>data</i>	the data to write (any supported type)
<i>idx</i>	the indices to which to write

Returns

The newly created DataSet

8.1.3.3 dump() [3/6]

```
template<class T >
DataSet H5Easy::dump (
    File & file,
    const std::string & path,
    const T & data,
    const std::initializer_list< size_t > & idx,
    const DumpOptions & options ) [inline]
```

Write a scalar to a (new, extendible) DataSet in an open HDF5 file.

Parameters

<i>file</i>	opened file (has to be writeable)
<i>path</i>	path of the DataSet
<i>data</i>	the data to write (any supported type)
<i>idx</i>	the indices to which to write
<i>options</i>	dump options

Returns

The newly created DataSet

8.1.3.4 dump() [4/6]

```
template<class T >
DataSet H5Easy::dump (
    File & file,
    const std::string & path,
    const T & data,
    const std::vector< size_t > & idx ) [inline]
```

Write a scalar to a (new, extendible) DataSet in an open HDF5 file.

Parameters

<i>file</i>	opened file (has to be writeable)
<i>path</i>	path of the DataSet
<i>data</i>	the data to write (any supported type)
<i>idx</i>	the indices to which to write

Returns

The newly created DataSet

8.1.3.5 dump() [5/6]

```
template<class T >
DataSet H5Easy::dump (
    File & file,
    const std::string & path,
    const T & data,
    const std::vector< size_t > & idx,
    const DumpOptions & options ) [inline]
```

Write a scalar to a (new, extendible) DataSet in an open HDF5 file.

Parameters

<i>file</i>	opened file (has to be writeable)
<i>path</i>	path of the DataSet
<i>data</i>	the data to write (any supported type)
<i>idx</i>	the indices to which to write
<i>options</i>	dump options

Returns

The newly created DataSet

8.1.3.6 dump() [6/6]

```
template<class T >
DataSet H5Easy::dump (
    File & file,
    const std::string & path,
    const T & data,
    DumpMode mode = DumpMode::Create ) [inline]
```

Write object (templated) to a (new) DataSet in an open HDF5 file.

Parameters

<i>file</i>	opened file (has to be writeable)
<i>path</i>	path of the DataSet
<i>data</i>	the data to write (any supported type)
<i>mode</i>	write mode

Returns

The newly created DataSet

8.1.3.7 dumpAttribute() [1/2]

```
template<class T >
Attribute H5Easy::dumpAttribute (
```

```

File & file,
const std::string & path,
const std::string & key,
const T & data,
const DumpOptions & options ) [inline]

```

Write object (templated) to a (new) Attribute in an open HDF5 file.

Parameters

<i>file</i>	opened file (has to be writeable)
<i>path</i>	path of the DataSet
<i>key</i>	name of the attribute
<i>data</i>	the data to write (any supported type)
<i>options</i>	dump options

Returns

The newly created DataSet

8.1.3.8 dumpAttribute() [2/2]

```

template<class T >
Attribute H5Easy::dumpAttribute (
    File & file,
    const std::string & path,
    const std::string & key,
    const T & data,
    DumpMode mode = DumpMode::Create ) [inline]

```

Write object (templated) to a (new) Attribute in an open HDF5 file.

Parameters

<i>file</i>	opened file (has to be writeable)
<i>path</i>	path of the DataSet
<i>key</i>	name of the attribute
<i>data</i>	the data to write (any supported type)
<i>mode</i>	write mode

Returns

The newly created DataSet

8.1.3.9 getShape()

```

std::vector< size_t > H5Easy::getShape (
    const File & file,
    const std::string & path ) [inline]

```

Get the shape of an existing DataSet in an readable file.

Parameters

<i>file</i>	opened file (has to be readable)
<i>path</i>	Path of the DataSet

Returns

the shape of the DataSet

8.1.3.10 getSize()

```
size_t H5Easy::getSize (
    const File & file,
    const std::string & path ) [inline]
```

Get the size of an existing DataSet in an open HDF5 file.

Parameters

<i>file</i>	opened file (has to be readable)
<i>path</i>	path of the DataSet

Returns

Size of the DataSet

8.1.3.11 load() [1/2]

```
template<class T >
T H5Easy::load (
    const File & file,
    const std::string & path ) [inline]
```

Load a DataSet in an open HDF5 file to an object (templated).

Parameters

<i>file</i>	opened file (has to be writeable)
<i>path</i>	path of the DataSet

Returns

The read data

8.1.3.12 load() [2/2]

```
template<class T >
T H5Easy::load (
```

```
const File & file,
const std::string & path,
const std::vector< size_t > & idx ) [inline]
```

Load entry {*i*, *j*, ...} from a DataSet in an open HDF5 file to a scalar.

Parameters

<i>file</i>	opened file (has to be writeable)
<i>idx</i>	the indices to load
<i>path</i>	path of the DataSet

Returns

The read data

8.1.3.13 loadAttribute()

```
template<class T >
T H5Easy::loadAttribute (
    const File & file,
    const std::string & path,
    const std::string & key ) [inline]
```

Load a Attribute in an open HDF5 file to an object (templated).

Parameters

<i>file</i>	opened file (has to be writeable)
<i>path</i>	path of the DataSet
<i>key</i>	name of the attribute

Returns

The read data

8.2 HighFive Namespace Reference

Classes

- class [AllocationTime](#)
When are datasets allocated?
- class [AnnotateTraits](#)
- class [AtomicType](#)
create an HDF5 [DataType](#) from a C++ type
- class [AtomicType< char\[StrLen\]>](#)
- class [AtomicType< FixedLenStringArray< StrLen > >](#)
- class [AtomicType< std::complex< T > >](#)

- class [Attribute](#)
Class representing an attribute of a dataset or group.
- class [AttributeException](#)
Exception specific to [HighFive Attribute](#) interface.
- class [Caching](#)
- class [Chunking](#)
- class [CompoundType](#)
Create a compound HDF5 datatype.
- class [CreateIntermediateGroup](#)
- struct [CreationOrder](#)
- class [DataSet](#)
Class representing a dataset.
- class [DataSetException](#)
Exception specific to [HighFive DataSet](#) interface.
- class [DataSpace](#)
Class representing the space (dimensions) of a dataset.
- class [DataSpaceException](#)
Exception specific to [HighFive DataSpace](#) interface.
- class [DataType](#)
HDF5 Data Type.
- class [DataTypeException](#)
Exception specific to [HighFive DataType](#) interface.
- class [Deflate](#)
- class [ElementSet](#)
- class [EnumType](#)
Create a enum HDF5 datatype.
- class [EstimatedLinkInfo](#)
Set hints as to how many links to expect and their average length.
- class [Exception](#)
Basic [HighFive Exception](#) class.
- class [File](#)
File class.
- class [FileDriver](#)
file driver base concept
- class [FileException](#)
Exception specific to [HighFive File](#) interface.
- class [FileVersionBounds](#)
Configure the version bounds for the file.
- class [FixedLenStringArray](#)
A structure representing a set of fixed-length strings.
- class [Group](#)
Represents an hdf5 group.
- class [GroupException](#)
Exception specific to [HighFive Group](#) interface.
- struct [HDF5ErrMapper](#)
- class [HyperSlab](#)
- class [LinkCreationOrder](#)
Track and index creation order time.
- class [Logger](#)
A logger with supporting basic functionality.
- class [MetadataBlockSize](#)

- Configure the metadata block size to use writing to files.*
- class [MPIOCollectiveMetadata](#)
 - Use collective MPI-IO for metadata read and write.*
- class [MPIOCollectiveMetadataRead](#)
 - Use collective MPI-IO for metadata read?*
- class [MPIOCollectiveMetadataWrite](#)
 - Use collective MPI-IO for metadata write?*
- class [MPIOFileAccess](#)
 - Configure MPI access for the file.*
- class [MPIOFileDriver](#)
 - MPIIO Driver for Parallel HDF5.*
- class [MpioNoCollectiveCause](#)
 - The cause for non-collective I/O.*
- class [NodeTraits](#)
 - NodeTraits: Base class for [Group](#) and [File](#).*
- class [Object](#)
- class [ObjectException](#)
 - Exception specific to [HighFive Object](#) interface.*
- class [ObjectInfo](#)
 - A class for accessing hdf5 objects info.*
- class [PathTraits](#)
- class [PropertyException](#)
 - Exception specific to [HighFive Property](#) interface.*
- class [PropertyList](#)
 - HDF5 property Lists.*
- class [PropertyListBase](#)
 - Base Class for Property lists, providing global default.*
- class [RawPropertyList](#)
- class [Reference](#)
 - An HDF5 (object) reference type.*
- class [ReferenceException](#)
 - Exception specific to [HighFive Reference](#) interface.*
- struct [RegularHyperSlab](#)
- class [Selection](#)
 - Selection: represent a view on a slice/part of a dataset.*
- class [Shuffle](#)
- class [SilenceHDF5](#)
 - Utility class to disable HDF5 stack printing inside a scope.*
- class [SliceTraits](#)
- class [Szip](#)
- class [UseCollectiveIO](#)

Typedefs

- `template<typename T>`
 - using `unqualified_t` = `typename std::remove_const< typename std::remove_reference< T >::type >::type`
- using `float16_t` = `half_float::half`
- using `ObjectCreateProps` = `PropertyList< PropertyType::OBJECT_CREATE >`
- using `FileCreateProps` = `PropertyList< PropertyType::FILE_CREATE >`
- using `FileAccessProps` = `PropertyList< PropertyType::FILE_ACCESS >`
- using `DataSetCreateProps` = `PropertyList< PropertyType::DATASET_CREATE >`

- using [DataSetAccessProps](#) = [PropertyList](#)< [PropertyType::DATASET_ACCESS](#) >
- using [DataTransferProps](#) = [PropertyList](#)< [PropertyType::DATASET_XFER](#) >
- using [GroupCreateProps](#) = [PropertyList](#)< [PropertyType::GROUP_CREATE](#) >
- using [GroupAccessProps](#) = [PropertyList](#)< [PropertyType::GROUP_ACCESS](#) >
- using [DataTypeCreateProps](#) = [PropertyList](#)< [PropertyType::DATATYPE_CREATE](#) >
- using [DataTypeAccessProps](#) = [PropertyList](#)< [PropertyType::DATATYPE_ACCESS](#) >
- using [StringCreateProps](#) = [PropertyList](#)< [PropertyType::STRING_CREATE](#) >
- using [AttributeCreateProps](#) = [PropertyList](#)< [PropertyType::ATTRIBUTE_CREATE](#) >
- using [ObjectCopyProps](#) = [PropertyList](#)< [PropertyType::OBJECT_COPY](#) >
- using [LinkCreateProps](#) = [PropertyList](#)< [PropertyType::LINK_CREATE](#) >
- using [LinkAccessProps](#) = [PropertyList](#)< [PropertyType::LINK_ACCESS](#) >

Enumerations

- enum class [IndexType](#) : [std::underlying_type](#)< [H5_index_t](#) >::type { [NAME](#) = [H5_INDEX_NAME](#) , [CRT_ORDER](#) = [H5_INDEX_CRT_ORDER](#) }
- enum class [LinkType](#) { [Hard](#) , [Soft](#) , [External](#) , [Other](#) }
The possible types of group entries (link concept)
- enum class [DataTypeClass](#) {
 [Time](#) = 1 << 1 , [Integer](#) = 1 << 2 , [Float](#) = 1 << 3 , [String](#) = 1 << 4 ,
 [BitField](#) = 1 << 5 , [Opaque](#) = 1 << 6 , [Compound](#) = 1 << 7 , [Reference](#) = 1 << 8 ,
 [Enum](#) = 1 << 9 , [VarLen](#) = 1 << 10 , [Array](#) = 1 << 11 , [Invalid](#) = 0 }
Enum of Fundamental data classes.
- enum class [ObjectType](#) {
 [File](#) , [Group](#) , [UserDataType](#) , [DataSpace](#) ,
 [Dataset](#) , [Attribute](#) , [Other](#) }
Enum of the types of objects (H5O api)
- enum class [PropertyType](#) : int {
 [OBJECT_CREATE](#) , [FILE_CREATE](#) , [FILE_ACCESS](#) , [DATASET_CREATE](#) ,
 [DATASET_ACCESS](#) , [DATASET_XFER](#) , [GROUP_CREATE](#) , [GROUP_ACCESS](#) ,
 [DATATYPE_CREATE](#) , [DATATYPE_ACCESS](#) , [STRING_CREATE](#) , [ATTRIBUTE_CREATE](#) ,
 [OBJECT_COPY](#) , [LINK_CREATE](#) , [LINK_ACCESS](#) }
Types of property lists.
- enum class [LogSeverity](#) { [Debug](#) = [HIGHFIVE_LOG_LEVEL_DEBUG](#) , [Info](#) = [HIGHFIVE_LOG_LEVEL_INFO](#) , [Warn](#) = [HIGHFIVE_LOG_LEVEL_WARN](#) , [Error](#) = [HIGHFIVE_LOG_LEVEL_ERROR](#) }

Functions

- [size_t compute_total_size](#) (const [std::vector](#)< [size_t](#) > &dims)
- [EnumType](#)< [details::Boolean](#) > [create_enum_boolean](#) ()
- [size_t find_first_atomic_member_size](#) ([hid_t](#) hid)
- [template](#)<typename T >
 [DataType create_datatype](#) ()
Create a [DataType](#) instance representing type T.
- [template](#)<typename T >
 [DataType create_and_check_datatype](#) ()
Create a [DataType](#) instance representing type T and perform a sanity check on its size.
- [std::vector](#)< [hsize_t](#) > [toHDF5SizeVector](#) (const [std::vector](#)< [size_t](#) > &from)
- [std::vector](#)< [size_t](#) > [toSTLSizeVector](#) (const [std::vector](#)< [hsize_t](#) > &from)
- [DataTypeClass](#) operator| ([DataTypeClass](#) lhs, [DataTypeClass](#) rhs)
- [DataTypeClass](#) operator& ([DataTypeClass](#) lhs, [DataTypeClass](#) rhs)
- [std::string to_string](#) ([LogSeverity](#) severity)

- void `default_logging_callback` (`LogSeverity` severity, const std::string &message, const std::string &file, int line)
- `Logger` & `get_global_logger` ()
Obtain a reference to the logger used by `HighFive`.
- void `register_logging_callback` (`Logger::callback_type` cb)
Sets the callback that's used by the logger.

8.2.1 Typedef Documentation

8.2.1.1 AttributeCreateProps

```
using HighFive::AttributeCreateProps = typedef PropertyList<PropertyType::ATTRIBUTE_CREATE>
```

8.2.1.2 DataSetAccessProps

```
using HighFive::DataSetAccessProps = typedef PropertyList<PropertyType::DATASET_ACCESS>
```

8.2.1.3 DataSetCreateProps

```
using HighFive::DataSetCreateProps = typedef PropertyList<PropertyType::DATASET_CREATE>
```

8.2.1.4 DataTransferProps

```
using HighFive::DataTransferProps = typedef PropertyList<PropertyType::DATASET_XFER>
```

8.2.1.5 DataTypeAccessProps

```
using HighFive::DataTypeAccessProps = typedef PropertyList<PropertyType::DATATYPE_ACCESS>
```

8.2.1.6 DataTypeCreateProps

```
using HighFive::DataTypeCreateProps = typedef PropertyList<PropertyType::DATATYPE_CREATE>
```

8.2.1.7 FileAccessProps

```
using HighFive::FileAccessProps = typedef PropertyList<PropertyType::FILE_ACCESS>
```

8.2.1.8 FileCreateProps

```
using HighFive::FileCreateProps = typedef PropertyList<PropertyType::FILE_CREATE>
```

8.2.1.9 float16_t

```
using HighFive::float16_t = typedef half_float::half
```

8.2.1.10 GroupAccessProps

```
using HighFive::GroupAccessProps = typedef PropertyList<PropertyType::GROUP_ACCESS>
```

8.2.1.11 GroupCreateProps

```
using HighFive::GroupCreateProps = typedef PropertyList<PropertyType::GROUP_CREATE>
```

8.2.1.12 LinkAccessProps

```
using HighFive::LinkAccessProps = typedef PropertyList<PropertyType::LINK_ACCESS>
```

8.2.1.13 LinkCreateProps

```
using HighFive::LinkCreateProps = typedef PropertyList<PropertyType::LINK_CREATE>
```

8.2.1.14 ObjectCopyProps

```
using HighFive::ObjectCopyProps = typedef PropertyList<PropertyType::OBJECT_COPY>
```

8.2.1.15 ObjectCreateProps

```
using HighFive::ObjectCreateProps = typedef PropertyList<PropertyType::OBJECT_CREATE>
```

8.2.1.16 StringCreateProps

```
using HighFive::StringCreateProps = typedef PropertyList<PropertyType::STRING_CREATE>
```

8.2.1.17 unqualified_t

```
template<typename T >  
using HighFive::unqualified_t = typedef typename std::remove_const<typename std::remove_↵  
reference<T>::type>::type
```

8.2.2 Enumeration Type Documentation

8.2.2.1 DataTypeClass

```
enum class HighFive::DataTypeClass [strong]
```

Enum of Fundamental data classes.

Enumerator

Time	
Integer	
Float	
String	
BitField	
Opaque	
Compound	
Reference	
Enum	
VarLen	
Array	
Invalid	

8.2.2.2 IndexType

```
enum class HighFive::IndexType : std::underlying_type< H5_index_t >::type [strong]
```

Enumerator

NAME	
CRT_ORDER	

8.2.2.3 LinkType

```
enum class HighFive::LinkType [strong]
```

The possible types of group entries (link concept)

Enumerator

Hard	
Soft	
External	
Other	

8.2.2.4 LogSeverity

```
enum class HighFive::LogSeverity [strong]
```

Enumerator

Debug	
Info	
Warn	
Error	

8.2.2.5 ObjectType

```
enum class HighFive::ObjectType [strong]
```

Enum of the types of objects (H5O api)

Enumerator

File	
Group	
UserDataTypes	
DataSpace	
Dataset	
Attribute	
Other	

8.2.2.6 PropertyType

```
enum class HighFive::PropertyType : int [strong]
```

Types of property lists.

Enumerator

OBJECT_CREATE	
FILE_CREATE	
FILE_ACCESS	
DATASET_CREATE	
DATASET_ACCESS	
DATASET_XFER	
GROUP_CREATE	
GROUP_ACCESS	
DATATYPE_CREATE	
DATATYPE_ACCESS	
STRING_CREATE	
ATTRIBUTE_CREATE	
OBJECT_COPY	
LINK_CREATE	
LINK_ACCESS	

8.2.3 Function Documentation

8.2.3.1 compute_total_size()

```
size_t HighFive::compute_total_size (  
    const std::vector< size_t > & dims ) [inline]
```

8.2.3.2 create_and_check_datatype()

```
template<typename T >
DataType HighFive::create_and_check_datatype ( ) [inline]
```

Create a [DataType](#) instance representing type T and perform a sanity check on its size.

8.2.3.3 create_datatype()

```
template<typename T >
DataType HighFive::create_datatype ( ) [inline]
```

Create a [DataType](#) instance representing type T.

8.2.3.4 create_enum_boolean()

```
EnumType< details::Boolean > HighFive::create_enum_boolean ( ) [inline]
```

8.2.3.5 default_logging_callback()

```
void HighFive::default_logging_callback (
    LogSeverity severity,
    const std::string & message,
    const std::string & file,
    int line ) [inline]
```

8.2.3.6 find_first_atomic_member_size()

```
size_t HighFive::find_first_atomic_member_size (
    hid_t hid ) [inline]
```

8.2.3.7 get_global_logger()

```
Logger & HighFive::get_global_logger ( ) [inline]
```

Obtain a reference to the logger used by [HighFive](#).

This uses a Meyers singleton, to ensure that the global logger is initialized with a safe default logger, before it is used.

Note: You probably don't need to call this function explicitly.

8.2.3.8 operator&()

```
DataTypeClass HighFive::operator& (
    DataTypeClass lhs,
    DataTypeClass rhs ) [inline]
```


8.2.3.9 operator" | ()

```
DataTypeClass HighFive::operator| (
    DataTypeClass lhs,
    DataTypeClass rhs ) [inline]
```

8.2.3.10 register_logging_callback()

```
void HighFive::register_logging_callback (
    Logger::callback_type cb ) [inline]
```

Sets the callback that's used by the logger.

8.2.3.11 to_string()

```
std::string HighFive::to_string (
    LogSeverity severity ) [inline]
```

8.2.3.12 toHDF5SizeVector()

```
std::vector< hsize_t > HighFive::toHDF5SizeVector (
    const std::vector< size_t > & from ) [inline]
```

8.2.3.13 toSTLSizeVector()

```
std::vector< size_t > HighFive::toSTLSizeVector (
    const std::vector< hsize_t > & from ) [inline]
```


Chapter 9

Class Documentation

9.1 HighFive::AllocationTime Class Reference

When are datasets allocated?

```
#include <H5PropertyList.hpp>
```

Public Member Functions

- [AllocationTime](#) (H5D_alloc_time_t alloc_time)
- [AllocationTime](#) (const [DataSetCreateProps](#) &dcpl)
- H5D_alloc_time_t [getAllocationTime](#) ()

9.1.1 Detailed Description

When are datasets allocated?

The precise time of when HDF5 requests space to store the dataset can be configured. Please, consider the upstream documentation for `H5Pset_alloc_time`.

9.1.2 Constructor & Destructor Documentation

9.1.2.1 AllocationTime() [1/2]

```
HighFive::AllocationTime::AllocationTime (  
    H5D_alloc_time_t alloc_time ) [inline], [explicit]
```

9.1.2.2 AllocationTime() [2/2]

```
HighFive::AllocationTime::AllocationTime (  
    const DataSetCreateProps & dcpl ) [inline], [explicit]
```

9.1.3 Member Function Documentation

9.1.3.1 getAllocationTime()

```
H5D_alloc_time_t HighFive::AllocationTime::getAllocationTime ( ) [inline]
```

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

- [highfive/H5PropertyList.hpp](#)
- [highfive/bits/H5PropertyList_misc.hpp](#)

9.2 HighFive::AnnotateTraits< Derivate > Class Template Reference

```
#include <H5Annotate_traits.hpp>
```

Public Member Functions

- [Attribute createAttribute](#) (const std::string &attribute_name, const [DataSpace](#) &space, const [DataType](#) &type)
create a new attribute with the name attribute_name
- template<typename Type >
[Attribute createAttribute](#) (const std::string &attribute_name, const [DataSpace](#) &space)
createAttribute create a new attribute on the current dataset with size specified by space
- template<typename T >
[Attribute createAttribute](#) (const std::string &attribute_name, const T &data)
createAttribute create a new attribute on the current dataset and write to it, inferring the [DataSpace](#) from data.
- void [deleteAttribute](#) (const std::string &attribute_name)
deleteAttribute let you delete an attribute by its name.
- [Attribute getAttribute](#) (const std::string &attribute_name) const
open an existing attribute with the name attribute_name
- size_t [getNumberAttributes](#) () const
return the number of attributes of the node / group
- std::vector< std::string > [listAttributeNames](#) () const
list all attribute name of the node / group
- bool [hasAttribute](#) (const std::string &attr_name) const
checks an attribute exists

9.2.1 Member Function Documentation

9.2.1.1 createAttribute() [1/3]

```
template<typename Derivate >
template<typename Type >
Attribute HighFive::AnnotateTraits< Derivate >::createAttribute (
    const std::string & attribute_name,
    const DataSpace & space ) [inline]
```

[createAttribute](#) create a new attribute on the current dataset with size specified by space

Parameters

<i>attribute_name</i>	identifier of the attribute
<i>space</i>	Associated DataSpace informations

Returns

[Attribute Object](#)

9.2.1.2 createAttribute() [2/3]

```
template<typename Derivate >
Attribute HighFive::AnnotateTraits< Derivate >::createAttribute (
    const std::string & attribute_name,
    const DataSpace & space,
    const DataType & type ) [inline]
```

create a new attribute with the name *attribute_name*

Parameters

<i>attribute_name</i>	identifier of the attribute
<i>space</i>	Associated DataSpace
<i>type</i>	

Returns

the attribute object

9.2.1.3 createAttribute() [3/3]

```
template<typename Derivate >
template<typename T >
Attribute HighFive::AnnotateTraits< Derivate >::createAttribute (
    const std::string & attribute_name,
    const T & data ) [inline]
```

createAttribute create a new attribute on the current dataset and write to it, inferring the [DataSpace](#) from data.

Parameters

<i>attribute_name</i>	identifier of the attribute
<i>data</i>	Associated data to write, must support DataSpace::From , see DataSpace for more information

Returns

[Attribute Object](#)

9.2.1.4 deleteAttribute()

```
template<typename Derivate >
void HighFive::AnnotateTraits< Derivate >::deleteAttribute (
    const std::string & attribute_name ) [inline]
```

deleteAttribute let you delete an attribute by its name.

Parameters

<i>attribute_name</i>	identifier of the attribute
-----------------------	-----------------------------

9.2.1.5 getAttribute()

```
template<typename Derivate >
Attribute HighFive::AnnotateTraits< Derivate >::getAttribute (
    const std::string & attribute_name ) const [inline]
```

open an existing attribute with the name attribute_name

Parameters

<i>attribute_name</i>	identifier of the attribute
-----------------------	-----------------------------

Returns

the attribute object

9.2.1.6 getNumberAttributes()

```
template<typename Derivate >
size_t HighFive::AnnotateTraits< Derivate >::getNumberAttributes [inline]
```

return the number of attributes of the node / group

Returns

number of attributes

9.2.1.7 hasAttribute()

```
template<typename Derivate >
bool HighFive::AnnotateTraits< Derivate >::hasAttribute (
    const std::string & attr_name ) const [inline]
```

checks an attribute exists

Returns

number of attributes

9.2.1.8 listAttributeNames()

```
template<typename Derivate >  
std::vector< std::string > HighFive::AnnotateTraits< Derivate >::listAttributeNames [inline]
```

list all attribute name of the node / group

Returns

number of attributes

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

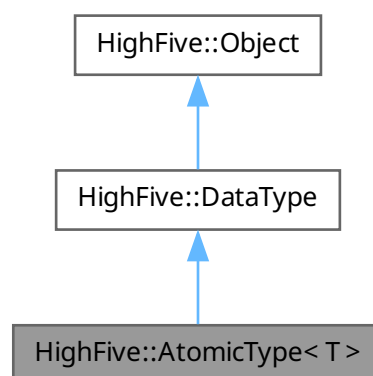
- [highfive/bits/H5_definitions.hpp](#)
- [highfive/bits/H5Annotate_traits.hpp](#)
- [highfive/bits/H5Annotate_traits_misc.hpp](#)

9.3 HighFive::AtomicType< T > Class Template Reference

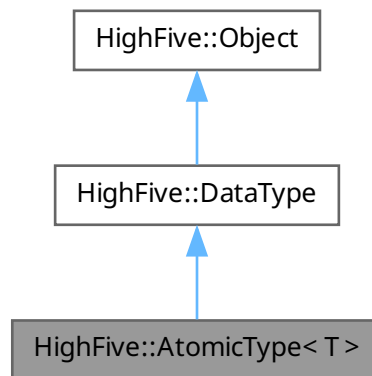
create an HDF5 [DataType](#) from a C++ type

```
#include <H5DataType.hpp>
```

Inheritance diagram for HighFive::AtomicType< T >:



Collaboration diagram for HighFive::AtomicType< T >:



Public Types

- using `basic_type` = T

Public Member Functions

- `AtomicType` ()
- `AtomicType` ()
- `AtomicType` ()
- `AtomicType` ()
- `AtomicType` ()
- `AtomicType` ()
- `AtomicType` ()
- `AtomicType` ()
- `AtomicType` ()
- `AtomicType` ()
- `AtomicType` ()
- `AtomicType` ()
- `AtomicType` ()
- `AtomicType` ()
- `AtomicType` ()
- `AtomicType` ()
- `AtomicType` ()
- `AtomicType` ()
- `AtomicType` ()

Public Member Functions inherited from HighFive::DataType

- bool `operator==` (const `DataType` &other) const
- bool `operator!=` (const `DataType` &other) const
- `DataTypeClass getClass` () const
Return the fundamental type.
- `size_t getSize` () const
Returns the length (in bytes) of this type elements.
- `std::string string` () const
Returns a friendly description of the type (e.g. Float32)
- bool `isVariableStr` () const
Returns whether the type is a variable-length string.
- bool `isFixedLenStr` () const
Returns whether the type is a fixed-length string.
- bool `empty` () const noexcept
Check the `DataType` was default constructed. Such value might represent auto-detection of the datatype from a buffer.
- bool `isReference` () const
Returns whether the type is a [Reference](#).
- `DataTypeCreateProps getCreatePropertyList` () const
Get the list of properties for creation of this `DataType`.

Public Member Functions inherited from HighFive::Object

- `Object` (`Object` &&other) noexcept
- `~Object` ()
- bool `isValid` () const noexcept
isValid
- `hid_t getId` () const noexcept
getId
- `ObjectInfo getInfo` () const
Retrieve several infos about the current object (address, dates, etc)
- `ObjectType getType` () const
Gets the fundamental type of the object (dataset, group, etc)
- bool `operator==` (const `Object` &other) const noexcept

Additional Inherited Members

Protected Member Functions inherited from HighFive::DataType

- `Object` (`Object` &&other) noexcept
- `Object` ()
- `Object` (const `Object` &other)
- `Object` (hid_t)

Protected Member Functions inherited from HighFive::Object

- `Object` ()
- `Object` (const `Object` &other)
- `Object` (hid_t)
- `Object & operator=` (const `Object` &other)

Protected Attributes inherited from [HighFive::Object](#)

- [hid_t _hid](#)

9.3.1 Detailed Description

```
template<typename T>
class HighFive::AtomicType< T >
```

create an HDF5 [DataType](#) from a C++ type

Support only basic data type

9.3.2 Member Typedef Documentation

9.3.2.1 `basic_type`

```
template<typename T >
using HighFive::AtomicType< T >::basic_type = T
```

9.3.3 Constructor & Destructor Documentation

9.3.3.1 `AtomicType()` [1/18]

```
template<typename T >
HighFive::AtomicType< T >::AtomicType
```

9.3.3.2 `AtomicType()` [2/18]

```
HighFive::AtomicType< char >::AtomicType ( ) [inline]
```

9.3.3.3 `AtomicType()` [3/18]

```
HighFive::AtomicType< signedchar >::AtomicType ( ) [inline]
```

9.3.3.4 `AtomicType()` [4/18]

```
HighFive::AtomicType< unsignedchar >::AtomicType ( ) [inline]
```

9.3.3.5 `AtomicType()` [5/18]

```
HighFive::AtomicType< short >::AtomicType ( ) [inline]
```

9.3.3.6 AtomicType() [6/18]

```
HighFive::AtomicType< unsignedshort >::AtomicType ( ) [inline]
```

9.3.3.7 AtomicType() [7/18]

```
HighFive::AtomicType< int >::AtomicType ( ) [inline]
```

9.3.3.8 AtomicType() [8/18]

```
HighFive::AtomicType< unsigned >::AtomicType ( ) [inline]
```

9.3.3.9 AtomicType() [9/18]

```
HighFive::AtomicType< long >::AtomicType ( ) [inline]
```

9.3.3.10 AtomicType() [10/18]

```
HighFive::AtomicType< unsignedlong >::AtomicType ( ) [inline]
```

9.3.3.11 AtomicType() [11/18]

```
HighFive::AtomicType< longlong >::AtomicType ( ) [inline]
```

9.3.3.12 AtomicType() [12/18]

```
HighFive::AtomicType< unsignedlonglong >::AtomicType ( ) [inline]
```

9.3.3.13 AtomicType() [13/18]

```
HighFive::AtomicType< float16_t >::AtomicType ( ) [inline]
```

9.3.3.14 AtomicType() [14/18]

```
HighFive::AtomicType< float >::AtomicType ( ) [inline]
```

9.3.3.15 AtomicType() [15/18]

```
HighFive::AtomicType< double >::AtomicType ( ) [inline]
```

9.3.3.16 AtomicType() [16/18]

```
HighFive::AtomicType< longdouble >::AtomicType ( ) [inline]
```

9.3.3.17 AtomicType() [17/18]

```
HighFive::AtomicType< std::string >::AtomicType ( ) [inline]
```

9.3.3.18 AtomicType() [18/18]

```
HighFive::AtomicType< Reference >::AtomicType ( ) [inline]
```

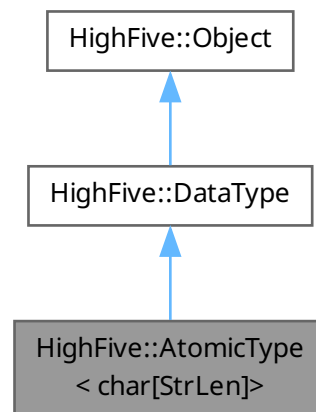
The documentation for this class was generated from the following files:

- [highfive/bits/H5_definitions.hpp](#)
- [highfive/H5DataType.hpp](#)
- [highfive/bits/H5DataType_misc.hpp](#)

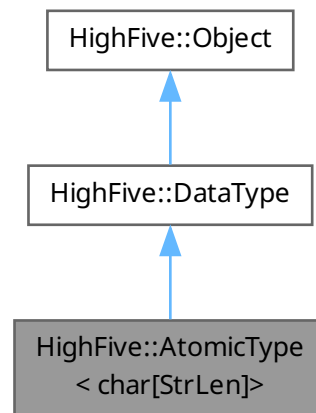
9.4 HighFive::AtomicType< char[StrLen]> Class Template Reference

```
#include <H5DataType_misc.hpp>
```

Inheritance diagram for HighFive::AtomicType< char[StrLen]>:



Collaboration diagram for HighFive::AtomicType< char[StrLen]>:



Public Member Functions

- [AtomicType](#) ()

Public Member Functions inherited from [HighFive::DataType](#)

- bool [operator==](#) (const [DataType](#) &other) const
- bool [operator!=](#) (const [DataType](#) &other) const
- [DataTypeClass](#) [getClass](#) () const
Return the fundamental type.
- size_t [getSize](#) () const
Returns the length (in bytes) of this type elements.
- std::string [string](#) () const
Returns a friendly description of the type (e.g. Float32)
- bool [isVariableStr](#) () const
Returns whether the type is a variable-length string.
- bool [isFixedLenStr](#) () const
Returns whether the type is a fixed-length string.
- bool [empty](#) () const noexcept
Check the [DataType](#) was default constructed. Such value might represent auto-detection of the datatype from a buffer.
- bool [isReference](#) () const
Returns whether the type is a [Reference](#).
- [DataTypeCreateProps](#) [getCreatePropertyList](#) () const
Get the list of properties for creation of this [DataType](#).

Public Member Functions inherited from [HighFive::Object](#)

- [Object](#) ([Object](#) &&other) noexcept
- [~Object](#) ()
- bool [isValid](#) () const noexcept
isValid
- hid_t [getIid](#) () const noexcept
getIid
- [ObjectInfo](#) [getInfo](#) () const
Retrieve several infos about the current object (address, dates, etc)
- [ObjectType](#) [getType](#) () const
Gets the fundamental type of the object (dataset, group, etc)
- bool [operator==](#) (const [Object](#) &other) const noexcept

Additional Inherited Members

Protected Member Functions inherited from [HighFive::DataType](#)

- [Object](#) ([Object](#) &&other) noexcept
- [Object](#) ()
- [Object](#) (const [Object](#) &other)
- [Object](#) (hid_t)

Protected Member Functions inherited from [HighFive::Object](#)

- [Object](#) ()
- [Object](#) (const [Object](#) &other)
- [Object](#) (hid_t)
- [Object](#) & [operator=](#) (const [Object](#) &other)

Protected Attributes inherited from [HighFive::Object](#)

- hid_t [_hid](#)

9.4.1 Constructor & Destructor Documentation

9.4.1.1 AtomicType()

```
template<size_t StrLen>
HighFive::AtomicType< char[StrLen]>::AtomicType ( ) [inline]
```

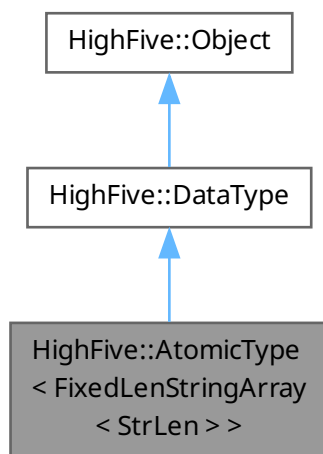
The documentation for this class was generated from the following file:

- [highfive/bits/H5DataType_misc.hpp](#)

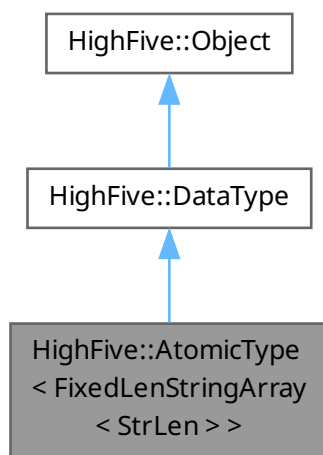
9.5 HighFive::AtomicType< FixedLenStringArray< StrLen > > Class Template Reference

```
#include <H5DataType_misc.hpp>
```

Inheritance diagram for HighFive::AtomicType< FixedLenStringArray< StrLen > >:



Collaboration diagram for HighFive::AtomicType< FixedLenStringArray< StrLen > >:



Public Member Functions

- [AtomicType](#) ()

Public Member Functions inherited from [HighFive::DataType](#)

- `bool operator== (const DataType &other) const`
- `bool operator!= (const DataType &other) const`
- `DataTypeClass getClass () const`
Return the fundamental type.
- `size_t getSize () const`
Returns the length (in bytes) of this type elements.
- `std::string string () const`
Returns a friendly description of the type (e.g. `Float32`)
- `bool isVariableStr () const`
Returns whether the type is a variable-length string.
- `bool isFixedLenStr () const`
Returns whether the type is a fixed-length string.
- `bool empty () const noexcept`
Check the [DataType](#) was default constructed. Such value might represent auto-detection of the datatype from a buffer.
- `bool isReference () const`
Returns whether the type is a [Reference](#).
- `DataTypeCreateProps getCreatePropertyList () const`
Get the list of properties for creation of this [DataType](#).

Public Member Functions inherited from [HighFive::Object](#)

- `Object (Object &&other) noexcept`
- `~Object ()`
- `bool isValid () const noexcept`
isValid
- `hid_t getIid () const noexcept`
getIid
- `ObjectInfo getInfo () const`
Retrieve several infos about the current object (address, dates, etc)
- `ObjectType getType () const`
Gets the fundamental type of the object (dataset, group, etc)
- `bool operator== (const Object &other) const noexcept`

Additional Inherited Members

Protected Member Functions inherited from [HighFive::DataType](#)

- `Object (Object &&other) noexcept`
- `Object ()`
- `Object (const Object &other)`
- `Object (hid_t)`

Protected Member Functions inherited from [HighFive::Object](#)

- `Object ()`
- `Object (const Object &other)`
- `Object (hid_t)`
- `Object & operator= (const Object &other)`

Protected Attributes inherited from [HighFive::Object](#)

- [hid_t _hid](#)

9.5.1 Constructor & Destructor Documentation

9.5.1.1 AtomicType()

```
template<size_t StrLen>  
HighFive::AtomicType< FixedLenStringArray< StrLen > >::AtomicType ( ) [inline]
```

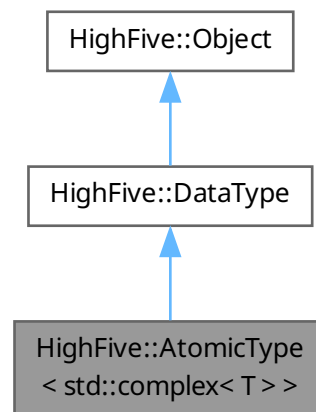
The documentation for this class was generated from the following file:

- [highfive/bits/H5DataType_misc.hpp](#)

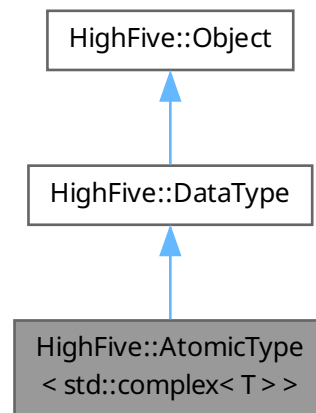
9.6 HighFive::AtomicType< std::complex< T > > Class Template Reference

```
#include <H5DataType_misc.hpp>
```

Inheritance diagram for HighFive::AtomicType< std::complex< T > >:



Collaboration diagram for `HighFive::AtomicType< std::complex< T > >`:



Public Member Functions

- [AtomicType](#) ()

Public Member Functions inherited from [HighFive::DataType](#)

- bool [operator==](#) (const [DataType](#) &other) const
- bool [operator!=](#) (const [DataType](#) &other) const
- [DataTypeClass](#) [getClass](#) () const
Return the fundamental type.
- size_t [getSize](#) () const
Returns the length (in bytes) of this type elements.
- std::string [string](#) () const
Returns a friendly description of the type (e.g. Float32)
- bool [isVariableStr](#) () const
Returns whether the type is a variable-length string.
- bool [isFixedLenStr](#) () const
Returns whether the type is a fixed-length string.
- bool [empty](#) () const noexcept
Check the [DataType](#) was default constructed. Such value might represent auto-detection of the datatype from a buffer.
- bool [isReference](#) () const
Returns whether the type is a [Reference](#).
- [DataTypeCreateProps](#) [getCreatePropertyList](#) () const
Get the list of properties for creation of this [DataType](#).

Public Member Functions inherited from [HighFive::Object](#)

- [Object](#) ([Object](#) &&other) noexcept
- [~Object](#) ()
- bool [isValid](#) () const noexcept
isValid
- hid_t [getId](#) () const noexcept
getId
- [ObjectInfo](#) [getInfo](#) () const
Retrieve several infos about the current object (address, dates, etc)
- [ObjectType](#) [getType](#) () const
Gets the fundamental type of the object (dataset, group, etc)
- bool [operator==](#) (const [Object](#) &other) const noexcept

Additional Inherited Members**Protected Member Functions inherited from [HighFive::DataType](#)**

- [Object](#) ([Object](#) &&other) noexcept
- [Object](#) ()
- [Object](#) (const [Object](#) &other)
- [Object](#) (hid_t)

Protected Member Functions inherited from [HighFive::Object](#)

- [Object](#) ()
- [Object](#) (const [Object](#) &other)
- [Object](#) (hid_t)
- [Object](#) & [operator=](#) (const [Object](#) &other)

Protected Attributes inherited from [HighFive::Object](#)

- hid_t [_hid](#)

9.6.1 Constructor & Destructor Documentation**9.6.1.1 AtomicType()**

```
template<typename T >
HighFive::AtomicType< std::complex< T > >::AtomicType ( ) [inline]
```

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

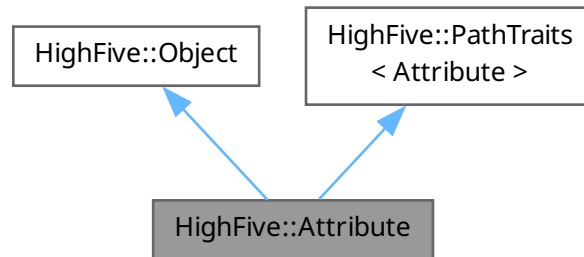
- highfive/bits/[H5DataType_misc.hpp](#)

9.7 HighFive::Attribute Class Reference

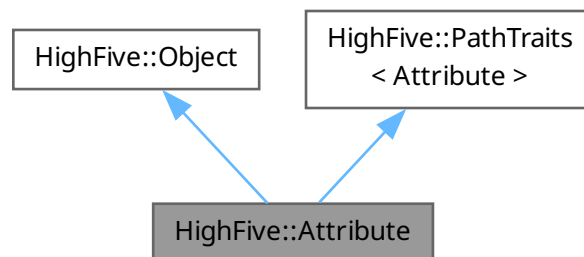
Class representing an attribute of a dataset or group.

```
#include <H5Attribute.hpp>
```

Inheritance diagram for HighFive::Attribute:



Collaboration diagram for HighFive::Attribute:



Public Member Functions

- `std::string getName () const`
return the name of the current attribute
- `size_t getStorageSize () const`
- `DataType getDataType () const`
getDataType
- `DataSpace getSpace () const`
getSpace
- `DataSpace getMemSpace () const`
getMemSpace

- `template<typename T >`
`T read () const`
Return the attribute.
- `template<typename T >`
`void read (T &array) const`
- `template<typename T >`
`void read (T *array, const DataType &dtype={}) const`
Read the attribute into a buffer.
- `template<typename T >`
`void write (const T &buffer)`
- `template<typename T >`
`void write_raw (const T *buffer, const DataType &dtype={})`
Write a buffer to this attribute.
- `AttributeCreateProps getCreatePropertyList () const`
Get the list of properties for creation of this attribute.
- `Attribute ()=delete`

Public Member Functions inherited from HighFive::Object

- `Object (Object &&other) noexcept`
- `~Object ()`
- `bool isValid () const noexcept`
isValid
- `hid_t getId () const noexcept`
getId
- `ObjectInfo getInfo () const`
Retrieve several infos about the current object (address, dates, etc)
- `ObjectType getType () const`
Gets the fundamental type of the object (dataset, group, etc)
- `bool operator== (const Object &other) const noexcept`

Public Member Functions inherited from HighFive::PathTraits< Attribute >

- `PathTraits ()`
- `std::string getPath () const`
return the path to the current object
- `File & getFile () const noexcept`
Return a reference to the File object this object belongs.

Static Public Attributes

- `static const ObjectType type = ObjectType::Attribute`

Protected Member Functions

- `Object (Object &&other) noexcept`
- `Object ()`
- `Object (const Object &other)`
- `Object (hid_t)`

Protected Member Functions inherited from [HighFive::Object](#)

- [Object](#) ()
- [Object](#) (const [Object](#) &other)
- [Object](#) (hid_t)
- [Object](#) & [operator=](#) (const [Object](#) &other)

Additional Inherited Members

Protected Attributes inherited from [HighFive::Object](#)

- [hid_t _hid](#)

Protected Attributes inherited from [HighFive::PathTraits< \[Attribute\]\(#\) >](#)

- `std::shared_ptr< File > _file_obj`

9.7.1 Detailed Description

Class representing an attribute of a dataset or group.

9.7.2 Constructor & Destructor Documentation

9.7.2.1 [Attribute\(\)](#)

```
HighFive::Attribute::Attribute ( ) [delete]
```

9.7.3 Member Function Documentation

9.7.3.1 [getCreatePropertyList\(\)](#)

```
AttributeCreateProps HighFive::Attribute::getCreatePropertyList ( ) const [inline]
```

Get the list of properties for creation of this attribute.

9.7.3.2 [getDataType\(\)](#)

```
DataType HighFive::Attribute::getDataType ( ) const [inline]
```

[getDataType](#)

Returns

return the datatype associated with this dataset

9.7.3.3 getMemSpace()

```
DataSpace HighFive::Attribute::getMemSpace ( ) const [inline]
```

getMemSpace

Returns

same than getSpace for [DataSet](#), compatibility with [Selection](#) class

9.7.3.4 getName()

```
std::string HighFive::Attribute::getName ( ) const [inline]
```

return the name of the current attribute

Returns

the name of the attribute

9.7.3.5 getSpace()

```
DataSpace HighFive::Attribute::getSpace ( ) const [inline]
```

getSpace

Returns

return the dataspace associated with this dataset

9.7.3.6 getStorageSize()

```
size_t HighFive::Attribute::getStorageSize ( ) const [inline]
```

9.7.3.7 Object() [1/4]

```
HighFive::Object::Object ( ) [inline], [protected]
```

9.7.3.8 Object() [2/4]

```
HighFive::Object::Object (
    const Object & other ) [inline], [protected]
```

9.7.3.9 Object() [3/4]

```
HighFive::Object::Object (
    hid_t hid ) [inline], [explicit], [protected]
```

9.7.3.10 Object() [4/4]

```
HighFive::Object::Object (
    Object && other ) [inline], [protected], [noexcept]
```

9.7.3.11 read() [1/3]

```
template<typename T >
T HighFive::Attribute::read [inline]
```

Return the attribute.

9.7.3.12 read() [2/3]

```
template<typename T >
void HighFive::Attribute::read (
    T & array ) const [inline]
```

Read the attribute into a buffer An exception is raised if the numbers of dimension of the buffer and of the attribute are different

The array type can be a N-pointer or a N-vector (e.g int** integer two dimensional array)

9.7.3.13 read() [3/3]

```
template<typename T >
void HighFive::Attribute::read (
    T * array,
    const DataType & dtype = {} ) const [inline]
```

Read the attribute into a buffer.

9.7.3.14 write()

```
template<typename T >
void HighFive::Attribute::write (
    const T & buffer ) [inline]
```

Write the integrality N-dimension buffer to this attribute An exception is raised if the numbers of dimension of the buffer and of the attribute are different

The array type can be a N-pointer or a N-vector (e.g int** integer two dimensional array)

9.7.3.15 write_raw()

```
template<typename T >
void HighFive::Attribute::write_raw (
    const T * buffer,
    const DataType & dtype = {} ) [inline]
```

Write a buffer to this attribute.

9.7.4 Member Data Documentation

9.7.4.1 type

```
const ObjectType HighFive::Attribute::type = ObjectType::Attribute [static]
```

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

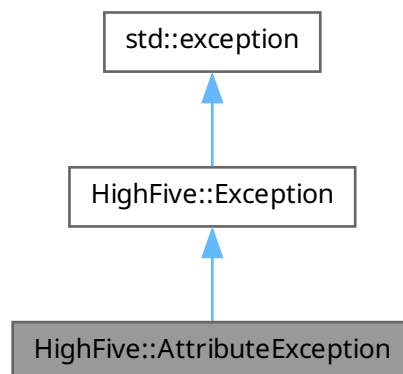
- [highfive/H5Attribute.hpp](#)
- [highfive/bits/H5Attribute_misc.hpp](#)

9.8 HighFive::AttributeException Class Reference

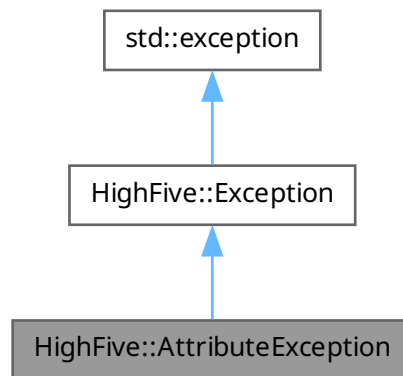
[Exception](#) specific to [HighFive Attribute](#) interface.

```
#include <H5Exception.hpp>
```

Inheritance diagram for HighFive::AttributeException:



Collaboration diagram for HighFive::AttributeException:



Public Member Functions

- [AttributeException](#) (const std::string &err_msg)

Public Member Functions inherited from [HighFive::Exception](#)

- [Exception](#) (const std::string &err_msg)
- virtual [~Exception](#) () throw ()
- const char * [what](#) () const override throw ()
get the current exception error message
- virtual void [setErrorMsg](#) (const std::string &errmsg)
define the error message
- [Exception](#) * [nextException](#) () const
nextException
- hid_t [getErrMajor](#) () const
HDF5 library error mapper.
- hid_t [getErrMinor](#) () const
HDF5 library error mapper.

Additional Inherited Members

Protected Attributes inherited from [HighFive::Exception](#)

- std::string [_errmsg](#)
- std::shared_ptr< [Exception](#) > [_next](#)
- hid_t [_err_major](#)
- hid_t [_err_minor](#)

9.8.1 Detailed Description

[Exception](#) specific to [HighFive Attribute](#) interface.

9.8.2 Constructor & Destructor Documentation

9.8.2.1 AttributeException()

```
HighFive::AttributeException::AttributeException (
    const std::string & err_msg ) [inline]
```

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

- [highfive/H5Exception.hpp](#)

9.9 HighFive::Caching Class Reference

```
#include <H5PropertyList.hpp>
```

Public Member Functions

- [Caching](#) (const size_t numSlots, const size_t cacheSize, const double w0=static_cast< double >(H5D_↵
CHUNK_CACHE_W0_DEFAULT))
- [Caching](#) (const [DataSetCreateProps](#) &dcpl)
- size_t [getNumSlots](#) () const
- size_t [getCacheSize](#) () const
- double [getW0](#) () const

9.9.1 Detailed Description

Dataset access property to control chunk cache configuration. Do not confuse with the similar file access property for `H5Pset_cache`

9.9.2 Constructor & Destructor Documentation

9.9.2.1 Caching() [1/2]

```
HighFive::Caching::Caching (
    const size_t numSlots,
    const size_t cacheSize,
    const double w0 = static_cast<double>(H5D_CHUNK_CACHE_W0_DEFAULT) ) [inline]
```

https://support.hdfgroup.org/HDF5/doc/RM/H5P/H5Pset_chunk_cache.html for details.

9.9.2.2 Caching() [2/2]

```
HighFive::Caching::Caching (
    const DataSetCreateProps & dcp1 ) [inline], [explicit]
```

9.9.3 Member Function Documentation

9.9.3.1 getCacheSize()

```
size_t HighFive::Caching::getCacheSize ( ) const [inline]
```

9.9.3.2 getNumSlots()

```
size_t HighFive::Caching::getNumSlots ( ) const [inline]
```

9.9.3.3 getW0()

```
double HighFive::Caching::getW0 ( ) const [inline]
```

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

- [highfive/H5PropertyList.hpp](#)
- [highfive/bits/H5PropertyList_misc.hpp](#)

9.10 HighFive::Chunking Class Reference

```
#include <H5PropertyList.hpp>
```

Public Member Functions

- [Chunking](#) (const std::vector< hsize_t > &dims)
- [Chunking](#) (const std::initializer_list< hsize_t > &items)
- [Chunking](#) (hsize_t item, Args... args)
- [Chunking](#) (DataSetCreateProps &plist, size_t max_dims=32)
- const std::vector< hsize_t > & [getDimensions](#) () const noexcept

9.10.1 Constructor & Destructor Documentation

9.10.1.1 Chunking() [1/4]

```
HighFive::Chunking::Chunking (
    const std::vector< hsize_t > & dims ) [inline], [explicit]
```

9.10.1.2 Chunking() [2/4]

```
HighFive::Chunking::Chunking (
    const std::initializer_list< hsize_t > & items ) [inline]
```

9.10.1.3 Chunking() [3/4]

```
template<typename... Args>
HighFive::Chunking::Chunking (
    hsize_t item,
    Args... args ) [inline], [explicit]
```

9.10.1.4 Chunking() [4/4]

```
HighFive::Chunking::Chunking (
    DataSetCreateProps & plist,
    size_t max_dims = 32 ) [inline], [explicit]
```

9.10.2 Member Function Documentation**9.10.2.1 getDimensions()**

```
const std::vector< hsize_t > & HighFive::Chunking::getDimensions ( ) const [inline], [noexcept]
```

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

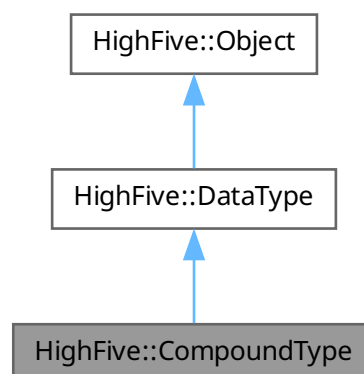
- [highfive/H5PropertyList.hpp](#)
- [highfive/bits/H5PropertyList_misc.hpp](#)

9.11 HighFive::CompoundType Class Reference

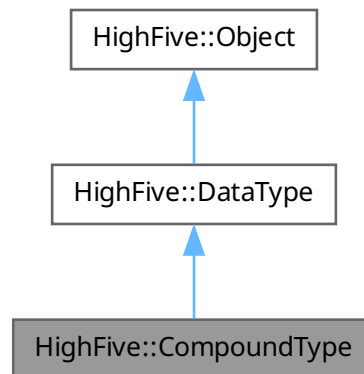
Create a compound HDF5 datatype.

```
#include <H5DataType.hpp>
```

Inheritance diagram for HighFive::CompoundType:



Collaboration diagram for HighFive::CompoundType:



Classes

- struct `member_def`
Use for defining a sub-type of compound type.

Public Member Functions

- `CompoundType` (const `CompoundType` &other)=default
- `CompoundType` (const std::vector< `member_def` > &t_members, size_t size=0)
Initializes a compound type from a vector of member definitions.
- `CompoundType` (std::vector< `member_def` > &&t_members, size_t size=0)
- `CompoundType` (const std::initializer_list< `member_def` > &t_members, size_t size=0)
- `CompoundType` (`DataType` &&type)
Initializes a compound type from a `DataType`.
- void `commit` (const `Object` &object, const std::string &name) const
Commit datatype into the given `Object`.
- const std::vector< `member_def` > & `getMembers` () const noexcept
Get read access to the `CompoundType` members.

Public Member Functions inherited from HighFive::DataType

- bool `operator==` (const `DataType` &other) const
- bool `operator!=` (const `DataType` &other) const
- `DataTypeClass` `getClass` () const
Return the fundamental type.
- size_t `getSize` () const
Returns the length (in bytes) of this type elements.
- std::string `string` () const
Returns a friendly description of the type (e.g. Float32)
- bool `isVariableStr` () const

- Returns whether the type is a variable-length string.*

 - bool `isFixedLenStr` () const

Returns whether the type is a fixed-length string.
- bool `empty` () const noexcept

Check the [DataType](#) was default constructed. Such value might represent auto-detection of the datatype from a buffer.
- bool `isReference` () const

Returns whether the type is a [Reference](#).
- [DataTypeCreateProps](#) `getCreatePropertyList` () const

Get the list of properties for creation of this [DataType](#).

Public Member Functions inherited from [HighFive::Object](#)

- [Object](#) ([Object](#) &&other) noexcept
- [~Object](#) ()
- bool `isValid` () const noexcept

isValid
- hid_t `getId` () const noexcept

getId
- [ObjectInfo](#) `getInfo` () const

Retrieve several infos about the current object (address, dates, etc)
- [ObjectType](#) `getType` () const

Gets the fundamental type of the object (dataset, group, etc)
- bool `operator==` (const [Object](#) &other) const noexcept

Additional Inherited Members

Protected Member Functions inherited from [HighFive::DataType](#)

- [Object](#) ([Object](#) &&other) noexcept
- [Object](#) ()
- [Object](#) (const [Object](#) &other)
- [Object](#) (hid_t)

Protected Member Functions inherited from [HighFive::Object](#)

- [Object](#) ()
- [Object](#) (const [Object](#) &other)
- [Object](#) (hid_t)
- [Object](#) & `operator=` (const [Object](#) &other)

Protected Attributes inherited from [HighFive::Object](#)

- hid_t `_hid`

9.11.1 Detailed Description

Create a compound HDF5 datatype.

9.11.2 Constructor & Destructor Documentation

9.11.2.1 CompoundType() [1/5]

```
HighFive::CompoundType::CompoundType (
    const CompoundType & other ) [default]
```

9.11.2.2 CompoundType() [2/5]

```
HighFive::CompoundType::CompoundType (
    const std::vector< member\_def > & t_members,
    size_t size = 0 ) [inline]
```

Initializes a compound type from a vector of member definitions.

Parameters

<i>t_members</i>	
<i>size</i>	

9.11.2.3 CompoundType() [3/5]

```
HighFive::CompoundType::CompoundType (
    std::vector< member\_def > && t_members,
    size_t size = 0 ) [inline]
```

9.11.2.4 CompoundType() [4/5]

```
HighFive::CompoundType::CompoundType (
    const std::initializer_list< member\_def > & t_members,
    size_t size = 0 ) [inline]
```

9.11.2.5 CompoundType() [5/5]

```
HighFive::CompoundType::CompoundType (
    DataType && type ) [inline]
```

Initializes a compound type from a [DataType](#).

Parameters

<i>type</i>	
-------------	--

9.11.3 Member Function Documentation

9.11.3.1 commit()

```
void HighFive::CompoundType::commit (
    const Object & object,
    const std::string & name ) const [inline]
```

Commit datatype into the given [Object](#).

Parameters

<i>object</i>	Location to commit object into
<i>name</i>	Name to give the datatype

9.11.3.2 getMembers()

```
const std::vector< member\_def > & HighFive::CompoundType::getMembers ( ) const [inline],
[noexcept]
```

Get read access to the [CompoundType](#) members.

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

- [highfive/H5DataType.hpp](#)
- [highfive/bits/H5DataType_misc.hpp](#)

9.12 H5Easy::Compression Class Reference

Signal to set compression level for written DataSets.

```
#include <H5Easy.hpp>
```

Public Member Functions

- [Compression](#) (bool enable=true)
Enable compression with the highest compression level (9). or disable compression (set compression level to 0).
- template<class T >
[Compression](#) (T level)
Set compression level.
- unsigned [get](#) () const
Return compression level.

9.12.1 Detailed Description

Signal to set compression level for written DataSets.

9.12.2 Constructor & Destructor Documentation

9.12.2.1 Compression() [1/2]

```
H5Easy::Compression::Compression (  
    bool enable = true )    [inline], [explicit]
```

Enable compression with the highest compression level (9). or disable compression (set compression level to 0).

Parameters

<i>enable</i>	true to enable with highest compression level
---------------	---

9.12.2.2 Compression() [2/2]

```
template<class T >
H5Easy::Compression::Compression (
    T level ) [inline]
```

Set compression level.

Parameters

<i>level</i>	the compression level
--------------	-----------------------

9.12.3 Member Function Documentation

9.12.3.1 get()

```
unsigned H5Easy::Compression::get ( ) const [inline]
```

Return compression level.

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

- [highfive/H5Easy.hpp](#)
- [highfive/h5easy_bits/H5Easy_public.hpp](#)

9.13 HighFive::CreateIntermediateGroup Class Reference

```
#include <H5PropertyList.hpp>
```

Public Member Functions

- [CreateIntermediateGroup](#) (bool create=true)
- [CreateIntermediateGroup](#) (const [ObjectCreateProps](#) &ocpl)
- [CreateIntermediateGroup](#) (const [LinkCreateProps](#) &lcpl)
- bool [isSet](#) () const

Protected Member Functions

- void [fromPropertyList](#) (hid_t hid)

9.13.1 Constructor & Destructor Documentation

9.13.1.1 CreateIntermediateGroup() [1/3]

```
HighFive::CreateIntermediateGroup::CreateIntermediateGroup (
    bool create = true ) [inline], [explicit]
```

9.13.1.2 CreateIntermediateGroup() [2/3]

```
HighFive::CreateIntermediateGroup::CreateIntermediateGroup (
    const ObjectCreateProps & ocpl ) [inline], [explicit]
```

9.13.1.3 CreateIntermediateGroup() [3/3]

```
HighFive::CreateIntermediateGroup::CreateIntermediateGroup (
    const LinkCreateProps & lcpl ) [explicit]
```

9.13.2 Member Function Documentation

9.13.2.1 fromPropertyList()

```
void HighFive::CreateIntermediateGroup::fromPropertyList (
    hid_t hid ) [inline], [protected]
```

9.13.2.2 isSet()

```
bool HighFive::CreateIntermediateGroup::isSet ( ) const [inline]
```

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

- [highfive/H5PropertyList.hpp](#)
- [highfive/bits/H5PropertyList_misc.hpp](#)

9.14 HighFive::CreationOrder Struct Reference

```
#include <H5PropertyList.hpp>
```

Public Types

- enum `_CreationOrder` { `Tracked` = H5P_CRT_ORDER_TRACKED, `Indexed` = H5P_CRT_ORDER_INDEXED }

9.14.1 Member Enumeration Documentation

9.14.1.1 _CreationOrder

```
enum HighFive::CreationOrder::_CreationOrder
```

Enumerator

Tracked	
Indexed	

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

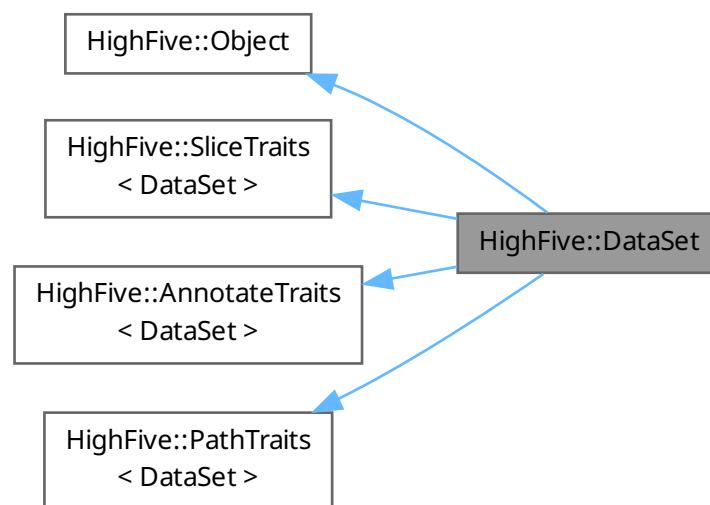
- [highfive/H5PropertyList.hpp](#)

9.15 HighFive::DataSet Class Reference

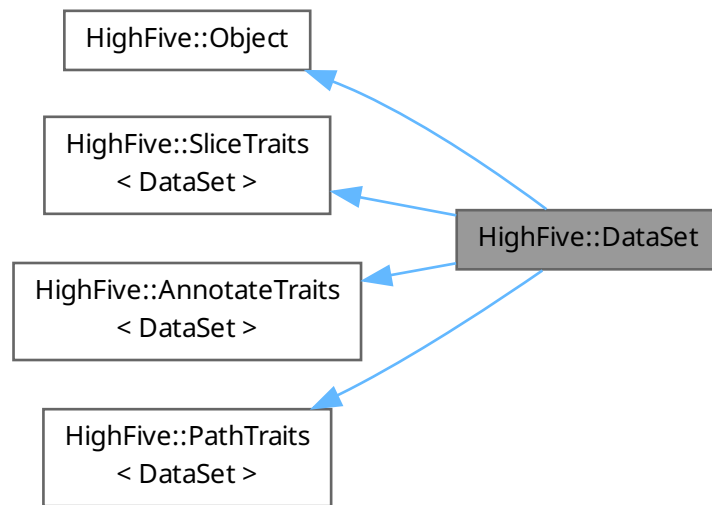
Class representing a dataset.

```
#include <H5DataSet.hpp>
```

Inheritance diagram for HighFive::DataSet:



Collaboration diagram for HighFive::DataSet:



Public Member Functions

- `uint64_t` [getStorageSize](#) () const
getStorageSize
- `uint64_t` [getOffset](#) () const
getOffset
- `DataType` [getDataType](#) () const
getDataType
- `DataSpace` [getSpace](#) () const
getSpace
- `DataSpace` [getMemSpace](#) () const
getMemSpace
- `void` [resize](#) (const std::vector< size_t > &dims)
Change the size of the dataset.
- `std::vector< size_t >` [getDimensions](#) () const
Get the dimensions of the whole [DataSet](#). This is a shorthand for [getSpace\(\).getDimensions\(\)](#)
- `size_t` [getElementCount](#) () const
Get the total number of elements in the current dataset. E.g. 2x2x2 matrix has size 8. This is a shorthand for [getSpace\(\).getTotalCount\(\)](#)
- `DataSetCreateProps` [getCreatePropertyList](#) () const
Get the list of properties for creation of this dataset.
- `DataSetAccessProps` [getAccessPropertyList](#) () const
Get the list of properties for accession of this dataset.
- `DataSet` ()=default

Public Member Functions inherited from HighFive::Object

- [Object](#) ([Object](#) &&other) noexcept
- [~Object](#) ()
- bool [isValid](#) () const noexcept
isValid
- hid_t [getId](#) () const noexcept
getId
- [ObjectInfo](#) [getInfo](#) () const
Retrieve several infos about the current object (address, dates, etc)
- [ObjectType](#) [getType](#) () const
Gets the fundamental type of the object (dataset, group, etc)
- bool [operator==](#) (const [Object](#) &other) const noexcept

Public Member Functions inherited from HighFive::SliceTraits< DataSet >

- [Selection](#) [select](#) (const [HyperSlab](#) &hyperslab) const
Select an hyperslab in the current Slice/Dataset.
- [Selection](#) [select](#) (const std::vector< size_t > &offset, const std::vector< size_t > &count, const std::vector< size_t > &stride={}, const std::vector< size_t > &block={}) const
Select a region in the current Slice/Dataset of count points at offset separated by stride. If strides are not provided they will default to 1 in all dimensions.
- [Selection](#) [select](#) (const std::vector< size_t > &columns) const
Select a set of columns in the last dimension of this dataset.
- [Selection](#) [select](#) (const [ElementSet](#) &elements) const
Select a region in the current Slice/Dataset out of a list of elements.
- T [read](#) (const [DataTransferProps](#) &xfer_props=[DataTransferProps](#)()) const
- void [read](#) (T &array, const [DataTransferProps](#) &xfer_props=[DataTransferProps](#)()) const
- void [read](#) (T *array, const [DataType](#) &dtype=[DataType](#)(), const [DataTransferProps](#) &xfer_props=[DataTransferProps](#)()) const
- void [write](#) (const T &buffer, const [DataTransferProps](#) &xfer_props=[DataTransferProps](#)())
- void [write_raw](#) (const T *buffer, const [DataType](#) &dtype=[DataType](#)(), const [DataTransferProps](#) &xfer_↵ props=[DataTransferProps](#)())

Public Member Functions inherited from HighFive::AnnotateTraits< DataSet >

- [Attribute](#) [createAttribute](#) (const std::string &attribute_name, const [DataSpace](#) &space, const [DataType](#) &type)
create a new attribute with the name attribute_name
- [Attribute](#) [createAttribute](#) (const std::string &attribute_name, const [DataSpace](#) &space)
createAttribute create a new attribute on the current dataset with size specified by space
- [Attribute](#) [createAttribute](#) (const std::string &attribute_name, const T &data)
createAttribute create a new attribute on the current dataset and write to it, inferring the DataSpace from data.
- void [deleteAttribute](#) (const std::string &attribute_name)
deleteAttribute let you delete an attribute by its name.
- [Attribute](#) [getAttribute](#) (const std::string &attribute_name) const
open an existing attribute with the name attribute_name
- size_t [getNumberAttributes](#) () const
return the number of attributes of the node / group
- std::vector< std::string > [listAttributeNames](#) () const
list all attribute name of the node / group
- bool [hasAttribute](#) (const std::string &attr_name) const
checks an attribute exists

Public Member Functions inherited from [HighFive::PathTraits< DataSet >](#)

- [PathTraits](#) ()
- `std::string` [getPath](#) () const
return the path to the current object
- [File](#) & [getFile](#) () const noexcept
Return a reference to the [File](#) object this object belongs.

Static Public Attributes

- static const [ObjectType](#) type = [ObjectType::Dataset](#)

Protected Member Functions

- [DataSet](#) ([Object](#) &&o) noexcept
- [Object](#) ([Object](#) &&other) noexcept
- [Object](#) ()
- [Object](#) (const [Object](#) &other)
- [Object](#) (hid_t)

Protected Member Functions inherited from [HighFive::Object](#)

- [Object](#) ()
- [Object](#) (const [Object](#) &other)
- [Object](#) (hid_t)
- [Object](#) & [operator=](#) (const [Object](#) &other)

Protected Member Functions inherited from [HighFive::SliceTraits< DataSet >](#)

- [Selection](#) [select_impl](#) (const [HyperSlab](#) &hyperslab, const [DataSpace](#) &memspace) const

Friends

- class [Reference](#)
- template<typename Derivate >
class [NodeTraits](#)

Additional Inherited Members

Protected Attributes inherited from [HighFive::Object](#)

- hid_t [_hid](#)

Protected Attributes inherited from [HighFive::PathTraits< DataSet >](#)

- `std::shared_ptr< File >` [_file_obj](#)

9.15.1 Detailed Description

Class representing a dataset.

9.15.2 Constructor & Destructor Documentation

9.15.2.1 DataSet() [1/2]

```
HighFive::DataSet::DataSet ( ) [default]
```

Deprecated Default constructor creates unsafe uninitialized objects

9.15.2.2 DataSet() [2/2]

```
HighFive::DataSet::DataSet (
    Object && o ) [inline], [protected], [noexcept]
```

9.15.3 Member Function Documentation

9.15.3.1 getAccessPropertyList()

```
DataSetAccessProps HighFive::DataSet::getAccessPropertyList ( ) const [inline]
```

Get the list of properties for accession of this dataset.

9.15.3.2 getCreatePropertyList()

```
DataSetCreateProps HighFive::DataSet::getCreatePropertyList ( ) const [inline]
```

Get the list of properties for creation of this dataset.

9.15.3.3 getDataType()

```
DataType HighFive::DataSet::getDataType ( ) const [inline]
```

getDataType

Returns

return the datatype associated with this dataset

9.15.3.4 getDimensions()

```
std::vector< size_t > HighFive::DataSet::getDimensions ( ) const [inline]
```

Get the dimensions of the whole [DataSet](#). This is a shorthand for [getSpace\(\).getDimensions\(\)](#)

Returns

The shape of the current [HighFive::DataSet](#)

9.15.3.5 getElementCount()

```
size_t HighFive::DataSet::getElementCount ( ) const [inline]
```

Get the total number of elements in the current dataset. E.g. 2x2x2 matrix has size 8. This is a shorthand for [getSpace\(\).getTotalCount\(\)](#)

Returns

The shape of the current [HighFive::DataSet](#)

9.15.3.6 getMemSpace()

```
DataSpace HighFive::DataSet::getMemSpace ( ) const [inline]
```

[getMemSpace](#)

Returns

same than [getSpace](#) for [DataSet](#), compatibility with [Selection](#) class

9.15.3.7 getOffset()

```
uint64_t HighFive::DataSet::getOffset ( ) const [inline]
```

[getOffset](#)

Returns

returns [DataSet](#) address in file

9.15.3.8 getSpace()

```
DataSpace HighFive::DataSet::getSpace ( ) const [inline]
```

[getSpace](#)

Returns

return the dataspace associated with this dataset

9.15.3.9 getStorageSize()

```
uint64_t HighFive::DataSet::getStorageSize ( ) const [inline]
```

getStorageSize

Returns

returns the amount of storage allocated for a dataset.

9.15.3.10 Object() [1/4]

```
HighFive::Object::Object ( ) [inline], [protected]
```

9.15.3.11 Object() [2/4]

```
HighFive::Object::Object (
    const Object & other ) [inline], [protected]
```

9.15.3.12 Object() [3/4]

```
HighFive::Object::Object (
    hid_t hid ) [inline], [explicit], [protected]
```

9.15.3.13 Object() [4/4]

```
HighFive::Object::Object (
    Object && other ) [inline], [protected], [noexcept]
```

9.15.3.14 resize()

```
void HighFive::DataSet::resize (
    const std::vector< size_t > & dims ) [inline]
```

Change the size of the dataset.

This requires that the dataset was created with chunking, and you would generally want to have set a larger maxdims setting

Parameters

<i>dims</i>	New size of the dataset
-------------	-------------------------

9.15.4 Friends And Related Symbol Documentation

9.15.4.1 NodeTraits

```
template<typename Derivate >  
friend class NodeTraits [friend]
```

9.15.4.2 Reference

```
friend class Reference [friend]
```

9.15.5 Member Data Documentation

9.15.5.1 type

```
const ObjectType HighFive::DataSet::type = ObjectType::Dataset [static]
```

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

- [highfive/H5DataSet.hpp](#)
- [highfive/bits/H5DataSet_misc.hpp](#)

9.16 HighFive::DataSetException Class Reference

Exception specific to [HighFive DataSet](#) interface.

```
#include <H5Exception.hpp>
```

Inheritance diagram for HighFive::DataSetException:



Collaboration diagram for HighFive::DataSetException:



Public Member Functions

- [DataSetException](#) (const std::string &err_msg)

Public Member Functions inherited from [HighFive::Exception](#)

- [Exception](#) (const std::string &err_msg)
- virtual [~Exception](#) () throw ()
- const char * [what](#) () const override throw ()
get the current exception error message
- virtual void [setErrorMsg](#) (const std::string &errmsg)
define the error message
- [Exception](#) * [nextException](#) () const
nextException
- hid_t [getErrMajor](#) () const
HDF5 library error mapper.
- hid_t [getErrMinor](#) () const
HDF5 library error mapper.

Additional Inherited Members

Protected Attributes inherited from [HighFive::Exception](#)

- std::string [_errmsg](#)
- std::shared_ptr< [Exception](#) > [_next](#)
- hid_t [_err_major](#)
- hid_t [_err_minor](#)

9.16.1 Detailed Description

[Exception](#) specific to [HighFive DataSet](#) interface.

9.16.2 Constructor & Destructor Documentation

9.16.2.1 DataSetException()

```
HighFive::DataSetException::DataSetException (
    const std::string & err_msg ) [inline]
```

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

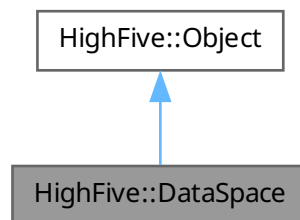
- [highfive/H5Exception.hpp](#)

9.17 HighFive::DataSpace Class Reference

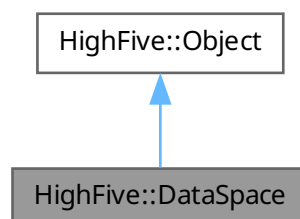
Class representing the space (dimensions) of a dataset.

```
#include <H5DataSpace.hpp>
```

Inheritance diagram for HighFive::DataSpace:



Collaboration diagram for HighFive::DataSpace:



Public Types

- enum [DataSpaceType](#) { [dataspace_scalar](#) , [dataspace_null](#) }
dataspace type

Public Member Functions

- [DataSpace](#) (const std::vector< size_t > &dims)
- template<size_t N>
[DataSpace](#) (const std::array< size_t, N > &dims)
- [DataSpace](#) (const std::initializer_list< size_t > &items)
- template<typename... Args>
[DataSpace](#) (size_t dim1, Args... dims)
- template<typename IT , typename = typename std::enable_if<!std::is_integral<IT>::value, IT>::type>
[DataSpace](#) (const IT begin, const IT end)
- [DataSpace](#) (const std::vector< size_t > &dims, const std::vector< size_t > &maxdims)
Create a resizable N-dimensional dataspace.
- [DataSpace](#) ([DataSpaceType](#) dtype)
DataSpace create a scalar dataspace or a null dataset.
- [DataSpace](#) clone () const
- size_t [getNumberDimensions](#) () const
getNumberDimensions
- std::vector< size_t > [getDimensions](#) () const
getDimensions
- size_t [getElementCount](#) () const
getElementCount
- std::vector< size_t > [getMaxDimensions](#) () const
getMaxDimensions

Public Member Functions inherited from [HighFive::Object](#)

- [Object](#) ([Object](#) &&other) noexcept
- ~[Object](#) ()
- bool [isValid](#) () const noexcept
isValid
- hid_t [getId](#) () const noexcept
getId
- [ObjectInfo](#) [getInfo](#) () const
Retrieve several infos about the current object (address, dates, etc)
- [ObjectType](#) [getType](#) () const
Gets the fundamental type of the object (dataset, group, etc)
- bool [operator==](#) (const [Object](#) &other) const noexcept

Static Public Member Functions

- template<typename T >
static [DataSpace](#) [From](#) (const T &value)
Create a dataspace matching a type accepted by details::inspector.
- template<std::size_t N, std::size_t Width>
static [DataSpace](#) [FromArrayStrings](#) (const char(&)[N][Width])

Static Public Attributes

- static const [ObjectType](#) type = [ObjectType::DataSpace](#)
- static const size_t [UNLIMITED](#) = [SIZE_MAX](#)

Protected Member Functions

- [DataSpace](#) ()=default

Protected Member Functions inherited from [HighFive::Object](#)

- [Object](#) ()
- [Object](#) (const [Object](#) &other)
- [Object](#) (hid_t)
- [Object](#) & [operator=](#) (const [Object](#) &other)

Friends

- class [Attribute](#)
- class [File](#)
- class [DataSet](#)

Additional Inherited Members

Protected Attributes inherited from [HighFive::Object](#)

- hid_t [_hid](#)

9.17.1 Detailed Description

Class representing the space (dimensions) of a dataset.

9.17.2 Member Enumeration Documentation

9.17.2.1 DataspaceType

enum [HighFive::DataSpace::DataspaceType](#)

dataspace type

Enumerator

dataspace_scalar	
dataspace_null	

9.17.3 Constructor & Destructor Documentation

9.17.3.1 DataSpace() [1/8]

```
HighFive::DataSpace::DataSpace (
    const std::vector< size_t > & dims ) [inline], [explicit]
```

create a dataspace of N-dimensions Each dimension is configured this way size(dim1) = vec[0] size(dim2) = vec[1] etc...

9.17.3.2 DataSpace() [2/8]

```
template<size_t N>
HighFive::DataSpace::DataSpace (
    const std::array< size_t, N > & dims ) [inline], [explicit]
```

9.17.3.3 DataSpace() [3/8]

```
HighFive::DataSpace::DataSpace (
    const std::initializer_list< size_t > & items ) [inline]
```

Make sure that `DataSpace({1,2,3})` works on GCC. This is the shortcut form of the vector initializer, but on some compilers (gcc) this does not resolve correctly without this constructor.

9.17.3.4 DataSpace() [4/8]

```
template<typename... Args>
HighFive::DataSpace::DataSpace (
    size_t dim1,
    Args... dims ) [inline], [explicit]
```

Allow directly listing 1 or more dimensions to initialize, that is, `DataSpace(1,2)` means `DataSpace(std::vector<size_t>{1,2})`.

9.17.3.5 DataSpace() [5/8]

```
template<class IT , typename >
HighFive::DataSpace::DataSpace (
    const IT begin,
    const IT end ) [inline]
```

Create a dataspace from an iterator pair

Explicitly disable `DataSpace(int_like, int_like)` from trying to use this constructor

9.17.3.6 DataSpace() [6/8]

```
HighFive::DataSpace::DataSpace (
    const std::vector< size_t > & dims,
    const std::vector< size_t > & maxdims ) [inline], [explicit]
```

Create a resizable N-dimensional dataspace.

Parameters

<i>dims</i>	Initial size of dataspace
<i>maxdims</i>	Maximum size of the dataspace

9.17.3.7 DataSpace() [7/8]

```
HighFive::DataSpace::DataSpace (
    DataSpace::DataspaceType dtype ) [inline], [explicit]
```

[DataSpace](#) create a scalar dataspace or a null dataset.

9.17.3.8 DataSpace() [8/8]

```
HighFive::DataSpace::DataSpace ( ) [protected], [default]
```

9.17.4 Member Function Documentation**9.17.4.1 clone()**

```
DataSpace HighFive::DataSpace::clone ( ) const [inline]
```

Create a new [DataSpace](#) with a different id available for modifications

9.17.4.2 From()

```
template<typename T >
DataSpace HighFive::DataSpace::From (
    const T & value ) [inline], [static]
```

Create a dataspace matching a type accepted by details::inspector.

9.17.4.3 FromCharArrayStrings()

```
template<std::size_t N, std::size_t Width>
DataSpace HighFive::DataSpace::FromCharArrayStrings (
    const char (&)[N][Width] ) [inline], [static]
```

9.17.4.4 getDimensions()

```
std::vector< size_t > HighFive::DataSpace::getDimensions ( ) const [inline]
```

getDimensions

Returns

return a vector of N-element, each element is the size of the associated dataset dimension

9.17.4.5 getElementCount()

```
size_t HighFive::DataSpace::getElementCount ( ) const [inline]
```

getElementCount

Returns

the total number of elements in the dataspace

9.17.4.6 getMaxDimensions()

```
std::vector< size_t > HighFive::DataSpace::getMaxDimensions ( ) const [inline]
```

getMaxDimensions

Returns

return a vector of N-element, each element is the size of the associated dataset maximum dimension

9.17.4.7 getNumberDimensions()

```
size_t HighFive::DataSpace::getNumberDimensions ( ) const [inline]
```

getNumberDimensions

Returns

the number of dimensions in the current dataspace

9.17.5 Friends And Related Symbol Documentation

9.17.5.1 Attribute

```
friend class Attribute [friend]
```

9.17.5.2 DataSet

```
friend class DataSet [friend]
```

9.17.5.3 File

```
friend class File [friend]
```

9.17.6 Member Data Documentation

9.17.6.1 type

```
const ObjectType HighFive::DataSpace::type = ObjectType::DataSpace [static]
```

9.17.6.2 UNLIMITED

```
const size_t HighFive::DataSpace::UNLIMITED = SIZE_MAX [static]
```

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

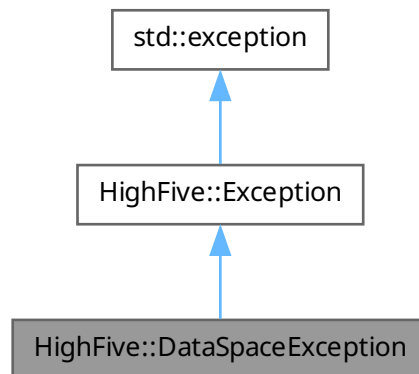
- [highfive/H5DataSpace.hpp](#)
- [highfive/bits/H5Dataspace_misc.hpp](#)

9.18 HighFive::DataSpaceException Class Reference

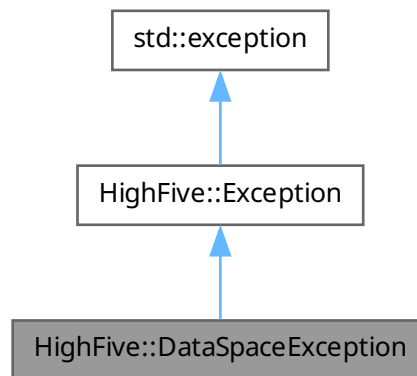
[Exception](#) specific to [HighFive DataSpace](#) interface.

```
#include <H5Exception.hpp>
```

Inheritance diagram for HighFive::DataSpaceException:



Collaboration diagram for HighFive::DataSpaceException:



Public Member Functions

- [DataSpaceException](#) (const std::string &err_msg)

Public Member Functions inherited from [HighFive::Exception](#)

- [Exception](#) (const std::string &err_msg)
- virtual [~Exception](#) () throw ()
- const char * [what](#) () const override throw ()
get the current exception error message
- virtual void [setErrorMsg](#) (const std::string &errmsg)
define the error message
- [Exception](#) * [nextException](#) () const
nextException
- hid_t [getErrMajor](#) () const
HDF5 library error mapper.
- hid_t [getErrMinor](#) () const
HDF5 library error mapper.

Additional Inherited Members

Protected Attributes inherited from [HighFive::Exception](#)

- std::string [_errmsg](#)
- std::shared_ptr< [Exception](#) > [_next](#)
- hid_t [_err_major](#)
- hid_t [_err_minor](#)

9.18.1 Detailed Description

Exception specific to [HighFive DataSpace](#) interface.

9.18.2 Constructor & Destructor Documentation

9.18.2.1 DataSpaceException()

```
HighFive::DataSpaceException::DataSpaceException (
    const std::string & err_msg ) [inline]
```

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

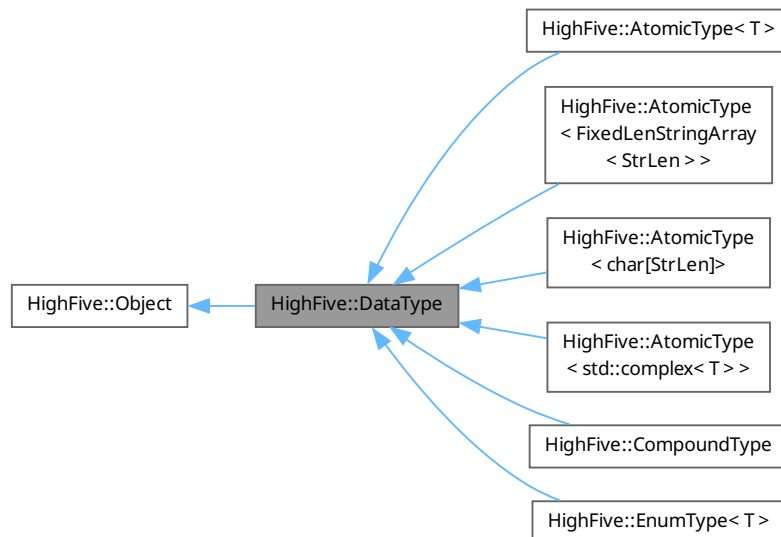
- [highfive/H5Exception.hpp](#)

9.19 HighFive::DataType Class Reference

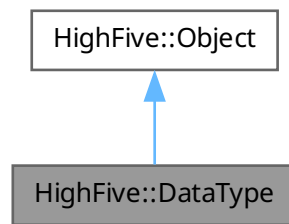
HDF5 Data Type.

```
#include <H5DataType.hpp>
```

Inheritance diagram for HighFive::DataType:



Collaboration diagram for HighFive::DataType:



Public Member Functions

- `bool operator== (const DataType &other) const`
- `bool operator!= (const DataType &other) const`
- `DataTypeClass getClass () const`
Return the fundamental type.
- `size_t getSize () const`
Returns the length (in bytes) of this type elements.
- `std::string string () const`
Returns a friendly description of the type (e.g. `Float32`)
- `bool isVariableStr () const`
Returns whether the type is a variable-length string.
- `bool isFixedLenStr () const`
Returns whether the type is a fixed-length string.
- `bool empty () const noexcept`
Check the [DataType](#) was default constructed. Such value might represent auto-detection of the datatype from a buffer.
- `bool isReference () const`
Returns whether the type is a [Reference](#).
- `DataTypeCreateProps getCreatePropertyList () const`
Get the list of properties for creation of this [DataType](#).

Public Member Functions inherited from [HighFive::Object](#)

- `Object (Object &&other) noexcept`
- `~Object ()`
- `bool isValid () const noexcept`
isValid
- `hid_t getId () const noexcept`
getId
- `ObjectInfo getInfo () const`
Retrieve several infos about the current object (address, dates, etc)
- `ObjectType getType () const`
Gets the fundamental type of the object (dataset, group, etc)
- `bool operator== (const Object &other) const noexcept`

Protected Member Functions

- [Object](#) ([Object](#) &&other) noexcept
- [Object](#) ()
- [Object](#) (const [Object](#) &other)
- [Object](#) (hid_t)

Protected Member Functions inherited from [HighFive::Object](#)

- [Object](#) ()
- [Object](#) (const [Object](#) &other)
- [Object](#) (hid_t)
- [Object](#) & [operator=](#) (const [Object](#) &other)

Friends

- class [Attribute](#)
- class [File](#)
- class [DataSet](#)
- class [CompoundType](#)

Additional Inherited Members

Protected Attributes inherited from [HighFive::Object](#)

- hid_t [_hid](#)

9.19.1 Detailed Description

HDF5 Data Type.

9.19.2 Member Function Documentation

9.19.2.1 empty()

```
bool HighFive::DataType::empty ( ) const [inline], [noexcept]
```

Check the [DataType](#) was default constructed. Such value might represent auto-detection of the datatype from a buffer.

9.19.2.2 getClass()

```
DataTypeClass HighFive::DataType::getClass ( ) const [inline]
```

Return the fundamental type.

9.19.2.3 getCreatePropertyList()

```
DataTypeCreateProps HighFive::DataType::getCreatePropertyList ( ) const [inline]
```

Get the list of properties for creation of this [DataType](#).

9.19.2.4 getSize()

```
size_t HighFive::DataType::getSize ( ) const [inline]
```

Returns the length (in bytes) of this type elements.

Notice that the size of variable length sequences may have limited applicability given that it refers to the size of the control structure. For info see https://support.hdfgroup.org/HDF5/doc/RM/RM_H5T.html#Datatype-GetSize

9.19.2.5 isFixedLenStr()

```
bool HighFive::DataType::isFixedLenStr ( ) const [inline]
```

Returns whether the type is a fixed-length string.

9.19.2.6 isReference()

```
bool HighFive::DataType::isReference ( ) const [inline]
```

Returns whether the type is a [Reference](#).

9.19.2.7 isVariableStr()

```
bool HighFive::DataType::isVariableStr ( ) const [inline]
```

Returns whether the type is a variable-length string.

9.19.2.8 Object() [1/4]

```
HighFive::Object::Object ( ) [inline], [protected]
```

9.19.2.9 Object() [2/4]

```
HighFive::Object::Object (
    const Object & other ) [inline], [protected]
```

9.19.2.10 Object() [3/4]

```
HighFive::Object::Object (
    hid_t hid ) [inline], [explicit], [protected]
```

9.19.2.11 Object() [4/4]

```
HighFive::Object::Object (
    Object && other ) [inline], [protected], [noexcept]
```

9.19.2.12 operator"!="()

```
bool HighFive::DataType::operator!= (
    const DataType & other ) const [inline]
```

9.19.2.13 operator==()

```
bool HighFive::DataType::operator== (
    const DataType & other ) const [inline]
```

9.19.2.14 string()

```
std::string HighFive::DataType::string ( ) const [inline]
```

Returns a friendly description of the type (e.g. Float32)

9.19.3 Friends And Related Symbol Documentation

9.19.3.1 Attribute

```
friend class Attribute [friend]
```

9.19.3.2 CompoundType

```
friend class CompoundType [friend]
```

9.19.3.3 DataSet

```
friend class DataSet [friend]
```

9.19.3.4 File

```
friend class File [friend]
```

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

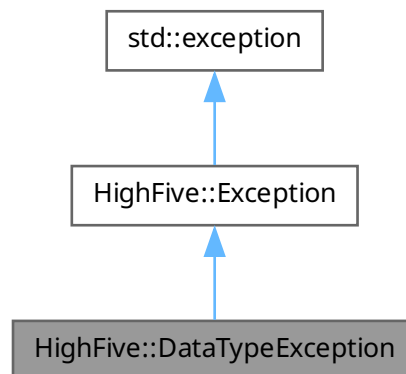
- [highfive/H5DataType.hpp](#)
- [highfive/bits/H5DataType_misc.hpp](#)

9.20 HighFive::DataTypeException Class Reference

Exception specific to [HighFive DataType](#) interface.

```
#include <H5Exception.hpp>
```

Inheritance diagram for HighFive::DataTypeException:



Collaboration diagram for HighFive::DataTypeException:



Public Member Functions

- [DataTypeException](#) (const std::string &err_msg)

Public Member Functions inherited from [HighFive::Exception](#)

- [Exception](#) (const std::string &err_msg)
- virtual [~Exception](#) () throw ()
- const char * [what](#) () const override throw ()
get the current exception error message
- virtual void [setErrMsg](#) (const std::string &errmsg)
define the error message
- [Exception](#) * [nextException](#) () const
nextException
- hid_t [getErrMajor](#) () const
HDF5 library error mapper.
- hid_t [getErrMinor](#) () const
HDF5 library error mapper.

Additional Inherited Members

Protected Attributes inherited from [HighFive::Exception](#)

- std::string [_errmsg](#)
- std::shared_ptr< [Exception](#) > [_next](#)
- hid_t [_err_major](#)
- hid_t [_err_minor](#)

9.20.1 Detailed Description

[Exception](#) specific to [HighFive DataType](#) interface.

9.20.2 Constructor & Destructor Documentation

9.20.2.1 DataTypeException()

```
HighFive::DataTypeException::DataTypeException (
    const std::string & err_msg ) [inline]
```

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

- [highfive/H5Exception.hpp](#)

9.21 HighFive::Deflate Class Reference

```
#include <H5PropertyList.hpp>
```

Public Member Functions

- [Deflate](#) (unsigned level)

9.21.1 Constructor & Destructor Documentation

9.21.1.1 Deflate()

```
HighFive::Deflate::Deflate (
    unsigned level ) [inline], [explicit]
```

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

- [highfive/H5PropertyList.hpp](#)
- [highfive/bits/H5PropertyList_misc.hpp](#)

9.22 H5Easy::DumpOptions Class Reference

Define options for dumping data.

```
#include <H5Easy.hpp>
```

Public Member Functions

- [DumpOptions](#) ()=default
Constructor: accept all default settings.
- `template<class... Args>`
[DumpOptions](#) (Args... args)
Constructor: overwrite (some of the) defaults.
- `void set (DumpMode mode)`
Overwrite [H5Easy::DumpMode](#) setting.
- `void set (Flush mode)`
Overwrite [H5Easy::Flush](#) setting.
- `void set (const Compression &level)`
Overwrite [H5Easy::Compression](#) setting.
- `template<class T, class... Args>`
`void set (T arg, Args... args)`
Overwrite any setting(s).
- `template<class T >`
`void setChunkSize (const std::vector< T > &shape)`
Set chunk-size. If the input is rank (size) zero, automatic chunking is enabled.
- `void setChunkSize (std::initializer_list< size_t > shape)`
Set chunk-size. If the input is rank (size) zero, automatic chunking is enabled.
- `bool overwrite () const`
Get overwrite-mode.
- `bool flush () const`
Get flush-mode.
- `bool compress () const`
Get compress-mode.
- `unsigned getCompressionLevel () const`
Get compression level.
- `bool isChunked () const`
Get chunking mode: `true` is manually set, `false` if chunk-size should be computed automatically.
- `std::vector< hsize_t > getChunkSize () const`
Get chunk size. Use [DumpOptions::getChunkSize](#) to check if chunk-size should be automatically computed.

9.22.1 Detailed Description

Define options for dumping data.

By default:

- [DumpMode::Create](#)
- [Flush::True](#)
- [Compression](#): false
- [ChunkSize](#): automatic

9.22.2 Constructor & Destructor Documentation

9.22.2.1 DumpOptions() [1/2]

```
H5Easy::DumpOptions::DumpOptions ( ) [default]
```

Constructor: accept all default settings.

9.22.2.2 DumpOptions() [2/2]

```
template<class... Args>
H5Easy::DumpOptions::DumpOptions (
    Args... args ) [inline]
```

Constructor: overwrite (some of the) defaults.

Parameters

<i>args</i>	any of DumpMode() , Flush() , Compression() in arbitrary number and order.
-------------	--

9.22.3 Member Function Documentation

9.22.3.1 compress()

```
bool H5Easy::DumpOptions::compress ( ) const [inline]
```

Get compress-mode.

Returns

bool

9.22.3.2 flush()

```
bool H5Easy::DumpOptions::flush ( ) const [inline]
```

Get flush-mode.

Returns

bool

9.22.3.3 getChunkSize()

```
std::vector< hsize_t > H5Easy::DumpOptions::getChunkSize ( ) const [inline]
```

Get chunk size. Use [DumpOptions::getChunkSize](#) to check if chunk-size should be automatically computed.

9.22.3.4 getCompressionLevel()

```
unsigned H5Easy::DumpOptions::getCompressionLevel ( ) const [inline]
```

Get compression level.

Returns

[0..9]

9.22.3.5 isChunked()

```
bool H5Easy::DumpOptions::isChunked ( ) const [inline]
```

Get chunking mode: `true` is manually set, `false` if chunk-size should be computed automatically.

Returns

bool

9.22.3.6 overwrite()

```
bool H5Easy::DumpOptions::overwrite ( ) const [inline]
```

Get overwrite-mode.

Returns

bool

9.22.3.7 set() [1/4]

```
void H5Easy::DumpOptions::set (
    const Compression & level ) [inline]
```

Overwrite [H5Easy::Compression](#) setting.

Parameters

<i>level</i>	Compression .
--------------	-------------------------------

9.22.3.8 set() [2/4]

```
void H5Easy::DumpOptions::set (
    DumpMode mode ) [inline]
```

Overwrite [H5Easy::DumpMode](#) setting.

Parameters

<i>mode</i>	DumpMode .
-------------	----------------------------

9.22.3.9 set() [3/4]

```
void H5Easy::DumpOptions::set (
    Flush mode ) [inline]
```

Overwrite [H5Easy::Flush](#) setting.

Parameters

<i>mode</i>	Flush .
-------------	-------------------------

9.22.3.10 set() [4/4]

```
template<class T , class... Args>
void H5Easy::DumpOptions::set (
    T arg,
    Args... args ) [inline]
```

Overwrite any setting(s).

Parameters

<i>arg</i>	any of DumpMode() , Flush() , Compression in arbitrary number and order.
<i>args</i>	any of DumpMode() , Flush() , Compression in arbitrary number and order.

9.22.3.11 setChunkSize() [1/2]

```
template<class T >
void H5Easy::DumpOptions::setChunkSize (
    const std::vector< T > & shape ) [inline]
```


Set chunk-size. If the input is rank (size) zero, automatic chunking is enabled.

Parameters

<i>shape</i>	Chunk size along each dimension.
--------------	----------------------------------

9.22.3.12 setChunkSize() [2/2]

```
void H5Easy::DumpOptions::setChunkSize (
    std::initializer_list< size_t > shape ) [inline]
```

Set chunk-size. If the input is rank (size) zero, automatic chunking is enabled.

Parameters

<i>shape</i>	Chunk size along each dimension.
--------------	----------------------------------

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

- [highfive/H5Easy.hpp](#)
- [highfive/h5easy_bits/H5Easy_public.hpp](#)

9.23 HighFive::ElementSet Class Reference

```
#include <H5Slice_traits.hpp>
```

Public Member Functions

- [ElementSet](#) (std::initializer_list< std::size_t > list)
Create a list of points of N-dimension for selection.
- [ElementSet](#) (std::initializer_list< std::vector< std::size_t > > list)
Create a list of points of N-dimension for selection.
- [ElementSet](#) (const std::vector< std::size_t > &element_ids)
Create a list of points of N-dimension for selection.
- [ElementSet](#) (const std::vector< std::vector< std::size_t > > &element_ids)
Create a list of points of N-dimension for selection.

Friends

- template<typename Derivate >
class [SliceTraits](#)

9.23.1 Constructor & Destructor Documentation**9.23.1.1 ElementSet()** [1/4]

```
HighFive::ElementSet::ElementSet (
    std::initializer_list< std::size_t > list ) [inline], [explicit]
```

Create a list of points of N-dimension for selection.

Parameters

<i>list</i>	List of continuous coordinates (e.g.: in 2 dimensions space <code>ElementSet{1, 2, 3, 4}</code> creates points (1, 2) and (3, 4)).
-------------	--

9.23.1.2 ElementSet() [2/4]

```
HighFive::ElementSet::ElementSet (
    std::initializer_list< std::vector< std::size_t > > list ) [inline], [explicit]
```

Create a list of points of N-dimension for selection.

Parameters

<i>list</i>	List of N-dim points.
-------------	-----------------------

9.23.1.3 ElementSet() [3/4]

```
HighFive::ElementSet::ElementSet (
    const std::vector< std::size_t > & element_ids ) [inline], [explicit]
```

Create a list of points of N-dimension for selection.

Parameters

<i>element_ids</i>	List of continuous coordinates (e.g.: in 2 dimensions space <code>ElementSet{1, 2, 3, 4}</code> creates points (1, 2) and (3, 4)).
--------------------	--

9.23.1.4 ElementSet() [4/4]

```
HighFive::ElementSet::ElementSet (
    const std::vector< std::vector< std::size_t > > & element_ids ) [inline], [explicit]
```

Create a list of points of N-dimension for selection.

Parameters

<i>element_ids</i>	List of N-dim points.
--------------------	-----------------------

9.23.2 Friends And Related Symbol Documentation**9.23.2.1 SliceTraits**

```
template<typename Derivate >
friend class SliceTraits [friend]
```

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

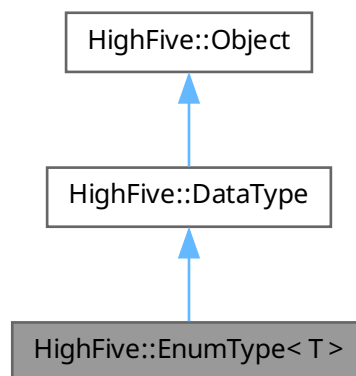
- [highfive/bits/H5Slice_traits.hpp](#)
- [highfive/bits/H5Slice_traits_misc.hpp](#)

9.24 HighFive::EnumType< T > Class Template Reference

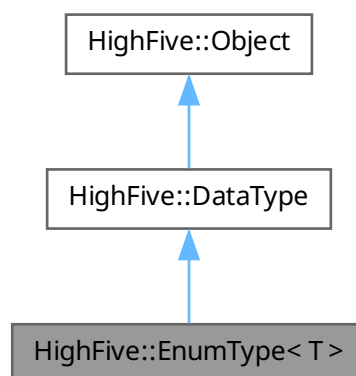
Create a enum HDF5 datatype.

```
#include <H5DataType.hpp>
```

Inheritance diagram for HighFive::EnumType< T >:



Collaboration diagram for HighFive::EnumType< T >:



Classes

- struct [member_def](#)

Use for defining a member of enum type.

Public Member Functions

- [EnumType](#) (const [EnumType](#) &other)=default
- [EnumType](#) (const std::vector< [member_def](#) > &t_members)
- [EnumType](#) (std::initializer_list< [member_def](#) > t_members)
- void [commit](#) (const [Object](#) &object, const std::string &name) const

Commit datatype into the given [Object](#).

Public Member Functions inherited from [HighFive::DataType](#)

- bool [operator==](#) (const [DataType](#) &other) const
- bool [operator!=](#) (const [DataType](#) &other) const
- [DataTypeClass](#) [getClass](#) () const
Return the fundamental type.
- size_t [getSize](#) () const
Returns the length (in bytes) of this type elements.
- std::string [string](#) () const
Returns a friendly description of the type (e.g. Float32)
- bool [isVariableStr](#) () const
Returns whether the type is a variable-length string.
- bool [isFixedLenStr](#) () const
Returns whether the type is a fixed-length string.
- bool [empty](#) () const noexcept
Check the [DataType](#) was default constructed. Such value might represent auto-detection of the datatype from a buffer.
- bool [isReference](#) () const
Returns whether the type is a [Reference](#).
- [DataTypeCreateProps](#) [getCreatePropertyList](#) () const
Get the list of properties for creation of this [DataType](#).

Public Member Functions inherited from [HighFive::Object](#)

- [Object](#) ([Object](#) &&other) noexcept
- [~Object](#) ()
- bool [isValid](#) () const noexcept
isValid
- hid_t [getId](#) () const noexcept
getId
- [ObjectInfo](#) [getInfo](#) () const
Retrieve several infos about the current object (address, dates, etc)
- [ObjectType](#) [getType](#) () const
Gets the fundamental type of the object (dataset, group, etc)
- bool [operator==](#) (const [Object](#) &other) const noexcept

Additional Inherited Members

Protected Member Functions inherited from [HighFive::DataType](#)

- [Object](#) ([Object](#) &&other) noexcept
- [Object](#) ()
- [Object](#) (const [Object](#) &other)
- [Object](#) (hid_t)

Protected Member Functions inherited from [HighFive::Object](#)

- [Object](#) ()
- [Object](#) (const [Object](#) &other)
- [Object](#) (hid_t)
- [Object](#) & [operator=](#) (const [Object](#) &other)

Protected Attributes inherited from [HighFive::Object](#)

- hid_t [_hid](#)

9.24.1 Detailed Description

```
template<typename T>
class HighFive::EnumType< T >
```

Create a enum HDF5 datatype.

```
enum class Position {
    FIRST = 1,
    SECOND = 2,
};

EnumType<Position> create_enum_position() {
    return {"FIRST", Position::FIRST},
           {"SECOND", Position::SECOND}};
}

// You have to register the type inside HighFive
HIGHFIVE_REGISTER_TYPE(Position, create_enum_position)

void write_first(H5::File& file) {
    auto dataset = file.createDataSet("/foo", Position::FIRST);
}
```

9.24.2 Constructor & Destructor Documentation

9.24.2.1 EnumType() [1/3]

```
template<typename T >
HighFive::EnumType< T >::EnumType (
    const EnumType< T > & other ) [default]
```

9.24.2.2 EnumType() [2/3]

```
template<typename T >
HighFive::EnumType< T >::EnumType (
    const std::vector< member_def > & t_members ) [inline]
```

9.24.2.3 EnumType() [3/3]

```
template<typename T >
HighFive::EnumType< T >::EnumType (
    std::initializer_list< member_def > t_members ) [inline]
```

9.24.3 Member Function Documentation**9.24.3.1 commit()**

```
template<typename T >
void HighFive::EnumType< T >::commit (
    const Object & object,
    const std::string & name ) const [inline]
```

Commit datatype into the given [Object](#).

Parameters

<i>object</i>	Location to commit object into
<i>name</i>	Name to give the datatype

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

- [highfive/H5DataType.hpp](#)
- [highfive/bits/H5DataType_misc.hpp](#)

9.25 HighFive::EstimatedLinkInfo Class Reference

Set hints as to how many links to expect and their average length.

```
#include <H5PropertyList.hpp>
```

Public Member Functions

- [EstimatedLinkInfo](#) (unsigned entries, unsigned length)
Create a property with the request parameters.
- [EstimatedLinkInfo](#) (const [GroupCreateProps](#) &gcpl)
- unsigned [getEntries](#) () const
The estimated number of links in a group.
- unsigned [getNameLength](#) () const
The estimated length of the names of links.

9.25.1 Detailed Description

Set hints as to how many links to expect and their average length.

9.25.2 Constructor & Destructor Documentation

9.25.2.1 EstimatedLinkInfo() [1/2]

```
HighFive::EstimatedLinkInfo::EstimatedLinkInfo (
    unsigned entries,
    unsigned length ) [inline], [explicit]
```

Create a property with the request parameters.

Parameters

<i>entries</i>	The estimated number of links in a group.
<i>length</i>	The estimated length of the names of links.

9.25.2.2 EstimatedLinkInfo() [2/2]

```
HighFive::EstimatedLinkInfo::EstimatedLinkInfo (
    const GroupCreateProps & gcpl ) [inline], [explicit]
```

9.25.3 Member Function Documentation

9.25.3.1 getEntries()

```
unsigned HighFive::EstimatedLinkInfo::getEntries ( ) const [inline]
```

The estimated number of links in a group.

9.25.3.2 getNameLength()

```
unsigned HighFive::EstimatedLinkInfo::getNameLength ( ) const [inline]
```

The estimated length of the names of links.

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

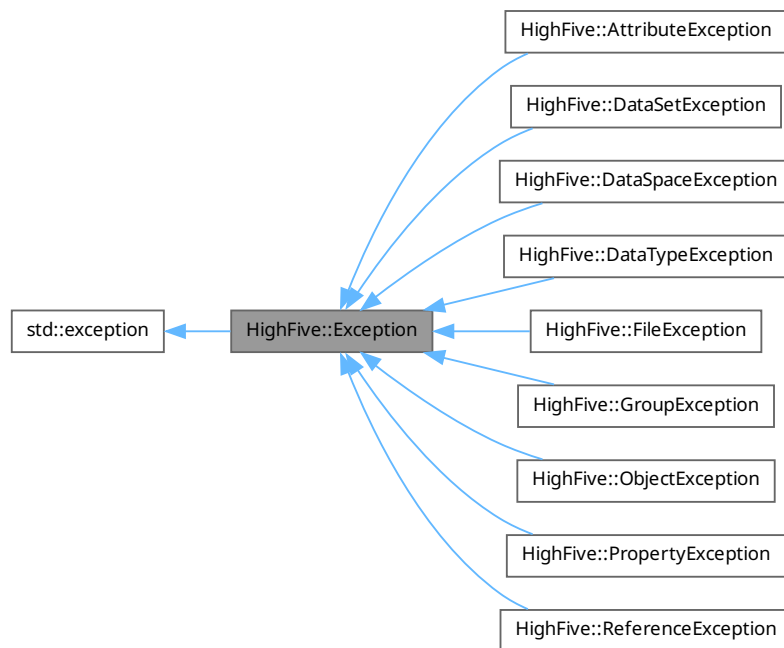
- [highfive/H5PropertyList.hpp](#)
- [highfive/bits/H5PropertyList_misc.hpp](#)

9.26 HighFive::Exception Class Reference

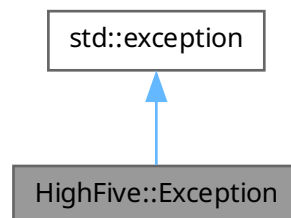
Basic [HighFive Exception](#) class.

```
#include <H5Exception.hpp>
```

Inheritance diagram for HighFive::Exception:



Collaboration diagram for HighFive::Exception:



Public Member Functions

- [Exception](#) (const std::string &err_msg)

- virtual [~Exception](#) () throw ()
- const char * [what](#) () const override throw ()
get the current exception error message
- virtual void [setErrMsg](#) (const std::string &errmsg)
define the error message
- [Exception](#) * [nextException](#) () const
nextException
- hid_t [getErrMajor](#) () const
HDF5 library error mapper.
- hid_t [getErrMinor](#) () const
HDF5 library error mapper.

Protected Attributes

- std::string [_errmsg](#)
- std::shared_ptr< [Exception](#) > [_next](#)
- hid_t [_err_major](#)
- hid_t [_err_minor](#)

Friends

- struct [HDF5ErrMapper](#)

9.26.1 Detailed Description

Basic [HighFive Exception](#) class.

9.26.2 Constructor & Destructor Documentation

9.26.2.1 Exception()

```
HighFive::Exception::Exception (
    const std::string & err_msg ) [inline]
```

9.26.2.2 ~Exception()

```
virtual HighFive::Exception::~~Exception ( ) throw ( ) [inline], [virtual]
```

9.26.3 Member Function Documentation

9.26.3.1 getErrMajor()

```
hid_t HighFive::Exception::getErrMajor ( ) const [inline]
```

HDF5 library error mapper.

Returns

HDF5 major error number

9.26.3.2 getErrMinor()

```
hid_t HighFive::Exception::getErrMinor ( ) const [inline]
```

HDF5 library error mapper.

Returns

HDF5 minor error number

9.26.3.3 nextException()

```
Exception * HighFive::Exception::nextException ( ) const [inline]
```

nextException

Returns

pointer to the next exception in the chain, or NULL if not existing

9.26.3.4 setErrorMsg()

```
virtual void HighFive::Exception::setErrorMsg (
    const std::string & errmsg ) [inline], [virtual]
```

define the error message

Parameters

<i>errmsg</i>	
---------------	--

9.26.3.5 what()

```
const char * HighFive::Exception::what ( ) const throw ( ) [inline], [override]
```

get the current exception error message

Returns

9.26.4 Friends And Related Symbol Documentation

9.26.4.1 HDF5ErrMapper

```
friend struct HDF5ErrMapper [friend]
```

9.26.5 Member Data Documentation

9.26.5.1 `_err_major`

```
hid_t HighFive::Exception::_err_major [protected]
```

9.26.5.2 `_err_minor`

```
hid_t HighFive::Exception::_err_minor [protected]
```

9.26.5.3 `_errmsg`

```
std::string HighFive::Exception::_errmsg [protected]
```

9.26.5.4 `_next`

```
std::shared_ptr<Exception> HighFive::Exception::_next [protected]
```

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

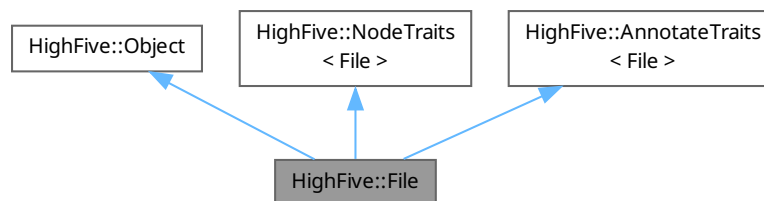
- [highfive/H5Exception.hpp](#)

9.27 HighFive::File Class Reference

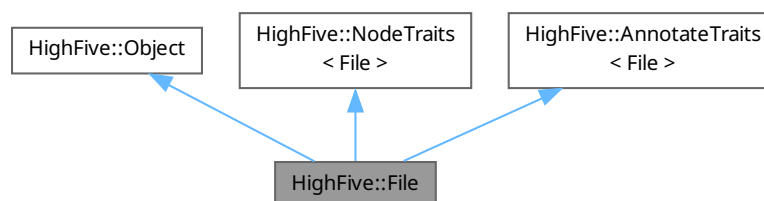
[File](#) class.

```
#include <H5File.hpp>
```

Inheritance diagram for HighFive::File:



Collaboration diagram for HighFive::File:



Public Types

- enum : unsigned {
[ReadOnly](#) = 0x00u , [ReadWrite](#) = 0x01u , [Truncate](#) = 0x02u , [Excl](#) = 0x04u ,
[Debug](#) = 0x08u , [Create](#) = 0x10u , [Overwrite](#) = Truncate , [OpenOrCreate](#) = ReadWrite | Create }

Public Member Functions

- [File](#) (const std::string &filename, unsigned openFlags=[ReadOnly](#), const [FileAccessProps](#) &fileAccessProps=[FileAccessProps::Default](#)())
[File](#).
- [File](#) (const std::string &filename, unsigned openFlags, const [FileCreateProps](#) &fileCreateProps, const [FileAccessProps](#) &fileAccessProps=[FileAccessProps::Default](#)())
[File](#).
- const std::string & [getName](#) () const noexcept
Return the name of the file.
- std::string [getPath](#) () const noexcept
[Object](#) path of a [File](#) is always "/".
- hsize_t [getMetadataBlockSize](#) () const
Returns the block size for metadata in bytes.
- std::pair< H5F_libver_t, H5F_libver_t > [getVersionBounds](#) () const
Returns the HDF5 version compatibility bounds.
- void [flush](#) ()
flush
- [FileCreateProps](#) [getCreatePropertyList](#) () const
Get the list of properties for creation of this file.
- [FileAccessProps](#) [getAccessPropertyList](#) () const
Get the list of properties for accession of this file.

Public Member Functions inherited from [HighFive::Object](#)

- [Object](#) ([Object](#) &&other) noexcept
- [~Object](#) ()
- bool [isValid](#) () const noexcept
[isValid](#)
- hid_t [getId](#) () const noexcept
[getId](#)
- [ObjectInfo](#) [getInfo](#) () const
Retrieve several infos about the current object (address, dates, etc)
- [ObjectType](#) [getType](#) () const
Gets the fundamental type of the object (dataset, group, etc)
- bool [operator==](#) (const [Object](#) &other) const noexcept

Public Member Functions inherited from `HighFive::NodeTraits< File >`

- `DataSet createDataSet` (const std::string &dataset_name, const `DataSetSpace` &space, const `DataSetType` &type, const `DataSetCreateProps` &createProps=`DataSetCreateProps::Default()`, const `DataSetAccessProps` &accessProps=`DataSetAccessProps::Default()`, bool parents=true)
createDataSet Create a new dataset in the current file of datatype type and of size space
- `DataSet createDataSet` (const std::string &dataset_name, const `DataSetSpace` &space, const `DataSetCreateProps` &createProps=`DataSetCreateProps::Default()`, const `DataSetAccessProps` &accessProps=`DataSetAccessProps::Default()`, bool parents=true)
createDataSet create a new dataset in the current file with a size specified by space
- `DataSet createDataSet` (const std::string &dataset_name, const `DataSetSpace` &space, const `DataSetCreateProps` &createProps=`DataSetCreateProps::Default()`, const `DataSetAccessProps` &accessProps=`DataSetAccessProps::Default()`, bool parents=true)
- `DataSet createDataSet` (const std::string &dataset_name, const T &data, const `DataSetCreateProps` &createProps=`DataSetCreateProps::Default()`, const `DataSetAccessProps` &accessProps=`DataSetAccessProps::Default()`, bool parents=true)
createDataSet create a new dataset in the current file and write to it, inferring the DataSetSpace from the data.
- `DataSet createDataSet` (const std::string &dataset_name, const `FixedLenStringArray< N >` &data, const `DataSetCreateProps` &createProps=`DataSetCreateProps::Default()`, const `DataSetAccessProps` &accessProps=`DataSetAccessProps::Default()`, bool parents=true)
- `DataSet getDataSet` (const std::string &dataset_name, const `DataSetAccessProps` &accessProps=`DataSetAccessProps::Default()`) const
get an existing dataset in the current file
- `Group createGroup` (const std::string &group_name, bool parents=true)
create a new group, and eventually intermediate groups
- `Group createGroup` (const std::string &group_name, const `GroupCreateProps` &createProps, bool parents=true)
create a new group, and eventually intermediate groups
- `Group getGroup` (const std::string &group_name) const
open an existing group with the name group_name
- `size_t getNumberObjects` () const
return the number of leaf objects of the node / group
- `std::string getObjectNames` (size_t index) const
return the name of the object with the given index
- `bool rename` (const std::string &src_path, const std::string &dest_path, bool parents=true) const
moves an object and its content within an HDF5 file.
- `std::vector< std::string > listObjectNames` (`IndexType` idx_type=`IndexType::NAME`) const
list all leaf objects name of the node / group
- `bool exist` (const std::string &node_name) const
check a dataset or group exists in the current node / group
- `void unlink` (const std::string &node_name) const
unlink the given dataset or group
- `LinkType getLinkType` (const std::string &node_name) const
Returns the kind of link of the given name (soft, hard...)
- `ObjectType getObjectType` (const std::string &node_name) const
A shorthand to get the kind of object pointed to (group, dataset, type...)
- `void createSoftLink` (const std::string &linkName, const T &obj)
A shorthand to create softlink to any object which provides getPath The link will be created with default properties along with required parent groups.
- `void createSoftLink` (const std::string &link_name, const std::string &obj_path, `LinkCreateProps` linkCreateProps=`LinkCreateProps()`, const `LinkAccessProps` &linkAccessProps=`LinkAccessProps()`, const bool parents=true)
Creates softlinks.
- `void createExternalLink` (const std::string &link_name, const std::string &h5_file, const std::string &obj_path, `LinkCreateProps` linkCreateProps=`LinkCreateProps()`, const `LinkAccessProps` &linkAccessProps=`LinkAccessProps()`, const bool parents=true)

Public Member Functions inherited from HighFive::AnnotateTraits< File >

- [Attribute createAttribute](#) (const std::string &attribute_name, const [DataSpace](#) &space, const [DataType](#) &type)
create a new attribute with the name attribute_name
- [Attribute createAttribute](#) (const std::string &attribute_name, const [DataSpace](#) &space)
createAttribute create a new attribute on the current dataset with size specified by space
- [Attribute createAttribute](#) (const std::string &attribute_name, const T &data)
createAttribute create a new attribute on the current dataset and write to it, inferring the [DataSpace](#) from data.
- void [deleteAttribute](#) (const std::string &attribute_name)
deleteAttribute let you delete an attribute by its name.
- [Attribute getAttribute](#) (const std::string &attribute_name) const
open an existing attribute with the name attribute_name
- size_t [getNumberAttributes](#) () const
return the number of attributes of the node / group
- std::vector< std::string > [listAttributeNames](#) () const
list all attribute name of the node / group
- bool [hasAttribute](#) (const std::string &attr_name) const
checks an attribute exists

Static Public Attributes

- static const [ObjectType](#) type = [ObjectType::File](#)

Protected Member Functions

- [File](#) ()=default
- [Object](#) ([Object](#) &&other) noexcept
- [Object](#) ()
- [Object](#) (const [Object](#) &other)
- [Object](#) (hid_t)

Protected Member Functions inherited from HighFive::Object

- [Object](#) ()
- [Object](#) (const [Object](#) &other)
- [Object](#) (hid_t)
- [Object](#) & [operator=](#) (const [Object](#) &other)

Friends

- template<typename >
class [PathTraits](#)

Additional Inherited Members

Protected Attributes inherited from HighFive::Object

- hid_t [_hid](#)

9.27.1 Detailed Description

[File](#) class.

9.27.2 Member Enumeration Documentation

9.27.2.1 anonymous enum

```
anonymous enum : unsigned
```

Enumerator

ReadOnly	Open flag: Read only access.
ReadWrite	Open flag: Read Write access.
Truncate	Open flag: Truncate a file if already existing.
Excl	Open flag: Open will fail if file already exist.
Debug	Open flag: Open in debug mode.
Create	Open flag: Create non existing file.
Overwrite	Derived open flag: common write mode (=ReadWrite Create Truncate)
OpenOrCreate	Derived open flag: Opens RW or exclusively creates.

9.27.3 Constructor & Destructor Documentation

9.27.3.1 File() [1/3]

```
HighFive::File::File (
    const std::string & filename,
    unsigned openFlags = ReadOnly,
    const FileAccessProps & fileAccessProps = FileAccessProps::Default\(\) ) [inline],
[explicit]
```

[File](#).

Parameters

<i>filename</i>	filepath of the HDF5 file
<i>openFlags</i>	Open mode / flags (ReadOnly , ReadWrite)
<i>fileAccessProps</i>	the file access properties

Open or create a new HDF5 file

9.27.3.2 File() [2/3]

```
HighFive::File::File (
    const std::string & filename,
    unsigned openFlags,
```



```
const FileCreateProps & fileCreateProps,
const FileAccessProps & fileAccessProps = FileAccessProps::Default() ) [inline]
```

File.

Parameters

<i>filename</i>	filepath of the HDF5 file
<i>openFlags</i>	Open mode / flags (ReadOnly, ReadWrite)
<i>fileCreateProps</i>	the file create properties
<i>fileAccessProps</i>	the file access properties

Open or create a new HDF5 file

9.27.3.3 File() [3/3]

```
HighFive::File::File ( ) [protected], [default]
```

9.27.4 Member Function Documentation

9.27.4.1 flush()

```
void HighFive::File::flush ( ) [inline]
```

flush

Flushes all buffers associated with a file to disk

9.27.4.2 getAccessPropertyList()

```
FileAccessProps HighFive::File::getAccessPropertyList ( ) const [inline]
```

Get the list of properties for accession of this file.

9.27.4.3 getCreatePropertyList()

```
FileCreateProps HighFive::File::getCreatePropertyList ( ) const [inline]
```

Get the list of properties for creation of this file.

9.27.4.4 getMetadataBlockSize()

```
hsize_t HighFive::File::getMetadataBlockSize ( ) const [inline]
```

Returns the block size for metadata in bytes.

9.27.4.5 getName()

```
const std::string & HighFive::File::getName ( ) const [inline], [noexcept]
```

Return the name of the file.

9.27.4.6 getPath()

```
std::string HighFive::File::getPath ( ) const [inline], [noexcept]
```

Object path of a File is always `"/"`.

9.27.4.7 getVersionBounds()

```
std::pair< H5F_libver_t, H5F_libver_t > HighFive::File::getVersionBounds ( ) const [inline]
```

Returns the HDF5 version compatibility bounds.

9.27.4.8 Object() [1/4]

```
HighFive::Object::Object ( ) [inline], [protected]
```

9.27.4.9 Object() [2/4]

```
HighFive::Object::Object (
    const Object & other ) [inline], [protected]
```

9.27.4.10 Object() [3/4]

```
HighFive::Object::Object (
    hid_t hid ) [inline], [explicit], [protected]
```

9.27.4.11 Object() [4/4]

```
HighFive::Object::Object (
    Object && other ) [inline], [protected], [noexcept]
```

9.27.5 Friends And Related Symbol Documentation

9.27.5.1 PathTraits

```
template<typename >
friend class PathTraits [friend]
```

9.27.6 Member Data Documentation

9.27.6.1 type

```
const ObjectType HighFive::File::type = ObjectType::File [static]
```

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

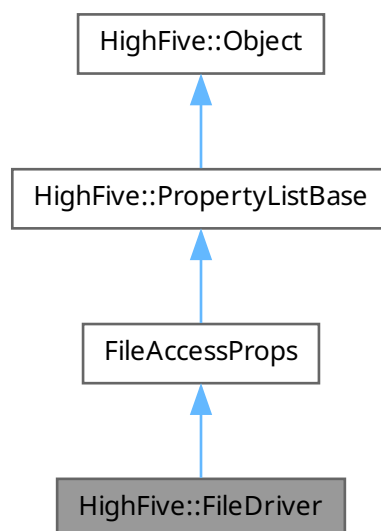
- [highfive/H5File.hpp](#)
- [highfive/bits/H5File_misc.hpp](#)

9.28 HighFive::FileDriver Class Reference

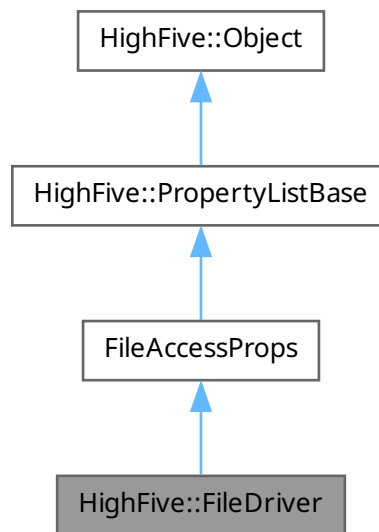
file driver base concept

```
#include <H5FileDriver.hpp>
```

Inheritance diagram for HighFive::FileDriver:



Collaboration diagram for HighFive::FileDriver:



Additional Inherited Members

Public Member Functions inherited from [HighFive::PropertyList< T >](#)

- constexpr [PropertyType](#) [getType](#) () const noexcept
return the type of this [PropertyList](#)
- template<typename P >
void [add](#) (const P &property)

Public Member Functions inherited from [HighFive::PropertyListBase](#)

- [PropertyListBase](#) () noexcept

Public Member Functions inherited from [HighFive::Object](#)

- [Object](#) ([Object](#) &&other) noexcept
- [~Object](#) ()
- bool [isValid](#) () const noexcept
isValid
- hid_t [getId](#) () const noexcept
getId
- [ObjectInfo](#) [getInfo](#) () const
Retrieve several infos about the current object (address, dates, etc)
- [ObjectType](#) [getType](#) () const
Gets the fundamental type of the object (dataset, group, etc)
- bool [operator==](#) (const [Object](#) &other) const noexcept

Static Public Member Functions inherited from [HighFive::PropertyList< T >](#)

- static const [PropertyList< T >](#) & [Default](#) () noexcept
Return the Default property type object.

Static Public Member Functions inherited from [HighFive::PropertyListBase](#)

- static const [PropertyListBase](#) & [Default](#) () noexcept

Protected Member Functions inherited from [HighFive::PropertyList< T >](#)

- void [_initializeIfNeeded](#) ()

Protected Member Functions inherited from [HighFive::Object](#)

- [Object](#) ()
- [Object](#) (const [Object](#) &other)
- [Object](#) (hid_t)
- [Object](#) & [operator=](#) (const [Object](#) &other)

Protected Attributes inherited from [HighFive::Object](#)

- hid_t [_hid](#)

9.28.1 Detailed Description

file driver base concept

Deprecated Use FileAccessProps directly

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

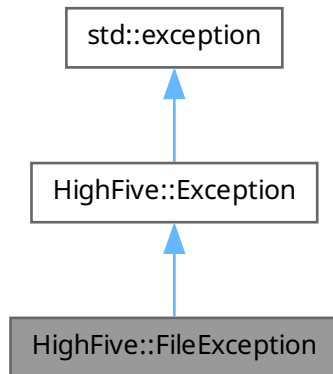
- [highfive/H5FileDriver.hpp](#)

9.29 HighFive::FileException Class Reference

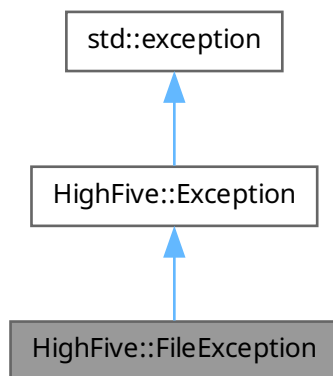
Exception specific to [HighFive File](#) interface.

```
#include <H5Exception.hpp>
```

Inheritance diagram for HighFive::FileException:



Collaboration diagram for HighFive::FileException:



Public Member Functions

- [FileException](#) (const std::string &err_msg)

Public Member Functions inherited from HighFive::Exception

- [Exception](#) (const std::string &err_msg)
- virtual [~Exception](#) () throw ()
- const char * [what](#) () const override throw ()
get the current exception error message
- virtual void [setErrorMsg](#) (const std::string &errmsg)
define the error message
- [Exception](#) * [nextException](#) () const
nextException
- hid_t [getErrMajor](#) () const
HDF5 library error mapper.
- hid_t [getErrMinor](#) () const
HDF5 library error mapper.

Additional Inherited Members

Protected Attributes inherited from HighFive::Exception

- std::string [_errmsg](#)
- std::shared_ptr< [Exception](#) > [_next](#)
- hid_t [_err_major](#)
- hid_t [_err_minor](#)

9.29.1 Detailed Description

[Exception](#) specific to [HighFive File](#) interface.

9.29.2 Constructor & Destructor Documentation

9.29.2.1 FileException()

```
HighFive::FileException::FileException (
    const std::string & err_msg ) [inline]
```

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

- [highfive/H5Exception.hpp](#)

9.30 HighFive::FileVersionBounds Class Reference

Configure the version bounds for the file.

```
#include <H5PropertyList.hpp>
```

Public Member Functions

- [FileVersionBounds](#) (H5F_libver_t low, H5F_libver_t high)
- [FileVersionBounds](#) (const [FileAccessProps](#) &fapl)
- std::pair< H5F_libver_t, H5F_libver_t > [getVersion](#) () const

9.30.1 Detailed Description

Configure the version bounds for the file.

Used to define the compatibility of objects created within HDF5 files, and affects the format of groups stored in the file.

See also the documentation of H5P_SET_LIBVER_BOUNDS in HDF5.

Possible values for low and high are:

- H5F_LIBVER_EARLIEST
- H5F_LIBVER_V18
- H5F_LIBVER_V110
- H5F_LIBVER_NBOUNDS
- H5F_LIBVER_LATEST currently defined as H5F_LIBVER_V110 within HDF5

9.30.2 Constructor & Destructor Documentation

9.30.2.1 FileVersionBounds() [1/2]

```
HighFive::FileVersionBounds::FileVersionBounds (
    H5F_libver_t low,
    H5F_libver_t high ) [inline]
```

9.30.2.2 FileVersionBounds() [2/2]

```
HighFive::FileVersionBounds::FileVersionBounds (
    const FileAccessProps & fapl ) [inline], [explicit]
```

9.30.3 Member Function Documentation

9.30.3.1 getVersion()

```
std::pair< H5F_libver_t, H5F_libver_t > HighFive::FileVersionBounds::getVersion ( ) const
[inline]
```

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

- highfive/H5PropertyList.hpp
- highfive/bits/H5PropertyList_misc.hpp

9.31 HighFive::FixedLenStringArray< N > Class Template Reference

A structure representing a set of fixed-length strings.

```
#include <H5DataType.hpp>
```

Public Types

- using `iterator` = typename vector_t::iterator
- using `const_iterator` = typename vector_t::const_iterator
- using `reverse_iterator` = typename vector_t::reverse_iterator
- using `const_reverse_iterator` = typename vector_t::const_reverse_iterator
- using `value_type` = typename vector_t::value_type

Public Member Functions

- `FixedLenStringArray` ()=default
- `FixedLenStringArray` (const char array[][N], std::size_t length)
Create a FixedStringArray from a raw contiguous buffer.
- `FixedLenStringArray` (const std::vector< std::string > &vec)
Create a FixedStringArray from a sequence of strings.
- `FixedLenStringArray` (const std::string *iter_begin, const std::string *iter_end)
- `FixedLenStringArray` (const std::initializer_list< std::string > &)
- void `push_back` (const std::string &)
Append an std::string to the buffer structure.
- void `push_back` (const std::array< char, N > &)
- std::string `getString` (std::size_t index) const
Retrieve a string from the structure as std::string.
- const char * `operator[]` (std::size_t i) const noexcept
- const char * `at` (std::size_t i) const
- bool `empty` () const noexcept
- std::size_t `size` () const noexcept
- void `resize` (std::size_t n)
- const char * `front` () const
- const char * `back` () const
- char * `data` () noexcept
- const char * `data` () const noexcept
- `iterator` `begin` () noexcept
- `iterator` `end` () noexcept
- `const_iterator` `begin` () const noexcept
- `const_iterator` `cbegin` () const noexcept
- `const_iterator` `end` () const noexcept
- `const_iterator` `cend` () const noexcept
- `reverse_iterator` `rbegin` () noexcept
- `reverse_iterator` `rend` () noexcept
- `const_reverse_iterator` `rbegin` () const noexcept
- `const_reverse_iterator` `rend` () const noexcept

9.31.1 Detailed Description

```
template<std::size_t N>
class HighFive::FixedLenStringArray< N >
```

A structure representing a set of fixed-length strings.

Although fixed-len arrays can be created 'raw' without the need for this structure, to retrieve results efficiently it must be used.

9.31.2 Member Typedef Documentation

9.31.2.1 const_iterator

```
template<std::size_t N>
using HighFive::FixedLenStringArray< N >::const_iterator = typename vector_t::const_iterator
```

9.31.2.2 const_reverse_iterator

```
template<std::size_t N>
using HighFive::FixedLenStringArray< N >::const_reverse_iterator = typename vector_t::const_↵
reverse_iterator
```

9.31.2.3 iterator

```
template<std::size_t N>
using HighFive::FixedLenStringArray< N >::iterator = typename vector_t::iterator
```

9.31.2.4 reverse_iterator

```
template<std::size_t N>
using HighFive::FixedLenStringArray< N >::reverse_iterator = typename vector_t::reverse_↵
iterator
```

9.31.2.5 value_type

```
template<std::size_t N>
using HighFive::FixedLenStringArray< N >::value_type = typename vector_t::value_type
```

9.31.3 Constructor & Destructor Documentation

9.31.3.1 FixedLenStringArray() [1/5]

```
template<std::size_t N>
HighFive::FixedLenStringArray< N >::FixedLenStringArray ( ) [default]
```

9.31.3.2 FixedLenStringArray() [2/5]

```
template<std::size_t N>
HighFive::FixedLenStringArray< N >::FixedLenStringArray (
    const char array[] [N],
    std::size_t length ) [inline]
```

Create a FixedStringArray from a raw contiguous buffer.

9.31.3.3 FixedLenStringArray() [3/5]

```
template<std::size_t N>
HighFive::FixedLenStringArray< N >::FixedLenStringArray (
    const std::vector< std::string > & vec ) [inline], [explicit]
```

Create a FixedStringArray from a sequence of strings.

Such conversion involves a copy, original vector is not modified

9.31.3.4 FixedLenStringArray() [4/5]

```
template<std::size_t N>
HighFive::FixedLenStringArray< N >::FixedLenStringArray (
    const std::string * iter_begin,
    const std::string * iter_end ) [inline]
```

9.31.3.5 FixedLenStringArray() [5/5]

```
template<std::size_t N>
HighFive::FixedLenStringArray< N >::FixedLenStringArray (
    const std::initializer_list< std::string > & init_list ) [inline]
```

9.31.4 Member Function Documentation**9.31.4.1 at()**

```
template<std::size_t N>
const char * HighFive::FixedLenStringArray< N >::at (
    std::size_t i ) const [inline]
```

9.31.4.2 back()

```
template<std::size_t N>
const char * HighFive::FixedLenStringArray< N >::back ( ) const [inline]
```

9.31.4.3 begin() [1/2]

```
template<std::size_t N>
const_iterator HighFive::FixedLenStringArray< N >::begin ( ) const [inline], [noexcept]
```

9.31.4.4 begin() [2/2]

```
template<std::size_t N>
iterator HighFive::FixedLenStringArray< N >::begin ( ) [inline], [noexcept]
```

9.31.4.5 cbegin()

```
template<std::size_t N>
const_iterator HighFive::FixedLenStringArray< N >::cbegin ( ) const [inline], [noexcept]
```

9.31.4.6 cend()

```
template<std::size_t N>
const_iterator HighFive::FixedLenStringArray< N >::cend ( ) const [inline], [noexcept]
```

9.31.4.7 data() [1/2]

```
template<std::size_t N>
const char * HighFive::FixedLenStringArray< N >::data ( ) const [inline], [noexcept]
```

9.31.4.8 data() [2/2]

```
template<std::size_t N>
char * HighFive::FixedLenStringArray< N >::data ( ) [inline], [noexcept]
```

9.31.4.9 empty()

```
template<std::size_t N>
bool HighFive::FixedLenStringArray< N >::empty ( ) const [inline], [noexcept]
```

9.31.4.10 end() [1/2]

```
template<std::size_t N>
const_iterator HighFive::FixedLenStringArray< N >::end ( ) const [inline], [noexcept]
```

9.31.4.11 end() [2/2]

```
template<std::size_t N>
iterator HighFive::FixedLenStringArray< N >::end ( ) [inline], [noexcept]
```

9.31.4.12 front()

```
template<std::size_t N>
const char * HighFive::FixedLenStringArray< N >::front ( ) const [inline]
```

9.31.4.13 getString()

```
template<std::size_t N>
std::string HighFive::FixedLenStringArray< N >::getString (
    std::size_t index ) const [inline]
```

Retrieve a string from the structure as std::string.

9.31.4.14 operator[]()

```
template<std::size_t N>
const char * HighFive::FixedLenStringArray< N >::operator[] (
    std::size_t i ) const [inline], [noexcept]
```

9.31.4.15 push_back() [1/2]

```
template<std::size_t N>
void HighFive::FixedLenStringArray< N >::push_back (
    const std::array< char, N > & src ) [inline]
```

9.31.4.16 push_back() [2/2]

```
template<std::size_t N>
void HighFive::FixedLenStringArray< N >::push_back (
    const std::string & src ) [inline]
```

Append an std::string to the buffer structure.

9.31.4.17 rbegin() [1/2]

```
template<std::size_t N>
const_reverse_iterator HighFive::FixedLenStringArray< N >::rbegin ( ) const [inline], [noexcept]
```

9.31.4.18 rbegin() [2/2]

```
template<std::size_t N>
reverse_iterator HighFive::FixedLenStringArray< N >::rbegin ( ) [inline], [noexcept]
```

9.31.4.19 rend() [1/2]

```
template<std::size_t N>
const_reverse_iterator HighFive::FixedLenStringArray< N >::rend ( ) const [inline], [noexcept]
```

9.31.4.20 `rend()` [2/2]

```
template<std::size_t N>
reverse_iterator HighFive::FixedLenStringArray< N >::rend ( ) [inline], [noexcept]
```

9.31.4.21 `resize()`

```
template<std::size_t N>
void HighFive::FixedLenStringArray< N >::resize (
    std::size_t n ) [inline]
```

9.31.4.22 `size()`

```
template<std::size_t N>
std::size_t HighFive::FixedLenStringArray< N >::size ( ) const [inline], [noexcept]
```

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

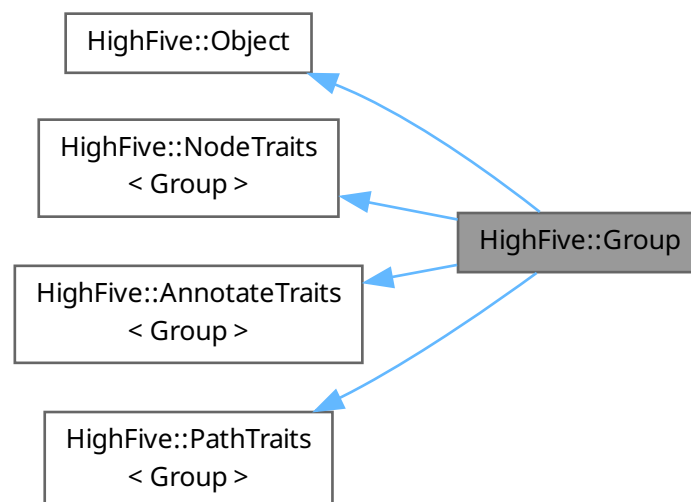
- [highfive/bits/H5_definitions.hpp](#)
- [highfive/H5DataType.hpp](#)
- [highfive/bits/H5DataType_misc.hpp](#)

9.32 HighFive::Group Class Reference

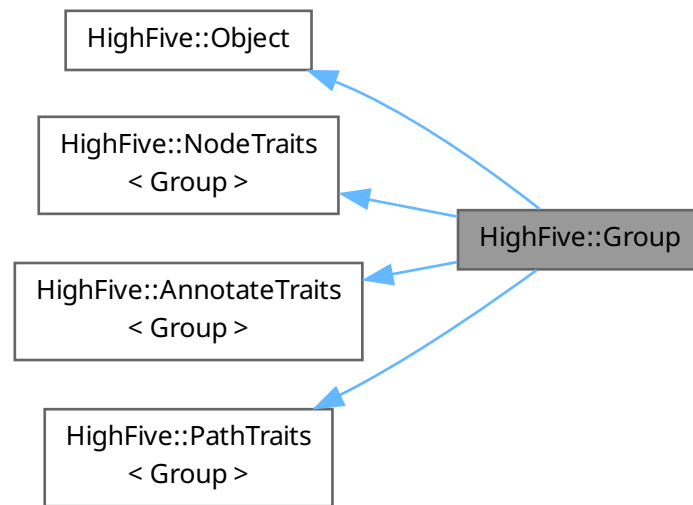
Represents an hdf5 group.

```
#include <H5Group.hpp>
```

Inheritance diagram for HighFive::Group:



Collaboration diagram for HighFive::Group:



Public Member Functions

- `Group ()`=default
- `std::pair< unsigned int, unsigned int > getEstimatedLinkInfo ()` const
- `GroupCreateProps getCreatePropertyList ()` const
Get the list of properties for creation of this group.
- `Group (Object &&o)` noexcept

Public Member Functions inherited from `HighFive::Object`

- `Object (Object &&other)` noexcept
- `~Object ()`
- `bool isValid ()` const noexcept
isValid
- `hid_t getId ()` const noexcept
getId
- `ObjectInfo getInfo ()` const
Retrieve several infos about the current object (address, dates, etc)
- `ObjectType getType ()` const
Gets the fundamental type of the object (dataset, group, etc)
- `bool operator== (const Object &other)` const noexcept

Public Member Functions inherited from `HighFive::NodeTraits< Group >`

- `DataSet createDataSet` (const std::string &dataset_name, const `DataSetSpace` &space, const `DataSetType` &type, const `DataSetCreateProps` &createProps=`DataSetCreateProps::Default()`, const `DataSetAccessProps` &accessProps=`DataSetAccessProps::Default()`, bool parents=true)
createDataSet Create a new dataset in the current file of datatype type and of size space
- `DataSet createDataSet` (const std::string &dataset_name, const `DataSetSpace` &space, const `DataSetCreateProps` &createProps=`DataSetCreateProps::Default()`, const `DataSetAccessProps` &accessProps=`DataSetAccessProps::Default()`, bool parents=true)
createDataSet create a new dataset in the current file with a size specified by space
- `DataSet createDataSet` (const std::string &dataset_name, const `DataSetSpace` &space, const `DataSetCreateProps` &createProps=`DataSetCreateProps::Default()`, const `DataSetAccessProps` &accessProps=`DataSetAccessProps::Default()`, bool parents=true)
- `DataSet createDataSet` (const std::string &dataset_name, const T &data, const `DataSetCreateProps` &createProps=`DataSetCreateProps::Default()`, const `DataSetAccessProps` &accessProps=`DataSetAccessProps::Default()`, bool parents=true)
createDataSet create a new dataset in the current file and write to it, inferring the DataSetSpace from the data.
- `DataSet createDataSet` (const std::string &dataset_name, const `FixedLenStringArray< N >` &data, const `DataSetCreateProps` &createProps=`DataSetCreateProps::Default()`, const `DataSetAccessProps` &accessProps=`DataSetAccessProps::Default()`, bool parents=true)
- `DataSet getDataSet` (const std::string &dataset_name, const `DataSetAccessProps` &accessProps=`DataSetAccessProps::Default()`) const
get an existing dataset in the current file
- `Group createGroup` (const std::string &group_name, bool parents=true)
create a new group, and eventually intermediate groups
- `Group createGroup` (const std::string &group_name, const `GroupCreateProps` &createProps, bool parents=true)
create a new group, and eventually intermediate groups
- `Group getGroup` (const std::string &group_name) const
open an existing group with the name group_name
- `size_t getNumberObjects` () const
return the number of leaf objects of the node / group
- `std::string getObjectNames` (size_t index) const
return the name of the object with the given index
- `bool rename` (const std::string &src_path, const std::string &dest_path, bool parents=true) const
moves an object and its content within an HDF5 file.
- `std::vector< std::string > listObjectNames` (`IndexType` idx_type=`IndexType::NAME`) const
list all leaf objects name of the node / group
- `bool exist` (const std::string &node_name) const
check a dataset or group exists in the current node / group
- `void unlink` (const std::string &node_name) const
unlink the given dataset or group
- `LinkType getLinkType` (const std::string &node_name) const
Returns the kind of link of the given name (soft, hard...)
- `ObjectType getObjectType` (const std::string &node_name) const
A shorthand to get the kind of object pointed to (group, dataset, type...)
- `void createSoftLink` (const std::string &linkName, const T &obj)
A shorthand to create softlink to any object which provides getPath The link will be created with default properties along with required parent groups.
- `void createSoftLink` (const std::string &link_name, const std::string &obj_path, `LinkCreateProps` linkCreateProps=`LinkCreateProps()`, const `LinkAccessProps` &linkAccessProps=`LinkAccessProps()`, const bool parents=true)
Creates softlinks.
- `void createExternalLink` (const std::string &link_name, const std::string &h5_file, const std::string &obj_path, `LinkCreateProps` linkCreateProps=`LinkCreateProps()`, const `LinkAccessProps` &linkAccessProps=`LinkAccessProps()`, const bool parents=true)

Public Member Functions inherited from [HighFive::AnnotateTraits< Group >](#)

- [Attribute createAttribute](#) (const std::string &attribute_name, const [DataSpace](#) &space, const [DataType](#) &type)
create a new attribute with the name attribute_name
- [Attribute createAttribute](#) (const std::string &attribute_name, const [DataSpace](#) &space)
createAttribute create a new attribute on the current dataset with size specified by space
- [Attribute createAttribute](#) (const std::string &attribute_name, const T &data)
createAttribute create a new attribute on the current dataset and write to it, inferring the [DataSpace](#) from data.
- void [deleteAttribute](#) (const std::string &attribute_name)
deleteAttribute let you delete an attribute by its name.
- [Attribute getAttribute](#) (const std::string &attribute_name) const
open an existing attribute with the name attribute_name
- size_t [getNumberAttributes](#) () const
return the number of attributes of the node / group
- std::vector< std::string > [listAttributeNames](#) () const
list all attribute name of the node / group
- bool [hasAttribute](#) (const std::string &attr_name) const
checks an attribute exists

Public Member Functions inherited from [HighFive::PathTraits< Group >](#)

- [PathTraits](#) ()
- std::string [getPath](#) () const
return the path to the current object
- [File](#) & [getFile](#) () const noexcept
Return a reference to the [File](#) object this object belongs.

Static Public Attributes

- static const [ObjectType](#) type = [ObjectType::Group](#)

Protected Member Functions

- [Object](#) ([Object](#) &&other) noexcept
- [Object](#) ()
- [Object](#) (const [Object](#) &other)
- [Object](#) (hid_t)

Protected Member Functions inherited from [HighFive::Object](#)

- [Object](#) ()
- [Object](#) (const [Object](#) &other)
- [Object](#) (hid_t)
- [Object](#) & [operator=](#) (const [Object](#) &other)

Friends

- class [File](#)
- class [Reference](#)

Additional Inherited Members

Protected Attributes inherited from [HighFive::Object](#)

- [hid_t _hid](#)

Protected Attributes inherited from [HighFive::PathTraits< Group >](#)

- `std::shared_ptr< File > _file_obj`

9.32.1 Detailed Description

Represents an hdf5 group.

9.32.2 Constructor & Destructor Documentation

9.32.2.1 Group() [1/2]

```
HighFive::Group::Group ( ) [default]
```

Deprecated Default constructor creates unsafe uninitialized objects

9.32.2.2 Group() [2/2]

```
HighFive::Group::Group (
    Object && o ) [inline], [noexcept]
```

9.32.3 Member Function Documentation

9.32.3.1 getCreatePropertyList()

```
GroupCreateProps HighFive::Group::getCreatePropertyList ( ) const [inline]
```

Get the list of properties for creation of this group.

9.32.3.2 getEstimatedLinkInfo()

```
std::pair< unsigned int, unsigned int > HighFive::Group::getEstimatedLinkInfo ( ) const [inline]
```

9.32.3.3 Object() [1/4]

```
HighFive::Object::Object ( ) [inline], [protected]
```

9.32.3.4 Object() [2/4]

```
HighFive::Object::Object (
    const Object & other ) [inline], [protected]
```

9.32.3.5 Object() [3/4]

```
HighFive::Object::Object (
    hid_t hid ) [inline], [explicit], [protected]
```

9.32.3.6 Object() [4/4]

```
HighFive::Object::Object (
    Object && other ) [inline], [protected], [noexcept]
```

9.32.4 Friends And Related Symbol Documentation

9.32.4.1 File

```
friend class File [friend]
```

9.32.4.2 Reference

```
friend class Reference [friend]
```

9.32.5 Member Data Documentation

9.32.5.1 type

```
const ObjectType HighFive::Group::type = ObjectType::Group [static]
```

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

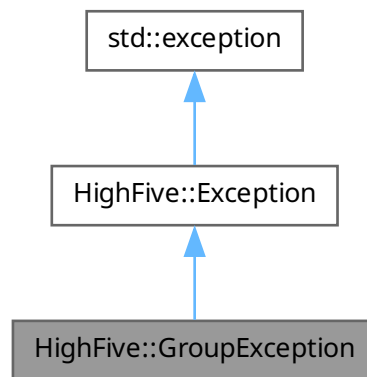
- [highfive/H5Group.hpp](#)

9.33 HighFive::GroupException Class Reference

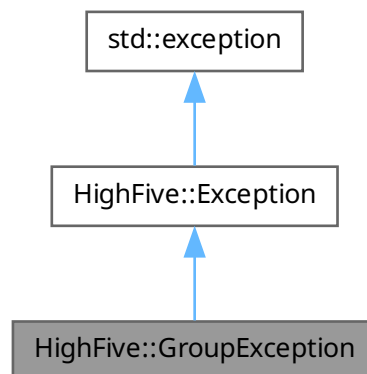
Exception specific to [HighFive Group](#) interface.

```
#include <H5Exception.hpp>
```

Inheritance diagram for HighFive::GroupException:



Collaboration diagram for HighFive::GroupException:



Public Member Functions

- [GroupException](#) (const std::string &err_msg)

Public Member Functions inherited from [HighFive::Exception](#)

- [Exception](#) (const std::string &err_msg)
- virtual [~Exception](#) () throw ()
- const char * [what](#) () const override throw ()
get the current exception error message
- virtual void [setErrorMsg](#) (const std::string &errmsg)
define the error message
- [Exception](#) * [nextException](#) () const
nextException
- hid_t [getErrMajor](#) () const
HDF5 library error mapper.
- hid_t [getErrMinor](#) () const
HDF5 library error mapper.

Additional Inherited Members

Protected Attributes inherited from [HighFive::Exception](#)

- std::string [_errmsg](#)
- std::shared_ptr< [Exception](#) > [_next](#)
- hid_t [_err_major](#)
- hid_t [_err_minor](#)

9.33.1 Detailed Description

[Exception](#) specific to [HighFive Group](#) interface.

9.33.2 Constructor & Destructor Documentation

9.33.2.1 GroupException()

```
HighFive::GroupException::GroupException (
    const std::string & err_msg ) [inline]
```

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

- [highfive/H5Exception.hpp](#)

9.34 HighFive::HDF5ErrMapper Struct Reference

```
#include <H5Exception_misc.hpp>
```

Static Public Member Functions

- `template<typename ExceptionType >`
`static herr_t stackWalk (unsigned n, const H5E_error2_t *err_desc, void *client_data)`
- `template<typename ExceptionType >`
`static void ToException (const std::string &prefix_msg)`

9.34.1 Member Function Documentation

9.34.1.1 `stackWalk()`

```
template<typename ExceptionType >
static herr_t HighFive::HDF5ErrMapper::stackWalk (
    unsigned n,
    const H5E_error2_t * err_desc,
    void * client_data ) [inline], [static]
```

9.34.1.2 `ToException()`

```
template<typename ExceptionType >
static void HighFive::HDF5ErrMapper::ToException (
    const std::string & prefix_msg ) [inline], [static]
```

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

- `highfive/bits/H5Exception\_misc.hpp`

9.35 HighFive::HyperSlab Class Reference

```
#include <H5Slice_traits.hpp>
```

Public Member Functions

- `HyperSlab ()`
- `HyperSlab (const RegularHyperSlab &sel)`
- `HyperSlab operator| (const RegularHyperSlab &sel) const`
- `HyperSlab & operator|= (const RegularHyperSlab &sel)`
- `HyperSlab operator& (const RegularHyperSlab &sel) const`
- `HyperSlab & operator&= (const RegularHyperSlab &sel)`
- `HyperSlab operator^ (const RegularHyperSlab &sel) const`
- `HyperSlab & operator^= (const RegularHyperSlab &sel)`
- `HyperSlab & notA (const RegularHyperSlab &sel)`
- `HyperSlab & notB (const RegularHyperSlab &sel)`
- `DataSpace apply (const DataSpace &space_) const`

9.35.1 Constructor & Destructor Documentation

9.35.1.1 HyperSlab() [1/2]

```
HighFive::HyperSlab::HyperSlab ( ) [inline]
```

9.35.1.2 HyperSlab() [2/2]

```
HighFive::HyperSlab::HyperSlab (
    const RegularHyperSlab & sel ) [inline], [explicit]
```

9.35.2 Member Function Documentation

9.35.2.1 apply()

```
DataSpace HighFive::HyperSlab::apply (
    const DataSpace & space_ ) const [inline]
```

9.35.2.2 notA()

```
HyperSlab & HighFive::HyperSlab::notA (
    const RegularHyperSlab & sel ) [inline]
```

9.35.2.3 notB()

```
HyperSlab & HighFive::HyperSlab::notB (
    const RegularHyperSlab & sel ) [inline]
```

9.35.2.4 operator&()

```
HyperSlab HighFive::HyperSlab::operator& (
    const RegularHyperSlab & sel ) const [inline]
```

9.35.2.5 operator&=()

```
HyperSlab & HighFive::HyperSlab::operator&= (
    const RegularHyperSlab & sel ) [inline]
```

9.35.2.6 operator^()

```
HyperSlab HighFive::HyperSlab::operator^ (
    const RegularHyperSlab & sel ) const [inline]
```

9.35.2.7 `operator^=()`

```
HyperSlab & HighFive::HyperSlab::operator^= (
    const RegularHyperSlab & sel ) [inline]
```

9.35.2.8 `operator" |()`

```
HyperSlab HighFive::HyperSlab::operator| (
    const RegularHyperSlab & sel ) const [inline]
```

9.35.2.9 `operator" |=()`

```
HyperSlab & HighFive::HyperSlab::operator|= (
    const RegularHyperSlab & sel ) [inline]
```

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

- [highfive/bits/H5Slice_traits.hpp](#)

9.36 HighFive::LinkCreationOrder Class Reference

Track and index creation order time.

```
#include <H5PropertyList.hpp>
```

Public Member Functions

- [LinkCreationOrder](#) (unsigned flags)
Create the property.
- [LinkCreationOrder](#) (const [FileCreateProps](#) &fcpl)
- [LinkCreationOrder](#) (const [GroupCreateProps](#) &gcpl)
- unsigned [getFlags](#) () const

Protected Member Functions

- void [fromPropertyList](#) (hid_t hid)

9.36.1 Detailed Description

Track and index creation order time.

Let user retrieve objects by creation order time instead of name.

9.36.2 Constructor & Destructor Documentation

9.36.2.1 `LinkCreationOrder()` [1/3]

```
HighFive::LinkCreationOrder::LinkCreationOrder (
    unsigned flags ) [inline], [explicit]
```

Create the property.

Parameters

<i>flags</i>	Should be a composition of HighFive::CreationOrder .
--------------	--

9.36.2.2 LinkCreationOrder() [2/3]

```
HighFive::LinkCreationOrder::LinkCreationOrder (
    const FileCreateProps & fcpl ) [inline], [explicit]
```

9.36.2.3 LinkCreationOrder() [3/3]

```
HighFive::LinkCreationOrder::LinkCreationOrder (
    const GroupCreateProps & gcpl ) [explicit]
```

9.36.3 Member Function Documentation**9.36.3.1 fromPropertyList()**

```
void HighFive::LinkCreationOrder::fromPropertyList (
    hid_t hid ) [inline], [protected]
```

9.36.3.2 getFlags()

```
unsigned HighFive::LinkCreationOrder::getFlags ( ) const [inline]
```

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

- [highfive/H5PropertyList.hpp](#)
- [highfive/bits/H5PropertyList_misc.hpp](#)

9.37 HighFive::Logger Class Reference

A logger with supporting basic functionality.

```
#include <H5Utility.hpp>
```

Public Types

- using [callback_type](#) = std::function< void([LogSeverity](#), const std::string &, const std::string &, int)>

Public Member Functions

- [Logger](#) ()=delete
- [Logger](#) (const [Logger](#) &)=delete
- [Logger](#) ([Logger](#) &&)=delete
- [Logger](#) ([callback_type](#) cb)
- [Logger](#) & [operator=](#) (const [Logger](#) &)=delete
- [Logger](#) & [operator=](#) ([Logger](#) &&)=delete
- void [log](#) ([LogSeverity](#) severity, const std::string &message, const std::string &file, int line)
- void [set_logging_callback](#) ([callback_type](#) cb)

9.37.1 Detailed Description

A logger with supporting basic functionality.

This logger delegates the logging task to a callback. This level of indirection enables using the default Python logger from C++; or integrating [HighFive](#) into some custom logging solution.

Using this class directly to log is not intended. Rather you should use

- `HIGHFIVE_LOG_DEBUG{, _IF}`
- `HIGHFIVE_LOG_INFO{, _IF}`
- `HIGHFIVE_LOG_WARNING{, _IF}`
- `HIGHFIVE_LOG_ERROR{, _IF}`

This is intended to be used as a singleton, via [get_global_logger\(\)](#).

9.37.2 Member Typedef Documentation

9.37.2.1 `callback_type`

```
using HighFive::Logger::callback_type = std::function<void(LogSeverity, const std::string&,
const std::string&, int)>
```

9.37.3 Constructor & Destructor Documentation

9.37.3.1 `Logger()` [1/4]

```
HighFive::Logger::Logger ( ) [delete]
```

9.37.3.2 `Logger()` [2/4]

```
HighFive::Logger::Logger (
    const Logger & ) [delete]
```

9.37.3.3 Logger() [3/4]

```
HighFive::Logger::Logger (
    Logger && ) [delete]
```

9.37.3.4 Logger() [4/4]

```
HighFive::Logger::Logger (
    callback_type cb ) [inline], [explicit]
```

9.37.4 Member Function Documentation

9.37.4.1 log()

```
void HighFive::Logger::log (
    LogSeverity severity,
    const std::string & message,
    const std::string & file,
    int line ) [inline]
```

9.37.4.2 operator=() [1/2]

```
Logger & HighFive::Logger::operator= (
    const Logger & ) [delete]
```

9.37.4.3 operator=() [2/2]

```
Logger & HighFive::Logger::operator= (
    Logger && ) [delete]
```

9.37.4.4 set_logging_callback()

```
void HighFive::Logger::set_logging_callback (
    callback_type cb ) [inline]
```

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

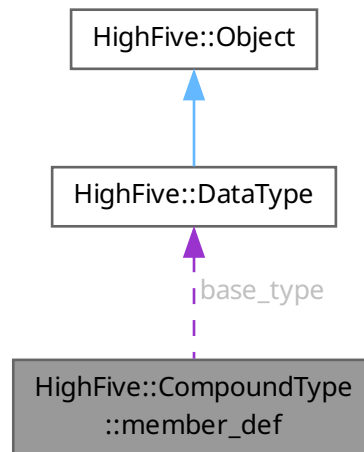
- [highfive/H5Utility.hpp](#)

9.38 HighFive::CompoundType::member_def Struct Reference

Use for defining a sub-type of compound type.

```
#include <H5DataType.hpp>
```

Collaboration diagram for HighFive::CompoundType::member_def:



Public Member Functions

- [member_def](#) (std::string t_name, [DataType](#) t_base_type, size_t t_offset=0)

Public Attributes

- std::string [name](#)
- [DataType](#) [base_type](#)
- size_t [offset](#)

9.38.1 Detailed Description

Use for defining a sub-type of compound type.

9.38.2 Constructor & Destructor Documentation

9.38.2.1 member_def()

```
HighFive::CompoundType::member_def::member_def (
    std::string t_name,
    DataType t_base_type,
    size_t t_offset = 0 ) [inline]
```

9.38.3 Member Data Documentation

9.38.3.1 base_type

`DataType` HighFive::CompoundType::member_def::base_type

9.38.3.2 name

`std::string` HighFive::CompoundType::member_def::name

9.38.3.3 offset

`size_t` HighFive::CompoundType::member_def::offset

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

- [highfive/H5DataType.hpp](#)

9.39 HighFive::EnumType< T >::member_def Struct Reference

Use for defining a member of enum type.

```
#include <H5DataType.hpp>
```

Public Member Functions

- [member_def](#) (const std::string &t_name, T t_value)

Public Attributes

- std::string [name](#)
- T [value](#)

9.39.1 Detailed Description

```
template<typename T>
struct HighFive::EnumType< T >::member_def
```

Use for defining a member of enum type.

9.39.2 Constructor & Destructor Documentation

9.39.2.1 member_def()

```
template<typename T >
HighFive::EnumType< T >::member_def::member_def (
    const std::string & t_name,
    T t_value )    [inline]
```

9.39.3 Member Data Documentation

9.39.3.1 name

```
template<typename T >
std::string HighFive::EnumType< T >::member_def::name
```

9.39.3.2 value

```
template<typename T >
T HighFive::EnumType< T >::member_def::value
```

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

- [highfive/H5DataType.hpp](#)

9.40 HighFive::MetadataBlockSize Class Reference

Configure the metadata block size to use writing to files.

```
#include <H5PropertyList.hpp>
```

Public Member Functions

- [MetadataBlockSize](#) (hsize_t size)
- [MetadataBlockSize](#) (const [FileAccessProps](#) &fapl)
- hsize_t [getSize](#) () const

9.40.1 Detailed Description

Configure the metadata block size to use writing to files.

Parameters

<i>size</i>	Metadata block size in bytes
-------------	------------------------------

9.40.2 Constructor & Destructor Documentation

9.40.2.1 MetadataBlockSize() [1/2]

```
HighFive::MetadataBlockSize::MetadataBlockSize (
    hsize_t size ) [inline], [explicit]
```

9.40.2.2 MetadataBlockSize() [2/2]

```
HighFive::MetadataBlockSize::MetadataBlockSize (
    const FileAccessProps & fapl ) [inline], [explicit]
```

9.40.3 Member Function Documentation

9.40.3.1 getSize()

```
hsize_t HighFive::MetadataBlockSize::getSize ( ) const [inline]
```

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

- [highfive/H5PropertyList.hpp](#)
- [highfive/bits/H5PropertyList_misc.hpp](#)

9.41 HighFive::MPIOCollectiveMetadata Class Reference

Use collective MPI-IO for metadata read and write.

```
#include <H5PropertyList.hpp>
```

Public Member Functions

- [MPIOCollectiveMetadata](#) (bool collective=true)
- [MPIOCollectiveMetadata](#) (const [FileAccessProps](#) &plist)
- bool [isCollectiveRead](#) () const
- bool [isCollectiveWrite](#) () const

9.41.1 Detailed Description

Use collective MPI-IO for metadata read and write.

See [MPIOCollectiveMetadataRead](#) and [MPIOCollectiveMetadataWrite](#).

9.41.2 Constructor & Destructor Documentation

9.41.2.1 MPIOCollectiveMetadata() [1/2]

```
HighFive::MPIOCollectiveMetadata::MPIOCollectiveMetadata (
    bool collective = true ) [inline], [explicit]
```

9.41.2.2 MPIOCollectiveMetadata() [2/2]

```
HighFive::MPIOCollectiveMetadata::MPIOCollectiveMetadata (
    const FileAccessProps & plist ) [inline], [explicit]
```

9.41.3 Member Function Documentation

9.41.3.1 isCollectiveRead()

```
bool HighFive::MPIOCollectiveMetadata::isCollectiveRead ( ) const [inline]
```

9.41.3.2 isCollectiveWrite()

```
bool HighFive::MPIOCollectiveMetadata::isCollectiveWrite ( ) const [inline]
```

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

- [highfive/H5PropertyList.hpp](#)
- [highfive/bits/H5PropertyList_misc.hpp](#)

9.42 HighFive::MPIOCollectiveMetadataRead Class Reference

Use collective MPI-IO for metadata read?

```
#include <H5PropertyList.hpp>
```

Public Member Functions

- [MPIOCollectiveMetadataRead](#) (bool *collective*=true)
- [MPIOCollectiveMetadataRead](#) (const [FileAccessProps](#) &*plist*)
- bool [isCollective](#) () const

9.42.1 Detailed Description

Use collective MPI-IO for metadata read?

Note that when used in a file access property list, this will force all reads of meta data to be collective. HDF5 function may implicitly perform metadata reads. These functions would become collective. A list of functions that perform metadata reads can be found in the HDF5 documentation, e.g. https://docs.hdfgroup.org/hdf5/v1_12/group___g_a_c_p_1.html

In [HighFive](#) setting collective read is (currently) only supported on file level.

Please also consult upstream documentation of `H5Pset_all_coll_metadata_ops`.

9.42.2 Constructor & Destructor Documentation

9.42.2.1 MPIOCollectiveMetadataRead() [1/2]

```
HighFive::MPIOCollectiveMetadataRead::MPIOCollectiveMetadataRead (
    bool collective = true ) [inline], [explicit]
```

9.42.2.2 MPIOCollectiveMetadataRead() [2/2]

```
HighFive::MPIOCollectiveMetadataRead::MPIOCollectiveMetadataRead (
    const FileAccessProps & plist ) [inline], [explicit]
```

9.42.3 Member Function Documentation

9.42.3.1 isCollective()

```
bool HighFive::MPIOCollectiveMetadataRead::isCollective ( ) const [inline]
```

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

- [highfive/H5PropertyList.hpp](#)
- [highfive/bits/H5PropertyList_misc.hpp](#)

9.43 HighFive::MPIOCollectiveMetadataWrite Class Reference

Use collective MPI-IO for metadata write?

```
#include <H5PropertyList.hpp>
```

Public Member Functions

- [MPIOCollectiveMetadataWrite](#) (bool collective=true)
- [MPIOCollectiveMetadataWrite](#) (const [FileAccessProps](#) &plist)
- bool [isCollective](#) () const

9.43.1 Detailed Description

Use collective MPI-IO for metadata write?

In order to keep the in-memory representation of the file structure consistent across MPI ranks, writing meta data is always a collective operation. Meaning all MPI ranks must participate. Passing this setting enables using MPI-IO collective operations for metadata writes.

Please also consult upstream documentation of `H5Pset_coll_metadata_write`.

9.43.2 Constructor & Destructor Documentation

9.43.2.1 MPIOCollectiveMetadataWrite() [1/2]

```
HighFive::MPIOCollectiveMetadataWrite::MPIOCollectiveMetadataWrite (
    bool collective = true ) [inline], [explicit]
```

9.43.2.2 MPIOCollectiveMetadataWrite() [2/2]

```
HighFive::MPIOCollectiveMetadataWrite::MPIOCollectiveMetadataWrite (
    const FileAccessProps & plist ) [inline], [explicit]
```

9.43.3 Member Function Documentation

9.43.3.1 isCollective()

```
bool HighFive::MPIOCollectiveMetadataWrite::isCollective ( ) const [inline]
```

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

- [highfive/H5PropertyList.hpp](#)
- [highfive/bits/H5PropertyList_misc.hpp](#)

9.44 HighFive::MPIOFileAccess Class Reference

Configure MPI access for the file.

```
#include <H5PropertyList.hpp>
```

Public Member Functions

- [MPIOFileAccess](#) (MPI_Comm comm, MPI_Info info)

9.44.1 Detailed Description

Configure MPI access for the file.

All further modifications to the structure of the file will have to be done with collective operations

9.44.2 Constructor & Destructor Documentation

9.44.2.1 MPIOFileAccess()

```
HighFive::MPIOFileAccess::MPIOFileAccess (
    MPI_Comm comm,
    MPI_Info info ) [inline]
```

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

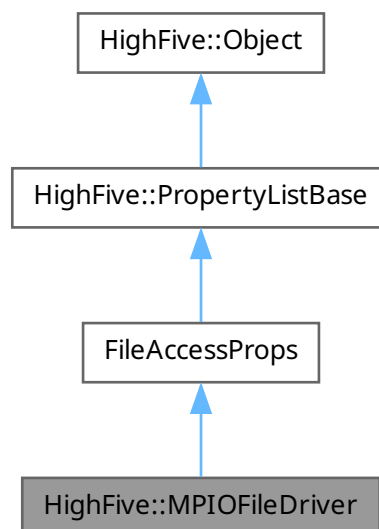
- [highfive/H5PropertyList.hpp](#)
- [highfive/bits/H5PropertyList_misc.hpp](#)

9.45 HighFive::MPIOFileDriver Class Reference

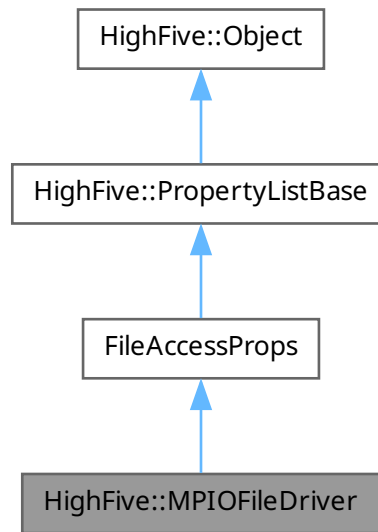
MPIIO Driver for Parallel HDF5.

```
#include <H5FileDriver.hpp>
```

Inheritance diagram for HighFive::MPIOFileDriver:



Collaboration diagram for HighFive::MPIOFilerDriver:



Public Member Functions

- [MPIOFilerDriver](#) (MPI_Comm mpi_comm, MPI_Info mpi_info)

Public Member Functions inherited from [HighFive::PropertyList< T >](#)

- constexpr [PropertyType](#) [getType](#) () const noexcept
return the type of this [PropertyList](#)
- template<typename P >
void [add](#) (const P &property)

Public Member Functions inherited from [HighFive::PropertyListBase](#)

- [PropertyListBase](#) () noexcept

Public Member Functions inherited from [HighFive::Object](#)

- [Object](#) ([Object](#) &&other) noexcept
- [~Object](#) ()
- bool [isValid](#) () const noexcept
isValid
- hid_t [getId](#) () const noexcept
getId
- [ObjectInfo](#) [getInfo](#) () const
Retrieve several infos about the current object (address, dates, etc)
- [ObjectType](#) [getType](#) () const
Gets the fundamental type of the object (dataset, group, etc)
- bool [operator==](#) (const [Object](#) &other) const noexcept

Additional Inherited Members

Static Public Member Functions inherited from [HighFive::PropertyList< T >](#)

- static const [PropertyList< T >](#) & [Default](#) () noexcept
Return the Default property type object.

Static Public Member Functions inherited from [HighFive::PropertyListBase](#)

- static const [PropertyListBase](#) & [Default](#) () noexcept

Protected Member Functions inherited from [HighFive::PropertyList< T >](#)

- void [_initializeIfNeeded](#) ()

Protected Member Functions inherited from [HighFive::Object](#)

- [Object](#) ()
- [Object](#) (const [Object](#) &other)
- [Object](#) (hid_t)
- [Object](#) & [operator=](#) (const [Object](#) &other)

Protected Attributes inherited from [HighFive::Object](#)

- hid_t [_hid](#)

9.45.1 Detailed Description

MPIIO Driver for Parallel HDF5.

Deprecated Add [MPIOFileAccess](#) directly to FileAccessProps

9.45.2 Constructor & Destructor Documentation

9.45.2.1 MPIOFileDriver()

```
HighFive::MPIOFileDriver::MPIOFileDriver (
    MPI_Comm mpi_comm,
    MPI_Info mpi_info ) [inline]
```

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

- highfive/H5FileDriver.hpp
- highfive/bits/H5FileDriver_misc.hpp

9.46 HighFive::MpioNoCollectiveCause Class Reference

The cause for non-collective I/O.

```
#include <H5PropertyList.hpp>
```

Public Member Functions

- [MpioNoCollectiveCause](#) (const [DataTransferProps](#) &dxpl)
- bool [wasCollective](#) () const
Was the datatransfer collective?
- uint32_t [getLocalCause](#) () const
The local cause for a non-collective I/O.
- uint32_t [getGlobalCause](#) () const
The global cause for a non-collective I/O.
- std::pair< uint32_t, uint32_t > [getCause](#) () const
A pair of the local and global cause for non-collective I/O.

9.46.1 Detailed Description

The cause for non-collective I/O.

The cause refers to the most recent I/O with data transfer property list dxpl at time of creation of this object. This object will not update automatically for later data transfers, i.e. `H5Pget_mpio_no_collective_cause` is called in the constructor, and not when fetching a value, such as `wasCollective`.

9.46.2 Constructor & Destructor Documentation

9.46.2.1 MpioNoCollectiveCause()

```
HighFive::MpioNoCollectiveCause::MpioNoCollectiveCause (
    const DataTransferProps & dxpl ) [inline], [explicit]
```

9.46.3 Member Function Documentation

9.46.3.1 getCause()

```
std::pair< uint32_t, uint32_t > HighFive::MpioNoCollectiveCause::getCause ( ) const [inline]
```

A pair of the local and global cause for non-collective I/O.

9.46.3.2 getGlobalCause()

```
uint32_t HighFive::MpioNoCollectiveCause::getGlobalCause ( ) const [inline]
```

The global cause for a non-collective I/O.

9.46.3.3 getLocalCause()

```
uint32_t HighFive::MpioNoCollectiveCause::getLocalCause ( ) const [inline]
```

The local cause for a non-collective I/O.

9.46.3.4 wasCollective()

```
bool HighFive::MpioNoCollectiveCause::wasCollective ( ) const [inline]
```

Was the datatransfer collective?

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

- [highfive/H5PropertyList.hpp](#)
- [highfive/bits/H5PropertyList_misc.hpp](#)

9.47 HighFive::NodeTraits< Derivate > Class Template Reference

NodeTraits: Base class for [Group](#) and [File](#).

```
#include <H5Node_traits.hpp>
```

Public Member Functions

- [DataSet](#) [createDataSet](#) (const std::string &dataset_name, const [DataSpace](#) &space, const [DataType](#) &type, const [DataSetCreateProps](#) &createProps=[DataSetCreateProps::Default](#)(), const [DataSetAccessProps](#) &accessProps=[DataSetAccessProps::Default](#)(), bool parents=true)
createDataSet Create a new dataset in the current file of datatype type and of size space
- template<typename T , typename std::enable_if< std::is_same< typename details::inspector< T >::base_type, details::Boolean >::value, int >::type * = nullptr>
[DataSet](#) [createDataSet](#) (const std::string &dataset_name, const [DataSpace](#) &space, const [DataSetCreateProps](#) &createProps=[DataSetCreateProps::Default](#)(), const [DataSetAccessProps](#) &accessProps=[DataSetAccessProps::Default](#)(), bool parents=true)
createDataSet create a new dataset in the current file with a size specified by space
- template<typename T , typename std::enable_if< !std::is_same< typename details::inspector< T >::base_type, details::Boolean >::value, int >::type * = nullptr>
[DataSet](#) [createDataSet](#) (const std::string &dataset_name, const [DataSpace](#) &space, const [DataSetCreateProps](#) &createProps=[DataSetCreateProps::Default](#)(), const [DataSetAccessProps](#) &accessProps=[DataSetAccessProps::Default](#)(), bool parents=true)
- template<typename T >
[DataSet](#) [createDataSet](#) (const std::string &dataset_name, const T &data, const [DataSetCreateProps](#) &createProps=[DataSetCreateProps::Default](#)(), const [DataSetAccessProps](#) &accessProps=[DataSetAccessProps::Default](#)(), bool parents=true)
createDataSet create a new dataset in the current file and write to it, inferring the [DataSpace](#) from the data.
- template<std::size_t N>
[DataSet](#) [createDataSet](#) (const std::string &dataset_name, const [FixedLenStringArray](#)< N > &data, const [DataSetCreateProps](#) &createProps=[DataSetCreateProps::Default](#)(), const [DataSetAccessProps](#) &accessProps=[DataSetAccessProps::Default](#)(), bool parents=true)
- [DataSet](#) [getDataSet](#) (const std::string &dataset_name, const [DataSetAccessProps](#) &accessProps=[DataSetAccessProps::Default](#)(), const

- get an existing dataset in the current file*
- [Group](#) [createGroup](#) (const std::string &group_name, bool parents=true)
create a new group, and eventually intermediate groups
- [Group](#) [createGroup](#) (const std::string &group_name, const [GroupCreateProps](#) &createProps, bool parents=true)
create a new group, and eventually intermediate groups
- [Group](#) [getGroup](#) (const std::string &group_name) const
open an existing group with the name group_name
- size_t [getNumberObjects](#) () const
return the number of leaf objects of the node / group
- std::string [getObjectName](#) (size_t index) const
return the name of the object with the given index
- bool [rename](#) (const std::string &src_path, const std::string &dest_path, bool parents=true) const
moves an object and its content within an HDF5 file.
- std::vector< std::string > [listObjectNames](#) ([IndexType](#) idx_type=[IndexType::NAME](#)) const
list all leaf objects name of the node / group
- bool [exist](#) (const std::string &node_name) const
check a dataset or group exists in the current node / group
- void [unlink](#) (const std::string &node_name) const
unlink the given dataset or group
- [LinkType](#) [getLinkType](#) (const std::string &node_name) const
Returns the kind of link of the given name (soft, hard...)
- [ObjectType](#) [getObjectType](#) (const std::string &node_name) const
A shorthand to get the kind of object pointed to (group, dataset, type...)
- template<typename T, typename = decltype(&T::getPath)>
void [createSoftLink](#) (const std::string &linkName, const T &obj)
A shorthand to create softlink to any object which provides `getPath`. The link will be created with default properties along with required parent groups.
- void [createSoftLink](#) (const std::string &link_name, const std::string &obj_path, [LinkCreateProps](#) linkCreateProps=[LinkCreateProps](#)(), const [LinkAccessProps](#) &linkAccessProps=[LinkAccessProps](#)(), const bool parents=true)
Creates softlinks.
- void [createExternalLink](#) (const std::string &link_name, const std::string &h5_file, const std::string &obj_path, [LinkCreateProps](#) linkCreateProps=[LinkCreateProps](#)(), const [LinkAccessProps](#) &linkAccessProps=[LinkAccessProps](#)(), const bool parents=true)

9.47.1 Detailed Description

```
template<typename Derivate>
class HighFive::NodeTraits< Derivate >
```

[NodeTraits](#): Base class for [Group](#) and [File](#).

9.47.2 Member Function Documentation

9.47.2.1 createDataSet() [1/5]

```
template<typename Derivate >
template<typename T , typename std::enable_if< !std::is_same< typename details::inspector< T
>::base_type, details::Boolean >::value, int >::type * >
DataSet HighFive::NodeTraits< Derivate >::createDataSet (
    const std::string & dataset_name,
    const DataSpace & space,
    const DataSetCreateProps & createProps = DataSetCreateProps::Default(),
    const DataSetAccessProps & accessProps = DataSetAccessProps::Default(),
    bool parents = true ) [inline]
```

createDataSet create a new dataset in the current file with a size specified by space

Parameters

<i>dataset_name</i>	identifier of the dataset
<i>space</i>	Associated DataSpace , see DataSpace for more information
<i>createProps</i>	A property list with data set creation properties
<i>accessProps</i>	A property list with data set access properties
<i>parents</i>	Create intermediate groups if needed. Default: true.

Returns

[DataSet Object](#)

9.47.2.2 createDataSet() [2/5]

```
template<typename Derivate >
template<typename T , typename std::enable_if< !std::is_same< typename details::inspector< T
>::base_type, details::Boolean >::value, int >::type * = nullptr>
DataSet HighFive::NodeTraits< Derivate >::createDataSet (
    const std::string & dataset_name,
    const DataSpace & space,
    const DataSetCreateProps & createProps = DataSetCreateProps::Default(),
    const DataSetAccessProps & accessProps = DataSetAccessProps::Default(),
    bool parents = true )
```

9.47.2.3 createDataSet() [3/5]

```
template<typename Derivate >
DataSet HighFive::NodeTraits< Derivate >::createDataSet (
    const std::string & dataset_name,
    const DataSpace & space,
    const DataType & type,
    const DataSetCreateProps & createProps = DataSetCreateProps::Default(),
    const DataSetAccessProps & accessProps = DataSetAccessProps::Default(),
    bool parents = true ) [inline]
```

createDataSet Create a new dataset in the current file of datatype type and of size space

Parameters

<i>dataset_name</i>	identifier of the dataset
<i>space</i>	Associated DataSpace , see DataSpace for more information
<i>type</i>	Type of Data
<i>createProps</i>	A property list with data set creation properties
<i>accessProps</i>	A property list with data set access properties
<i>parents</i>	Create intermediate groups if needed. Default: true.

Returns

[DataSet Object](#)

9.47.2.4 createDataSet() [4/5]

```
template<typename Derivate >
template<std::size_t N>
DataSet HighFive::NodeTraits< Derivate >::createDataSet (
    const std::string & dataset_name,
    const FixedLenStringArray< N > & data,
    const DataSetCreateProps & createProps = DataSetCreateProps::Default(),
    const DataSetAccessProps & accessProps = DataSetAccessProps::Default(),
    bool parents = true ) [inline]
```

9.47.2.5 createDataSet() [5/5]

```
template<typename Derivate >
template<typename T >
DataSet HighFive::NodeTraits< Derivate >::createDataSet (
    const std::string & dataset_name,
    const T & data,
    const DataSetCreateProps & createProps = DataSetCreateProps::Default(),
    const DataSetAccessProps & accessProps = DataSetAccessProps::Default(),
    bool parents = true ) [inline]
```

createDataSet create a new dataset in the current file and write to it, inferring the [DataSpace](#) from the data.

Parameters

<i>dataset_name</i>	identifier of the dataset
<i>data</i>	Associated data, must support DataSpace::From , see DataSpace for more information
<i>createProps</i>	A property list with data set creation properties
<i>accessProps</i>	A property list with data set access properties
<i>parents</i>	Create intermediate groups if needed. Default: true.

Returns

[DataSet Object](#)

9.47.2.6 createExternalLink()

```
template<typename Derivate >
void HighFive::NodeTraits< Derivate >::createExternalLink (
    const std::string & link_name,
    const std::string & h5_file,
    const std::string & obj_path,
    LinkCreateProps linkCreateProps = LinkCreateProps(),
    const LinkAccessProps & linkAccessProps = LinkAccessProps(),
    const bool parents = true ) [inline]
```

9.47.2.7 createGroup() [1/2]

```
template<typename Derivate >
Group HighFive::NodeTraits< Derivate >::createGroup (
    const std::string & group_name,
    bool parents = true ) [inline]
```

create a new group, and eventually intermediate groups

Parameters

<i>group_name</i>	
<i>parents</i>	Create intermediate groups if needed. Default: true.

Returns

the group object

9.47.2.8 createGroup() [2/2]

```
template<typename Derivate >
Group HighFive::NodeTraits< Derivate >::createGroup (
    const std::string & group_name,
    const GroupCreateProps & createProps,
    bool parents = true ) [inline]
```

create a new group, and eventually intermediate groups

Parameters

<i>group_name</i>	
<i>createProps</i>	A property list with group creation properties
<i>parents</i>	Create intermediate groups if needed. Default: true.

Returns

the group object

9.47.2.9 createSoftLink() [1/2]

```
template<typename Derivate >
void HighFive::NodeTraits< Derivate >::createSoftLink (
    const std::string & link_name,
    const std::string & obj_path,
    LinkCreateProps linkCreateProps = LinkCreateProps(),
    const LinkAccessProps & linkAccessProps = LinkAccessProps(),
    const bool parents = true ) [inline]
```

Creates softlinks.

Parameters

<i>link_name</i>	The name of the link
<i>obj_path</i>	The target object path
<i>linkCreateProps</i>	A Link_Create property list. Notice "parents=true" overrides
<i>linkAccessProps</i>	The Link_Access property list
<i>parents</i>	Whether parent groups should be created: Default: true

9.47.2.10 createSoftLink() [2/2]

```
template<typename Derivate >
template<typename T , typename = decltype(&T::getPath)>
void HighFive::NodeTraits< Derivate >::createSoftLink (
    const std::string & linkName,
    const T & obj ) [inline]
```

A shorthand to create softlink to any object which provides `getPath`. The link will be created with default properties along with required parent groups.

9.47.2.11 exist()

```
template<typename Derivate >
bool HighFive::NodeTraits< Derivate >::exist (
    const std::string & node_name ) const [inline]
```

check a dataset or group exists in the current node / group

Parameters

<i>node_name</i>	dataset/group name to check
------------------	-----------------------------

Returns

true if a dataset/group with the associated name exists, or false

9.47.2.12 getDataSet()

```
template<typename Derivate >
```

```
DataSet HighFive::NodeTraits< Derivate >::getDataSet (
    const std::string & dataset_name,
    const DataSetAccessProps & accessProps = DataSetAccessProps::Default() ) const
[inline]
```

get an existing dataset in the current file

Parameters

<i>dataset_name</i>	
<i>accessProps</i>	property list to configure dataset chunk cache

Returns

return the named dataset, or throw exception if not found

9.47.2.13 getGroup()

```
template<typename Derivate >
Group HighFive::NodeTraits< Derivate >::getGroup (
    const std::string & group_name ) const [inline]
```

open an existing group with the name group_name

Parameters

<i>group_name</i>	
-------------------	--

Returns

the group object

9.47.2.14 getLinkType()

```
template<typename Derivate >
LinkType HighFive::NodeTraits< Derivate >::getLinkType (
    const std::string & node_name ) const [inline]
```

Returns the kind of link of the given name (soft, hard...)

Parameters

<i>node_name</i>	The entry to check, path relative to the current group
------------------	--

9.47.2.15 getNumberObjects()

```
template<typename Derivate >
size_t HighFive::NodeTraits< Derivate >::getNumberObjects [inline]
```

return the number of leaf objects of the node / group

Returns

number of leaf objects

9.47.2.16 getObjectName()

```
template<typename Derivate >
std::string HighFive::NodeTraits< Derivate >::getObjectName (
    size_t index ) const [inline]
```

return the name of the object with the given index

Returns

the name of the object

9.47.2.17 getObjectType()

```
template<typename Derivate >
ObjectType HighFive::NodeTraits< Derivate >::getObjectType (
    const std::string & node_name ) const [inline]
```

A shorthand to get the kind of object pointed to (group, dataset, type...)

Parameters

<i>node_name</i>	The entry to check, path relative to the current group
------------------	--

9.47.2.18 listObjectNames()

```
template<typename Derivate >
std::vector< std::string > HighFive::NodeTraits< Derivate >::listObjectNames (
    IndexType idx_type = IndexType::NAME ) const [inline]
```

list all leaf objects name of the node / group

Parameters

<i>idx_type</i>	tell if the list should be ordered by Name or CreationOrderTime. CreationOrderTime can be use only if the file/group has been created with the HighFive::LinkCreationTime property.
-----------------	---

Returns

number of leaf objects

9.47.2.19 rename()

```
template<typename Derivate >
bool HighFive::NodeTraits< Derivate >::rename (
    const std::string & src_path,
    const std::string & dest_path,
    bool parents = true ) const [inline]
```

moves an object and its content within an HDF5 file.

Parameters

<i>src_path</i>	relative path of the object to current File/Group
<i>dest_path</i>	new relative path of the object to current File/Group
<i>parents</i>	Create intermediate groups if needed. Default: true.

Returns

boolean that is true if the move was successful

9.47.2.20 unlink()

```
template<typename Derivate >
void HighFive::NodeTraits< Derivate >::unlink (
    const std::string & node_name ) const [inline]
```

unlink the given dataset or group

Parameters

<i>node_name</i>	dataset/group name to unlink
------------------	------------------------------

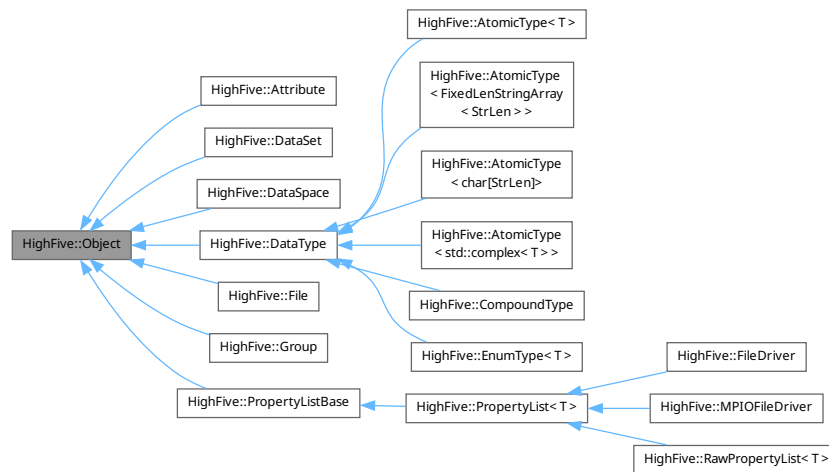
The documentation for this class was generated from the following files:

- [highfive/bits/H5_definitions.hpp](#)
- [highfive/bits/H5Node_traits.hpp](#)
- [highfive/bits/H5Node_traits_misc.hpp](#)

9.48 HighFive::Object Class Reference

```
#include <H5Object.hpp>
```

Inheritance diagram for HighFive::Object:



Public Member Functions

- [Object](#) ([Object](#) &&other) noexcept
- [~Object](#) ()
- bool [isValid](#) () const noexcept
isValid
- hid_t [getId](#) () const noexcept
getId
- [ObjectInfo](#) [getInfo](#) () const
Retrieve several infos about the current object (address, dates, etc)
- [ObjectType](#) [getType](#) () const
Gets the fundamental type of the object (dataset, group, etc)
- bool [operator==](#) (const [Object](#) &other) const noexcept

Protected Member Functions

- [Object](#) ()
- [Object](#) (const [Object](#) &other)
- [Object](#) (hid_t)
- [Object](#) & [operator=](#) (const [Object](#) &other)

Protected Attributes

- hid_t [_hid](#)

Friends

- class [Reference](#)
- class [CompoundType](#)

9.48.1 Constructor & Destructor Documentation

9.48.1.1 Object() [1/4]

```
HighFive::Object::Object (
    Object && other ) [inline], [noexcept]
```

9.48.1.2 ~Object()

```
HighFive::Object::~~Object ( ) [inline]
```

9.48.1.3 Object() [2/4]

```
HighFive::Object::Object ( ) [inline], [protected]
```

9.48.1.4 Object() [3/4]

```
HighFive::Object::Object (
    const Object & other ) [inline], [protected]
```

9.48.1.5 Object() [4/4]

```
HighFive::Object::Object (
    hid_t hid ) [inline], [explicit], [protected]
```

9.48.2 Member Function Documentation

9.48.2.1 getId()

```
hid_t HighFive::Object::getId ( ) const [inline], [noexcept]
```

getId

Returns

internal HDF5 id to the object provided for C API compatibility

9.48.2.2 getInfo()

```
ObjectInfo HighFive::Object::getInfo ( ) const [inline]
```

Retrieve several infos about the current object (address, dates, etc)

9.48.2.3 getType()

```
ObjectType HighFive::Object::getType ( ) const [inline]
```

Gets the fundamental type of the object (dataset, group, etc)

Exceptions

ObjectException	when the <code>_hid</code> is negative or the type is custom and not registered yet
---------------------------------	---

9.48.2.4 isValid()

```
bool HighFive::Object::isValid ( ) const [inline], [noexcept]
```

isValid

Returns

true if current [Object](#) is a valid HDF5Object

9.48.2.5 operator=()

```
Object & HighFive::Object::operator= (
    const Object & other ) [inline], [protected]
```

9.48.2.6 operator==()

```
bool HighFive::Object::operator== (
    const Object & other ) const [inline], [noexcept]
```

9.48.3 Friends And Related Symbol Documentation**9.48.3.1 CompoundType**

```
friend class CompoundType [friend]
```

9.48.3.2 Reference

```
friend class Reference [friend]
```

9.48.4 Member Data Documentation**9.48.4.1 _hid**

```
hid_t HighFive::Object::_hid [protected]
```

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

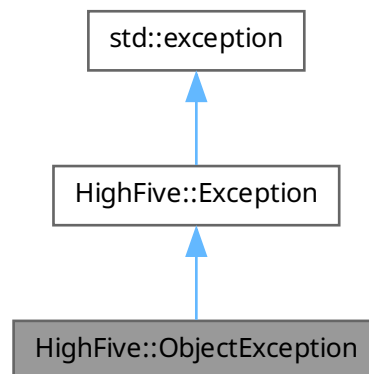
- [highfive/H5Object.hpp](#)
- [highfive/bits/H5Object_misc.hpp](#)

9.49 HighFive::ObjectException Class Reference

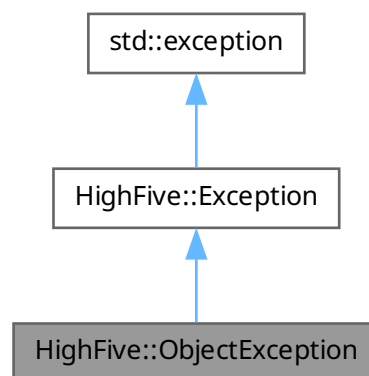
Exception specific to [HighFive Object](#) interface.

```
#include <H5Exception.hpp>
```

Inheritance diagram for HighFive::ObjectException:



Collaboration diagram for HighFive::ObjectException:



Public Member Functions

- [ObjectException](#) (const std::string &err_msg)

Public Member Functions inherited from [HighFive::Exception](#)

- [Exception](#) (const std::string &err_msg)
- virtual [~Exception](#) () throw ()
- const char * [what](#) () const override throw ()
get the current exception error message
- virtual void [setErrMsg](#) (const std::string &errmsg)
define the error message
- [Exception](#) * [nextException](#) () const
nextException
- hid_t [getErrMajor](#) () const
HDF5 library error mapper.
- hid_t [getErrMinor](#) () const
HDF5 library error mapper.

Additional Inherited Members

Protected Attributes inherited from [HighFive::Exception](#)

- std::string [_errmsg](#)
- std::shared_ptr< [Exception](#) > [_next](#)
- hid_t [_err_major](#)
- hid_t [_err_minor](#)

9.49.1 Detailed Description

[Exception](#) specific to [HighFive Object](#) interface.

9.49.2 Constructor & Destructor Documentation

9.49.2.1 ObjectException()

```
HighFive::ObjectException::ObjectException (
    const std::string & err_msg ) [inline]
```

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

- [highfive/H5Exception.hpp](#)

9.50 HighFive::ObjectInfo Class Reference

A class for accessing hdf5 objects info.

```
#include <H5Object.hpp>
```

Public Member Functions

- `haddr_t getAddress () const noexcept`
Retrieve the address of the object (within its file)
- `size_t getRefCount () const noexcept`
Retrieve the number of references to this object.
- `time_t getCreationTime () const noexcept`
Retrieve the object's creation time.
- `time_t getModificationTime () const noexcept`
Retrieve the object's last modification time.

Protected Attributes

- `H5O_info_t raw_info`

Friends

- class `Object`

9.50.1 Detailed Description

A class for accessing hdf5 objects info.

9.50.2 Member Function Documentation

9.50.2.1 getAddress()

```
haddr_t HighFive::ObjectInfo::getAddress ( ) const [inline], [noexcept]
```

Retrieve the address of the object (within its file)

Deprecated Deprecated since [HighFive 2.2](#). Soon supporting VOL tokens

9.50.2.2 getCreationTime()

```
time_t HighFive::ObjectInfo::getCreationTime ( ) const [inline], [noexcept]
```

Retrieve the object's creation time.

9.50.2.3 getModificationTime()

```
time_t HighFive::ObjectInfo::getModificationTime ( ) const [inline], [noexcept]
```

Retrieve the object's last modification time.

9.50.2.4 getRefCount()

```
size_t HighFive::ObjectInfo::getRefCount ( ) const [inline], [noexcept]
```

Retrieve the number of references to this object.

9.50.3 Friends And Related Symbol Documentation

9.50.3.1 Object

```
friend class Object [friend]
```

9.50.4 Member Data Documentation

9.50.4.1 raw_info

```
H5O_info_t HighFive::ObjectInfo::raw_info [protected]
```

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

- [highfive/H5Object.hpp](#)
- [highfive/bits/H5Object_misc.hpp](#)

9.51 HighFive::PathTraits< Derivate > Class Template Reference

```
#include <H5Path_traits.hpp>
```

Public Member Functions

- [PathTraits](#) ()
- `std::string` [getPath](#) () const
return the path to the current object
- `File &` [getFile](#) () const noexcept
Return a reference to the [File](#) object this object belongs.

Protected Attributes

- `std::shared_ptr< File >` [_file_obj](#)

9.51.1 Constructor & Destructor Documentation

9.51.1.1 PathTraits()

```
template<typename Derivate >  
HighFive::PathTraits< Derivate >::PathTraits [inline]
```

9.51.2 Member Function Documentation

9.51.2.1 getFile()

```
template<typename Derivate >  
File & HighFive::PathTraits< Derivate >::getFile [inline], [noexcept]
```

Return a reference to the [File](#) object this object belongs.

Returns

the [File](#) object ref

9.51.2.2 getPath()

```
template<typename Derivate >  
std::string HighFive::PathTraits< Derivate >::getPath [inline]
```

return the path to the current object

Returns

the path to the object

9.51.3 Member Data Documentation

9.51.3.1 _file_obj

```
template<typename Derivate >  
std::shared_ptr<File> HighFive::PathTraits< Derivate >::_file_obj [protected]
```

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

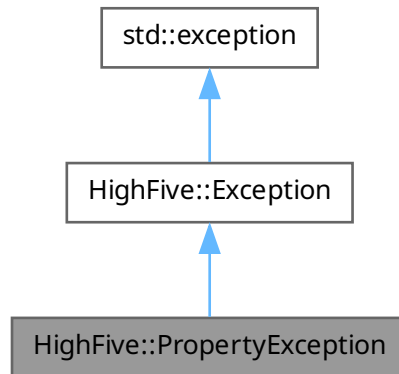
- [highfive/bits/H5Path_traits.hpp](#)
- [highfive/bits/H5Path_traits_misc.hpp](#)

9.52 HighFive::PropertyException Class Reference

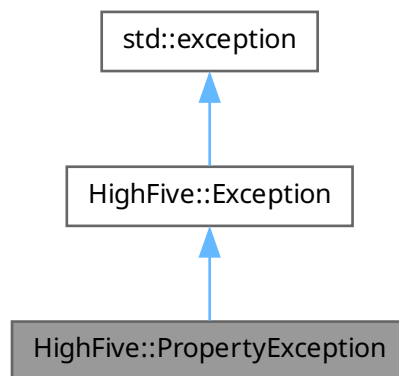
Exception specific to [HighFive](#) Property interface.

```
#include <H5Exception.hpp>
```

Inheritance diagram for HighFive::PropertyException:



Collaboration diagram for HighFive::PropertyException:



Public Member Functions

- [PropertyException](#) (const std::string &err_msg)

Public Member Functions inherited from HighFive::Exception

- [Exception](#) (const std::string &err_msg)
- virtual [~Exception](#) () throw ()
- const char * [what](#) () const override throw ()
get the current exception error message
- virtual void [setErrorMsg](#) (const std::string &errmsg)
define the error message
- [Exception](#) * [nextException](#) () const
nextException
- hid_t [getErrMajor](#) () const
HDF5 library error mapper.
- hid_t [getErrMinor](#) () const
HDF5 library error mapper.

Additional Inherited Members

Protected Attributes inherited from HighFive::Exception

- std::string [_errmsg](#)
- std::shared_ptr< [Exception](#) > [_next](#)
- hid_t [_err_major](#)
- hid_t [_err_minor](#)

9.52.1 Detailed Description

[Exception](#) specific to [HighFive](#) Property interface.

9.52.2 Constructor & Destructor Documentation

9.52.2.1 PropertyException()

```
HighFive::PropertyException::PropertyException (
    const std::string & err_msg ) [inline]
```

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

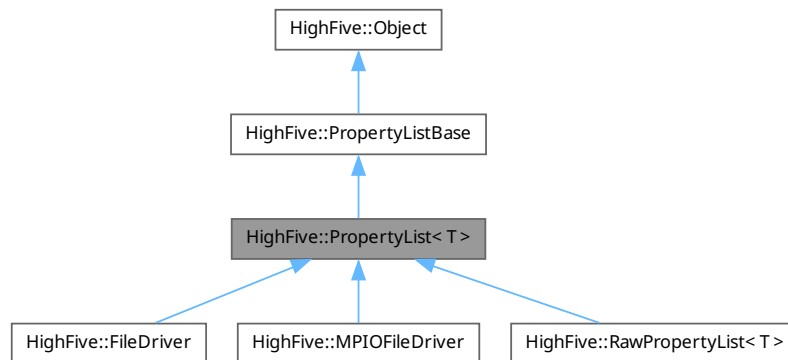
- [highfive/H5Exception.hpp](#)

9.53 HighFive::PropertyList< T > Class Template Reference

HDF5 property Lists.

```
#include <H5PropertyList.hpp>
```

Inheritance diagram for HighFive::PropertyList< T >:



Collaboration diagram for HighFive::PropertyList< T >:



Public Member Functions

- constexpr [PropertyType](#) [getType](#) () const noexcept
return the type of this [PropertyList](#)
- template<typename P >
void [add](#) (const P &property)

Public Member Functions inherited from [HighFive::PropertyListBase](#)

- [PropertyListBase](#) () noexcept

Public Member Functions inherited from [HighFive::Object](#)

- [Object](#) ([Object](#) &&other) noexcept
- [~Object](#) ()
- bool [isValid](#) () const noexcept
isValid
- hid_t [getId](#) () const noexcept
getId
- [ObjectInfo](#) [getInfo](#) () const
Retrieve several infos about the current object (address, dates, etc)
- [ObjectType](#) [getType](#) () const
Gets the fundamental type of the object (dataset, group, etc)
- bool [operator==](#) (const [Object](#) &other) const noexcept

Static Public Member Functions

- static const [PropertyList](#)< T > & [Default](#) () noexcept
Return the Default property type object.

Static Public Member Functions inherited from [HighFive::PropertyListBase](#)

- static const [PropertyListBase](#) & [Default](#) () noexcept

Protected Member Functions

- void [_initializeIfNeeded](#) ()

Protected Member Functions inherited from [HighFive::Object](#)

- [Object](#) ()
- [Object](#) (const [Object](#) &other)
- [Object](#) (hid_t)
- [Object](#) & [operator=](#) (const [Object](#) &other)

Additional Inherited Members**Protected Attributes inherited from [HighFive::Object](#)**

- hid_t [_hid](#)

9.53.1 Detailed Description

```
template<PropertyType T>
class HighFive::PropertyList< T >
```

HDF5 property Lists.

9.53.2 Member Function Documentation

9.53.2.1 `_initializeIfNeeded()`

```
template<PropertyType T>
void HighFive::PropertyList< T >::_initializeIfNeeded [inline], [protected]
```

9.53.2.2 `add()`

```
template<PropertyType T>
template<typename P >
void HighFive::PropertyList< T >::add (
    const P & property ) [inline]
```

Add a property to this property list. A property is an object which is expected to have a method with the following signature `void apply(hid_t hid) const`

9.53.2.3 `Default()`

```
template<PropertyType T>
static const PropertyList< T > & HighFive::PropertyList< T >::Default ( ) [inline], [static],
[noexcept]
```

Return the Default property type object.

9.53.2.4 `getType()`

```
template<PropertyType T>
constexpr PropertyType HighFive::PropertyList< T >::getType ( ) const [inline], [constexpr],
[noexcept]
```

return the type of this [PropertyList](#)

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

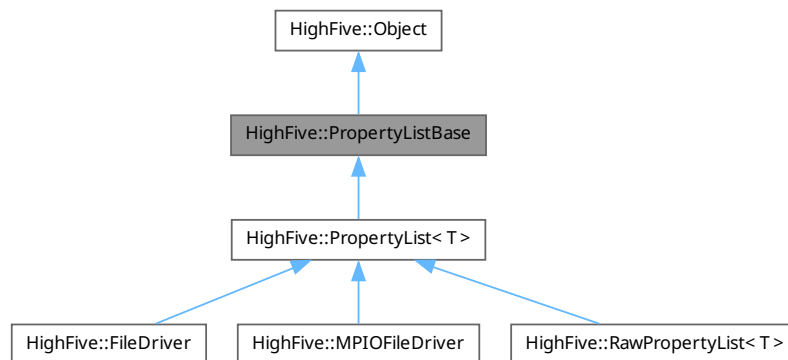
- [highfive/bits/H5_definitions.hpp](#)
- [highfive/H5PropertyList.hpp](#)
- [highfive/bits/H5PropertyList_misc.hpp](#)

9.54 HighFive::PropertyListBase Class Reference

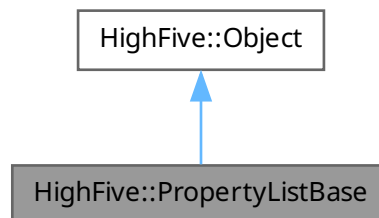
Base Class for Property lists, providing global default.

```
#include <H5PropertyList.hpp>
```

Inheritance diagram for HighFive::PropertyListBase:



Collaboration diagram for HighFive::PropertyListBase:



Public Member Functions

- [PropertyListBase](#) () noexcept

Public Member Functions inherited from [HighFive::Object](#)

- [Object](#) ([Object](#) &&other) noexcept
 - [~Object](#) ()
 - bool [isValid](#) () const noexcept
- isValid*

- `hid_t getIid ()` `const noexcept`
getIid
- `ObjectInfo getInfo ()` `const`
Retrieve several infos about the current object (address, dates, etc)
- `ObjectType getType ()` `const`
Gets the fundamental type of the object (dataset, group, etc)
- `bool operator== (const Object &other)` `const noexcept`

Static Public Member Functions

- `static const PropertyListBase & Default ()` `noexcept`

Additional Inherited Members

Protected Member Functions inherited from `HighFive::Object`

- `Object ()`
- `Object (const Object &other)`
- `Object (hid_t)`
- `Object & operator= (const Object &other)`

Protected Attributes inherited from `HighFive::Object`

- `hid_t _hid`

9.54.1 Detailed Description

Base Class for Property lists, providing global default.

9.54.2 Constructor & Destructor Documentation

9.54.2.1 `PropertyListBase()`

```
HighFive::PropertyListBase::PropertyListBase ( ) [inline], [noexcept]
```

9.54.3 Member Function Documentation

9.54.3.1 `Default()`

```
static const PropertyListBase & HighFive::PropertyListBase::Default ( ) [inline], [static], [noexcept]
```

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

- `highfive/H5PropertyList.hpp`
- `highfive/bits/H5PropertyList_misc.hpp`

9.55 HighFive::RawPropertyList< T > Class Template Reference

```
#include <H5PropertyList.hpp>
```

Inheritance diagram for HighFive::RawPropertyList< T >:



Collaboration diagram for HighFive::RawPropertyList< T >:



Public Member Functions

- `template<typename F, typename... Args>`
`void add (const F &funct, const Args &... args)`

Public Member Functions inherited from [HighFive::PropertyList< T >](#)

- `constexpr PropertyType getType ()` `const noexcept`
return the type of this [PropertyList](#)
- `template<typename P >`
`void add (const P &property)`

Public Member Functions inherited from [HighFive::PropertyListBase](#)

- [PropertyListBase](#) () `noexcept`

Public Member Functions inherited from [HighFive::Object](#)

- [Object](#) ([Object](#) &&other) `noexcept`
- [~Object](#) ()
- `bool isValid ()` `const noexcept`
isValid
- `hid_t getId ()` `const noexcept`
getId
- [ObjectInfo](#) `getInfo ()` `const`
Retrieve several infos about the current object (address, dates, etc)
- [ObjectType](#) `getType ()` `const`
Gets the fundamental type of the object (dataset, group, etc)
- `bool operator== (const Object &other)` `const noexcept`

Additional Inherited Members

Static Public Member Functions inherited from [HighFive::PropertyList< T >](#)

- `static const PropertyList< T > & Default ()` `noexcept`
Return the Default property type object.

Static Public Member Functions inherited from [HighFive::PropertyListBase](#)

- `static const PropertyListBase & Default ()` `noexcept`

Protected Member Functions inherited from [HighFive::PropertyList< T >](#)

- `void _initializeIfNeeded ()`

Protected Member Functions inherited from [HighFive::Object](#)

- [Object](#) ()
- [Object](#) (const [Object](#) &other)
- [Object](#) (hid_t)
- [Object](#) & [operator=](#) (const [Object](#) &other)

Protected Attributes inherited from [HighFive::Object](#)

- `hid_t _hid`

9.55.1 Detailed Description

```
template<PropertyType T>
class HighFive::RawPropertyList< T >
```

RawPropertyLists are to be used when advanced H5 properties are desired and are not part of the [HighFive](#) API. Therefore this class is mainly for internal use.

9.55.2 Member Function Documentation

9.55.2.1 add()

```
template<PropertyType T>
template<typename F , typename... Args>
void HighFive::RawPropertyList< T >::add (
    const F & funct,
    const Args &... args ) [inline]
```

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

- [highfive/H5PropertyList.hpp](#)
- [highfive/bits/H5PropertyList_misc.hpp](#)

9.56 HighFive::Reference Class Reference

An HDF5 (object) reference type.

```
#include <H5Reference.hpp>
```

Public Member Functions

- [Reference](#) ()=default
Create an empty [Reference](#) to be initialized later.
- [Reference](#) (const [Object](#) &location, const [Object](#) &object)
Create a [Reference](#) to an object residing at a given location.
- template<typename T >
T [dereference](#) (const [Object](#) &location) const
Retrieve the [Object](#) being referenced by the [Reference](#).
- [ObjectType](#) [getType](#) (const [Object](#) &location) const
Get only the type of the referenced [Object](#).

Protected Member Functions

- [Reference](#) (const hobj_ref_t h5_ref)
Create a [Reference](#) from a low-level HDF5 object reference.
- void [create_ref](#) (hobj_ref_t *refptr) const
Create the low-level reference and store it at refptr.

9.56.1 Detailed Description

An HDF5 (object) reference type.

HDF5 object references allow pointing to groups, datasets (and compound types). They differ from links in their ability to be stored and retrieved as data from the HDF5 file in datasets themselves.

9.56.2 Constructor & Destructor Documentation

9.56.2.1 [Reference\(\)](#) [1/3]

```
HighFive::Reference::Reference ( ) [default]
```

Create an empty [Reference](#) to be initialized later.

9.56.2.2 [Reference\(\)](#) [2/3]

```
HighFive::Reference::Reference (
    const Object & location,
    const Object & object ) [inline]
```

Create a [Reference](#) to an object residing at a given location.

Parameters

<i>location</i>	A File or Group where the object being referenced to resides
<i>object</i>	A Dataset or Group to be referenced

9.56.2.3 [Reference\(\)](#) [3/3]

```
HighFive::Reference::Reference (
    const hobj_ref_t h5_ref ) [inline], [explicit], [protected]
```

Create a [Reference](#) from a low-level HDF5 object reference.

9.56.3 Member Function Documentation

9.56.3.1 [create_ref\(\)](#)

```
void HighFive::Reference::create_ref (
    hobj_ref_t * refptr ) const [inline], [protected]
```

Create the low-level reference and store it at refptr.

Parameters

<i>refptr</i>	Pointer to a memory location where the created HDF5 reference will be stored
---------------	--

9.56.3.2 dereference()

```
template<typename T >
T HighFive::Reference::dereference (
    const Object & location ) const [inline]
```

Retrieve the [Object](#) being referenced by the [Reference](#).

Template Parameters

<i>T</i>	the appropriate HighFive Container (either DataSet or Group)
----------	---

Parameters

<i>location</i>	the location where the referenced object is to be found (a File)
-----------------	---

Returns

the dereferenced [Object](#) (either a [Group](#) or [DataSet](#))

9.56.3.3 getType()

```
ObjectType HighFive::Reference::getType (
    const Object & location ) const [inline]
```

Get only the type of the referenced [Object](#).

Parameters

<i>location</i>	the location where the referenced object is to be found (a File)
-----------------	---

Returns

the [ObjectType](#) of the referenced object

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

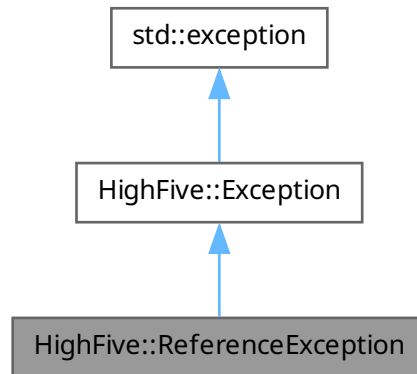
- [highfive/H5Reference.hpp](#)
- [highfive/bits/H5Reference_misc.hpp](#)

9.57 HighFive::ReferenceException Class Reference

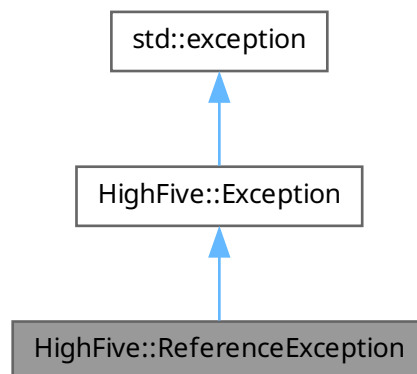
Exception specific to [HighFive Reference](#) interface.

```
#include <H5Exception.hpp>
```

Inheritance diagram for HighFive::ReferenceException:



Collaboration diagram for HighFive::ReferenceException:



Public Member Functions

- [ReferenceException](#) (const std::string &err_msg)

Public Member Functions inherited from [HighFive::Exception](#)

- [Exception](#) (const std::string &err_msg)
- virtual [~Exception](#) () throw ()
- const char * [what](#) () const override throw ()
get the current exception error message
- virtual void [setErrorMsg](#) (const std::string &errmsg)
define the error message
- [Exception](#) * [nextException](#) () const
nextException
- hid_t [getErrMajor](#) () const
HDF5 library error mapper.
- hid_t [getErrMinor](#) () const
HDF5 library error mapper.

Additional Inherited Members

Protected Attributes inherited from [HighFive::Exception](#)

- std::string [_errmsg](#)
- std::shared_ptr< [Exception](#) > [_next](#)
- hid_t [_err_major](#)
- hid_t [_err_minor](#)

9.57.1 Detailed Description

[Exception](#) specific to [HighFive Reference](#) interface.

9.57.2 Constructor & Destructor Documentation

9.57.2.1 ReferenceException()

```
HighFive::ReferenceException::ReferenceException (
    const std::string & err_msg ) [inline]
```

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

- [highfive/H5Exception.hpp](#)

9.58 HighFive::RegularHyperSlab Struct Reference

```
#include <H5Slice_traits.hpp>
```

Public Member Functions

- [RegularHyperSlab](#) ()=default
- [RegularHyperSlab](#) (std::vector< size_t > offset_, std::vector< size_t > count_={}, std::vector< size_t > stride_={}, std::vector< size_t > block_={})
- size_t [rank](#) () const
- std::vector< size_t > [packedDims](#) () const
Dimensions when all gaps are removed.

Static Public Member Functions

- static [RegularHyperSlab fromHDF5Sizes](#) (std::vector< hsize_t > offset_, std::vector< hsize_t > count_={}, std::vector< hsize_t > stride_={}, std::vector< hsize_t > block_={})

Public Attributes

- std::vector< hsize_t > [offset](#)
- std::vector< hsize_t > [count](#)
- std::vector< hsize_t > [stride](#)
- std::vector< hsize_t > [block](#)

9.58.1 Constructor & Destructor Documentation

9.58.1.1 RegularHyperSlab() [1/2]

```
HighFive::RegularHyperSlab::RegularHyperSlab ( ) [default]
```

9.58.1.2 RegularHyperSlab() [2/2]

```
HighFive::RegularHyperSlab::RegularHyperSlab (
    std::vector< size_t > offset_,
    std::vector< size_t > count_ = {},
    std::vector< size_t > stride_ = {},
    std::vector< size_t > block_ = {} ) [inline]
```

9.58.2 Member Function Documentation

9.58.2.1 fromHDF5Sizes()

```
static RegularHyperSlab HighFive::RegularHyperSlab::fromHDF5Sizes (
    std::vector< hsize_t > offset_,
    std::vector< hsize_t > count_ = {},
    std::vector< hsize_t > stride_ = {},
    std::vector< hsize_t > block_ = {} ) [inline], [static]
```

9.58.2.2 packedDims()

```
std::vector< size_t > HighFive::RegularHyperSlab::packedDims ( ) const [inline]
```

Dimensions when all gaps are removed.

9.58.2.3 rank()

```
size_t HighFive::RegularHyperSlab::rank ( ) const [inline]
```

9.58.3 Member Data Documentation

9.58.3.1 block

```
std::vector<hsize_t> HighFive::RegularHyperSlab::block
```

9.58.3.2 count

```
std::vector<hsize_t> HighFive::RegularHyperSlab::count
```

9.58.3.3 offset

```
std::vector<hsize_t> HighFive::RegularHyperSlab::offset
```

9.58.3.4 stride

```
std::vector<hsize_t> HighFive::RegularHyperSlab::stride
```

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

- [highfive/bits/H5Slice_traits.hpp](#)

9.59 HighFive::Selection Class Reference

[Selection](#): represent a view on a slice/part of a dataset.

```
#include <H5Selection.hpp>
```

Inheritance diagram for HighFive::Selection:



Collaboration diagram for HighFive::Selection:



Public Member Functions

- [DataSet](#) [getSpace](#) () const noexcept
getSpace
- [DataSet](#) [getMemSpace](#) () const noexcept
getMemSpace
- [DataSet](#) & [getDataset](#) () noexcept
getDataSet
- const [DataSet](#) & [getDataset](#) () const noexcept
- const [DataType](#) [getDataType](#) () const
return the datatype of the selection

Public Member Functions inherited from HighFive::SliceTraits< Selection >

- [Selection select](#) (const [HyperSlab](#) &hyperslab) const
Select an hyperslab in the current Slice/Dataset.
- [Selection select](#) (const std::vector< size_t > &offset, const std::vector< size_t > &count, const std::vector< size_t > &stride={}, const std::vector< size_t > &block={}) const
Select a region in the current Slice/Dataset of count points at offset separated by stride. If strides are not provided they will default to 1 in all dimensions.
- [Selection select](#) (const std::vector< size_t > &columns) const
Select a set of columns in the last dimension of this dataset.
- [Selection select](#) (const [ElementSet](#) &elements) const
Select a region in the current Slice/Dataset out of a list of elements.
- [T read](#) (const [DataTransferProps](#) &xfer_props=[DataTransferProps](#)()) const
- [void read](#) (T &array, const [DataTransferProps](#) &xfer_props=[DataTransferProps](#)()) const
- [void read](#) (T *array, const [DataType](#) &dtype=[DataType](#)(), const [DataTransferProps](#) &xfer_props=[DataTransferProps](#)()) const
- [void write](#) (const T &buffer, const [DataTransferProps](#) &xfer_props=[DataTransferProps](#)())
- [void write_raw](#) (const T *buffer, const [DataType](#) &dtype=[DataType](#)(), const [DataTransferProps](#) &xfer_props=[DataTransferProps](#)())

Protected Member Functions

- [Selection](#) (const [DataSpace](#) &memspace, const [DataSpace](#) &file_space, const [DataSet](#) &set)

Protected Member Functions inherited from HighFive::SliceTraits< Selection >

- [Selection select_impl](#) (const [HyperSlab](#) &hyperslab, const [DataSpace](#) &memspace) const

9.59.1 Detailed Description

[Selection](#): represent a view on a slice/part of a dataset.

A [Selection](#) is valid only if its parent dataset is valid

9.59.2 Constructor & Destructor Documentation

9.59.2.1 Selection()

```
HighFive::Selection::Selection (
    const DataSpace & memspace,
    const DataSpace & file_space,
    const DataSet & set ) [inline], [protected]
```

9.59.3 Member Function Documentation

9.59.3.1 getDataset() [1/2]

```
const DataSet & HighFive::Selection::getDataset ( ) const [inline], [noexcept]
```

9.59.3.2 `getDataset()` [2/2]

```
DataSet & HighFive::Selection::getDataset ( ) [inline], [noexcept]
```

`getDataSet`

Returns

parent dataset of this selection

9.59.3.3 `getDataType()`

```
const DataType HighFive::Selection::getDataType ( ) const [inline]
```

return the datatype of the selection

Returns

return the datatype of the selection

9.59.3.4 `getMemSpace()`

```
DataSpace HighFive::Selection::getMemSpace ( ) const [inline], [noexcept]
```

`getMemSpace`

Returns

Dataspace associated with the memory representation of this selection

9.59.3.5 `getSpace()`

```
DataSpace HighFive::Selection::getSpace ( ) const [inline], [noexcept]
```

`getSpace`

Returns

Dataspace associated with this selection

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

- [highfive/H5Selection.hpp](#)
- [highfive/bits/H5Selection_misc.hpp](#)

9.60 HighFive::Shuffle Class Reference

```
#include <H5PropertyList.hpp>
```

Public Member Functions

- [Shuffle](#) ()=default

9.60.1 Constructor & Destructor Documentation

9.60.1.1 Shuffle()

```
HighFive::Shuffle::Shuffle ( ) [default]
```

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

- [highfive/H5PropertyList.hpp](#)
- [highfive/bits/H5PropertyList_misc.hpp](#)

9.61 HighFive::SilenceHDF5 Class Reference

Utility class to disable HDF5 stack printing inside a scope.

```
#include <H5Utility.hpp>
```

Public Member Functions

- [SilenceHDF5](#) (bool enable=true)
- [~SilenceHDF5](#) ()

9.61.1 Detailed Description

Utility class to disable HDF5 stack printing inside a scope.

9.61.2 Constructor & Destructor Documentation

9.61.2.1 SilenceHDF5()

```
HighFive::SilenceHDF5::SilenceHDF5 (  
    bool enable = true ) [inline]
```

9.61.2.2 ~SilenceHDF5()

```
HighFive::SilenceHDF5::~~SilenceHDF5 ( ) [inline]
```

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

- [highfive/H5Utility.hpp](#)

9.62 HighFive::SliceTraits< Derivate > Class Template Reference

```
#include <H5Slice_traits.hpp>
```

Public Member Functions

- [Selection select](#) (const [HyperSlab](#) &hyperslab) const
Select an hyperslab in the current Slice/Dataset.
- [Selection select](#) (const std::vector< size_t > &offset, const std::vector< size_t > &count, const std::vector< size_t > &stride={}, const std::vector< size_t > &block={}) const
Select a region in the current Slice/Dataset of count points at offset separated by stride. If strides are not provided they will default to 1 in all dimensions.
- [Selection select](#) (const std::vector< size_t > &columns) const
Select a set of columns in the last dimension of this dataset.
- [Selection select](#) (const [ElementSet](#) &elements) const
Select a region in the current Slice/Dataset out of a list of elements.
- template<typename T >
T [read](#) (const [DataTransferProps](#) &xfer_props=[DataTransferProps](#)()) const
- template<typename T >
void [read](#) (T &array, const [DataTransferProps](#) &xfer_props=[DataTransferProps](#)()) const
- template<typename T >
void [read](#) (T *array, const [DataType](#) &dtype=[DataType](#)(), const [DataTransferProps](#) &xfer_props=[DataTransferProps](#)()) const
- template<typename T >
void [write](#) (const T &buffer, const [DataTransferProps](#) &xfer_props=[DataTransferProps](#)())
- template<typename T >
void [write_raw](#) (const T *buffer, const [DataType](#) &dtype=[DataType](#)(), const [DataTransferProps](#) &xfer_props=[DataTransferProps](#)())

Protected Member Functions

- [Selection select_impl](#) (const [HyperSlab](#) &hyperslab, const [DataSpace](#) &memspace) const

9.62.1 Member Function Documentation

9.62.1.1 read() [1/3]

```
template<typename Derivate >
template<typename T >
T HighFive::SliceTraits< Derivate >::read (
    const DataTransferProps & xfer_props = DataTransferProps() ) const [inline]
```

9.62.1.2 read() [2/3]

```
template<typename Derivate >
template<typename T >
void HighFive::SliceTraits< Derivate >::read (
    T & array,
    const DataTransferProps & xfer_props = DataTransferProps() ) const [inline]
```

Read the entire dataset into a buffer An exception is raised is if the numbers of dimension of the buffer and of the dataset are different.

The array type can be a N-pointer or a N-vector. For plain pointers not dimensionality checking will be performed, it is the user's responsibility to ensure that the right amount of space has been allocated.

9.62.1.3 read() [3/3]

```
template<typename Derivate >
template<typename T >
void HighFive::SliceTraits< Derivate >::read (
    T * array,
    const DataType & dtype = DataType(),
    const DataTransferProps & xfer_props = DataTransferProps() ) const [inline]
```

Read the entire dataset into a raw buffer

No dimensionality checks will be performed, it is the user's responsibility to ensure that the right amount of space has been allocated.

Parameters

<i>array</i>	A buffer containing enough space for the data
<i>dtype</i>	The type of the data, in case it cannot be automatically guessed
<i>xfer_props</i>	Data Transfer properties

9.62.1.4 select() [1/4]

```
template<typename Derivate >
Selection HighFive::SliceTraits< Derivate >::select (
    const ElementSet & elements ) const [inline]
```

Select a region in the current Slice/Dataset out of a list of elements.

9.62.1.5 select() [2/4]

```
template<typename Derivate >
Selection HighFive::SliceTraits< Derivate >::select (
    const HyperSlab & hyperslab ) const [inline]
```

Select an hyperslab in the current Slice/Dataset.

HyperSlabs can be either regular or irregular. Irregular hyperslabs are typically generated by taking the union of regular hyperslabs. An irregular hyperslab, in general, does not fit nicely into a multi-dimensional array, but only a subset of such an array.

Therefore, the only memspaces supported for general hyperslabs are one-dimensional arrays.

9.62.1.6 select() [3/4]

```
template<typename Derivate >
Selection HighFive::SliceTraits< Derivate >::select (
    const std::vector< size_t > & columns ) const [inline]
```

Select a set of columns in the last dimension of this dataset.

The column indices must be smaller than the dimension size.

9.62.1.7 select() [4/4]

```
template<typename Derivate >
Selection HighFive::SliceTraits< Derivate >::select (
    const std::vector< size_t > & offset,
    const std::vector< size_t > & count,
    const std::vector< size_t > & stride = {},
    const std::vector< size_t > & block = {} ) const [inline]
```

Select a region in the current Slice/Dataset of count points at offset separated by stride. If strides are not provided they will default to 1 in all dimensions.

vector offset and count have to be from the same dimension

9.62.1.8 select_impl()

```
template<typename Derivate >
Selection HighFive::SliceTraits< Derivate >::select_impl (
    const HyperSlab & hyperslab,
    const DataSpace & memspace ) const [inline], [protected]
```

9.62.1.9 write()

```
template<typename Derivate >
template<typename T >
void HighFive::SliceTraits< Derivate >::write (
    const T & buffer,
    const DataTransferProps & xfer_props = DataTransferProps() ) [inline]
```

Write the integrality N-dimension buffer to this dataset An exception is raised is if the numbers of dimension of the buffer and of the dataset are different

The array type can be a N-pointer or a N-vector (e.g int** integer two dimensional array)

9.62.1.10 write_raw()

```
template<typename Derivate >
template<typename T >
void HighFive::SliceTraits< Derivate >::write_raw (
    const T * buffer,
    const DataType & dtype = DataType(),
    const DataTransferProps & xfer_props = DataTransferProps() ) [inline]
```

Write from a raw buffer into this dataset

No dimensionality checks will be performed, it is the user's responsibility to ensure that the buffer holds the right amount of elements. For n-dimensional matrices the buffer layout follows H5 default conventions.

Parameters

<i>buffer</i>	A buffer containing the data to be written
<i>dtype</i>	The type of the data, in case it cannot be automatically guessed
<i>xfer_props</i>	The HDF5 data transfer properties, e.g. collective MPI-IO.

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

- [highfive/bits/H5Slice_traits.hpp](#)
- [highfive/bits/H5Slice_traits_misc.hpp](#)

9.63 HighFive::Szip Class Reference

```
#include <H5PropertyList.hpp>
```

Public Member Functions

- [Szip](#) (unsigned options_mask=H5_SZIP_EC_OPTION_MASK, unsigned pixels_per_block=H5_SZIP_MAX_PIXELS_PER_BLOCK)
- unsigned [getOptionsMask](#) () const
- unsigned [getPixelsPerBlock](#) () const

9.63.1 Constructor & Destructor Documentation

9.63.1.1 Szip()

```
HighFive::Szip::Szip (
    unsigned options_mask = H5_SZIP_EC_OPTION_MASK,
    unsigned pixels_per_block = H5_SZIP_MAX_PIXELS_PER_BLOCK ) [inline], [explicit]
```

9.63.2 Member Function Documentation

9.63.2.1 getOptionsMask()

```
unsigned HighFive::Szip::getOptionsMask ( ) const [inline]
```

9.63.2.2 getPixelsPerBlock()

```
unsigned HighFive::Szip::getPixelsPerBlock ( ) const [inline]
```

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

- [highfive/H5PropertyList.hpp](#)
- [highfive/bits/H5PropertyList_misc.hpp](#)

9.64 HighFive::UseCollectiveIO Class Reference

```
#include <H5PropertyList.hpp>
```

Public Member Functions

- [UseCollectiveIO](#) (bool enable=true)
- [UseCollectiveIO](#) (const [DataTransferProps](#) &dxml)
- bool [isCollective](#) () const

Does the property request collective IO?

9.64.1 Constructor & Destructor Documentation

9.64.1.1 UseCollectiveIO() [1/2]

```
HighFive::UseCollectiveIO::UseCollectiveIO (  
    bool enable = true ) [inline], [explicit]
```

9.64.1.2 UseCollectiveIO() [2/2]

```
HighFive::UseCollectiveIO::UseCollectiveIO (  
    const DataTransferProps & dxml ) [inline], [explicit]
```

9.64.2 Member Function Documentation

9.64.2.1 isCollective()

```
bool HighFive::UseCollectiveIO::isCollective ( ) const [inline]
```

Does the property request collective IO?

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

- [highfive/H5PropertyList.hpp](#)
- [highfive/bits/H5PropertyList_misc.hpp](#)

File Documentation

10.2 highfive/bits/H5_definitions.hpp File Reference

[illegible]

Namespaces

- namespace [HighFive](#)

Macros

- `#define` [H5_DEPRECATED](#)(msg)

10.2.1 Macro Definition Documentation

10.2.1.1 H5_DEPRECATED

```
#define H5_DEPRECATED(  
    msg )
```

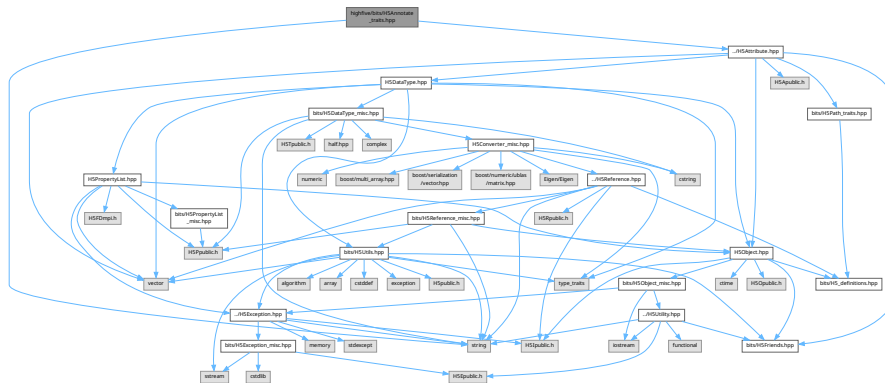
10.3 H5_definitions.hpp

[Go to the documentation of this file.](#)

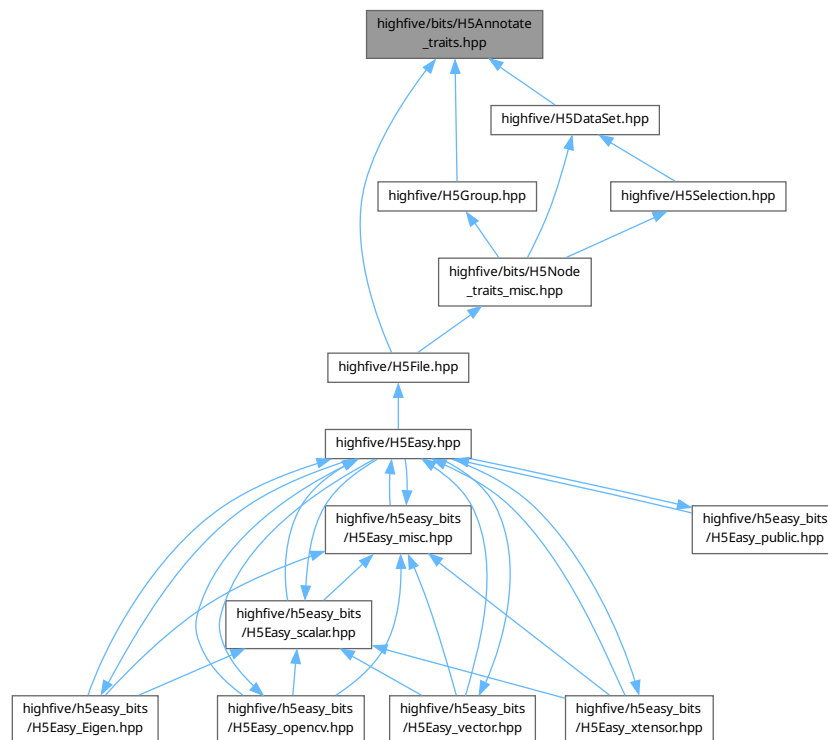
```
00001 #pragma once
00002
00003 #if defined(__GNUC__) || defined(__clang__)
00004 #define H5_DEPRECATED(msg) __attribute__((deprecated(#msg)))
00005 #elif defined(_MSC_VER)
00006 #define H5_DEPRECATED(msg) __declspec(deprecated(#msg))
00007 #else
00008 #pragma message("WARNING: Compiler doesnt support deprecation")
00009 #define H5_DEPRECATED(msg)
00010 #endif
00011
00012
00013 // Forward declarations
00014
00015 namespace HighFive {
00016
00017     enum class LinkType;
00018     enum class ObjectType;
00019     enum class PropertyType;
00020
00021     class Attribute;
00022     class DataSet;
00023     class DataSpace;
00024     class DataType;
00025     class Exception;
00026     class File;
00027     class FileDriver;
00028     class Group;
00029     class Object;
00030     class ObjectInfo;
00031     class Reference;
00032     class Selection;
00033     class SilenceHDF5;
00034
00035     template <typename T>
00036     class AtomicType;
00037
00038     template <typename Derivate>
00039     class AnnotateTraits;
00040
00041     template <std::size_t N>
00042     class FixedLenStringArray;
00043
00044     template <typename Derivate>
00045     class NodeTraits;
00046
00047     template <PropertyType T>
00048     class PropertyList;
00049
00050 } // namespace HighFive
```

10.4 highfive/bits/H5Annotate_traits.hpp File Reference

```
#include <string>
#include "../H5Attribute.hpp"
Include dependency graph for H5Annotate_traits.hpp:
```



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



Classes

- class [HighFive::AnnotateTraits< Derivate >](#)

Namespaces

- namespace [HighFive](#)

10.5 H5Annotate_traits.hpp

[Go to the documentation of this file.](#)

```

00001  /*
00002  *   Copyright (c), 2017, Adrien Devresse <adrien.devresse@epfl.ch>
00003  *
00004  *   Distributed under the Boost Software License, Version 1.0.
00005  *   (See accompanying file LICENSE_1_0.txt or copy at
00006  *    http://www.boost.org/LICENSE_1_0.txt)
00007  *
00008  */
00009  #pragma once
00010
00011  #include <string>
00012
00013  #include "../H5Attribute.hpp"
00014
00015  namespace HighFive {
00016
00017  template <typename Derivate>
00018  class AnnotateTraits {
00019  public:
00027      Attribute createAttribute(const std::string& attribute_name,
00028                              const DataSpace& space,
00029                              const DataType& type);
00030
00038      template <typename Type>
00039      Attribute createAttribute(const std::string& attribute_name, const DataSpace& space);
00040
00049      template <typename T>
00050      Attribute createAttribute(const std::string& attribute_name, const T& data);
00051
00055      void deleteAttribute(const std::string& attribute_name);
00056
00061      Attribute getAttribute(const std::string& attribute_name) const;
00062
00066      size_t getNumberAttributes() const;
00067
00071      std::vector<std::string> listAttributeNames() const;
00072
00076      bool hasAttribute(const std::string& attr_name) const;
00077
00078  private:
00079      using derivate_type = Derivate;
00080  };
00081  } // namespace HighFive

```

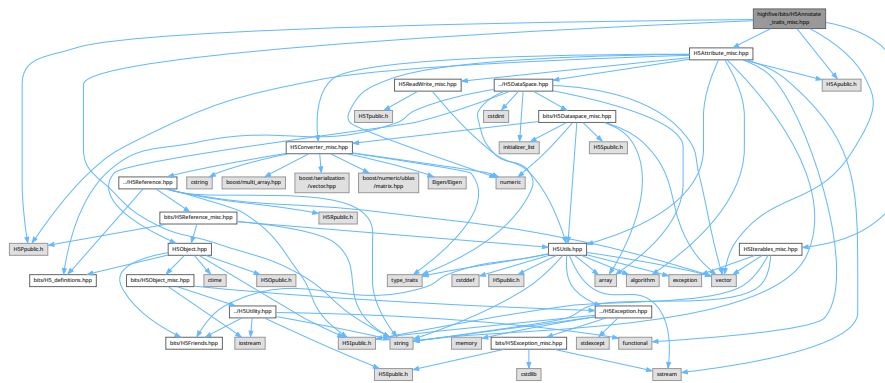
10.6 highfive/bits/H5Annotate_traits_misc.hpp File Reference

```

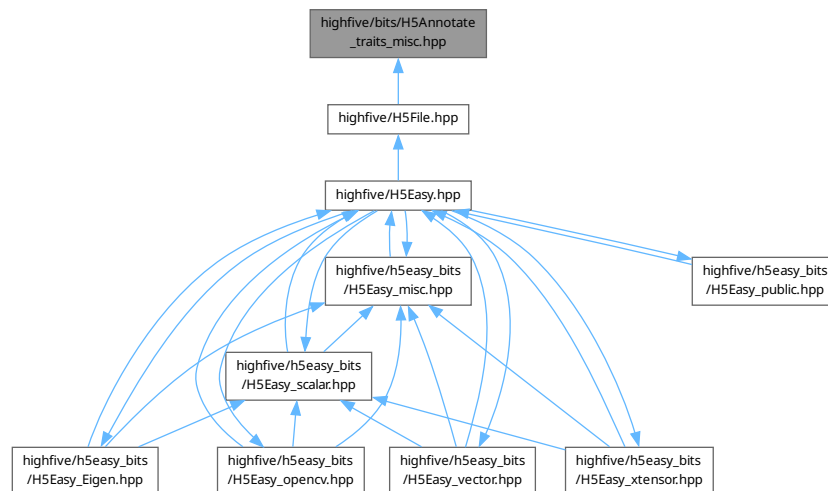
#include <string>
#include <vector>
#include <H5Apublic.h>
#include <H5Ppublic.h>
#include "H5Attribute_misc.hpp"
#include "H5Iterables_misc.hpp"

```

Include dependency graph for H5Annotate_traits_misc.hpp:



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



Namespaces

- namespace [HighFive](#)

10.7 H5Annotate_traits_misc.hpp

[Go to the documentation of this file.](#)

```
00001 /*
00002  * Copyright (c), 2017, Adrien Devresse <adrien.devresse@epfl.ch>
00003  *
00004  * Distributed under the Boost Software License, Version 1.0.
00005  * (See accompanying file LICENSE_1_0.txt or copy at
00006  * http://www.boost.org/LICENSE_1_0.txt)
00007  *
00008  */
00009 #pragma once
00010
```

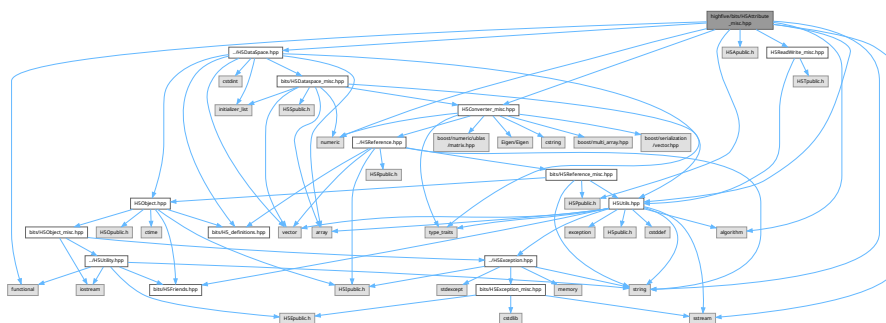
```

00011 #include <string>
00012 #include <vector>
00013
00014 #include <H5Apublic.h>
00015 #include <H5Ppublic.h>
00016
00017 #include "H5Attribute_misc.hpp"
00018 #include "H5Iterables_misc.hpp"
00019
00020 namespace HighFive {
00021
00022 template <typename Derivate>
00023 inline Attribute AnnotateTraits<Derivate>::createAttribute(const std::string& attribute_name,
00024                                                         const DataSpace& space,
00025                                                         const DataType& dtype) {
00026     auto attr_id = H5Acreate2(static_cast<Derivate*>(this)->getId(),
00027                             attribute_name.c_str(),
00028                             dtype.getId(),
00029                             space.getId(),
00030                             H5P_DEFAULT,
00031                             H5P_DEFAULT);
00032     if (attr_id < 0) {
00033         HDF5ErrMapper::ToException<AttributeException>(
00034             std::string("Unable to create the attribute \"" + attribute_name + "\"");
00035     }
00036     return detail::make_attribute(attr_id);
00037 }
00038
00039 template <typename Derivate>
00040 template <typename Type>
00041 inline Attribute AnnotateTraits<Derivate>::createAttribute(const std::string& attribute_name,
00042                                                         const DataSpace& space) {
00043     return createAttribute(attribute_name, space, create_and_check_datatype<Type>());
00044 }
00045
00046 template <typename Derivate>
00047 template <typename T>
00048 inline Attribute AnnotateTraits<Derivate>::createAttribute(const std::string& attribute_name,
00049                                                         const T& data) {
00050     Attribute att =
00051         createAttribute(attribute_name,
00052                         DataSpace::From(data),
00053                         create_and_check_datatype<typename details::inspector<T>::base_type>());
00054     att.write(data);
00055     return att;
00056 }
00057
00058 template <typename Derivate>
00059 inline void AnnotateTraits<Derivate>::deleteAttribute(const std::string& attribute_name) {
00060     if (H5Adelete(static_cast<const Derivate*>(this)->getId(), attribute_name.c_str()) < 0) {
00061         HDF5ErrMapper::ToException<AttributeException>(
00062             std::string("Unable to delete attribute \"" + attribute_name + "\"");
00063     }
00064 }
00065
00066 template <typename Derivate>
00067 inline Attribute AnnotateTraits<Derivate>::getAttribute(const std::string& attribute_name) const {
00068     const auto attr_id =
00069         H5Aopen(static_cast<const Derivate*>(this)->getId(), attribute_name.c_str(), H5P_DEFAULT);
00070     if (attr_id < 0) {
00071         HDF5ErrMapper::ToException<AttributeException>(
00072             std::string("Unable to open the attribute \"" + attribute_name + "\"");
00073     }
00074     return detail::make_attribute(attr_id);
00075 }
00076
00077 template <typename Derivate>
00078 inline size_t AnnotateTraits<Derivate>::getNumberAttributes() const {
00079     int res = H5Aget_num_attrs(static_cast<const Derivate*>(this)->getId());
00080     if (res < 0) {
00081         HDF5ErrMapper::ToException<AttributeException>(
00082             std::string("Unable to count attributes in existing group or file"));
00083     }
00084     return static_cast<size_t>(res);
00085 }
00086
00087 template <typename Derivate>
00088 inline std::vector<std::string> AnnotateTraits<Derivate>::listAttributeNames() const {
00089     std::vector<std::string> names;
00090     details::HighFiveIterateData iterateData(names);
00091
00092     size_t num_objs = getNumberAttributes();
00093     names.reserve(num_objs);
00094
00095     if (H5Aiterate2(static_cast<const Derivate*>(this)->getId(),
00096                   H5_INDEX_NAME,
00097                   H5_ITER_INC,

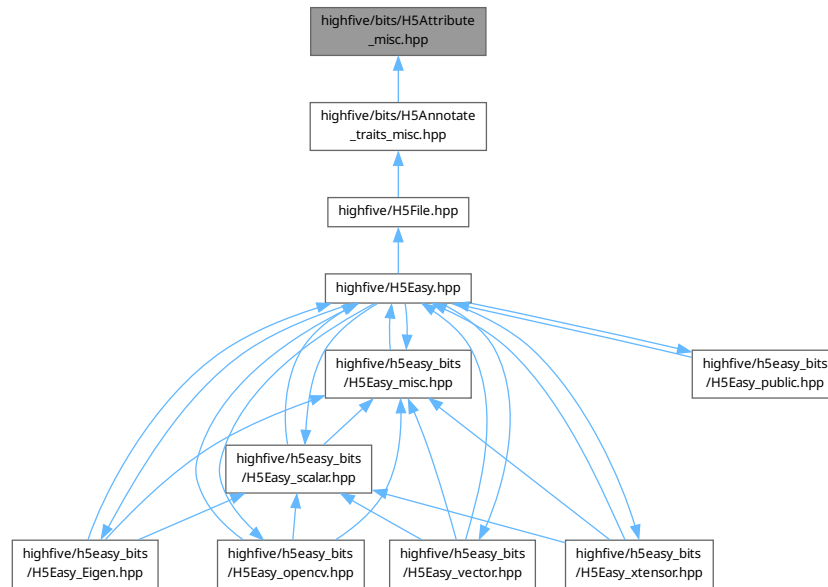
```

10.8 highfive/bits/H5Attribute misc.hpp File Reference

Include dependency graph for H5Attribute misc.hpp:



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



Namespaces

- namespace [HighFive](#)

10.9 H5Attribute_misc.hpp

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c), 2017, Ali Can Demiralp <ali.demiralp@rwth-aachen.de>
00003  *
00004  * Distributed under the Boost Software License, Version 1.0.
00005  * (See accompanying file LICENSE_1_0.txt or copy at
00006  * http://www.boost.org/LICENSE_1_0.txt)
00007  *
00008  */
00009 #pragma once
00010
00011 #include <algorithm>
00012 #include <functional>
00013 #include <numeric>
00014 #include <sstream>
00015 #include <string>
00016
00017 #include <H5Apublic.h>
00018 #include <H5Ppublic.h>
00019
00020 #include "../H5DataSpace.hpp"
00021 #include "H5Converter_misc.hpp"
00022 #include "H5ReadWrite_misc.hpp"
00023 #include "H5Utils.hpp"
00024
00025 namespace HighFive {
00026
00027 inline std::string Attribute::getName() const {
00028     return details::get_name(
00029         [&](char* buffer, size_t length) { return H5Aget_name(_hid, length, buffer); });
00030 }
00031
00032 inline size_t Attribute::getStorageSize() const {
00033     return static_cast<size_t>(H5Aget_storage_size(_hid));

```



```

00034 }
00035
00036 inline DataType Attribute::getDataType() const {
00037     DataType res;
00038     res._hid = H5Aget_type(_hid);
00039     return res;
00040 }
00041
00042 inline DataSpace Attribute::getSpace() const {
00043     DataSpace space;
00044     if ((space._hid = H5Aget_space(_hid)) < 0) {
00045         HDF5ErrMapper::ToException<AttributeException>("Unable to get DataSpace out of Attribute");
00046     }
00047     return space;
00048 }
00049
00050 inline DataSpace Attribute::getMemSpace() const {
00051     return getSpace();
00052 }
00053
00054 template <typename T>
00055 inline T Attribute::read() const {
00056     T array;
00057     read(array);
00058     return array;
00059 }
00060
00061 template <typename T>
00062 inline void Attribute::read(T& array) const {
00063     const DataSpace& mem_space = getMemSpace();
00064     const details::BufferInfo<T> buffer_info(
00065         getDataType(),
00066         [this]() -> std::string { return this->getName(); },
00067         details::BufferInfo<T>::read);
00068
00069     if (!details::checkDimensions(mem_space, buffer_info.n_dimensions)) {
00070         std::ostringstream ss;
00071         ss << "Impossible to read DataSet of dimensions " << mem_space.getNumberDimensions()
00072            << " into arrays of dimensions " << buffer_info.n_dimensions;
00073         throw DataSpaceException(ss.str());
00074     }
00075     auto dims = mem_space.getDimensions();
00076
00077     if (mem_space.getElementCount() == 0) {
00078         auto effective_dims = details::squeezeDimensions(dims,
00079                                                         details::inspector<T>::recursive_ndim);
00080
00081         details::inspector<T>::prepare(array, effective_dims);
00082         return;
00083     }
00084
00085     auto r = details::data_converter::get_reader<T>(dims, array);
00086     read(r.get_pointer(), buffer_info.data_type);
00087     // re-arrange results
00088     r.unserialize();
00089     auto t = create_datatype<typename details::inspector<T>::base_type>();
00090     auto c = t.getClass();
00091     if (c == DataTypeClass::VarLen || t.isVariableStr()) {
00092 #if H5_VERSION_GE(1, 12, 0)
00093         // This one have been created in 1.12.0
00094         (void) H5Tclaim(t.getId(), mem_space.getId(), H5P_DEFAULT, r.get_pointer());
00095 #else
00096         // This one is deprecated since 1.12.0
00097         (void) H5Tvlen_reclaim(t.getId(), mem_space.getId(), H5P_DEFAULT, r.get_pointer());
00098 #endif
00099     }
00100 }
00101
00102 template <typename T>
00103 inline void Attribute::read(T* array, const DataType& dtype) const {
00104     static_assert(!std::is_const<T>::value,
00105                 "read() requires a non-const structure to read data into");
00106     using element_type = typename details::inspector<T>::base_type;
00107     // Auto-detect mem datatype if not provided
00108     const DataType& mem_datatype = dtype.empty() ? create_and_check_datatype<element_type>()
00109                                                  : dtype;
00110
00111     if (H5Aread(getId(), mem_datatype.getId(), static_cast<void*>(array)) < 0) {
00112         HDF5ErrMapper::ToException<AttributeException>("Error during HDF5 Read: ");
00113     }
00114 }
00115
00116 template <typename T>
00117 inline void Attribute::write(const T& buffer) {
00118     const DataSpace& mem_space = getMemSpace();
00119
00120     if (mem_space.getElementCount() == 0) {

```

```

00121         return;
00122     }
00123
00124     const details::BufferInfo<T> buffer_info(
00125         getDataType(),
00126         [this]() -> std::string { return this->getName(); },
00127         details::BufferInfo<T>::write);
00128
00129     if (!details::checkDimensions(mem_space, buffer_info.n_dimensions)) {
00130         std::ostringstream ss;
00131         ss << "Impossible to write buffer of dimensions " << buffer_info.n_dimensions
00132             << " into dataset of dimensions " << mem_space.getNumberDimensions();
00133         throw DataSpaceException(ss.str());
00134     }
00135     auto w = details::data_converter::serialize<T>(buffer);
00136     write_raw(w.get_pointer(), buffer_info.data_type);
00137 }
00138
00139 template <typename T>
00140 inline void Attribute::write_raw(const T* buffer, const DataType& dtype) {
00141     using element_type = typename details::inspector<T>::base_type;
00142     const auto& mem_datatype = dtype.empty() ? create_and_check_datatype<element_type>() : dtype;
00143
00144     if (H5Awrite(getId(), mem_datatype.getId(), buffer) < 0) {
00145         HDF5ErrMapper::ToException<DataSetException>("Error during HDF5 Write: ");
00146     }
00147 }
00148
00149 } // namespace HighFive

```

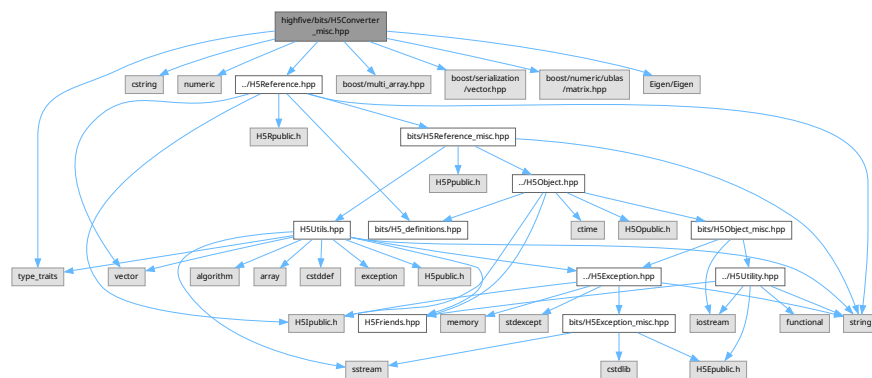
10.10 highfive/bits/H5Converter_misc.hpp File Reference

```

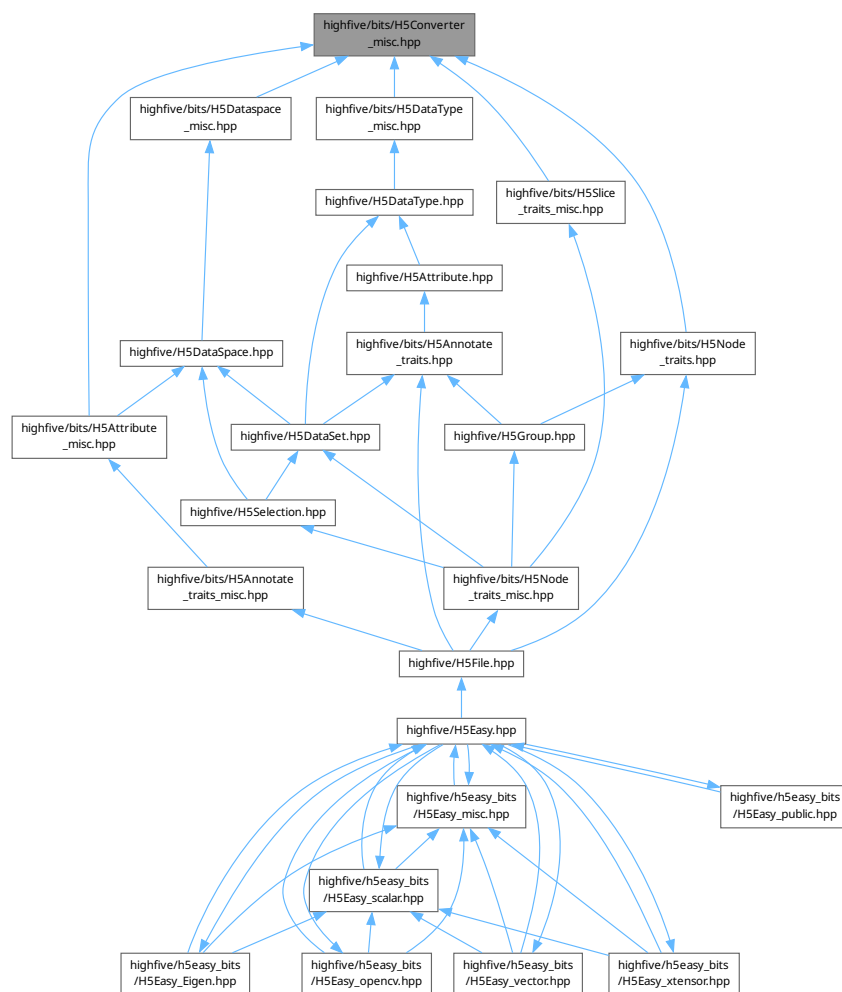
#include <type_traits>
#include <cstring>
#include <numeric>
#include "../H5Reference.hpp"
#include <boost/multi_array.hpp>
#include <boost/serialization/vector.hpp>
#include <boost/numeric/ublas/matrix.hpp>
#include <Eigen/Eigen>

```

Include dependency graph for H5Converter_misc.hpp:



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



Namespaces

- namespace `HighFive`

Typedefs

- `template<typename T >`
`using HighFive::unqualified_t = typename std::remove_const< typename std::remove_reference< T >::type`
`>::type`

Functions

- `size_t HighFive::compute_total_size (const std::vector< size_t > &dims)`

10.11 H5Converter_misc.hpp

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c) 2022 Blue Brain Project
00003  *
00004  * Distributed under the Boost Software License, Version 1.0.
00005  * (See accompanying file LICENSE_1_0.txt or copy at
00006  * http://www.boost.org/LICENSE_1_0.txt)
00007  *
00008  */
00009 #pragma once
00010
00011 #include <type_traits>
00012 #include <cstring>
00013 #include <numeric>
00014
00015 #include "../H5Reference.hpp"
00016 #ifdef H5_USE_BOOST
00017 #include <boost/multi_array.hpp>
00018 // starting Boost 1.64, serialization header must come before ublas
00019 #include <boost/serialization/vector.hpp>
00020 #include <boost/numeric/ublas/matrix.hpp>
00021 #endif
00022 #ifdef H5_USE_EIGEN
00023 #include <Eigen/Eigen>
00024 #endif
00025
00026 namespace HighFive {
00027 namespace details {
00028
00029 inline bool checkDimensions(const std::vector<size_t>& dims, size_t n_dim_requested) {
00030     size_t n_dim_actual = dims.size();
00031
00032     // We should allow reading scalar from shapes like `(1, 1, 1)`.
00033     if (n_dim_requested == 0) {
00034         if (n_dim_actual == 0ul) {
00035             return true;
00036         }
00037
00038         return size_t(std::count(dims.begin(), dims.end(), 1ul)) == n_dim_actual;
00039     }
00040
00041     // For non-scalar datasets, we can squeeze away singleton dimension, but
00042     // we never add any.
00043     if (n_dim_actual < n_dim_requested) {
00044         return false;
00045     }
00046
00047     // Special case for 1-dimensional arrays, which can squeeze `1`s from either
00048     // side simultaneously if needed.
00049     if (n_dim_requested == 1ul) {
00050         return n_dim_actual >= 1ul &&
00051             size_t(std::count(dims.begin(), dims.end(), 1ul)) >= n_dim_actual - 1ul;
00052     }
00053
00054     // All other cases strip front only. This avoid unstable behaviour when
00055     // squeezing singleton dimensions.
00056     size_t n_dim_excess = n_dim_actual - n_dim_requested;
00057
00058     bool squeeze_back = true;
00059     for (size_t i = 1; i <= n_dim_excess; ++i) {
00060         if (dims[n_dim_actual - i] != 1) {
00061             squeeze_back = false;
00062             break;
00063         }
00064     }
00065
00066     return squeeze_back;
00067 }
00068
00069 inline std::vector<size_t> squeezeDimensions(const std::vector<size_t>& dims,
00070                                             size_t n_dim_requested) {
00071     auto format_error_message = [&]() -> std::string {
00072         return "Can't interpret dims = " + format_vector(dims) + " as " +
00073             std::to_string(n_dim_requested) + "-dimensional.";
00074     };
00075
00076     if (n_dim_requested == 0) {
00077         if (!checkDimensions(dims, n_dim_requested)) {
00078             throw std::invalid_argument(format_error_message());
00079         }
00080     }
00081
00082     return {1ul};

```

```

00083     }
00084
00085     auto n_dim = dims.size();
00086     if (n_dim < n_dim_requested) {
00087         throw std::invalid_argument(format_error_message());
00088     }
00089
00090     if (n_dim_requested == 1ul) {
00091         size_t non_singleton_dim = size_t(-1);
00092         for (size_t i = 0; i < n_dim; ++i) {
00093             if (dims[i] != 1ul) {
00094                 if (non_singleton_dim == size_t(-1)) {
00095                     non_singleton_dim = i;
00096                 } else {
00097                     throw std::invalid_argument(format_error_message());
00098                 }
00099             }
00100         }
00101
00102         return {dims[std::min(non_singleton_dim, n_dim - 1)]};
00103     }
00104
00105     size_t n_dim_excess = dims.size() - n_dim_requested;
00106     for (size_t i = 1; i <= n_dim_excess; ++i) {
00107         if (dims[n_dim - i] != 1) {
00108             throw std::invalid_argument(format_error_message());
00109         }
00110     }
00111
00112     return std::vector<size_t>(dims.begin(),
00113                               dims.end() - static_cast<std::ptrdiff_t>(n_dim_excess));
00114 }
00115 } // namespace details
00116
00117 inline size_t compute_total_size(const std::vector<size_t>& dims) {
00118     return std::accumulate(dims.begin(), dims.end(), size_t{1u}, std::multiplies<size_t>());
00119 }
00120
00121 template <typename T>
00122 using unqualified_t = typename std::remove_const<typename std::remove_reference<T>::type>::type;
00123
00124 /*****
00125 inspector<T> {
00126     using type = T
00127     // base_type is the base type inside c++ (e.g. std::vector<int> => int)
00128     using base_type
00129     // hdf5_type is the base read by hdf5 (c-type) (e.g. std::vector<std::string> => const char*)
00130     using hdf5_type
00131
00132     // Number of dimensions starting from here
00133     static constexpr size_t recursive_ndim
00134     // Is the inner type trivially copyable for optimisation
00135     // If this value is true: data() is mandatory
00136     // If this value is false: getSizeVal, getSize, serialize, unserialize are mandatory
00137     static constexpr bool is_trivially_copyable
00138
00139     // Reading:
00140     // Allocate the value following dims (should be recursive)
00141     static void prepare(type& val, const std::vector<std::size_t> dims)
00142     // Return the size of the vector pass to/from hdf5 from a vector of dims
00143     static size_t getSize(const std::vector<size_t>& dims)
00144     // Return a pointer of the first value of val (for reading)
00145     static hdf5_type* data(type& val)
00146     // Take a serialized vector 'in', some dims and copy value to val (for reading)
00147     static void unserialize(const hdf5_type* in, const std::vector<size_t>&i, type& val)
00148
00149
00150
00151     // Writing:
00152     // Return the size of the vector pass to/from hdf5 from a value
00153     static size_t getSizeVal(const type& val)
00154     // Return a point of the first value of val
00155     static const hdf5_type* data(const type& val)
00156     // Take a val and serialize it inside 'out'
00157     static void serialize(const type& val, hdf5_type* out)
00158     // Return an array of dimensions of the space needed for writing val
00159     static std::vector<size_t> getDimensions(const type& val)
00160 }
00161 *****/
00162
00163 namespace details {
00164 template <typename T>
00165 struct type_helper {
00166     using type = unqualified_t<T>;
00167     using base_type = unqualified_t<T>;
00168     using hdf5_type = base_type;

```

```

00170
00171     static constexpr size_t ndim = 0;
00172     static constexpr size_t recursive_ndim = ndim;
00173     static constexpr bool is_trivially_copyable = std::is_trivially_copyable<type>::value;
00174
00175     static std::vector<size_t> getDimensions(const type& /* val */) {
00176         return {};
00177     }
00178
00179     static size_t getSizeVal(const type& val) {
00180         return compute_total_size(getDimensions(val));
00181     }
00182
00183     static size_t getSize(const std::vector<size_t>& dims) {
00184         return compute_total_size(dims);
00185     }
00186
00187     static void prepare(type& /* val */, const std::vector<size_t>& /* dims */) {}
00188
00189     static hdf5_type* data(type& val) {
00190         static_assert(is_trivially_copyable, "The type is not trivially copyable");
00191         return &val;
00192     }
00193
00194     static const hdf5_type* data(const type& val) {
00195         static_assert(is_trivially_copyable, "The type is not trivially copyable");
00196         return &val;
00197     }
00198
00199     static void serialize(const type& val, hdf5_type* m) {
00200         static_assert(is_trivially_copyable, "The type is not trivially copyable");
00201         *m = val;
00202     }
00203
00204     static void unserialize(const hdf5_type* vec,
00205                             const std::vector<size_t>& /* dims */,
00206                             type& val) {
00207         static_assert(is_trivially_copyable, "The type is not trivially copyable");
00208         val = vec[0];
00209     }
00210 };
00211
00212 template <typename T>
00213 struct inspector: type_helper<T> {};
00214
00215 enum class Boolean : int8_t {
00216     HighFiveFalse = 0,
00217     HighFiveTrue = 1,
00218 };
00219
00220 template <>
00221 struct inspector<bool>: type_helper<bool> {
00222     using base_type = Boolean;
00223     using hdf5_type = int8_t;
00224
00225     static constexpr bool is_trivially_copyable = false;
00226
00227     static hdf5_type* data(type& /* val */) {
00228         throw DataSpaceException("A boolean cannot be read directly.");
00229     }
00230
00231     static const hdf5_type* data(const type& /* val */) {
00232         throw DataSpaceException("A boolean cannot be written directly.");
00233     }
00234
00235     static void unserialize(const hdf5_type* vec,
00236                             const std::vector<size_t>& /* dims */,
00237                             type& val) {
00238         val = vec[0] != 0 ? true : false;
00239     }
00240
00241     static void serialize(const type& val, hdf5_type* m) {
00242         *m = val ? 1 : 0;
00243     }
00244 };
00245
00246 template <>
00247 struct inspector<std::string>: type_helper<std::string> {
00248     using hdf5_type = const char*;
00249
00250     static hdf5_type* data(type& /* val */) {
00251         throw DataSpaceException("A std::string cannot be read directly.");
00252     }
00253
00254     static const hdf5_type* data(const type& /* val */) {
00255         throw DataSpaceException("A std::string cannot be written directly.");
00256     }

```

```

00257
00258     static void serialize(const type& val, hdf5_type* m) {
00259         *m = val.c_str();
00260     }
00261
00262     static void unserialize(const hdf5_type* vec,
00263                             const std::vector<size_t>& /* dims */,
00264                             type& val) {
00265         val = vec[0];
00266     }
00267 };
00268
00269 template <>
00270 struct inspector<Reference>: type_helper<Reference> {
00271     using hdf5_type = hobj_ref_t;
00272
00273     static constexpr bool is_trivially_copyable = false;
00274
00275     static hdf5_type* data(type& /* val */) {
00276         throw DataSpaceException("A Reference cannot be read directly.");
00277     }
00278
00279     static const hdf5_type* data(const type& /* val */) {
00280         throw DataSpaceException("A Reference cannot be written directly.");
00281     }
00282
00283     static void serialize(const type& val, hdf5_type* m) {
00284         hobj_ref_t ref;
00285         val.create_ref(&ref);
00286         *m = ref;
00287     }
00288
00289     static void unserialize(const hdf5_type* vec,
00290                             const std::vector<size_t>& /* dims */,
00291                             type& val) {
00292         val = type{vec[0]};
00293     }
00294 };
00295
00296 template <size_t N>
00297 struct inspector<FixedLenStringArray<N>> {
00298     using type = FixedLenStringArray<N>;
00299     using value_type = char*;
00300     using base_type = FixedLenStringArray<N>;
00301     using hdf5_type = char;
00302
00303     static constexpr size_t ndim = 1;
00304     static constexpr size_t recursive_ndim = ndim;
00305     static constexpr bool is_trivially_copyable = false;
00306
00307     static std::vector<size_t> getDimensions(const type& val) {
00308         return std::vector<size_t>{val.size()};
00309     }
00310
00311     static size_t getSizeVal(const type& val) {
00312         return N * compute_total_size(getDimensions(val));
00313     }
00314
00315     static size_t getSize(const std::vector<size_t>& dims) {
00316         return N * compute_total_size(dims);
00317     }
00318
00319     static void prepare(type& /* val */, const std::vector<size_t>& dims) {
00320         if (dims[0] > N) {
00321             std::ostringstream os;
00322             os << "Size of FixedlenStringArray (" << N << ") is too small for dims (" << dims[0]
00323                 << ").";
00324             throw DataSpaceException(os.str());
00325         }
00326     }
00327
00328     static hdf5_type* data(type& val) {
00329         return val.data();
00330     }
00331
00332     static const hdf5_type* data(const type& val) {
00333         return val.data();
00334     }
00335
00336     static void serialize(const type& val, hdf5_type* m) {
00337         for (size_t i = 0; i < val.size(); ++i) {
00338             std::memcpy(m + i * N, val[i], N);
00339         }
00340     }
00341
00342     static void unserialize(const hdf5_type* vec, const std::vector<size_t>& dims, type& val) {
00343         for (size_t i = 0; i < dims[0]; ++i) {

```

```

00344         std::array<char, N> s;
00345         std::memcpy(s.data(), vec + (i * N), N);
00346         val.push_back(s);
00347     }
00348 }
00349 };
00350
00351 template <typename T>
00352 struct inspector<std::vector<T> > {
00353     using type = std::vector<T>;
00354     using value_type = unqualified_t<T>;
00355     using base_type = typename inspector<value_type>::base_type;
00356     using hdf5_type = typename inspector<value_type>::hdf5_type;
00357
00358     static constexpr size_t ndim = 1;
00359     static constexpr size_t recursive_ndim = ndim + inspector<value_type>::recursive_ndim;
00360     static constexpr bool is_trivially_copyable = std::is_trivially_copyable<value_type>::value &&
00361         inspector<value_type>::is_trivially_copyable;
00362
00363     static std::vector<size_t> getDimensions(const type& val) {
00364         std::vector<size_t> sizes(recursive_ndim, 1ul);
00365         sizes[0] = val.size();
00366         if (!val.empty()) {
00367             auto s = inspector<value_type>::getDimensions(val[0]);
00368             std::copy(s.begin(), s.end(), sizes.begin() + 1);
00369         }
00370         return sizes;
00371     }
00372
00373     static size_t getSizeVal(const type& val) {
00374         return compute_total_size(getDimensions(val));
00375     }
00376
00377     static size_t getSize(const std::vector<size_t>& dims) {
00378         return compute_total_size(dims);
00379     }
00380
00381     static void prepare(type& val, const std::vector<size_t>& dims) {
00382         val.resize(dims[0]);
00383         std::vector<size_t> next_dims(dims.begin() + 1, dims.end());
00384         for (auto&& e: val) {
00385             inspector<value_type>::prepare(e, next_dims);
00386         }
00387     }
00388
00389     static hdf5_type* data(type& val) {
00390         return inspector<value_type>::data(val[0]);
00391     }
00392
00393     static const hdf5_type* data(const type& val) {
00394         return inspector<value_type>::data(val[0]);
00395     }
00396
00397     static void serialize(const type& val, hdf5_type* m) {
00398         size_t subsize = inspector<value_type>::getSizeVal(val[0]);
00399         for (auto&& e: val) {
00400             inspector<value_type>::serialize(e, m);
00401             m += subsize;
00402         }
00403     }
00404
00405     static void unserialize(const hdf5_type* vec_align,
00406                             const std::vector<size_t>& dims,
00407                             type& val) {
00408         std::vector<size_t> next_dims(dims.begin() + 1, dims.end());
00409         size_t next_size = compute_total_size(next_dims);
00410         for (size_t i = 0; i < dims[0]; ++i) {
00411             inspector<value_type>::unserialize(vec_align + i * next_size, next_dims, val[i]);
00412         }
00413     }
00414 };
00415
00416 template <>
00417 struct inspector<std::vector<bool> > {
00418     using type = std::vector<bool>;
00419     using value_type = bool;
00420     using base_type = Boolean;
00421     using hdf5_type = uint8_t;
00422
00423     static constexpr size_t ndim = 1;
00424     static constexpr size_t recursive_ndim = ndim;
00425     static constexpr bool is_trivially_copyable = false;
00426
00427     static std::vector<size_t> getDimensions(const type& val) {
00428         std::vector<size_t> sizes{val.size()};
00429         return sizes;
00430     }

```



```

00431
00432     static size_t getSizeVal(const type& val) {
00433         return val.size();
00434     }
00435
00436     static size_t getSize(const std::vector<size_t>& dims) {
00437         if (dims.size() > 1) {
00438             throw DataSpaceException("std::vector<bool> is only 1 dimension.");
00439         }
00440         return dims[0];
00441     }
00442
00443     static void prepare(type& val, const std::vector<size_t>& dims) {
00444         if (dims.size() > 1) {
00445             throw DataSpaceException("std::vector<bool> is only 1 dimension.");
00446         }
00447         val.resize(dims[0]);
00448     }
00449
00450     static hdf5_type* data(type& /* val */) {
00451         throw DataSpaceException("A std::vector<bool> cannot be read directly.");
00452     }
00453
00454     static const hdf5_type* data(const type& /* val */) {
00455         throw DataSpaceException("A std::vector<bool> cannot be written directly.");
00456     }
00457
00458     static void serialize(const type& val, hdf5_type* m) {
00459         for (size_t i = 0; i < val.size(); ++i) {
00460             m[i] = val[i] ? 1 : 0;
00461         }
00462     }
00463
00464     static void unserialize(const hdf5_type* vec_align,
00465                             const std::vector<size_t>& dims,
00466                             type& val) {
00467         for (size_t i = 0; i < dims[0]; ++i) {
00468             val[i] = vec_align[i] != 0 ? true : false;
00469         }
00470     }
00471 };
00472
00473 template <typename T, size_t N>
00474 struct inspector<std::array<T, N> {
00475     using type = std::array<T, N>;
00476     using value_type = unqualified_t<T>;
00477     using base_type = typename inspector<value_type>::base_type;
00478     using hdf5_type = typename inspector<value_type>::hdf5_type;
00479
00480     static constexpr size_t ndim = 1;
00481     static constexpr size_t recursive_ndim = ndim + inspector<value_type>::recursive_ndim;
00482     static constexpr bool is_trivially_copyable = std::is_trivially_copyable<value_type>::value &&
00483         inspector<value_type>::is_trivially_copyable;
00484
00485     static std::vector<size_t> getDimensions(const type& val) {
00486         std::vector<size_t> sizes{N};
00487         if (!val.empty()) {
00488             auto s = inspector<value_type>::getDimensions(val[0]);
00489             sizes.insert(sizes.end(), s.begin(), s.end());
00490         }
00491         return sizes;
00492     }
00493
00494     static size_t getSizeVal(const type& val) {
00495         return compute_total_size(getDimensions(val));
00496     }
00497
00498     static size_t getSize(const std::vector<size_t>& dims) {
00499         return compute_total_size(dims);
00500     }
00501
00502     static void prepare(type& /* val */, const std::vector<size_t>& dims) {
00503         if (dims[0] > N) {
00504             std::ostringstream os;
00505             os << "Size of std::array (" << N << ") is too small for dims (" << dims[0] << ").";
00506             throw DataSpaceException(os.str());
00507         }
00508     }
00509
00510     static hdf5_type* data(type& val) {
00511         return inspector<value_type>::data(val[0]);
00512     }
00513
00514     static const hdf5_type* data(const type& val) {
00515         return inspector<value_type>::data(val[0]);
00516     }
00517

```

```

00518     static void serialize(const type& val, hdf5_type* m) {
00519         size_t subsize = inspector<value_type>::getSizeVal(val[0]);
00520         for (auto& e: val) {
00521             inspector<value_type>::serialize(e, m);
00522             m += subsize;
00523         }
00524     }
00525
00526     static void unserialize(const hdf5_type* vec_align,
00527                             const std::vector<size_t>& dims,
00528                             type& val) {
00529         if (dims[0] != N) {
00530             std::ostringstream os;
00531             os << "Impossible to pair DataSet with " << dims[0] << " elements into an array with "
00532                 << N << " elements.";
00533             throw DataSpaceException(os.str());
00534         }
00535         std::vector<size_t> next_dims(dims.begin() + 1, dims.end());
00536         size_t next_size = compute_total_size(next_dims);
00537         for (size_t i = 0; i < dims[0]; ++i) {
00538             inspector<value_type>::unserialize(vec_align + i * next_size, next_dims, val[i]);
00539         }
00540     }
00541 };
00542
00543 // Cannot be use for reading
00544 template <typename T>
00545 struct inspector<T*> {
00546     using type = T*;
00547     using value_type = unqualified_t<T>;
00548     using base_type = typename inspector<value_type>::base_type;
00549     using hdf5_type = typename inspector<value_type>::hdf5_type;
00550
00551     static constexpr size_t ndim = 1;
00552     static constexpr size_t recursive_ndim = ndim + inspector<value_type>::recursive_ndim;
00553     static constexpr bool is_trivially_copyable = std::is_trivially_copyable<value_type>::value &&
00554         inspector<value_type>::is_trivially_copyable;
00555
00556     static size_t getSizeVal(const type& /* val */) {
00557         throw DataSpaceException("Not possible to have size of a T*");
00558     }
00559
00560     static std::vector<size_t> getDimensions(const type& /* val */) {
00561         throw DataSpaceException("Not possible to have size of a T*");
00562     }
00563
00564     static const hdf5_type* data(const type& val) {
00565         return reinterpret_cast<const hdf5_type*>(val);
00566     }
00567
00568     /* it works because there is only T[][][] currently
00569        we will fix it one day */
00570     static void serialize(const type& /* val */, hdf5_type* /* m */) {
00571         throw DataSpaceException("Not possible to serialize a T*");
00572     }
00573 };
00574
00575 // Cannot be use for reading
00576 template <typename T, size_t N>
00577 struct inspector<T[N]> {
00578     using type = T[N];
00579     using value_type = unqualified_t<T>;
00580     using base_type = typename inspector<value_type>::base_type;
00581     using hdf5_type = typename inspector<value_type>::hdf5_type;
00582
00583     static constexpr size_t ndim = 1;
00584     static constexpr size_t recursive_ndim = ndim + inspector<value_type>::recursive_ndim;
00585     static constexpr bool is_trivially_copyable = std::is_trivially_copyable<value_type>::value &&
00586         inspector<value_type>::is_trivially_copyable;
00587
00588     static size_t getSizeVal(const type& val) {
00589         return compute_total_size(getDimensions(val));
00590     }
00591
00592     static std::vector<size_t> getDimensions(const type& val) {
00593         std::vector<size_t> sizes{N};
00594         if (N > 0) {
00595             auto s = inspector<value_type>::getDimensions(val[0]);
00596             sizes.insert(sizes.end(), s.begin(), s.end());
00597         }
00598         return sizes;
00599     }
00600
00601     static const hdf5_type* data(const type& val) {
00602         return inspector<value_type>::data(val[0]);
00603     }
00604

```

```

00605     /* it works because there is only T[][][] currently
00606     we will fix it one day */
00607     static void serialize(const type& val, hdf5_type* m) {
00608         size_t subsize = inspector<value_type>::getSizeVal(val[0]);
00609         for (size_t i = 0; i < N; ++i) {
00610             inspector<value_type>::serialize(val[i], m + i * subsize);
00611         }
00612     }
00613 };
00614
00615 #ifdef H5_USE_EIGEN
00616 template <typename T, int M, int N>
00617 struct inspector<Eigen::Matrix<T, M, N> > {
00618     using type = Eigen::Matrix<T, M, N>;
00619     using value_type = T;
00620     using base_type = typename inspector<value_type>::base_type;
00621     using hdf5_type = base_type;
00622
00623     static constexpr size_t ndim = 2;
00624     static constexpr size_t recursive_ndim = ndim + inspector<value_type>::recursive_ndim;
00625     static constexpr bool is_trivially_copyable = std::is_trivially_copyable<value_type>::value &&
00626         inspector<value_type>::is_trivially_copyable;
00627
00628     static std::vector<size_t> getDimensions(const type& val) {
00629         std::vector<size_t> sizes{static_cast<size_t>(val.rows()), static_cast<size_t>(val.cols())};
00630         auto s = inspector<value_type>::getDimensions(val.data()[0]);
00631         sizes.insert(sizes.end(), s.begin(), s.end());
00632         return sizes;
00633     }
00634
00635     static size_t getSizeVal(const type& val) {
00636         return compute_total_size(getDimensions(val));
00637     }
00638
00639     static size_t getSize(const std::vector<size_t>& dims) {
00640         return compute_total_size(dims);
00641     }
00642
00643     static void prepare(type& val, const std::vector<size_t>& dims) {
00644         if (dims[0] != static_cast<size_t>(val.rows()) ||
00645             dims[1] != static_cast<size_t>(val.cols())) {
00646             val.resize(static_cast<typename type::Index>(dims[0]),
00647                       static_cast<typename type::Index>(dims[1]));
00648         }
00649     }
00650
00651     static hdf5_type* data(type& val) {
00652         return inspector<value_type>::data(*val.data());
00653     }
00654
00655     static const hdf5_type* data(const type& val) {
00656         return inspector<value_type>::data(*val.data());
00657     }
00658
00659     static void serialize(const type& val, hdf5_type* m) {
00660         std::memcpy(m, val.data(), static_cast<size_t>(val.size()) * sizeof(hdf5_type));
00661     }
00662
00663     static void unserialize(const hdf5_type* vec_align,
00664                             const std::vector<size_t>& dims,
00665                             type& val) {
00666         if (dims.size() < 2) {
00667             std::ostringstream os;
00668             os << "Impossible to pair DataSet with " << dims.size()
00669               << " dimensions into an eigen-matrix.";
00670             throw DataSpaceException(os.str());
00671         }
00672         std::memcpy(val.data(), vec_align, compute_total_size(dims) * sizeof(hdf5_type));
00673     }
00674 };
00675 #endif
00676
00677 #ifdef H5_USE_BOOST
00678 template <typename T, size_t Dims>
00679 struct inspector<boost::multi_array<T, Dims> > {
00680     using type = boost::multi_array<T, Dims>;
00681     using value_type = T;
00682     using base_type = typename inspector<value_type>::base_type;
00683     using hdf5_type = typename inspector<value_type>::hdf5_type;
00684
00685     static constexpr size_t ndim = Dims;
00686     static constexpr size_t recursive_ndim = ndim + inspector<value_type>::recursive_ndim;
00687     static constexpr bool is_trivially_copyable = std::is_trivially_copyable<value_type>::value &&
00688         inspector<value_type>::is_trivially_copyable;
00689
00690     static std::vector<size_t> getDimensions(const type& val) {
00691         std::vector<size_t> sizes;

```

```

00692         for (size_t i = 0; i < ndim; ++i) {
00693             sizes.push_back(val.shape()[i]);
00694         }
00695         auto s = inspector<value_type>::getDimensions(val.data()[0]);
00696         sizes.insert(sizes.end(), s.begin(), s.end());
00697         return sizes;
00698     }
00699
00700     static size_t getSizeVal(const type& val) {
00701         return compute_total_size(getDimensions(val));
00702     }
00703
00704     static size_t getSize(const std::vector<size_t>& dims) {
00705         return compute_total_size(dims);
00706     }
00707
00708     static void prepare(type& val, const std::vector<size_t>& dims) {
00709         if (dims.size() < ndim) {
00710             std::ostringstream os;
00711             os << "Only '" << dims.size() << "' given but boost::multi_array is of size '" << ndim
00712                << "'.";
00713             throw DataSpaceException(os.str());
00714         }
00715         boost::array<typename type::index, Dims> ext;
00716         std::copy(dims.begin(), dims.begin() + ndim, ext.begin());
00717         val.resize(ext);
00718         std::vector<size_t> next_dims(dims.begin() + Dims, dims.end());
00719         std::size_t size = std::accumulate(dims.begin(),
00720                                           dims.begin() + Dims,
00721                                           std::size_t{1},
00722                                           std::multiplies<size_t>());
00723         for (size_t i = 0; i < size; ++i) {
00724             inspector<value_type>::prepare(*(val.origin() + i), next_dims);
00725         }
00726     }
00727
00728     static hdf5_type* data(type& val) {
00729         return inspector<value_type>::data(*val.data());
00730     }
00731
00732     static const hdf5_type* data(const type& val) {
00733         return inspector<value_type>::data(*val.data());
00734     }
00735
00736     static void serialize(const type& val, hdf5_type* m) {
00737         size_t size = val.num_elements();
00738         size_t subsize = inspector<value_type>::getSizeVal(*val.origin());
00739         for (size_t i = 0; i < size; ++i) {
00740             inspector<value_type>::serialize(*(val.origin() + i), m + i * subsize);
00741         }
00742     }
00743
00744     static void unserialize(const hdf5_type* vec_align,
00745                             const std::vector<size_t>& dims,
00746                             type& val) {
00747         std::vector<size_t> next_dims(dims.begin() + ndim, dims.end());
00748         size_t subsize = compute_total_size(next_dims);
00749         for (size_t i = 0; i < val.num_elements(); ++i) {
00750             inspector<value_type>::unserialize(vec_align + i * subsize,
00751                                                next_dims,
00752                                                *(val.origin() + i));
00753         }
00754     }
00755 };
00756
00757 template <typename T>
00758 struct inspector<boost::numeric::ublas::matrix<T>> {
00759     using type = boost::numeric::ublas::matrix<T>;
00760     using value_type = unqualified_t<T>;
00761     using base_type = typename inspector<value_type>::base_type;
00762     using hdf5_type = typename inspector<value_type>::hdf5_type;
00763
00764     static constexpr size_t ndim = 2;
00765     static constexpr size_t recursive_ndim = ndim + inspector<value_type>::recursive_ndim;
00766     static constexpr bool is_trivially_copyable = std::is_trivially_copyable<value_type>::value &&
00767                                                    inspector<value_type>::is_trivially_copyable;
00768
00769     static std::vector<size_t> getDimensions(const type& val) {
00770         std::vector<size_t> sizes{val.size1(), val.size2()};
00771         auto s = inspector<value_type>::getDimensions(val(0, 0));
00772         sizes.insert(sizes.end(), s.begin(), s.end());
00773         return sizes;
00774     }
00775
00776     static size_t getSizeVal(const type& val) {
00777         return compute_total_size(getDimensions(val));
00778     }

```

```

00779
00780     static size_t getSize(const std::vector<size_t>& dims) {
00781         return compute_total_size(dims);
00782     }
00783
00784     static void prepare(type& val, const std::vector<size_t>& dims) {
00785         if (dims.size() < ndim) {
00786             std::ostringstream os;
00787             os << "Impossible to pair DataSet with " << dims.size() << " dimensions into a " << ndim
00788                 << " boost::numeric::ublas::matrix";
00789             throw DataSpaceException(os.str());
00790         }
00791         val.resize(dims[0], dims[1], false);
00792     }
00793
00794     static hdf5_type* data(type& val) {
00795         return inspector<value_type>::data(val(0, 0));
00796     }
00797
00798     static const hdf5_type* data(const type& val) {
00799         return inspector<value_type>::data(val(0, 0));
00800     }
00801
00802     static void serialize(const type& val, hdf5_type* m) {
00803         size_t size = val.size1() * val.size2();
00804         size_t subsize = inspector<value_type>::getSizeVal(val(0, 0));
00805         for (size_t i = 0; i < size; ++i) {
00806             inspector<value_type>::serialize(&val(0, 0) + i, m + i * subsize);
00807         }
00808     }
00809
00810     static void unserialize(const hdf5_type* vec_align,
00811                             const std::vector<size_t>& dims,
00812                             type& val) {
00813         std::vector<size_t> next_dims(dims.begin() + ndim, dims.end());
00814         size_t subsize = compute_total_size(next_dims);
00815         size_t size = val.size1() * val.size2();
00816         for (size_t i = 0; i < size; ++i) {
00817             inspector<value_type>::unserialize(vec_align + i * subsize,
00818                                                next_dims,
00819                                                &val(0, 0) + i);
00820         }
00821     }
00822 };
00823 #endif
00824
00825 template <typename T>
00826 struct Writer {
00827     using hdf5_type = typename inspector<T>::hdf5_type;
00828     const hdf5_type* get_pointer() {
00829         if (vec.empty()) {
00830             return ptr;
00831         } else {
00832             return vec.data();
00833         }
00834     }
00835     std::vector<hdf5_type> vec{};
00836     const hdf5_type* ptr{nullptr};
00837 };
00838
00839 template <typename T>
00840 struct Reader {
00841     using type = unqualified_t<T>;
00842     using hdf5_type = typename inspector<type>::hdf5_type;
00843
00844     Reader(const std::vector<size_t>& _dims, type& _val)
00845         : dims(_dims)
00846         , val(_val) {}
00847
00848     hdf5_type* get_pointer() {
00849         if (vec.empty()) {
00850             return inspector<type>::data(val);
00851         } else {
00852             return vec.data();
00853         }
00854     }
00855
00856     void unserialize() {
00857         if (!vec.empty()) {
00858             inspector<type>::unserialize(vec.data(), dims, val);
00859         }
00860     }
00861
00862     std::vector<size_t> dims{};
00863     std::vector<hdf5_type> vec{};
00864     type& val{};
00865 };

```

```

00866
00867 struct data_converter {
00868     template <typename T>
00869     static typename std::enable_if<inspector<T>::is_trivially_copyable, Writer<T>::type serialize(
00870         const typename inspector<T>::type& val) {
00871         Writer<T> w;
00872         w.ptr = inspector<T>::data(val);
00873         return w;
00874     }
00875
00876     template <typename T>
00877     static typename std::enable_if<!inspector<T>::is_trivially_copyable, Writer<T>::type serialize(
00878         const typename inspector<T>::type& val) {
00879         Writer<T> w;
00880         w.vec.resize(inspector<T>::getSizeVal(val));
00881         inspector<T>::serialize(val, w.vec.data());
00882         return w;
00883     }
00884
00885     template <typename T>
00886     static
00887         typename std::enable_if<inspector<unqualified_t<T>::is_trivially_copyable, Reader<T>::type
00888         get_reader(const std::vector<size_t>& dims, T& val) {
00889             auto effective_dims = details::squeezeDimensions(dims, inspector<T>::recursive_ndim);
00890             Reader<T> r(effective_dims, val);
00891             inspector<T>::prepare(r.val, effective_dims);
00892             return r;
00893         }
00894
00895     template <typename T>
00896     static typename std::enable_if<!inspector<unqualified_t<T>::is_trivially_copyable,
00897         Reader<T>::type
00898         get_reader(const std::vector<size_t>& dims, T& val) {
00899         auto effective_dims = details::squeezeDimensions(dims, inspector<T>::recursive_ndim);
00900
00901         Reader<T> r(effective_dims, val);
00902         inspector<T>::prepare(r.val, effective_dims);
00903         r.vec.resize(inspector<T>::getSize(effective_dims));
00904         return r;
00905     }
00906 };
00907
00908 } // namespace details
00909 } // namespace HighFive

```

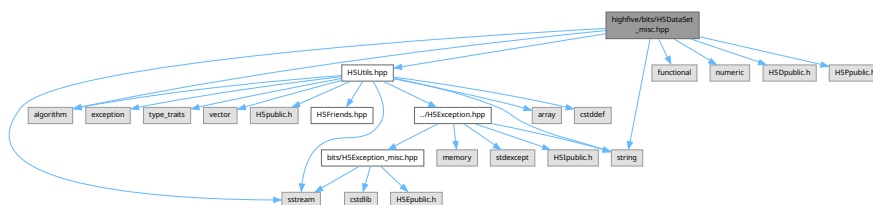
10.12 highfive/bits/H5DataSet_misc.hpp File Reference

```

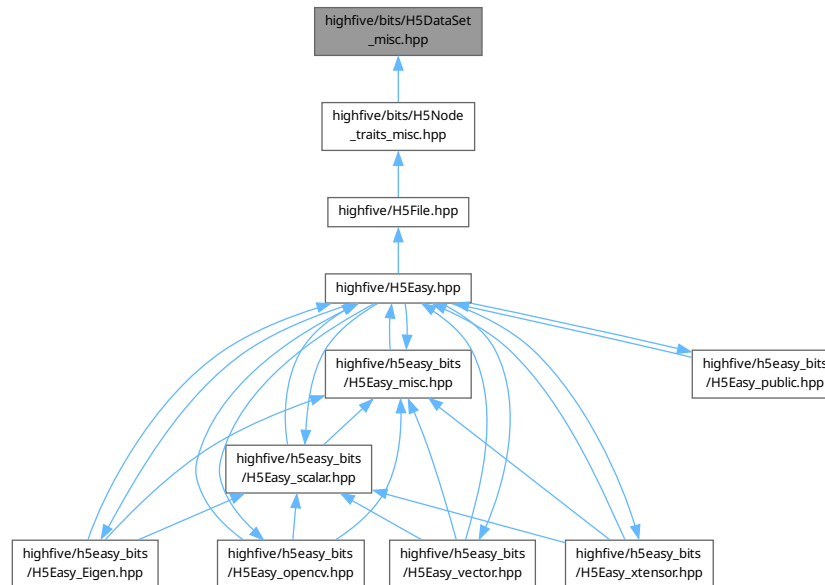
#include <algorithm>
#include <functional>
#include <numeric>
#include <sstream>
#include <string>
#include <H5Dpublic.h>
#include <H5Ppublic.h>
#include "H5Utils.hpp"

```

Include dependency graph for H5DataSet_misc.hpp:



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



Namespaces

- namespace [HighFive](#)

10.13 H5DataSet_misc.hpp

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c), 2017, Adrien Devresse <adrien.devresse@epfl.ch>
00003  *
00004  * Distributed under the Boost Software License, Version 1.0.
00005  * (See accompanying file LICENSE_1_0.txt or copy at
00006  * http://www.boost.org/LICENSE_1_0.txt)
00007  *
00008  */
00009 #pragma once
00010
00011 #include <algorithm>
00012 #include <functional>
00013 #include <numeric>
00014 #include <sstream>
00015 #include <string>
00016
00017 #include <H5Dpublic.h>
00018 #include <H5Ppublic.h>
00019
00020 #include "H5Utils.hpp"
00021
00022 namespace HighFive {
00023
00024 inline uint64_t DataSet::getStorageSize() const {
00025     return H5Dget_storage_size(_hid);
00026 }
00027
00028 inline DataType DataSet::getDataType() const {
00029     return DataType(H5Dget_type(_hid));
00030 }
00031
00032 inline DataSpace DataSet::getSpace() const {
00033     DataSpace space;

```

```

00034     if ((space._hid = H5Dget_space(_hid)) < 0) {
00035         HDF5ErrMapper::ToException<DataSetException>("Unable to get DataSpace out of DataSet");
00036     }
00037     return space;
00038 }
00039
00040 inline DataSpace DataSet::getMemSpace() const {
00041     return getSpace();
00042 }
00043
00044 inline uint64_t DataSet::getOffset() const {
00045     uint64_t addr = H5Dget_offset(_hid);
00046     if (addr == HADDR_UNDEF) {
00047         HDF5ErrMapper::ToException<DataSetException>("Cannot get offset of DataSet.");
00048     }
00049     return addr;
00050 }
00051
00052 inline void DataSet::resize(const std::vector<size_t>& dims) {
00053     const size_t numDimensions = getSpace().getDimensions().size();
00054     if (dims.size() != numDimensions) {
00055         HDF5ErrMapper::ToException<DataSetException>("Invalid dataspace dimensions, got " +
00056             std::to_string(dims.size()) + " expected " +
00057             std::to_string(numDimensions));
00058     }
00059     std::vector<hsize_t> real_dims(dims.begin(), dims.end());
00060
00061     if (H5Dset_extent(getId(), real_dims.data()) < 0) {
00062         HDF5ErrMapper::ToException<DataSetException>("Could not resize dataset.");
00063     }
00064 }
00065 }
00066
00067 } // namespace HighFive

```

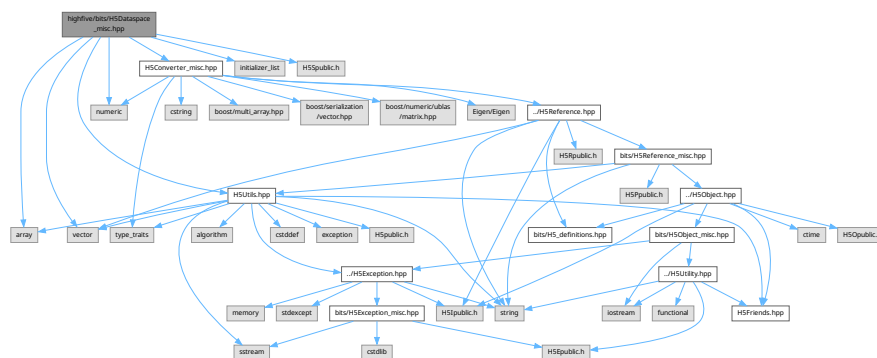
10.14 highfive/bits/H5Dataspace_misc.hpp File Reference

```

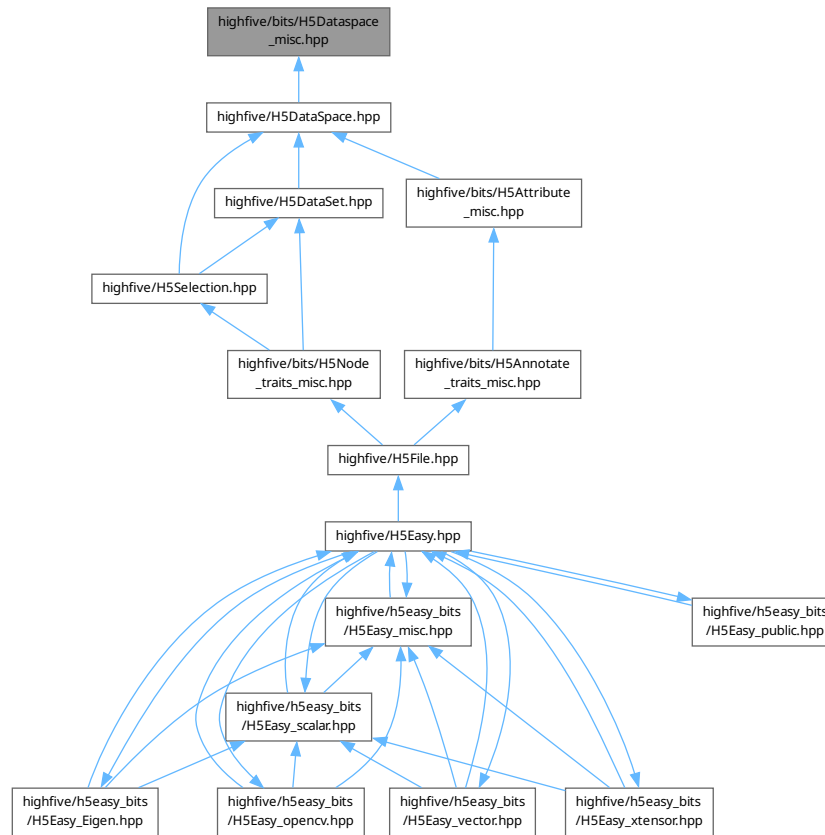
#include <array>
#include <initializer_list>
#include <vector>
#include <numeric>
#include <H5Spublic.h>
#include "H5Utils.hpp"
#include "H5Converter_misc.hpp"

```

Include dependency graph for H5Dataspace_misc.hpp:



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



Namespaces

- namespace [HighFive](#)

10.15 H5Dataspace_misc.hpp

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c), 2017, Adrien Devresse <adrien.devresse@epfl.ch>
00003  *
00004  * Distributed under the Boost Software License, Version 1.0.
00005  * (See accompanying file LICENSE_1_0.txt or copy at
00006  * http://www.boost.org/LICENSE_1_0.txt)
00007  *
00008  */
00009 #pragma once
00010
00011 #include <array>
00012 #include <initializer_list>
00013 #include <vector>
00014 #include <numeric>
00015
00016 #include <H5Spublic.h>
00017
00018 #include "H5Utils.hpp"
00019 #include "H5Converter_misc.hpp"
00020
00021 namespace HighFive {

```

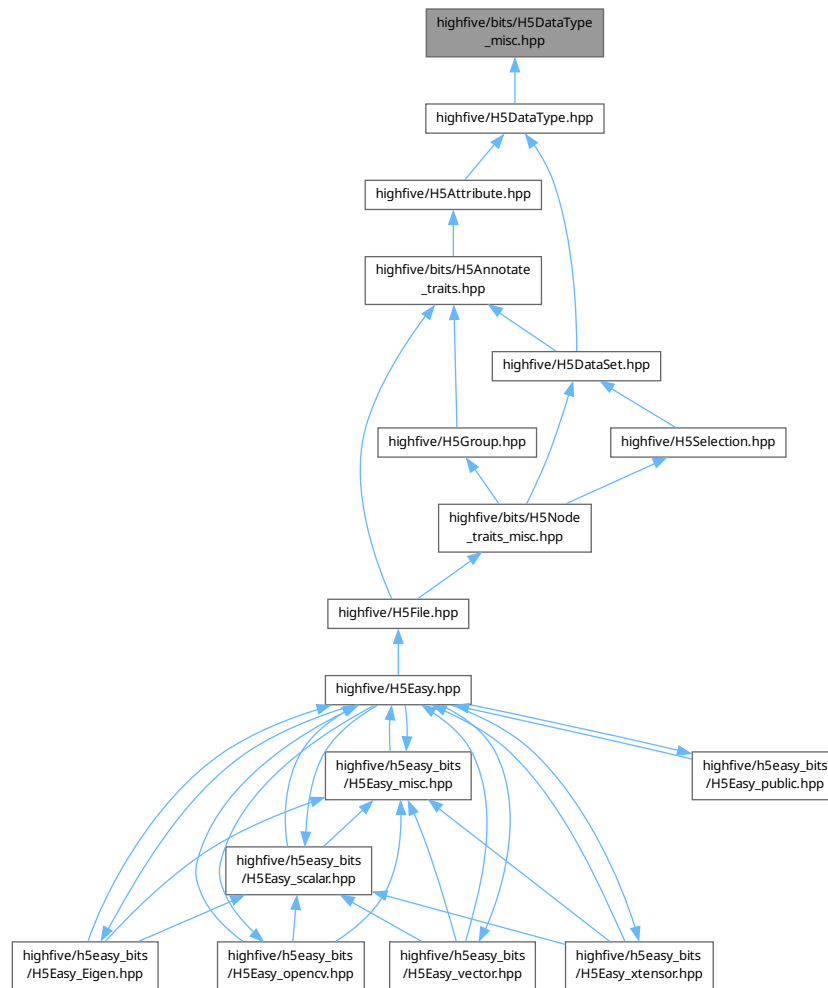
```

00022
00023 inline DataSpace::DataSpace(const std::vector<size_t>& dims)
00024 : DataSpace(dims.begin(), dims.end()) {}
00025
00026 template <size_t N>
00027 inline DataSpace::DataSpace(const std::array<size_t, N>& dims)
00028 : DataSpace(dims.begin(), dims.end()) {}
00029
00030 inline DataSpace::DataSpace(const std::initializer_list<size_t>& items)
00031 : DataSpace(std::vector<size_t>(items)) {}
00032
00033 template <typename... Args>
00034 inline DataSpace::DataSpace(size_t dim1, Args... dims)
00035 : DataSpace(std::vector<size_t>(dim1, static_cast<size_t>(dims)...)) {}
00036
00037 template <class IT, typename>
00038 inline DataSpace::DataSpace(const IT begin, const IT end) {
00039     std::vector<hsize_t> real_dims(begin, end);
00040
00041     if ((_hid = H5Screate_simple(int(real_dims.size()), real_dims.data(), NULL)) < 0) {
00042         throw DataSpaceException("Impossible to create dataspace");
00043     }
00044 }
00045
00046 inline DataSpace::DataSpace(const std::vector<size_t>& dims, const std::vector<size_t>& maxdims) {
00047     if (dims.size() != maxdims.size()) {
00048         throw DataSpaceException("dims and maxdims must be the same length.");
00049     }
00050
00051     std::vector<hsize_t> real_dims(dims.begin(), dims.end());
00052     std::vector<hsize_t> real_maxdims(maxdims.begin(), maxdims.end());
00053
00054     // Replace unlimited flag with actual HDF one
00055     std::replace(real_maxdims.begin(),
00056                 real_maxdims.end(),
00057                 static_cast<hsize_t>(DataSpace::UNLIMITED),
00058                 H5S_UNLIMITED);
00059
00060     if ((_hid = H5Screate_simple(int(dims.size()), real_dims.data(), real_maxdims.data())) < 0) {
00061         throw DataSpaceException("Impossible to create dataspace");
00062     }
00063 } // namespace HighFive
00064
00065 inline DataSpace::DataSpace(DataSpace::DataspaceType dtype) {
00066     H5S_class_t h5_dataspace_type;
00067     switch (dtype) {
00068     case DataSpace::dataspace_scalar:
00069         h5_dataspace_type = H5S_SCALAR;
00070         break;
00071     case DataSpace::dataspace_null:
00072         h5_dataspace_type = H5S_NULL;
00073         break;
00074     default:
00075         throw DataSpaceException(
00076             "Invalid dataspace type: should be "
00077             "dataspace_scalar or dataspace_null");
00078     }
00079
00080     if ((_hid = H5Screate(h5_dataspace_type)) < 0) {
00081         throw DataSpaceException("Unable to create dataspace");
00082     }
00083 }
00084
00085 inline DataSpace DataSpace::clone() const {
00086     DataSpace res;
00087     if ((res._hid = H5Scopy(_hid)) < 0) {
00088         throw DataSpaceException("Unable to copy dataspace");
00089     }
00090     return res;
00091 }
00092
00093 inline size_t DataSpace::getNumberDimensions() const {
00094     const int ndim = H5Sget_simple_extent_ndims(_hid);
00095     if (ndim < 0) {
00096         HDF5ErrMapper::ToException<DataSetException>(
00097             "Unable to get dataspace number of dimensions");
00098     }
00099     return size_t(ndim);
00100 }
00101
00102 inline std::vector<size_t> DataSpace::getDimensions() const {
00103     std::vector<hsize_t> dims(getNumberDimensions());
00104     if (!dims.empty()) {
00105         if (H5Sget_simple_extent_dims(_hid, dims.data(), NULL) < 0) {
00106             HDF5ErrMapper::ToException<DataSetException>("Unable to get dataspace dimensions");
00107         }
00108     }

```

10.16 highfive/bits/H5DataType_misc.hpp File Reference

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



Classes

- class [HighFive::AtomicType< char\[StrLen\]>](#)
- class [HighFive::AtomicType< FixedLenStringArray< StrLen > >](#)
- class [HighFive::AtomicType< std::complex< T > >](#)

Namespaces

- namespace [HighFive](#)

Macros

- `#define \_H5\_STRUCT\_PADDING(current_size, member_size)`

Typedefs

- using [HighFive::float16_t](#) = `half_float::half`

Functions

- [EnumType< details::Boolean > HighFive::create_enum_boolean \(\)](#)
- [size_t HighFive::find_first_atomic_member_size \(hid_t hid\)](#)
- [template<typename T > DataType HighFive::create_datatype \(\)](#)
Create a [DataType](#) instance representing type *T*.
- [template<typename T > DataType HighFive::create_and_check_datatype \(\)](#)
Create a [DataType](#) instance representing type *T* and perform a sanity check on its size.

10.16.1 Macro Definition Documentation

10.16.1.1 _H5_STRUCT_PADDING

```
#define _H5_STRUCT_PADDING(
    current_size,
    member_size )
```

Value:

```
((member_size) >= (current_size))
? (((member_size) - (current_size)) % (member_size))
: (((member_size) - (((current_size) - (member_size)) % (member_size))) % \
(member_size)))
```

10.17 H5DataType_misc.hpp

[Go to the documentation of this file.](#)

```
00001 /*
00002  * Copyright (c), 2017, Adrien Devresse <adrien.devresse@epfl.ch>
00003  *
00004  * Distributed under the Boost Software License, Version 1.0.
00005  * (See accompanying file LICENSE_1_0.txt or copy at
00006  * http://www.boost.org/LICENSE_1_0.txt)
00007  *
00008  */
00009 #pragma once
00010
00011 #include <string>
00012 #include <complex>
00013 #include <cstring>
00014 #if HIGHFIVE_CXX_STD >= 17
00015 #include <cstdint>
00016 #endif
00017
00018 #include <H5Ppublic.h>
00019 #include <H5Tpublic.h>
00020
00021 #ifdef H5_USE_HALF_FLOAT
00022 #include <half.hpp>
00023 #endif
00024
00025 #include "H5Converter_misc.hpp"
00026
00027 namespace HighFive {
00028     namespace { // unnamed
00029         inline DataTypeClass convert_type_class(const H5T_class_t& tclass);
00030         inline std::string type_class_string(DataTypeClass);
00031         inline hid_t create_string(std::size_t length);
00032     } // namespace
00033
00034     inline bool DataType::empty() const noexcept {
00035         return _hid == H5I_INVALID_HID;
00036     }
00037
00038     inline DataTypeClass DataType::getClass() const {
```

```

00040     return convert_type_class(H5Tget_class(_hid));
00041 }
00042
00043 inline size_t DataType::getSize() const {
00044     return H5Tget_size(_hid);
00045 }
00046
00047 inline bool DataType::operator==(const DataType& other) const {
00048     return H5Tequal(_hid, other._hid) > 0;
00049 }
00050
00051 inline bool DataType::operator!=(const DataType& other) const {
00052     return !(*this == other);
00053 }
00054
00055 inline bool DataType::isVariableStr() const {
00056     auto var_value = H5Tis_variable_str(_hid);
00057     if (var_value < 0) {
00058         HDF5ErrMapper::ToException<DataTypeException>("Unable to define datatype size to variable");
00059     }
00060     return static_cast<bool>(var_value);
00061 }
00062
00063 inline bool DataType::isFixedLenStr() const {
00064     return getClass() == DataTypeClass::String && !isVariableStr();
00065 }
00066
00067 inline bool DataType::isReference() const {
00068     return H5Tequal(_hid, H5T_STD_REF_OBJ) > 0;
00069 }
00070
00071 inline std::string DataType::string() const {
00072     return type_class_string(getClass()) + std::to_string(getSize() * 8);
00073 }
00074
00075 // char mapping
00076 template <>
00077 inline AtomicType<char>::AtomicType() {
00078     _hid = H5Tcopy(H5T_NATIVE_CHAR);
00079 }
00080
00081 template <>
00082 inline AtomicType<signed char>::AtomicType() {
00083     _hid = H5Tcopy(H5T_NATIVE_SCHAR);
00084 }
00085
00086 template <>
00087 inline AtomicType<unsigned char>::AtomicType() {
00088     _hid = H5Tcopy(H5T_NATIVE_UCHAR);
00089 }
00090
00091 // short mapping
00092 template <>
00093 inline AtomicType<short>::AtomicType() {
00094     _hid = H5Tcopy(H5T_NATIVE_SHORT);
00095 }
00096
00097 template <>
00098 inline AtomicType<unsigned short>::AtomicType() {
00099     _hid = H5Tcopy(H5T_NATIVE_USHORT);
00100 }
00101
00102 // integer mapping
00103 template <>
00104 inline AtomicType<int>::AtomicType() {
00105     _hid = H5Tcopy(H5T_NATIVE_INT);
00106 }
00107
00108 template <>
00109 inline AtomicType<unsigned>::AtomicType() {
00110     _hid = H5Tcopy(H5T_NATIVE_UINT);
00111 }
00112
00113 // long mapping
00114 template <>
00115 inline AtomicType<long>::AtomicType() {
00116     _hid = H5Tcopy(H5T_NATIVE_LONG);
00117 }
00118
00119 template <>
00120 inline AtomicType<unsigned long>::AtomicType() {
00121     _hid = H5Tcopy(H5T_NATIVE_ULONG);
00122 }
00123
00124 // long long mapping
00125 template <>
00126 inline AtomicType<long long>::AtomicType() {

```

```

00127     _hid = H5Tcopy(H5T_NATIVE_LLONG);
00128 }
00129
00130 template <>
00131 inline AtomicType<unsigned long long>::AtomicType() {
00132     _hid = H5Tcopy(H5T_NATIVE_ULLONG);
00133 }
00134
00135 // half-float, float, double and long double mapping
00136 #ifdef H5_USE_HALF_FLOAT
00137 using float16_t = half_float::half;
00138
00139 template <>
00140 inline AtomicType<float16_t>::AtomicType() {
00141     _hid = H5Tcopy(H5T_NATIVE_FLOAT);
00142     // Sign position, exponent position, exponent size, mantissa position, mantissa size
00143     H5Tset_fields(_hid, 15, 10, 5, 0, 10);
00144     // Total datatype size (in bytes)
00145     H5Tset_size(_hid, 2);
00146     // Floating point exponent bias
00147     H5Tset_ebias(_hid, 15);
00148 }
00149 #endif
00150
00151 template <>
00152 inline AtomicType<float>::AtomicType() {
00153     _hid = H5Tcopy(H5T_NATIVE_FLOAT);
00154 }
00155
00156 template <>
00157 inline AtomicType<double>::AtomicType() {
00158     _hid = H5Tcopy(H5T_NATIVE_DOUBLE);
00159 }
00160
00161 template <>
00162 inline AtomicType<long double>::AtomicType() {
00163     _hid = H5Tcopy(H5T_NATIVE_LDOUBLE);
00164 }
00165
00166 // std string
00167 template <>
00168 inline AtomicType<std::string>::AtomicType() {
00169     _hid = create_string(H5T_VARIABLE);
00170 }
00171
00172 #if HIGHFIVE_CXX_STD >= 17
00173 // std byte
00174 template <>
00175 inline AtomicType<std::byte>::AtomicType() {
00176     _hid = H5Tcopy(H5T_NATIVE_B8);
00177 }
00178 #endif
00179
00180 // Fixed-Length strings
00181 // require class specialization templated for the char length
00182 template <size_t StrLen>
00183 class AtomicType<char[StrLen]>: public DataType {
00184 public:
00185     inline AtomicType()
00186         : DataType(create_string(StrLen)) {}
00187 };
00188
00189 template <size_t StrLen>
00190 class AtomicType<FixedLenStringArray<StrLen>>: public DataType {
00191 public:
00192     inline AtomicType()
00193         : DataType(create_string(StrLen)) {}
00194 };
00195
00196 template <typename T>
00197 class AtomicType<std::complex<T>>: public DataType {
00198 public:
00199     inline AtomicType()
00200         : DataType(
00201             CompoundType({{"r", create_datatype<T>(), 0}, {"i", create_datatype<T>(), sizeof(T)}},
00202                 sizeof(std::complex<T>)) {
00203         static_assert(std::is_floating_point<T>::value,
00204             "std::complex accepts only floating point numbers.");
00205     }
00206 };
00207
00208 // For boolean we act as h5py
00209 inline EnumType<details::Boolean> create_enum_boolean() {
00210     return {"FALSE", details::Boolean::HighFiveFalse}, {"TRUE", details::Boolean::HighFiveTrue};
00211 }
00212
00213 // Other cases not supported. Fail early with a user message

```

```

00214 template <typename T>
00215 AtomicType<T>::AtomicType() {
00216     static_assert(details::inspector<T>::recursive_ndim == 0,
00217         "Atomic types cant be arrays, except for char[] (fixed-length strings)");
00218     static_assert(details::inspector<T>::recursive_ndim > 0, "Type not supported");
00219 }
00220
00221
00222 // class FixedLenStringArray<N>
00223
00224 template <std::size_t N>
00225 inline FixedLenStringArray<N>::FixedLenStringArray(const char array[][N], std::size_t length) {
00226     datavec.resize(length);
00227     std::memcpy(datavec[0].data(), array[0].data(), N * length);
00228 }
00229
00230 template <std::size_t N>
00231 inline FixedLenStringArray<N>::FixedLenStringArray(const std::string* iter_begin,
00232     const std::string* iter_end) {
00233     datavec.resize(static_cast<std::size_t>(iter_end - iter_begin));
00234     for (auto& dst_array: datavec) {
00235         const char* src = (iter_begin++)->c_str();
00236         const size_t length = std::min(N - 1, std::strlen(src));
00237         std::memcpy(dst_array.data(), src, length);
00238         dst_array[length] = 0;
00239     }
00240 }
00241
00242 template <std::size_t N>
00243 inline FixedLenStringArray<N>::FixedLenStringArray(const std::vector<std::string>& vec)
00244     : FixedLenStringArray(&vec.front(), &vec.back()) {}
00245
00246 template <std::size_t N>
00247 inline FixedLenStringArray<N>::FixedLenStringArray(
00248     const std::initializer_list<std::string>& init_list)
00249     : FixedLenStringArray(init_list.begin(), init_list.end()) {}
00250
00251 template <std::size_t N>
00252 inline void FixedLenStringArray<N>::push_back(const std::string& src) {
00253     datavec.emplace_back();
00254     const size_t length = std::min(N - 1, src.length());
00255     std::memcpy(datavec.back().data(), src.c_str(), length);
00256     datavec.back()[length] = 0;
00257 }
00258
00259 template <std::size_t N>
00260 inline void FixedLenStringArray<N>::push_back(const std::array<char, N>& src) {
00261     datavec.emplace_back();
00262     std::copy(src.begin(), src.end(), datavec.back().data());
00263 }
00264
00265 template <std::size_t N>
00266 inline std::string FixedLenStringArray<N>::getString(std::size_t i) const {
00267     return std::string(datavec[i].data());
00268 }
00269
00270 // Internal
00271 // Reference mapping
00272 template <>
00273 inline AtomicType<Reference>::AtomicType() {
00274     _hid = H5Tcopy(H5T_STD_REF_OBJ);
00275 }
00276
00277 inline size_t find_first_atomic_member_size(hid_t hid) {
00278     // Recursive exit condition
00279     if (H5Tget_class(hid) == H5T_COMPOUND) {
00280         auto number_of_members = H5Tget_nmembers(hid);
00281         if (number_of_members == -1) {
00282             throw DataTypeException("Cannot get members of CompoundType with hid: " +
00283                 std::to_string(hid));
00284         }
00285         if (number_of_members == 0) {
00286             throw DataTypeException("No members defined for CompoundType with hid: " +
00287                 std::to_string(hid));
00288         }
00289
00290         auto member_type = H5Tget_member_type(hid, 0);
00291         auto size = find_first_atomic_member_size(member_type);
00292         H5Tclose(member_type);
00293         return size;
00294     } else if (H5Tget_class(hid) == H5T_STRING) {
00295         return 1;
00296     }
00297     return H5Tget_size(hid);
00298 }
00299
00300 // Calculate the padding required to align an element of a struct

```



```

00301 // For padding see explanation here: https://en.cppreference.com/w/cpp/language/object#Alignment
00302 // It is to compute padding following last element inserted inside a struct
00303 // 1) We want to push back an element padded to the structure
00304 // 'current_size' is the size of the structure before adding the new element.
00305 // 'member_size' the size of the element we want to add.
00306 // 2) We want to compute the final padding for the global structure
00307 // 'current_size' is the size of the whole structure without final padding
00308 // 'member_size' is the maximum size of all element of the struct
00309 //
00310 // The basic formula is only to know how much we need to add to 'current_size' to fit
00311 // 'member_size'.
00312 // And at the end, we do another computation because the end padding, should fit the biggest
00313 // element of the struct.
00314 //
00315 // As we are with `size_t` element, we need to compute everything inside R+
00316 #define _H5_STRUCT_PADDING(current_size, member_size) \
00317     (((member_size) >= (current_size)) \
00318      ? ((member_size) - (current_size)) % (member_size) \
00319      : (((member_size) - ((current_size) - (member_size)) % (member_size))) % \
00320        (member_size)))
00321
00322 inline void CompoundType::create(size_t size) {
00323     if (size == 0) {
00324         size_t current_size = 0, max_atomic_size = 0;
00325
00326         // Do a first pass to find the total size of the compound datatype
00327         for (auto& member: members) {
00328             size_t member_size = H5Tget_size(member.base_type.getId());
00329
00330             if (member_size == 0) {
00331                 throw DataTypeException("Cannot get size of DataType with hid: " +
00332                                         std::to_string(member.base_type.getId()));
00333             }
00334
00335             size_t first_atomic_size = find_first_atomic_member_size(member.base_type.getId());
00336
00337             // Set the offset of this member within the struct according to the
00338             // standard alignment rules. The c++ standard specifies that:
00339             // > objects have an alignment requirement of which their size is a multiple
00340             member.offset = current_size + _H5_STRUCT_PADDING(current_size, first_atomic_size);
00341
00342             // Set the current size to the end of the new member
00343             current_size = member.offset + member_size;
00344
00345             // Keep track of the highest atomic member size because it's needed
00346             // for the padding of the complete compound type.
00347             max_atomic_size = std::max(max_atomic_size, first_atomic_size);
00348         }
00349
00350         size = current_size + _H5_STRUCT_PADDING(current_size, max_atomic_size);
00351     }
00352
00353     // Create the HDF5 type
00354     if ((_hid = H5Tcreate(H5T_COMPOUND, size)) < 0) {
00355         HDF5ErrMapper::ToException<DataTypeException>("Could not create new compound datatype");
00356     }
00357
00358     // Loop over all the members and insert them into the datatype
00359     for (const auto& member: members) {
00360         if (H5Tinsert(_hid, member.name.c_str(), member.offset, member.base_type.getId()) < 0) {
00361             HDF5ErrMapper::ToException<DataTypeException>("Could not add new member to datatype");
00362         }
00363     }
00364 }
00365
00366 #undef _H5_STRUCT_PADDING
00367
00368 inline void CompoundType::commit(const Object& object, const std::string& name) const {
00369     H5Tcommit2(object.getId(), name.c_str(), getId(), H5P_DEFAULT, H5P_DEFAULT, H5P_DEFAULT);
00370 }
00371
00372 template <typename T>
00373 inline void EnumType<T>::create() {
00374     // Create the HDF5 type
00375     if ((_hid = H5Tenum_create(AtomicType<typename std::underlying_type<T>::type>{}.getId())) < 0) {
00376         HDF5ErrMapper::ToException<DataTypeException>("Could not create new enum datatype");
00377     }
00378
00379     // Loop over all the members and insert them into the datatype
00380     for (const auto& member: members) {
00381         if (H5Tenum_insert(_hid, member.name.c_str(), &(member.value)) < 0) {
00382             HDF5ErrMapper::ToException<DataTypeException>(
00383                 "Could not add new member to this enum datatype");
00384         }
00385     }
00386 }
00387

```

```

00388 template <typename T>
00389 inline void EnumType<T>::commit(const Object& object, const std::string& name) const {
00390     H5Tcommit2(object.getId(), name.c_str(), getId(), H5P_DEFAULT, H5P_DEFAULT, H5P_DEFAULT);
00391 }
00392
00393 namespace {
00394
00395 inline hid_t create_string(size_t length) {
00396     hid_t _hid = H5Tcopy(H5T_C_S1);
00397     if (H5Tset_size(_hid, length) < 0) {
00398         HDF5ErrMapper::ToException<DataTypeException>("Unable to define datatype size to variable");
00399     }
00400     // define encoding to UTF-8 by default
00401     H5Tset_cset(_hid, H5T_CSET_UTF8);
00402     return _hid;
00403 }
00404
00405 inline DataTypeClass convert_type_class(const H5T_class_t& tclass) {
00406     switch (tclass) {
00407     case H5T_TIME:
00408         return DataTypeClass::Time;
00409     case H5T_INTEGER:
00410         return DataTypeClass::Integer;
00411     case H5T_FLOAT:
00412         return DataTypeClass::Float;
00413     case H5T_STRING:
00414         return DataTypeClass::String;
00415     case H5T_BITFIELD:
00416         return DataTypeClass::BitField;
00417     case H5T_OPAQUE:
00418         return DataTypeClass::Opaque;
00419     case H5T_COMPOUND:
00420         return DataTypeClass::Compound;
00421     case H5T_REFERENCE:
00422         return DataTypeClass::Reference;
00423     case H5T_ENUM:
00424         return DataTypeClass::Enum;
00425     case H5T_VLEN:
00426         return DataTypeClass::VarLen;
00427     case H5T_ARRAY:
00428         return DataTypeClass::Array;
00429     case H5T_NO_CLASS:
00430     case H5T_NCLASSES:
00431     default:
00432         return DataTypeClass::Invalid;
00433     }
00434 }
00435 }
00436
00437 inline std::string type_class_string(DataTypeClass tclass) {
00438     switch (tclass) {
00439     case DataTypeClass::Time:
00440         return "Time";
00441     case DataTypeClass::Integer:
00442         return "Integer";
00443     case DataTypeClass::Float:
00444         return "Float";
00445     case DataTypeClass::String:
00446         return "String";
00447     case DataTypeClass::BitField:
00448         return "BitField";
00449     case DataTypeClass::Opaque:
00450         return "Opaque";
00451     case DataTypeClass::Compound:
00452         return "Compound";
00453     case DataTypeClass::Reference:
00454         return "Reference";
00455     case DataTypeClass::Enum:
00456         return "Enum";
00457     case DataTypeClass::VarLen:
00458         return "Varlen";
00459     case DataTypeClass::Array:
00460         return "Array";
00461     default:
00462         return "(Invalid)";
00463     }
00464 }
00465 }
00466
00467 } // unnamed namespace
00468
00469 template <typename T>
00470 inline DataType create_datatype() {
00471     return AtomicType<T>();
00472 }
00473
00474 }
00475

```

```

00476
00477 template <typename T>
00478 inline DataType create_and_check_datatype() {
00479     DataType t = create_datatype<T>();
00480     if (t.empty()) {
00481         throw DataTypeException("Type given to create_and_check_datatype is not valid");
00482     }
00483     // Skip check if the base type is a variable length string
00484     if (t.isVariableStr()) {
00485         return t;
00486     }
00487     // Check that the size of the template type matches the size that HDF5 is
00488     // expecting.
00489     if (t.isReference() || t.isFixedLenStr()) {
00490         return t;
00491     }
00492     if (sizeof(T) != t.getSize()) {
00493         std::ostringstream ss;
00494         ss << "Size of array type " << sizeof(T) << " != that of memory datatype " << t.getSize()
00495         << std::endl;
00496         throw DataTypeException(ss.str());
00497     }
00498     return t;
00499 }
00500
00501 // namespace HighFive
00502 HIGHFIVE_REGISTER_TYPE(HighFive::details::Boolean, HighFive::create_enum_boolean)

```

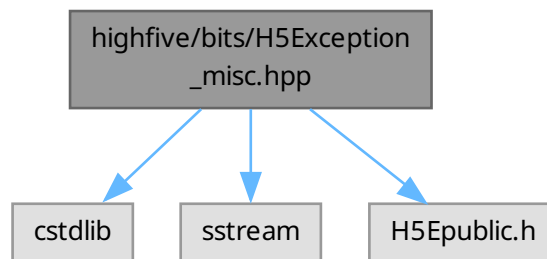
10.18 highfive/bits/H5Exception_misc.hpp File Reference

```

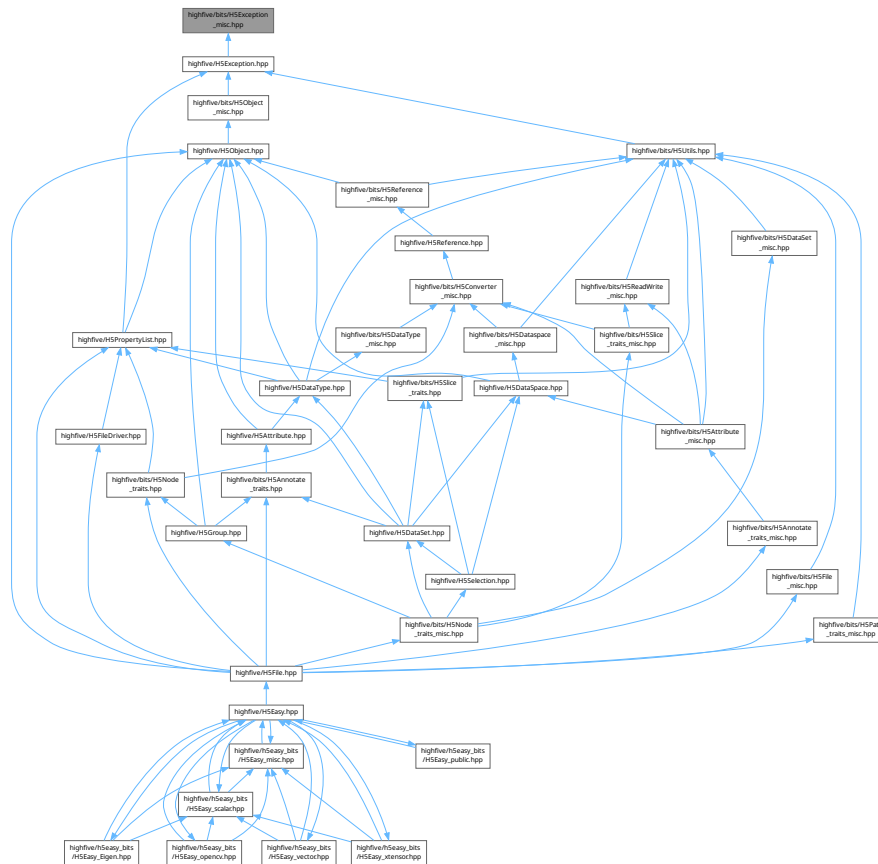
#include <cstdlib>
#include <sstream>
#include <H5Epublic.h>

```

Include dependency graph for H5Exception_misc.hpp:



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



Classes

- struct [HighFive::HDF5ErrMapper](#)

Namespaces

- namespace [HighFive](#)

10.19 H5Exception_misc.hpp

[Go to the documentation of this file.](#)

```

00001  /*
00002   * Copyright (c), 2017, Adrien Devresse <adrien.devresse@epfl.ch>
00003   *
00004   * Distributed under the Boost Software License, Version 1.0.
00005   * (See accompanying file LICENSE_1_0.txt or copy at
00006   * http://www.boost.org/LICENSE_1_0.txt)
00007   *
00008   */
00009  #pragma once
00010
00011  #include <cstdlib>
00012  #include <sstream>
00013
00014  #include <H5Epublic.h>
00015

```

```

00016 namespace HighFive {
00017
00018 struct HDF5ErrMapper {
00019     template <typename ExceptionType>
00020     static inline herr_t stackWalk(unsigned n, const H5E_error2_t* err_desc, void* client_data) {
00021         auto* e_iter = static_cast<ExceptionType**>(client_data);
00022         (void) n;
00023
00024         const char* major_err = H5Eget_major(err_desc->maj_num);
00025         const char* minor_err = H5Eget_minor(err_desc->min_num);
00026
00027         std::ostringstream oss;
00028         oss << '(' << major_err << ") " << minor_err;
00029
00030         H5free_memory((void*) major_err);
00031         H5free_memory((void*) minor_err);
00032
00033         auto* e = new ExceptionType(oss.str());
00034         e->_err_major = err_desc->maj_num;
00035         e->_err_minor = err_desc->min_num;
00036         (*e_iter)->_next.reset(e);
00037         *e_iter = e;
00038         return 0;
00039     }
00040
00041     template <typename ExceptionType>
00042     [[noreturn]] static inline void ToException(const std::string& prefix_msg) {
00043         hid_t err_stack = H5Eget_current_stack();
00044         if (err_stack >= 0) {
00045             ExceptionType e("");
00046             ExceptionType* e_iter = &e;
00047
00048             H5Ewalk2(err_stack, H5E_WALK_UPWARD, &HDF5ErrMapper::stackWalk<ExceptionType>, &e_iter);
00049             H5Eclear2(err_stack);
00050
00051             const char* next_err_msg = (e.nextException() != NULL) ? (e.nextException()->what())
00052                                                         : ("");
00053
00054             e.setErrorMsg(prefix_msg + " " + next_err_msg);
00055             throw e;
00056         }
00057         // throw generic error, unrecognized error
00058         throw ExceptionType(prefix_msg + ": Unknown HDF5 error");
00059     }
00060 };
00061
00062 } // namespace HighFive

```

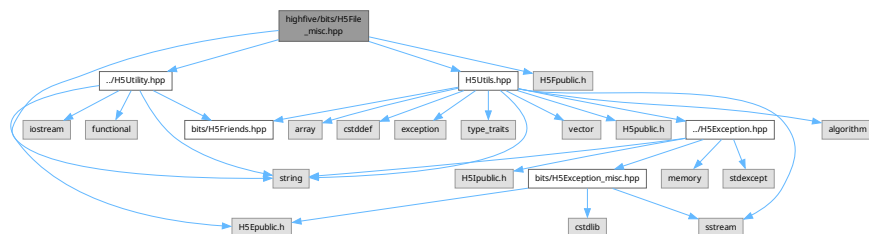
10.20 highfive/bits/H5File_misc.hpp File Reference

```

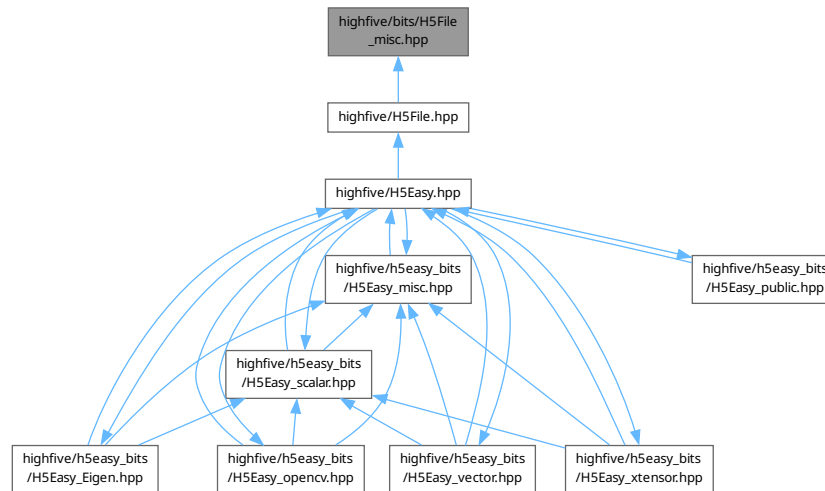
#include <string>
#include <H5Fpublic.h>
#include "../H5Utility.hpp"
#include "H5Utils.hpp"

```

Include dependency graph for H5File_misc.hpp:



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



Namespaces

- namespace [HighFive](#)

10.21 H5File_misc.hpp

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c), 2017, Adrien Devresse <adrien.devresse@epfl.ch>
00003  *
00004  * Distributed under the Boost Software License, Version 1.0.
00005  * (See accompanying file LICENSE_1_0.txt or copy at
00006  * http://www.boost.org/LICENSE_1_0.txt)
00007  *
00008  */
00009 #pragma once
00010
00011 #include <string>
00012
00013 #include <H5Fpublic.h>
00014
00015 #include "../H5Utility.hpp"
00016 #include "H5Utils.hpp"
00017
00018 namespace HighFive {
00019
00020 namespace { // unnamed
00021
00022 // libhdf5 uses a preprocessor trick on their oflags
00023 // we can not declare them constant without a mapper
00024 inline unsigned convert_open_flag(unsigned openFlags) {
00025     unsigned res_open = 0;
00026     if (openFlags & File::ReadOnly)
00027         res_open |= H5F_ACC_RDONLY;
00028     if (openFlags & File::ReadWrite)
00029         res_open |= H5F_ACC_RDWR;
00030     if (openFlags & File::Create)
00031         res_open |= H5F_ACC_CREAT;
00032     if (openFlags & File::Truncate)
00033         res_open |= H5F_ACC_TRUNC;
00034     if (openFlags & File::Excl)
00035         res_open |= H5F_ACC_EXCL;
00036     return res_open;
00037 }

```

```

00038 } // namespace
00039
00040 inline File::File(const std::string& filename,
00041                  unsigned openFlags,
00042                  const FileAccessProps& fileAccessProps)
00043 : File(filename, openFlags, FileCreateProps::Default(), fileAccessProps) {}
00044
00045
00046 inline File::File(const std::string& filename,
00047                  unsigned openFlags,
00048                  const FileCreateProps& fileCreateProps,
00049                  const FileAccessProps& fileAccessProps) {
00050     openFlags = convert_open_flag(openFlags);
00051
00052     unsigned createMode = openFlags & (H5F_ACC_TRUNC | H5F_ACC_EXCL);
00053     unsigned openMode = openFlags & (H5F_ACC_RDWR | H5F_ACC_RDONLY);
00054     bool mustCreate = createMode > 0;
00055     bool openOrCreate = (openFlags & H5F_ACC_CREAT) > 0;
00056
00057     // open is default. It's skipped only if flags require creation
00058     // If open fails it will try create() if H5F_ACC_CREAT is set
00059     if (!mustCreate) {
00060         // Silence open errors if create is allowed
00061         std::unique_ptr<SilenceHDF5> silencer;
00062         if (openOrCreate)
00063             silencer.reset(new SilenceHDF5());
00064
00065         _hid = H5Fopen(filename.c_str(), openMode, fileAccessProps.getId());
00066
00067         if (isValid())
00068             return; // Done
00069
00070         if (openOrCreate) {
00071             // Will attempt to create ensuring wont clobber any file
00072             createMode = H5F_ACC_EXCL;
00073         } else {
00074             HDF5ErrMapper::ToException<FileException>(
00075                 std::string("Unable to open file " + filename));
00076         }
00077     }
00078
00079     auto fcpl = fileCreateProps.getId();
00080     auto fapl = fileAccessProps.getId();
00081     if ((_hid = H5Fcreate(filename.c_str(), createMode, fcpl, fapl)) < 0) {
00082         HDF5ErrMapper::ToException<FileException>(std::string("Unable to create file " + filename));
00083     }
00084 }
00085
00086 inline const std::string& File::getName() const noexcept {
00087     if (_filename.empty()) {
00088         _filename = details::get_name(
00089             [this](char* buffer, size_t length) { return H5Fget_name(getId(), buffer, length); });
00090     }
00091     return _filename;
00092 }
00093
00094 inline hsize_t File::getMetadataBlockSize() const {
00095     auto fapl = getAccessPropertyList();
00096     return MetadataBlockSize(fapl).getSize();
00097 }
00098
00099 inline std::pair<H5F_libver_t, H5F_libver_t> File::getVersionBounds() const {
00100     auto fapl = getAccessPropertyList();
00101     auto fileVer = FileVersionBounds(fapl);
00102     return fileVer.getVersion();
00103 }
00104
00105 #if H5_VERSION_GE(1, 10, 1)
00106 inline H5F_fspace_strategy_t File::getFileSpaceStrategy() const {
00107     auto fcpl = getCreatePropertyList();
00108     FileSpaceStrategy spaceStrategy(fcpl);
00109     return spaceStrategy.getStrategy();
00110 }
00111
00112 inline hsize_t File::getFileSpacePageSize() const {
00113     auto fcpl = getCreatePropertyList();
00114
00115     if (getFileSpaceStrategy() != H5F_FSPACE_STRATEGY_PAGE) {
00116         HDF5ErrMapper::ToException<FileException>(
00117             std::string("Cannot obtain page size as paged allocation is not used."));
00118     }
00119
00120     return FileSpacePageSize(fcpl).getPageSize();
00121 }
00122 #endif
00123
00124 inline void File::flush() {

```

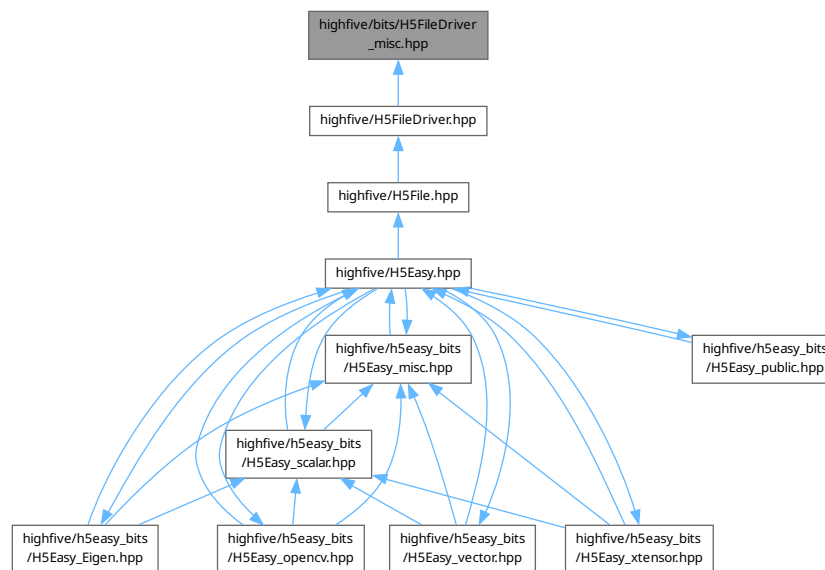
```

00125     if (H5Fflush(_hid, H5F_SCOPE_GLOBAL) < 0) {
00126         HDF5ErrMapper::ToException<FileException>(std::string("Unable to flush file " + getName()));
00127     }
00128 }
00129
00130 } // namespace HighFive

```

10.22 highfive/bits/H5FileDriver_misc.hpp File Reference

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



Namespaces

- namespace [HighFive](#)

10.23 H5FileDriver_misc.hpp

[Go to the documentation of this file.](#)

```

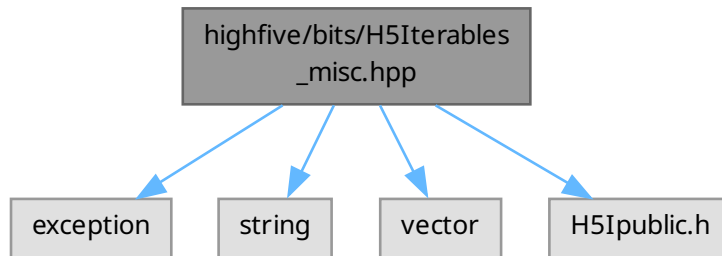
00001 /*
00002  * Copyright (c), 2017–2018, Adrien Devresse <adrien.devresse@epfl.ch>
00003  * Juan Hernando <juan.hernando@epfl.ch>
00004  *
00005  * Distributed under the Boost Software License, Version 1.0.
00006  * (See accompanying file LICENSE_1_0.txt or copy at
00007  * http://www.boost.org/LICENSE_1_0.txt)
00008  */
00009 #pragma once
00010
00011 namespace HighFive {
00012
00013 #ifdef H5_HAVE_PARALLEL
00014 inline MPIIOFileDriver::MPIIOFileDriver(MPI_Comm comm, MPI_Info info) {
00015     add(MPIIOFileAccess(comm, info));
00016 }
00017 #endif
00018
00019 } // namespace HighFive

```

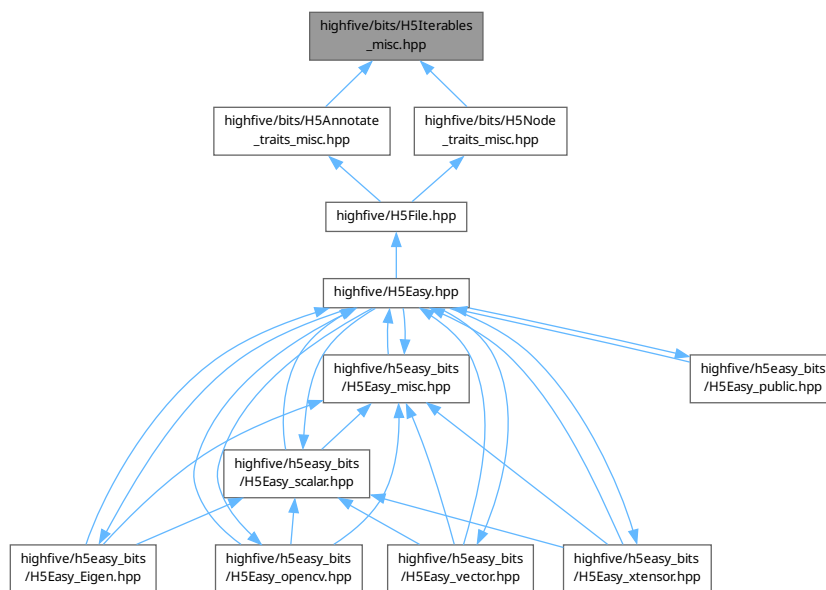


```
#include <H5Ipublic.h>
```

Include dependency graph for H5Iterables_misc.hpp:



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



Namespaces

- namespace `HighFive`

10.27 H5Iterables_misc.hpp

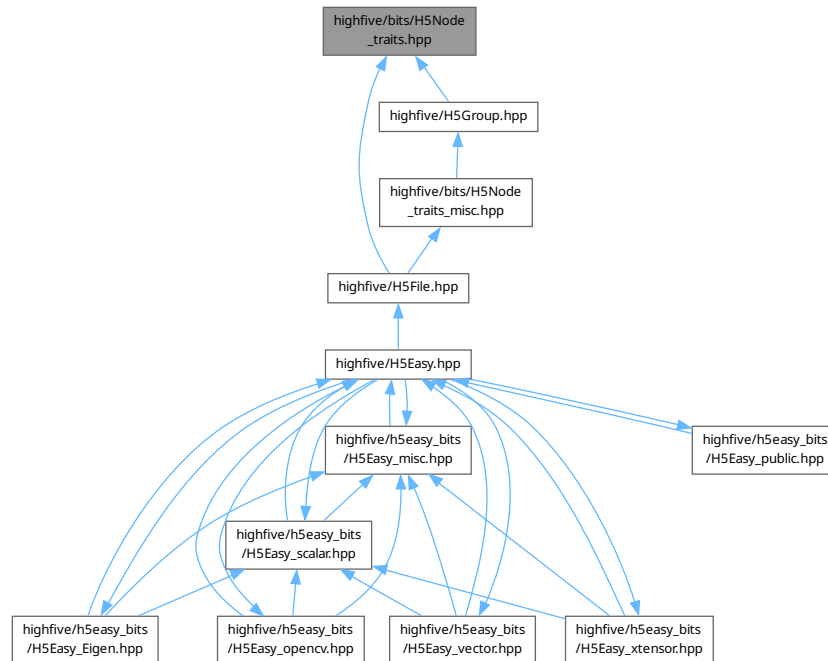
[Go to the documentation of this file.](#)

```
00001 /*
00002  * Copyright (c), 2017, Adrien Devresse <adrien.devresse@epfl.ch>
```

10.28 highfive/bits/H5Node_traits.hpp File Reference

[illegible]

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



Classes

- class [HighFive::NodeTraits< Derivate >](#)
NodeTraits: Base class for *Group* and *File*.

Namespaces

- namespace [HighFive](#)

Enumerations

- enum class [HighFive::IndexType](#) : std::underlying_type< H5_index_t >::type { [HighFive::NAME](#) = H5_INDEX_NAME , [HighFive::CRT_ORDER](#) = H5_INDEX_CRT_ORDER }
- enum class [HighFive::LinkType](#) { [HighFive::Hard](#) , [HighFive::Soft](#) , [HighFive::External](#) , [HighFive::Other](#) }
The possible types of group entries (link concept)

10.29 H5Node_traits.hpp

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c), 2017, Adrien Devresse <adrien.devresse@epfl.ch>
00003  *
00004  * Distributed under the Boost Software License, Version 1.0.
00005  * (See accompanying file LICENSE_1_0.txt or copy at
00006  * http://www.boost.org/LICENSE_1_0.txt)
00007  *

```

```

00008  */
00009  #pragma once
00010
00011  #include <string>
00012
00013  #include "../H5PropertyList.hpp"
00014  #include "H5_definitions.hpp"
00015  #include "H5Converter_misc.hpp"
00016
00017  namespace HighFive {
00018
00019  enum class IndexType : std::underlying_type<H5_index_t>::type {
00020      NAME = H5_INDEX_NAME,
00021      CRT_ORDER = H5_INDEX_CRT_ORDER,
00022  };
00023
00027  template <typename Derivate>
00028  class NodeTraits {
00029  public:
00040      DataSet createDataSet(const std::string& dataset_name,
00041                          const DataSpace& space,
00042                          const DataType& type,
00043                          const DataSetCreateProps& createProps = DataSetCreateProps::Default(),
00044                          const DataSetAccessProps& accessProps = DataSetAccessProps::Default(),
00045                          bool parents = true);
00046
00056      template <typename T,
00057              typename std::enable_if<
00058                  std::is_same<typename details::inspector<T>::base_type, details::Boolean>::value,
00059                  int>::type* = nullptr>
00060      DataSet createDataSet(const std::string& dataset_name,
00061                          const DataSpace& space,
00062                          const DataSetCreateProps& createProps = DataSetCreateProps::Default(),
00063                          const DataSetAccessProps& accessProps = DataSetAccessProps::Default(),
00064                          bool parents = true);
00065
00066      template <typename T,
00067              typename std::enable_if<
00068                  !std::is_same<typename details::inspector<T>::base_type, details::Boolean>::value,
00069                  int>::type* = nullptr>
00070      DataSet createDataSet(const std::string& dataset_name,
00071                          const DataSpace& space,
00072                          const DataSetCreateProps& createProps = DataSetCreateProps::Default(),
00073                          const DataSetAccessProps& accessProps = DataSetAccessProps::Default(),
00074                          bool parents = true);
00075
00086      template <typename T>
00087      DataSet createDataSet(const std::string& dataset_name,
00088                          const T& data,
00089                          const DataSetCreateProps& createProps = DataSetCreateProps::Default(),
00090                          const DataSetAccessProps& accessProps = DataSetAccessProps::Default(),
00091                          bool parents = true);
00092
00093
00094      template <std::size_t N>
00095      DataSet createDataSet(const std::string& dataset_name,
00096                          const FixedLenStringArray<N>& data,
00097                          const DataSetCreateProps& createProps = DataSetCreateProps::Default(),
00098                          const DataSetAccessProps& accessProps = DataSetAccessProps::Default(),
00099                          bool parents = true);
00100
00106      DataSet getDataSet(const std::string& dataset_name,
00107                      const DataSetAccessProps& accessProps = DataSetAccessProps::Default()) const;
00108
00114      Group createGroup(const std::string& group_name, bool parents = true);
00115
00122      Group createGroup(const std::string& group_name,
00123                      const GroupCreateProps& createProps,
00124                      bool parents = true);
00125
00130      Group getGroup(const std::string& group_name) const;
00131
00135      size_t getNumberObjects() const;
00136
00140      std::string getObjectNames(size_t index) const;
00141
00148      bool rename(const std::string& src_path,
00149                const std::string& dest_path,
00150                bool parents = true) const;
00151
00158      std::vector<std::string> listObjectNames(IndexType idx_type = IndexType::NAME) const;
00159
00164      bool exist(const std::string& node_name) const;
00165
00169      void unlink(const std::string& node_name) const;
00170
00174      LinkType getLinkType(const std::string& node_name) const;

```

```

00175
00179     ObjectType getObjectType(const std::string& node_name) const;
00180
00184     template <typename T, typename = decltype(T::getPath)>
00185     void createSoftLink(const std::string& linkName, const T& obj) {
00186         static_assert(!std::is_same<T, Attribute>::value,
00187             "hdf5 doesn't support soft links to Attributes");
00188         createSoftLink(linkName, obj.getPath());
00189     }
00190
00198     void createSoftLink(const std::string& link_name,
00199         const std::string& obj_path,
00200         LinkCreateProps linkCreateProps = LinkCreateProps(),
00201         const LinkAccessProps& linkAccessProps = LinkAccessProps(),
00202         const bool parents = true);
00203
00204     void createExternalLink(const std::string& link_name,
00205         const std::string& h5_file,
00206         const std::string& obj_path,
00207         LinkCreateProps linkCreateProps = LinkCreateProps(),
00208         const LinkAccessProps& linkAccessProps = LinkAccessProps(),
00209         const bool parents = true);
00210
00211 private:
00212     using derivate_type = Derivate;
00213
00214     // A wrapper over the low-level H5Lexist
00215     // It makes behavior consistent among versions and by default transforms
00216     // errors to exceptions
00217     bool _exist(const std::string& node_name, bool raise_errors = true) const;
00218
00219     // Opens an arbitrary object to obtain info
00220     Object _open(const std::string& node_name,
00221         const DataSetAccessProps& accessProps = DataSetAccessProps::Default()) const;
00222 };
00223
00224
00228 enum class LinkType {
00229     Hard,
00230     Soft,
00231     External,
00232     Other // Reserved or User-defined
00233 };
00234
00235
00236 } // namespace HighFive

```

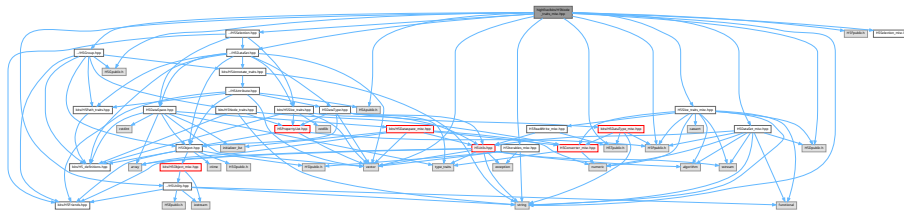
10.30 highfive/bits/H5Node_traits_misc.hpp File Reference

```

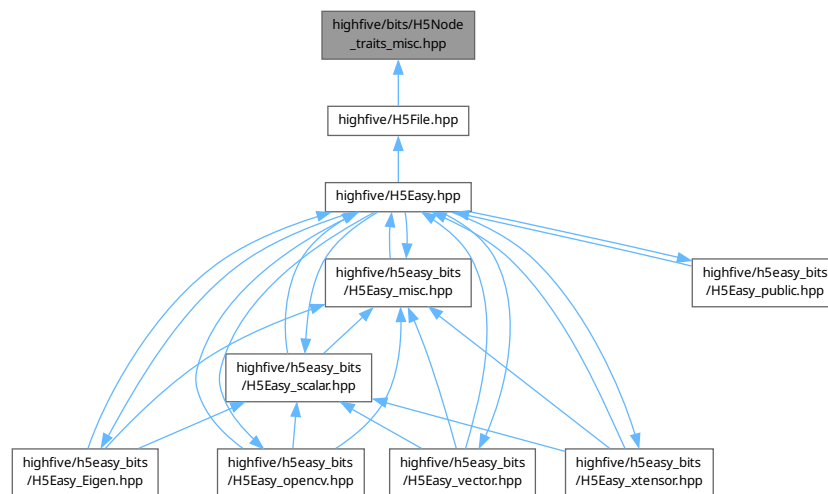
#include <string>
#include <vector>
#include <H5Apublic.h>
#include <H5Dpublic.h>
#include <H5Fpublic.h>
#include <H5Gpublic.h>
#include <H5Ppublic.h>
#include <H5Tpublic.h>
#include "../H5DataSet.hpp"
#include "../H5Group.hpp"
#include "../H5Selection.hpp"
#include "../H5Utility.hpp"
#include "H5DataSet_misc.hpp"
#include "H5Iterables_misc.hpp"
#include "H5Selection_misc.hpp"
#include "H5Slice_traits_misc.hpp"

```

Include dependency graph for H5Node_traits_misc.hpp:



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



Namespaces

- namespace [HighFive](#)

10.31 H5Node_traits_misc.hpp

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c), 2017, Adrien Devresse <adrien.devresse@epfl.ch>
00003  *
00004  * Distributed under the Boost Software License, Version 1.0.
00005  * (See accompanying file LICENSE_1_0.txt or copy at
00006  * http://www.boost.org/LICENSE_1_0.txt)
00007  *
00008  */
00009 #pragma once
00010
00011 #include <string>
00012 #include <vector>
00013
00014 #include <H5Apublic.h>
00015 #include <H5Dpublic.h>
00016 #include <H5Fpublic.h>
00017 #include <H5Gpublic.h>
00018 #include <H5Ppublic.h>
00019 #include <H5Tpublic.h>

```

```

00020
00021 #include "../H5DataSet.hpp"
00022 #include "../H5Group.hpp"
00023 #include "../H5Selection.hpp"
00024 #include "../H5Utility.hpp"
00025 #include "H5DataSet_misc.hpp"
00026 #include "H5Iterables_misc.hpp"
00027 #include "H5Selection_misc.hpp"
00028 #include "H5Slice_traits_misc.hpp"
00029
00030 namespace HighFive {
00031
00032
00033 template <typename Derivate>
00034 inline DataSet NodeTraits<Derivate>::createDataSet(const std::string& dataset_name,
00035                                                    const DataSpace& space,
00036                                                    const DataType& dtype,
00037                                                    const DataSetCreateProps& createProps,
00038                                                    const DataSetAccessProps& accessProps,
00039                                                    bool parents) {
00040     LinkCreateProps lcpl;
00041     lcpl.add(CreateIntermediateGroup(parents));
00042     const auto hid = H5Dcreate2(static_cast<Derivate*>(this)->getId(),
00043                                dataset_name.c_str(),
00044                                dtype.getId(),
00045                                space.getId(),
00046                                lcpl.getId(),
00047                                createProps.getId(),
00048                                accessProps.getId());
00049     if (hid < 0) {
00050         HDF5ErrMapper::ToException<DataSetException>(
00051             std::string("Unable to create the dataset \"" + dataset_name + "\"");
00052         )
00053     }
00054     return DataSet(hid);
00055 }
00056
00057 template <typename Derivate>
00058 template <typename T,
00059          typename std::enable_if<
00060             std::is_same<typename details::inspector<T>::base_type, details::Boolean>::value,
00061             int>::type*>
00062 inline DataSet NodeTraits<Derivate>::createDataSet(const std::string& dataset_name,
00063                                                    const DataSpace& space,
00064                                                    const DataSetCreateProps& createProps,
00065                                                    const DataSetAccessProps& accessProps,
00066                                                    bool parents) {
00067     return createDataSet(dataset_name,
00068                          space,
00069                          create_and_check_datatype<typename details::inspector<T>::base_type>(),
00070                          createProps,
00071                          accessProps,
00072                          parents);
00073 }
00074
00075 template <typename Derivate>
00076 template <typename T,
00077          typename std::enable_if<
00078             !std::is_same<typename details::inspector<T>::base_type, details::Boolean>::value,
00079             int>::type*>
00080 inline DataSet NodeTraits<Derivate>::createDataSet(const std::string& dataset_name,
00081                                                    const DataSpace& space,
00082                                                    const DataSetCreateProps& createProps,
00083                                                    const DataSetAccessProps& accessProps,
00084                                                    bool parents) {
00085     return createDataSet(
00086         dataset_name, space, create_and_check_datatype<T>(), createProps, accessProps, parents);
00087 }
00088
00089 template <typename Derivate>
00090 template <typename T>
00091 inline DataSet NodeTraits<Derivate>::createDataSet(const std::string& dataset_name,
00092                                                    const T& data,
00093                                                    const DataSetCreateProps& createProps,
00094                                                    const DataSetAccessProps& accessProps,
00095                                                    bool parents) {
00096     DataSet ds =
00097         createDataSet(dataset_name,
00098                      DataSpace::From(data),
00099                      create_and_check_datatype<typename details::inspector<T>::base_type>(),
00100                      createProps,
00101                      accessProps,
00102                      parents);
00103     ds.write(data);
00104     return ds;
00105 }
00106
00107 template <typename Derivate>

```



```

00107 template <std::size_t N>
00108 inline DataSet NodeTraits<Derivate>::createDataSet(const std::string& dataset_name,
00109                                                    const FixedLenStringArray<N>& data,
00110                                                    const DataSetCreateProps& createProps,
00111                                                    const DataSetAccessProps& accessProps,
00112                                                    bool parents) {
00113     DataSet ds = createDataSet<char[N]> (
00114         dataset_name, DataSpace(data.size()), createProps, accessProps, parents);
00115     ds.write(data);
00116     return ds;
00117 }
00118
00119 template <typename Derivate>
00120 inline DataSet NodeTraits<Derivate>::getDataSet(const std::string& dataset_name,
00121                                                  const DataSetAccessProps& accessProps) const {
00122     const auto hid = H5Dopen2(static_cast<const Derivate*>(this)->getId(),
00123                               dataset_name.c_str(),
00124                               accessProps.getId());
00125     if (hid < 0) {
00126         HDF5ErrMapper::ToException<DataSetException>(std::string("Unable to open the dataset \"") +
00127                                                       dataset_name + "\"");
00128     }
00129     return DataSet(hid);
00130 }
00131
00132 template <typename Derivate>
00133 inline Group NodeTraits<Derivate>::createGroup(const std::string& group_name, bool parents) {
00134     LinkCreateProps lcpl;
00135     lcpl.add(CreateIntermediateGroup(parents));
00136     const auto hid = H5Gcreate2(static_cast<Derivate*>(this)->getId(),
00137                                group_name.c_str(),
00138                                lcpl.getId(),
00139                                H5P_DEFAULT,
00140                                H5P_DEFAULT);
00141     if (hid < 0) {
00142         HDF5ErrMapper::ToException<GroupException>(std::string("Unable to create the group \"") +
00143                                                     group_name + "\"");
00144     }
00145     return detail::make_group(hid);
00146 }
00147
00148 template <typename Derivate>
00149 inline Group NodeTraits<Derivate>::createGroup(const std::string& group_name,
00150                                                  const GroupCreateProps& createProps,
00151                                                  bool parents) {
00152     LinkCreateProps lcpl;
00153     lcpl.add(CreateIntermediateGroup(parents));
00154     const auto hid = H5Gcreate2(static_cast<Derivate*>(this)->getId(),
00155                                group_name.c_str(),
00156                                lcpl.getId(),
00157                                createProps.getId(),
00158                                H5P_DEFAULT);
00159     if (hid < 0) {
00160         HDF5ErrMapper::ToException<GroupException>(std::string("Unable to create the group \"") +
00161                                                     group_name + "\"");
00162     }
00163     return detail::make_group(hid);
00164 }
00165
00166 template <typename Derivate>
00167 inline Group NodeTraits<Derivate>::getGroup(const std::string& group_name) const {
00168     const auto hid =
00169         H5Gopen2(static_cast<const Derivate*>(this)->getId(), group_name.c_str(), H5P_DEFAULT);
00170     if (hid < 0) {
00171         HDF5ErrMapper::ToException<GroupException>(std::string("Unable to open the group \"") +
00172                                                     group_name + "\"");
00173     }
00174     return detail::make_group(hid);
00175 }
00176
00177 template <typename Derivate>
00178 inline size_t NodeTraits<Derivate>::getNumberObjects() const {
00179     hsize_t res;
00180     if (H5Gget_num_objs(static_cast<const Derivate*>(this)->getId(), &res) < 0) {
00181         HDF5ErrMapper::ToException<GroupException>(
00182             std::string("Unable to count objects in existing group or file"));
00183     }
00184     return static_cast<size_t>(res);
00185 }
00186
00187 template <typename Derivate>
00188 inline std::string NodeTraits<Derivate>::getObjectName(size_t index) const {
00189     return details::get_name([&](char* buffer, size_t length) {
00190         return H5Lget_name_by_idx(static_cast<const Derivate*>(this)->getId(),
00191                                   ".",
00192                                   H5_INDEX_NAME,
00193                                   H5_ITER_INC,

```

```

00194             index,
00195             buffer,
00196             length,
00197             H5P_DEFAULT);
00198     });
00199 }
00200
00201 template <typename Derivate>
00202 inline bool NodeTraits<Derivate>::rename(const std::string& src_path,
00203                                         const std::string& dst_path,
00204                                         bool parents) const {
00205     LinkCreateProps lcpl;
00206     lcpl.add(CreateIntermediateGroup(parents));
00207     herr_t status = H5Lmove(static_cast<const Derivate*>(this)->getId(),
00208                           src_path.c_str(),
00209                           static_cast<const Derivate*>(this)->getId(),
00210                           dst_path.c_str(),
00211                           lcpl.getId(),
00212                           H5P_DEFAULT);
00213     if (status < 0) {
00214         HDF5ErrMapper::ToException<GroupException>(std::string("Unable to move link to \"" +
00215                                                         dst_path + "\"");
00216         return false;
00217     }
00218     return true;
00219 }
00220
00221 template <typename Derivate>
00222 inline std::vector<std::string> NodeTraits<Derivate>::listObjectNames(IndexType idx_type) const {
00223     std::vector<std::string> names;
00224     details::HighFiveIterateData iterateData(names);
00225
00226     size_t num_objs = getNumberObjects();
00227     names.reserve(num_objs);
00228
00229     if (H5Literate(static_cast<const Derivate*>(this)->getId(),
00230                   static_cast<H5_index_t>(idx_type),
00231                   H5_ITER_INC,
00232                   NULL,
00233                   &details::internal_high_five_iterate<H5L_info_t>,
00234                   static_cast<void*>(&iterateData)) < 0) {
00235         HDF5ErrMapper::ToException<GroupException>(std::string("Unable to list objects in group"));
00236     }
00237
00238     return names;
00239 }
00240
00241 template <typename Derivate>
00242 inline bool NodeTraits<Derivate>::_exist(const std::string& node_name, bool raise_errors) const {
00243     SilenceHDF5 silencer{};
00244     const auto val =
00245         H5Lexists(static_cast<const Derivate*>(this)->getId(), node_name.c_str(), H5P_DEFAULT);
00246     if (val < 0) {
00247         if (raise_errors) {
00248             HDF5ErrMapper::ToException<GroupException>("Invalid link for exist()");
00249         } else {
00250             return false;
00251         }
00252     }
00253
00254     // The root path always exists, but H5Lexists return 0 or 1
00255     // depending of the version of HDF5, so always return true for it
00256     // We had to call H5Lexists anyway to check that there are no errors
00257     return (node_name == "/" ) ? true : (val > 0);
00258 }
00259
00260 template <typename Derivate>
00261 inline bool NodeTraits<Derivate>::exist(const std::string& group_path) const {
00262     // When there are slashes, first check everything is fine
00263     // so that subsequent errors are only due to missing intermediate groups
00264     if (group_path.find('/') != std::string::npos) {
00265         _exist("/"); // Shall not throw under normal circumstances
00266         // Unless "/" (already checked), verify path exists (not throwing errors)
00267         return (group_path == "/" ) ? true : _exist(group_path, false);
00268     }
00269     return _exist(group_path);
00270 }
00271
00272
00273 template <typename Derivate>
00274 inline void NodeTraits<Derivate>::unlink(const std::string& node_name) const {
00275     const herr_t val =
00276         H5Ldelete(static_cast<const Derivate*>(this)->getId(), node_name.c_str(), H5P_DEFAULT);
00277     if (val < 0) {
00278         HDF5ErrMapper::ToException<GroupException>(std::string("Invalid name for unlink() "));
00279     }
00280 }

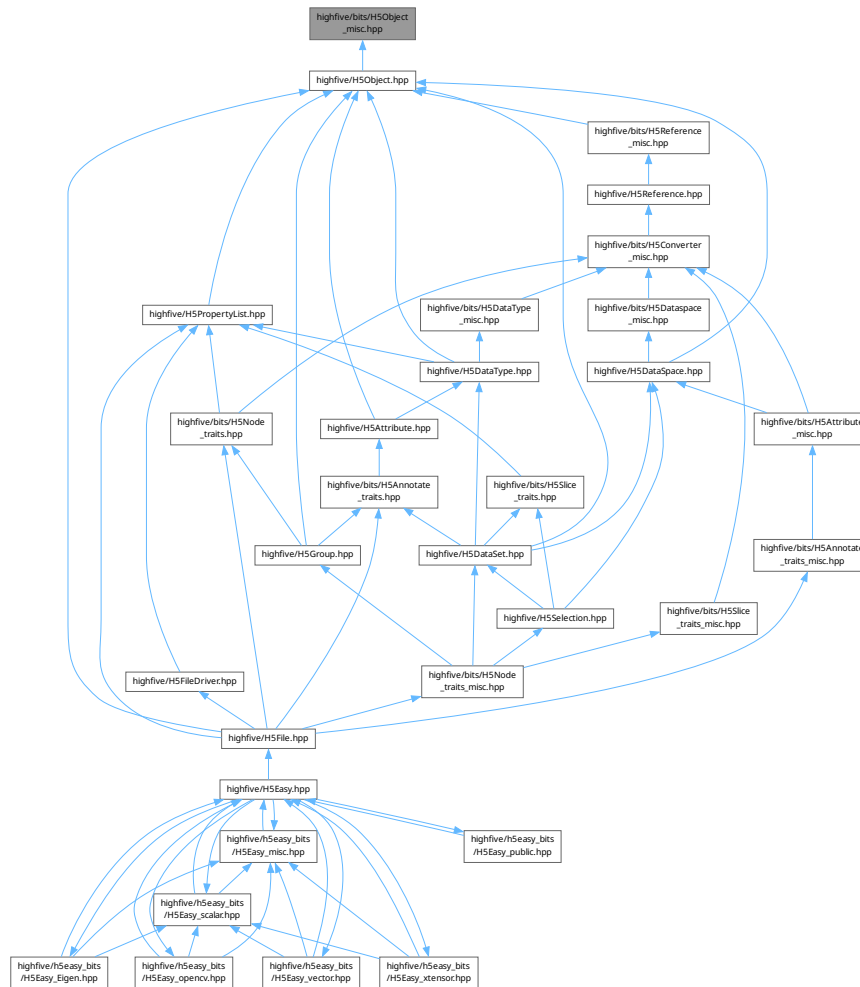
```

```

00281
00282
00283 // convert internal link types to enum class.
00284 // This function is internal, so H5L_TYPE_ERROR shall be handled in the calling context
00285 static inline LinkType _convert_link_type(const H5L_type_t& ltype) noexcept {
00286     switch (ltype) {
00287         case H5L_TYPE_HARD:
00288             return LinkType::Hard;
00289         case H5L_TYPE_SOFT:
00290             return LinkType::Soft;
00291         case H5L_TYPE_EXTERNAL:
00292             return LinkType::External;
00293         default:
00294             // Other link types are possible but are considered strange to HighFive.
00295             // see https://support.hdfgroup.org/HDF5/doc/RM/H5L/H5Lregister.htm
00296             return LinkType::Other;
00297     }
00298 }
00299
00300 template <typename Derivate>
00301 inline LinkType NodeTraits<Derivate>::getLinkType(const std::string& node_name) const {
00302     H5L_info_t linkinfo;
00303     if (H5Lget_info(static_cast<const Derivate*>(this)->getId(),
00304                     node_name.c_str(),
00305                     &linkinfo,
00306                     H5P_DEFAULT) < 0 ||
00307         linkinfo.type == H5L_TYPE_ERROR) {
00308         HDF5ErrMapper::ToException<GroupException>(std::string("Unable to obtain info for link ") +
00309                                                    node_name);
00310     }
00311     return _convert_link_type(linkinfo.type);
00312 }
00313
00314 template <typename Derivate>
00315 inline ObjectType NodeTraits<Derivate>::getObjectType(const std::string& node_name) const {
00316     return _open(node_name).getType();
00317 }
00318
00319
00320 template <typename Derivate>
00321 inline void NodeTraits<Derivate>::createSoftLink(const std::string& link_name,
00322                                                  const std::string& obj_path,
00323                                                  LinkCreateProps linkCreateProps,
00324                                                  const LinkAccessProps& linkAccessProps,
00325                                                  const bool parents) {
00326     if (parents) {
00327         linkCreateProps.add(CreateIntermediateGroup{});
00328     }
00329     auto status = H5Lcreate_soft(obj_path.c_str(),
00330                                 static_cast<const Derivate*>(this)->getId(),
00331                                 link_name.c_str(),
00332                                 linkCreateProps.getId(),
00333                                 linkAccessProps.getId());
00334     if (status < 0) {
00335         HDF5ErrMapper::ToException<GroupException>(std::string("Unable to create soft link: "));
00336     }
00337 }
00338
00339
00340 template <typename Derivate>
00341 inline void NodeTraits<Derivate>::createExternalLink(const std::string& link_name,
00342                                                      const std::string& h5_file,
00343                                                      const std::string& obj_path,
00344                                                      LinkCreateProps linkCreateProps,
00345                                                      const LinkAccessProps& linkAccessProps,
00346                                                      const bool parents) {
00347     if (parents) {
00348         linkCreateProps.add(CreateIntermediateGroup{});
00349     }
00350     auto status = H5Lcreate_external(h5_file.c_str(),
00351                                     obj_path.c_str(),
00352                                     static_cast<const Derivate*>(this)->getId(),
00353                                     link_name.c_str(),
00354                                     linkCreateProps.getId(),
00355                                     linkAccessProps.getId());
00356     if (status < 0) {
00357         HDF5ErrMapper::ToException<GroupException>(std::string("Unable to create external link: "));
00358     }
00359 }
00360
00361
00362 template <typename Derivate>
00363 inline Object NodeTraits<Derivate>::_open(const std::string& node_name,
00364                                           const DataSetAccessProps& accessProps) const {
00365     const auto id = H5Oopen(static_cast<const Derivate*>(this)->getId(),
00366                             node_name.c_str(),
00367                             accessProps.getId());

```


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



Namespaces

- namespace [HighFive](#)

10.33 H5Object_misc.hpp

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c), 2017, Adrien Devresse <adrien.devresse@epfl.ch>
00003  *
00004  * Distributed under the Boost Software License, Version 1.0.
00005  * (See accompanying file LICENSE_1_0.txt or copy at
00006  * http://www.boost.org/LICENSE_1_0.txt)
00007  *
00008  */
00009 #pragma once
00010
00011 #include <iostream>
00012
00013 #include "../H5Exception.hpp"
00014 #include "../H5Utility.hpp"
00015
00016 namespace HighFive {

```

```

00017 namespace detail {
00018 inline Object make_object(hid_t hid) {
00019     return Object(hid);
00020 }
00021 } // namespace detail
00022
00023
00024 inline Object::Object()
00025     : _hid(H5I_INVALID_HID) {}
00026
00027 inline Object::Object(hid_t hid)
00028     : _hid(hid) {}
00029
00030 inline Object::Object(const Object& other)
00031     : _hid(other._hid) {
00032     if (other.isValid() && H5Iinc_ref(_hid) < 0) {
00033         throw ObjectException("Reference counter increase failure");
00034     }
00035 }
00036
00037 inline Object::Object(Object&& other) noexcept
00038     : _hid(other._hid) {
00039     other._hid = H5I_INVALID_HID;
00040 }
00041
00042 inline Object& Object::operator=(const Object& other) {
00043     if (this != &other) {
00044         if (isValid())
00045             H5Idec_ref(_hid);
00046
00047         _hid = other._hid;
00048         if (other.isValid() && H5Iinc_ref(_hid) < 0) {
00049             throw ObjectException("Reference counter increase failure");
00050         }
00051     }
00052     return *this;
00053 }
00054
00055 inline Object::~Object() {
00056     if (isValid() && H5Idec_ref(_hid) < 0) {
00057         HIGHFIVE_LOG_ERROR("HighFive::~Object: reference counter decrease failure");
00058     }
00059 }
00060
00061 inline bool Object::isValid() const noexcept {
00062     return (_hid != H5I_INVALID_HID) && (H5Iis_valid(_hid) != false);
00063 }
00064
00065 inline hid_t Object::getId() const noexcept {
00066     return _hid;
00067 }
00068
00069 static inline ObjectType _convert_object_type(const H5I_type_t& h5type) {
00070     switch (h5type) {
00071     case H5I_FILE:
00072         return ObjectType::File;
00073     case H5I_GROUP:
00074         return ObjectType::Group;
00075     case H5I_DATATYPE:
00076         return ObjectType::UserDataType;
00077     case H5I_DATASPACE:
00078         return ObjectType::DataSpace;
00079     case H5I_DATASET:
00080         return ObjectType::Dataset;
00081     case H5I_ATTR:
00082         return ObjectType::Attribute;
00083     default:
00084         return ObjectType::Other;
00085     }
00086 }
00087
00088 inline ObjectType Object::getType() const {
00089     // H5Iget_type is a very lightweight func which extracts the type from the id
00090     H5I_type_t h5type;
00091     if ((h5type = H5Iget_type(_hid)) == H5I_BADID) {
00092         HDF5ErrMapper::ToException<ObjectException>("Invalid hid or object type");
00093     }
00094     return _convert_object_type(h5type);
00095 }
00096
00097 inline ObjectInfo Object::getInfo() const {
00098     ObjectInfo info;
00099     #if (H5Oget_info_vers < 3)
00100     if (H5Oget_info(_hid, &info.raw_info) < 0) {
00101     #else
00102     if (H5Oget_info1(_hid, &info.raw_info) < 0) {
00103     #endif

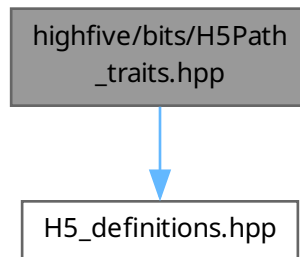
```

```
00104         HDF5ErrMapper::ToException<ObjectException>("Unable to obtain info for object");
00105     }
00106     return info;
00107 }
00108
00109 inline haddr_t ObjectInfo::getAddress() const noexcept {
00110     return raw_info.addr;
00111 }
00112 inline size_t ObjectInfo::getRefCount() const noexcept {
00113     return raw_info.rc;
00114 }
00115 inline time_t ObjectInfo::getCreationTime() const noexcept {
00116     return raw_info.btime;
00117 }
00118 inline time_t ObjectInfo::getModificationTime() const noexcept {
00119     return raw_info.mtime;
00120 }
00121
00122
00123 } // namespace HighFive
```

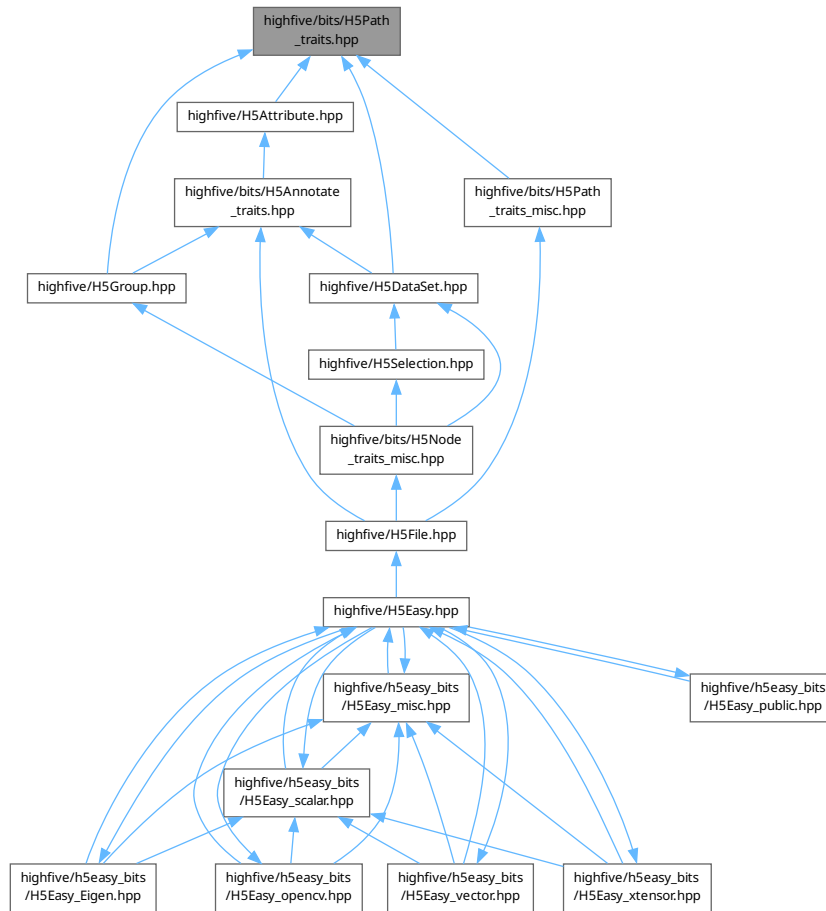
10.34 highfive/bits/H5Path_traits.hpp File Reference

#include "H5_definitions.hpp"

Include dependency graph for H5Path_traits.hpp:



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



Classes

- class [HighFive::PathTraits](#)< Derivate >

Namespaces

- namespace [HighFive](#)

10.35 H5Path_traits.hpp

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c), 2020, EPFL - Blue Brain Project
00003  *
00004  * Distributed under the Boost Software License, Version 1.0.
00005  * (See accompanying file LICENSE_1_0.txt or copy at
00006  * http://www.boost.org/LICENSE_1_0.txt)
00007  *
00008  */
00009 #pragma once
00010

```



```

00011 #include "H5_definitions.hpp"
00012
00013 namespace HighFive {
00014
00015 template <typename Derivate>
00016 class PathTraits {
00017 public:
00018     PathTraits();
00019
00023     std::string getPath() const;
00024
00028     File& getFile() const noexcept;
00029
00030 protected:
00032     std::shared_ptr<File> _file_obj; // keep a ref to file so we keep its ref count > 0
00033 };
00034
00035 } // namespace HighFive

```

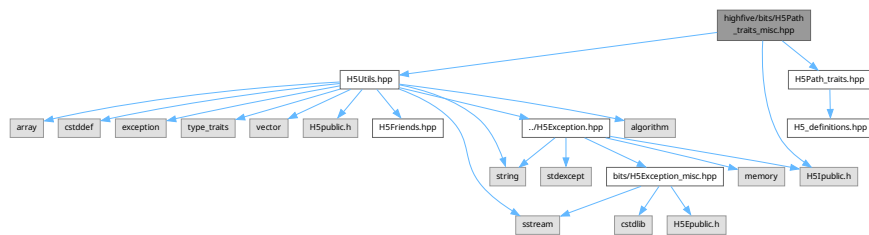
10.36 highfive/bits/H5Path_traits_misc.hpp File Reference

```

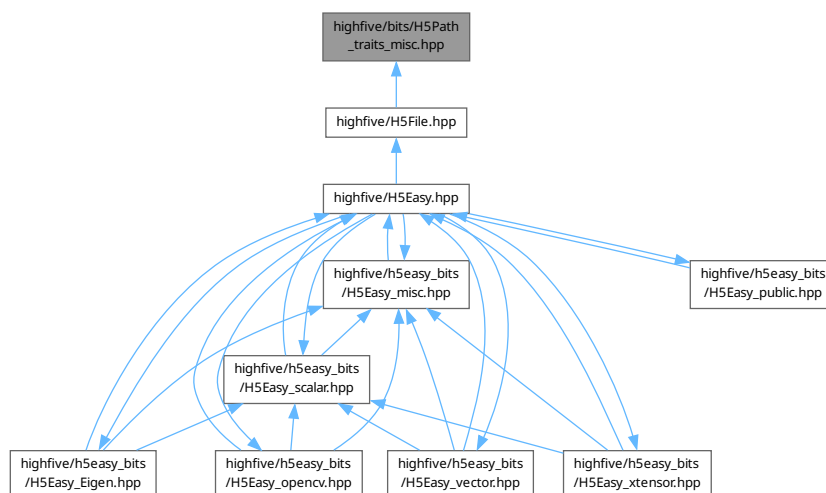
#include <H5Ipublic.h>
#include "H5Utils.hpp"
#include "H5Path_traits.hpp"

```

Include dependency graph for H5Path_traits_misc.hpp:



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



Namespaces

- namespace [HighFive](#)

10.37 H5Path_traits_misc.hpp

[Go to the documentation of this file.](#)

```

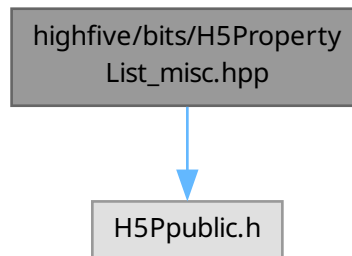
00001  /*
00002  *   Copyright (c), 2020, EPFL - Blue Brain Project
00003  *
00004  *   Distributed under the Boost Software License, Version 1.0.
00005  *   (See accompanying file LICENSE_1_0.txt or copy at
00006  *    http://www.boost.org/LICENSE_1_0.txt)
00007  *
00008  */
00009  #pragma once
00010
00011  #include <H5Ipublic.h>
00012
00013  #include "H5Utils.hpp"
00014  #include "H5Path_traits.hpp"
00015
00016  namespace HighFive {
00017
00018  template <typename Derivate>
00019  inline PathTraits<Derivate>::PathTraits() {
00020      static_assert(std::is_same<Derivate, Group>::value || std::is_same<Derivate, DataSet>::value ||
00021                  std::is_same<Derivate, Attribute>::value,
00022                  "PathTraits can only be applied to Group, DataSet and Attribute");
00023      const auto& obj = static_cast<const Derivate&>(*this);
00024      if (!obj.isValid()) {
00025          return;
00026      }
00027      const hid_t file_id = H5Iget_file_id(obj.getId());
00028      if (file_id < 0) {
00029          HDF5ErrMapper::ToException<PropertyException>("getFile(): Could not obtain file of object");
00030      }
00031      _file_obj.reset(new File(file_id));
00032  }
00033
00034  template <typename Derivate>
00035  inline std::string PathTraits<Derivate>::getPath() const {
00036      return details::get_name([this](char* buffer, size_t length) {
00037          return H5Iget_name(static_cast<const Derivate*>(this)->getId(), buffer, length);
00038      });
00039  }
00040
00041  template <typename Derivate>
00042  inline File& PathTraits<Derivate>::getFile() const noexcept {
00043      return *_file_obj;
00044  }
00045
00046  } // namespace HighFive

```

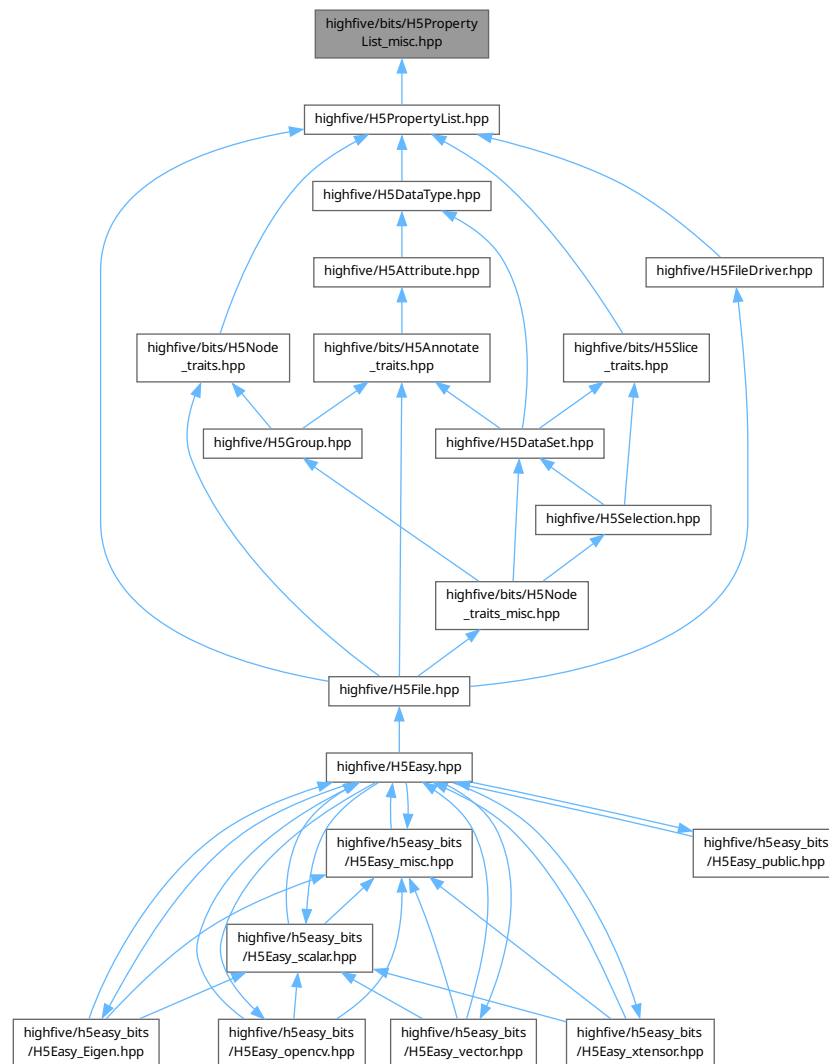
10.38 highfive/bits/H5PropertyList_misc.hpp File Reference

```
#include <H5Ppublic.h>
```

Include dependency graph for H5PropertyList_misc.hpp:



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



Namespaces

- namespace [HighFive](#)

10.39 H5PropertyList_misc.hpp

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c), 2017-2018, Adrien Devresse <adrien.devresse@epfl.ch>
00003  *                               Juan Hernando <juan.hernando@epfl.ch>
00004  * Distributed under the Boost Software License, Version 1.0.
00005  * (See accompanying file LICENSE_1_0.txt or copy at
00006  *  http://www.boost.org/LICENSE_1_0.txt)
00007  *
00008  */
00009 #pragma once
00010
00011 #include <H5Ppublic.h>

```

```

00012
00013 namespace HighFive {
00014
00015 namespace {
00016 inline hid_t convert_plist_type(PropertyType propertyType) {
00017     // The HP5_XXX are macros with function calls so we can't assign
00018     // them as the enum values
00019     switch (propertyType) {
00020     case PropertyType::OBJECT_CREATE:
00021         return H5P_OBJECT_CREATE;
00022     case PropertyType::FILE_CREATE:
00023         return H5P_FILE_CREATE;
00024     case PropertyType::FILE_ACCESS:
00025         return H5P_FILE_ACCESS;
00026     case PropertyType::DATASET_CREATE:
00027         return H5P_DATASET_CREATE;
00028     case PropertyType::DATASET_ACCESS:
00029         return H5P_DATASET_ACCESS;
00030     case PropertyType::DATASET_XFER:
00031         return H5P_DATASET_XFER;
00032     case PropertyType::GROUP_CREATE:
00033         return H5P_GROUP_CREATE;
00034     case PropertyType::GROUP_ACCESS:
00035         return H5P_GROUP_ACCESS;
00036     case PropertyType::DATATYPE_CREATE:
00037         return H5P_DATATYPE_CREATE;
00038     case PropertyType::DATATYPE_ACCESS:
00039         return H5P_DATATYPE_ACCESS;
00040     case PropertyType::STRING_CREATE:
00041         return H5P_STRING_CREATE;
00042     case PropertyType::ATTRIBUTE_CREATE:
00043         return H5P_ATTRIBUTE_CREATE;
00044     case PropertyType::OBJECT_COPY:
00045         return H5P_OBJECT_COPY;
00046     case PropertyType::LINK_CREATE:
00047         return H5P_LINK_CREATE;
00048     case PropertyType::LINK_ACCESS:
00049         return H5P_LINK_ACCESS;
00050     default:
00051         HDF5ErrMapper::ToException<PropertyException>("Unsupported property list type");
00052     }
00053 }
00054
00055 } // namespace
00056
00057
00058 inline PropertyListBase::PropertyListBase() noexcept
00059     : Object(H5P_DEFAULT) {}
00060
00061
00062 template <PropertyType T>
00063 inline void PropertyList<T>::_initializeIfNeeded() {
00064     if (_hid != H5P_DEFAULT) {
00065         return;
00066     }
00067     if ((_hid = H5Pcreate(convert_plist_type(T))) < 0) {
00068         HDF5ErrMapper::ToException<PropertyException>("Unable to create property list");
00069     }
00070 }
00071
00072 template <PropertyType T>
00073 template <typename P>
00074 inline void PropertyList<T>::add(const P& property) {
00075     _initializeIfNeeded();
00076     property.apply(_hid);
00077 }
00078
00079 template <PropertyType T>
00080 template <typename F, typename... Args>
00081 inline void RawPropertyList<T>::add(const F& funct, const Args&... args) {
00082     this->_initializeIfNeeded();
00083     if (funct(this->_hid, args...) < 0) {
00084         HDF5ErrMapper::ToException<PropertyException>("Error setting raw hdf5 property.");
00085     }
00086 }
00087
00088 // Specific options to be added to Property Lists
00089 #if H5_VERSION_GE(1, 10, 1)
00090 inline FileSpaceStrategy::FileSpaceStrategy(H5F_fspace_strategy_t strategy,
00091                                             hbool_t persist,
00092                                             hsize_t threshold)
00093     : _strategy(strategy)
00094     , _persist(persist)
00095     , _threshold(threshold) {}
00096
00097 inline FileSpaceStrategy::FileSpaceStrategy(const FileCreateProps& fcpl) {
00098     if (H5Pget_file_space_strategy(fcpl.getId(), &_amp;_strategy, &_amp;_persist, &_amp;_threshold) < 0) {

```

```

00099         HDF5ErrMapper::ToException<PropertyException>("Unable to get file space strategy");
00100     }
00101 }
00102
00103 inline void FileSpaceStrategy::apply(const hid_t list) const {
00104     if (H5Pset_file_space_strategy(list, _strategy, _persist, _threshold) < 0) {
00105         HDF5ErrMapper::ToException<PropertyException>("Error setting file space strategy.");
00106     }
00107 }
00108
00109 inline H5F_fspace_strategy_t FileSpaceStrategy::getStrategy() const {
00110     return _strategy;
00111 }
00112
00113 inline hbool_t FileSpaceStrategy::getPersist() const {
00114     return _persist;
00115 }
00116
00117 inline hsize_t FileSpaceStrategy::getThreshold() const {
00118     return _threshold;
00119 }
00120
00121 inline FileSpacePageSize::FileSpacePageSize(hsize_t page_size)
00122     : _page_size(page_size) {}
00123
00124 inline void FileSpacePageSize::apply(const hid_t list) const {
00125     if (H5Pset_file_space_page_size(list, _page_size) < 0) {
00126         HDF5ErrMapper::ToException<PropertyException>("Error setting file space page size.");
00127     }
00128 }
00129
00130 inline FileSpacePageSize::FileSpacePageSize(const FileCreateProps& fcpl) {
00131     if (H5Pget_file_space_page_size(fcpl.getId(), &_page_size) < 0) {
00132         HDF5ErrMapper::ToException<PropertyException>("Unable to get file space page size");
00133     }
00134 }
00135
00136 inline hsize_t FileSpacePageSize::getPageSize() const {
00137     return _page_size;
00138 }
00139
00140 #ifndef H5_HAVE_PARALLEL
00141 inline PageBufferSize::PageBufferSize(size_t page_buffer_size,
00142                                         unsigned min_meta_percent,
00143                                         unsigned min_raw_percent)
00144     : _page_buffer_size(page_buffer_size)
00145     , _min_meta(min_meta_percent)
00146     , _min_raw(min_raw_percent) {}
00147
00148 inline PageBufferSize::PageBufferSize(const FileAccessProps& plist) {
00149     if (H5Pget_page_buffer_size(plist.getId(), &_page_buffer_size, &_min_meta, &_min_raw) < 0) {
00150         HDF5ErrMapper::ToException<PropertyException>("Error setting page buffer size.");
00151     }
00152 }
00153
00154 inline void PageBufferSize::apply(const hid_t list) const {
00155     if (H5Pset_page_buffer_size(list, _page_buffer_size, _min_meta, _min_raw) < 0) {
00156         HDF5ErrMapper::ToException<PropertyException>("Error setting page buffer size.");
00157     }
00158 }
00159
00160 inline size_t PageBufferSize::getPageBufferSize() const {
00161     return _page_buffer_size;
00162 }
00163
00164 inline unsigned PageBufferSize::getMinMetaPercent() const {
00165     return _min_meta;
00166 }
00167
00168 inline unsigned PageBufferSize::getMinRawPercent() const {
00169     return _min_raw;
00170 }
00171 #endif
00172 #endif
00173
00174 #ifdef H5_HAVE_PARALLEL
00175
00176 inline MPIIOFileAccess::MPIIOFileAccess(MPI_Comm comm, MPI_Info info)
00177     : _comm(comm)
00178     , _info(info) {}
00179
00180 inline void MPIIOFileAccess::apply(const hid_t list) const {
00181     if (H5Pset_fapl_mpio(list, _comm, _info) < 0) {
00182         HDF5ErrMapper::ToException<FileException>("Unable to set-up MPIIO Driver configuration");
00183     }
00184 }
00185

```

```

00186 inline void MPIOCollectiveMetadata::apply(const hid_t plist) const {
00187     auto read = MPIOCollectiveMetadataRead{collective_read_};
00188     auto write = MPIOCollectiveMetadataWrite{collective_write_};
00189
00190     read.apply(plist);
00191     write.apply(plist);
00192 }
00193
00194 inline MPIOCollectiveMetadata::MPIOCollectiveMetadata(bool collective)
00195     : collective_read_(collective)
00196     , collective_write_(collective) {}
00197
00198
00199 inline MPIOCollectiveMetadata::MPIOCollectiveMetadata(const FileAccessProps& plist)
00200     : collective_read_(MPIOCollectiveMetadataRead(plist).isCollective())
00201     , collective_write_(MPIOCollectiveMetadataWrite(plist).isCollective()) {}
00202
00203 inline bool MPIOCollectiveMetadata::isCollectiveRead() const {
00204     return collective_read_;
00205 }
00206
00207 inline bool MPIOCollectiveMetadata::isCollectiveWrite() const {
00208     return collective_write_;
00209 }
00210
00211
00212 inline void MPIOCollectiveMetadataRead::apply(const hid_t plist) const {
00213     if (H5Pset_all_coll_metadata_ops(plist, collective_) < 0) {
00214         HDF5ErrMapper::ToException<FileException>("Unable to request collective metadata reads");
00215     }
00216 }
00217
00218 inline bool MPIOCollectiveMetadataRead::isCollective() const {
00219     return collective_;
00220 }
00221
00222 inline MPIOCollectiveMetadataRead::MPIOCollectiveMetadataRead(const FileAccessProps& plist) {
00223     if (H5Pget_all_coll_metadata_ops(plist.getId(), &collective_) < 0) {
00224         HDF5ErrMapper::ToException<PropertyException>("Error loading MPI metadata read.");
00225     }
00226 }
00227
00228 inline MPIOCollectiveMetadataRead::MPIOCollectiveMetadataRead(bool collective)
00229     : collective_(collective) {}
00230
00231 inline void MPIOCollectiveMetadataWrite::apply(const hid_t plist) const {
00232     if (H5Pset_coll_metadata_write(plist, collective_) < 0) {
00233         HDF5ErrMapper::ToException<FileException>("Unable to request collective metadata writes");
00234     }
00235 }
00236
00237 inline bool MPIOCollectiveMetadataWrite::isCollective() const {
00238     return collective_;
00239 }
00240
00241 inline MPIOCollectiveMetadataWrite::MPIOCollectiveMetadataWrite(const FileAccessProps& plist) {
00242     if (H5Pget_coll_metadata_write(plist.getId(), &collective_) < 0) {
00243         HDF5ErrMapper::ToException<PropertyException>("Error loading MPI metadata write.");
00244     }
00245 }
00246
00247 inline MPIOCollectiveMetadataWrite::MPIOCollectiveMetadataWrite(bool collective)
00248     : collective_(collective) {}
00249
00250 #endif
00251
00252 inline FileVersionBounds::FileVersionBounds(H5F_libver_t low, H5F_libver_t high)
00253     : _low(low)
00254     , _high(high) {}
00255
00256 inline FileVersionBounds::FileVersionBounds(const FileAccessProps& fapl) {
00257     if (H5Pget_libver_bounds(fapl.getId(), &_low, &_high) < 0) {
00258         HDF5ErrMapper::ToException<PropertyException>("Unable to access file version bounds");
00259     }
00260 }
00261
00262 inline std::pair<H5F_libver_t, H5F_libver_t> FileVersionBounds::getVersion() const {
00263     return std::make_pair(_low, _high);
00264 }
00265
00266 inline void FileVersionBounds::apply(const hid_t list) const {
00267     if (H5Pset_libver_bounds(list, _low, _high) < 0) {
00268         HDF5ErrMapper::ToException<PropertyException>("Error setting file version bounds");
00269     }
00270 }
00271
00272 inline MetadataBlockSize::MetadataBlockSize(hsize_t size)

```

```

00273     : _size(size) {}
00274
00275 inline MetadataBlockSize::MetadataBlockSize(const FileAccessProps& fapl) {
00276     if (H5Pget_meta_block_size(fapl.getId(), &_size) < 0) {
00277         HDF5ErrMapper::ToException<PropertyException>("Unable to access file metadata block size");
00278     }
00279 }
00280
00281 inline void MetadataBlockSize::apply(const hid_t list) const {
00282     if (H5Pset_meta_block_size(list, _size) < 0) {
00283         HDF5ErrMapper::ToException<PropertyException>("Error setting metadata block size");
00284     }
00285 }
00286
00287 inline hsize_t MetadataBlockSize::getSize() const {
00288     return _size;
00289 }
00290
00291 inline void EstimatedLinkInfo::apply(const hid_t hid) const {
00292     if (H5Pset_est_link_info(hid, _entries, _length) < 0) {
00293         HDF5ErrMapper::ToException<PropertyException>("Error setting estimated link info");
00294     }
00295 }
00296
00297 inline EstimatedLinkInfo::EstimatedLinkInfo(unsigned entries, unsigned length)
00298     : _entries(entries)
00299     , _length(length) {}
00300
00301 inline EstimatedLinkInfo::EstimatedLinkInfo(const GroupCreateProps& gcpl) {
00302     if (H5Pget_est_link_info(gcpl.getId(), &_entries, &_length) < 0) {
00303         HDF5ErrMapper::ToException<PropertyException>("Unable to access group link size property");
00304     }
00305 }
00306
00307 inline unsigned EstimatedLinkInfo::getEntries() const {
00308     return _entries;
00309 }
00310
00311 inline unsigned EstimatedLinkInfo::getNameLength() const {
00312     return _length;
00313 }
00314
00315 inline void Chunking::apply(const hid_t hid) const {
00316     if (H5Pset_chunk(hid, static_cast<int>(_dims.size()), _dims.data()) < 0) {
00317         HDF5ErrMapper::ToException<PropertyException>("Error setting chunk property");
00318     }
00319 }
00320
00321 inline Chunking::Chunking(const std::vector<hsize_t>& dims)
00322     : _dims(dims) {}
00323
00324 inline Chunking::Chunking(const std::initializer_list<hsize_t>& items)
00325     : Chunking(std::vector<hsize_t>(items)) {}
00326
00327 inline Chunking::Chunking(DataSetCreateProps& plist, size_t max_dims)
00328     : _dims(max_dims + 1) {
00329     auto n_loaded = H5Pget_chunk(plist.getId(), static_cast<int>(_dims.size()), _dims.data());
00330     if (n_loaded < 0) {
00331         HDF5ErrMapper::ToException<PropertyException>("Error getting chunk size");
00332     }
00333
00334     if (n_loaded >= static_cast<int>(_dims.size())) {
00335         *this = Chunking(plist, 8 * max_dims);
00336     } else {
00337         _dims.resize(static_cast<size_t>(n_loaded));
00338     }
00339 }
00340
00341 inline const std::vector<hsize_t>& Chunking::getDimensions() const noexcept {
00342     return _dims;
00343 }
00344
00345 template <typename... Args>
00346 inline Chunking(hsize_t item, Args... args)
00347     : Chunking(std::vector<hsize_t>(item, static_cast<hsize_t>(args)...)) {}
00348
00349 inline void Deflate::apply(const hid_t hid) const {
00350     if (!H5Zfilter_avail(H5Z_FILTER_DEFLATE) || H5Pset_deflate(hid, _level) < 0) {
00351         HDF5ErrMapper::ToException<PropertyException>("Error setting deflate property");
00352     }
00353 }
00354
00355 inline Deflate::Deflate(unsigned int level)
00356     : _level(level) {}
00357
00358 inline void Szip::apply(const hid_t hid) const {
00359     if (!H5Zfilter_avail(H5Z_FILTER_SZIP)) {

```



```

00360         HDF5ErrMapper::ToException<PropertyException>("Error setting szip property");
00361     }
00362
00363     if (H5Pset_szip(hid, _options_mask, _pixels_per_block) < 0) {
00364         HDF5ErrMapper::ToException<PropertyException>("Error setting szip property");
00365     }
00366 }
00367
00368 inline Szip::Szip(unsigned int options_mask, unsigned int pixels_per_block)
00369 : _options_mask(options_mask)
00370 , _pixels_per_block(pixels_per_block) {}
00371
00372 inline unsigned Szip::getOptionsMask() const {
00373     return _options_mask;
00374 }
00375
00376 inline unsigned Szip::getPixelsPerBlock() const {
00377     return _pixels_per_block;
00378 }
00379
00380 inline void Shuffle::apply(const hid_t hid) const {
00381     if (!H5Zfilter_avail(H5Z_FILTER_SHUFFLE)) {
00382         HDF5ErrMapper::ToException<PropertyException>("Error setting shuffle property");
00383     }
00384
00385     if (H5Pset_shuffle(hid) < 0) {
00386         HDF5ErrMapper::ToException<PropertyException>("Error setting shuffle property");
00387     }
00388 }
00389
00390 inline AllocationTime::AllocationTime(H5D_alloc_time_t alloc_time)
00391 : _alloc_time(alloc_time) {}
00392
00393 inline AllocationTime::AllocationTime(const DataSetCreateProps& dcpl) {
00394     if (H5Pget_alloc_time(dcpl.getId(), &_alloc_time) < 0) {
00395         HDF5ErrMapper::ToException<PropertyException>("Error getting allocation time");
00396     }
00397 }
00398
00399 inline void AllocationTime::apply(hid_t dcpl) const {
00400     if (H5Pset_alloc_time(dcpl, _alloc_time) < 0) {
00401         HDF5ErrMapper::ToException<PropertyException>("Error setting allocation time");
00402     }
00403 }
00404
00405 inline H5D_alloc_time_t AllocationTime::getAllocationTime() {
00406     return _alloc_time;
00407 }
00408
00409 inline Caching::Caching(const DataSetCreateProps& dcpl) {
00410     if (H5Pget_chunk_cache(dcpl.getId(), &_numSlots, &_cacheSize, &_w0) < 0) {
00411         HDF5ErrMapper::ToException<PropertyException>("Error getting dataset cache parameters");
00412     }
00413 }
00414
00415 inline void Caching::apply(const hid_t hid) const {
00416     if (H5Pset_chunk_cache(hid, _numSlots, _cacheSize, _w0) < 0) {
00417         HDF5ErrMapper::ToException<PropertyException>("Error setting dataset cache parameters");
00418     }
00419 }
00420
00421 inline Caching::Caching(const size_t numSlots, const size_t cacheSize, const double w0)
00422 : _numSlots(numSlots)
00423 , _cacheSize(cacheSize)
00424 , _w0(w0) {}
00425
00426 inline size_t Caching::getNumSlots() const {
00427     return _numSlots;
00428 }
00429
00430 inline size_t Caching::getCacheSize() const {
00431     return _cacheSize;
00432 }
00433
00434 inline double Caching::getW0() const {
00435     return _w0;
00436 }
00437
00438 inline CreateIntermediateGroup::CreateIntermediateGroup(bool create)
00439 : _create(create) {}
00440
00441 inline CreateIntermediateGroup::CreateIntermediateGroup(const ObjectCreateProps& ocpl) {
00442     fromPropertyList(ocpl.getId());
00443 }
00444
00445
00446 inline void CreateIntermediateGroup::apply(const hid_t hid) const {

```

```

00447     if (H5Pset_create_intermediate_group(hid, _create ? 1 : 0) < 0) {
00448         HDF5ErrMapper::ToException<PropertyException>(
00449             "Error setting property for create intermediate groups");
00450     }
00451 }
00452
00453 inline CreateIntermediateGroup::CreateIntermediateGroup(const LinkCreateProps& lcpl) {
00454     fromPropertyList(lcpl.getId());
00455 }
00456
00457 inline void CreateIntermediateGroup::fromPropertyList(hid_t hid) {
00458     unsigned c_bool = 0;
00459     if (H5Pget_create_intermediate_group(hid, &c_bool) < 0) {
00460         HDF5ErrMapper::ToException<PropertyException>(
00461             "Error getting property for create intermediate groups");
00462     }
00463
00464     _create = bool(c_bool);
00465 }
00466
00467 inline bool CreateIntermediateGroup::isSet() const {
00468     return _create;
00469 }
00470
00471 #ifdef H5_HAVE_PARALLEL
00472 inline UseCollectiveIO::UseCollectiveIO(bool enable)
00473     : _enable(enable) {}
00474
00475 inline void UseCollectiveIO::apply(const hid_t hid) const {
00476     if (H5Pset_dxpl_mpio(hid, _enable ? H5FD_MPIO_COLLECTIVE : H5FD_MPIO_INDEPENDENT) < 0) {
00477         HDF5ErrMapper::ToException<PropertyException>("Error setting H5Pset_dxpl_mpio.");
00478     }
00479 }
00480
00481 inline UseCollectiveIO::UseCollectiveIO(const DataTransferProps& dxpl) {
00482     H5FD_mpio_xfer_t collective;
00483
00484     if (H5Pget_dxpl_mpio(dxpl.getId(), &collective) < 0) {
00485         HDF5ErrMapper::ToException<PropertyException>("Error getting H5Pset_dxpl_mpio.");
00486     }
00487
00488     if (collective != H5FD_MPIO_COLLECTIVE && collective != H5FD_MPIO_INDEPENDENT) {
00489         throw std::logic_error("H5Pget_dxpl_mpio returned something strange.");
00490     }
00491
00492     _enable = collective == H5FD_MPIO_COLLECTIVE;
00493 }
00494
00495 inline bool UseCollectiveIO::isCollective() const {
00496     return _enable;
00497 }
00498
00499 inline MpioNoCollectiveCause::MpioNoCollectiveCause(const DataTransferProps& dxpl) {
00500     if (H5Pget_mpio_no_collective_cause(dxpl.getId(), &_local_cause, &_global_cause) < 0) {
00501         HDF5ErrMapper::ToException<PropertyException>("Failed to check mpio_no_collective_cause.");
00502     }
00503 }
00504
00505 inline bool MpioNoCollectiveCause::wasCollective() const {
00506     return _local_cause == 0 && _global_cause == 0;
00507 }
00508
00509 inline uint32_t MpioNoCollectiveCause::getLocalCause() const {
00510     return _local_cause;
00511 }
00512
00513 inline uint32_t MpioNoCollectiveCause::getGlobalCause() const {
00514     return _global_cause;
00515 }
00516
00517 inline std::pair<uint32_t, uint32_t> MpioNoCollectiveCause::getCause() const {
00518     return {_local_cause, _global_cause};
00519 }
00520 #endif
00521
00522 inline LinkCreationOrder::LinkCreationOrder(const FileCreateProps& fcpl) {
00523     fromPropertyList(fcpl.getId());
00524 }
00525
00526 inline LinkCreationOrder::LinkCreationOrder(const GroupCreateProps& gcpl) {
00527     fromPropertyList(gcpl.getId());
00528 }
00529
00530 inline unsigned LinkCreationOrder::getFlags() const {
00531     return _flags;
00532 }
00533

```

```

00534 inline void LinkCreationOrder::apply(const hid_t hid) const {
00535     if (H5Pset_link_creation_order(hid, _flags) < 0) {
00536         HDF5ErrMapper::ToException<PropertyException>("Error setting LinkCreationOrder.");
00537     }
00538 }
00539
00540 inline void LinkCreationOrder::fromPropertyList(hid_t hid) {
00541     if (H5Pget_link_creation_order(hid, &_amp;_flags) < 0) {
00542         HDF5ErrMapper::ToException<PropertyException>("Error getting property for link creation order");
00543     }
00544 }
00545 }
00546 } // namespace HighFive

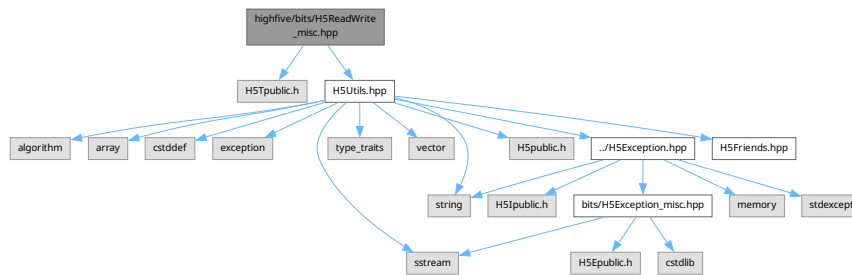
```

10.40 highfive/bits/H5ReadWrite_misc.hpp File Reference

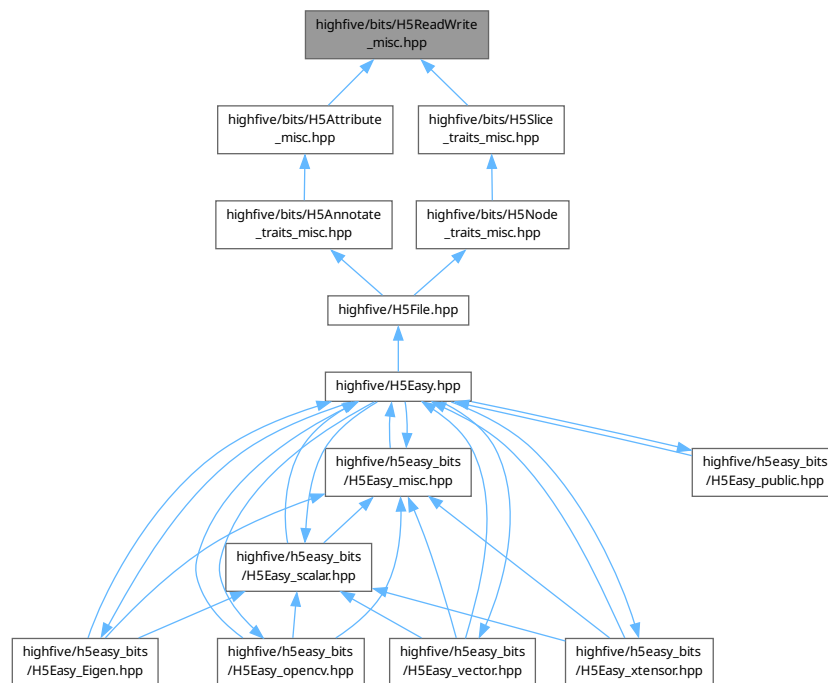
```
#include <H5Tpublic.h>
```

```
#include "H5Utils.hpp"
```

Include dependency graph for H5ReadWrite_misc.hpp:



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



Namespaces

- namespace [HighFive](#)

10.41 H5ReadWrite_misc.hpp

[Go to the documentation of this file.](#)

```

00001  /*
00002   * Copyright (c) 2020 Blue Brain Project
00003   *
00004   * Distributed under the Boost Software License, Version 1.0.
00005   * (See accompanying file LICENSE_1_0.txt or copy at
00006   * http://www.boost.org/LICENSE_1_0.txt)
00007   *
00008   */
00009 #pragma once
00010
00011 #include <H5Tpublic.h>
00012 #include "H5Utils.hpp"
00013
00014 namespace HighFive {
00015
00016 namespace details {
00017
00018 template <typename T>
00019 using unqualified_t = typename std::remove_const<typename std::remove_reference<T>::type>::type;
00020
00021 // Find the type of an eventual char array, otherwise void
00022 template <typename>
00023 struct type_char_array {
00024     using type = void;
00025 };
00026
00027 template <typename T>
00028 struct type_char_array<T*> {
00029     using type = typename std::conditional<std::is_same<unqualified_t<T>, char>::value,
00030                                           char*,
00031                                           typename type_char_array<T>::type>::type;
00032 };
00033
00034 template <typename T, std::size_t N>
00035 struct type_char_array<T[N]> {
00036     using type = typename std::conditional<std::is_same<unqualified_t<T>, char>::value,
00037                                           char[N],
00038                                           typename type_char_array<T>::type>::type;
00039 };
00040
00041 template <typename T>
00042 struct BufferInfo {
00043     using type_no_const = typename std::remove_const<T>::type;
00044     using elem_type = typename details::inspector<type_no_const>::base_type;
00045     using char_array_t = typename details::type_char_array<type_no_const>::type;
00046     static constexpr bool is_char_array = !std::is_same<char_array_t, void>::value;
00047
00048     enum Operation { read, write };
00049     const Operation op;
00050
00051     template <class F>
00052     BufferInfo(const DataType& dtype, F getName, Operation _op);
00053
00054     // member data for info depending on the destination dataset type
00055     const bool is_fixed_len_string;
00056     const size_t n_dimensions;
00057     const DataType data_type;
00058 };
00059
00060 // details implementation
00061 template <typename SrcStrT>
00062 struct string_type_checker {
00063     static DataType getDataType(const DataType&, const DataType&);
00064 };
00065
00066 template <>
00067 struct string_type_checker<void> {
00068     inline static DataType getDataType(const DataType& element_type, const DataType& dtype) {
00069         // TEMP. CHANGE: Ensure that the character set is properly configured to prevent
00070         // converter issues on HDF5 <=v1.12.0 when loading ASCII strings first.
00071         // See https://github.com/HDFGroup/hdf5/issues/544 for further information.
00072         if (H5Tget_class(element_type.getId()) == H5T_STRING &&
00073             H5Tget_cset(dtype.getId()) == H5T_CSET_ASCII) {

```

```

00074         H5Tset_cset(element_type.getId(), H5T_CSET_ASCII);
00075     }
00076     return element_type;
00077 }
00078 };
00079
00080 template <std::size_t FixedLen>
00081 struct string_type_checker<char[FixedLen]> {
00082     inline static DataType getDataType(const DataType& element_type, const DataType& dtype) {
00083         DataType return_type = (dtype.isFixedLenStr()) ? AtomicType<char[FixedLen]>()
00084             : element_type;
00085         // TEMP. CHANGE: See string_type_checker<void> definition
00086         if (H5Tget_cset(dtype.getId()) == H5T_CSET_ASCII) {
00087             H5Tset_cset(return_type.getId(), H5T_CSET_ASCII);
00088         }
00089         return return_type;
00090     }
00091 };
00092
00093 template <>
00094 struct string_type_checker<char*> {
00095     inline static DataType getDataType(const DataType&, const DataType& dtype) {
00096         if (dtype.isFixedLenStr())
00097             throw DataSetException("Can't output variable-length to fixed-length strings");
00098         // TEMP. CHANGE: See string_type_checker<void> definition
00099         DataType return_type = AtomicType<std::string>();
00100         if (H5Tget_cset(dtype.getId()) == H5T_CSET_ASCII) {
00101             H5Tset_cset(return_type.getId(), H5T_CSET_ASCII);
00102         }
00103         return return_type;
00104     }
00105 };
00106
00107 template <typename T>
00108 template <class F>
00109 BufferInfo<T>::BufferInfo(const DataType& dtype, F getName, Operation _op)
00110     : op(_op)
00111     , is_fixed_len_string(dtype.isFixedLenStr())
00112     , // In case we are using Fixed-len strings we need to subtract one dimension
00113     , n_dimensions(details::inspector<type_no_const>::recursive_ndim -
00114         ((is_fixed_len_string && is_char_array) ? 1 : 0))
00115     , data_type(
00116         string_type_checker<char_array_t>::getDataType(create_datatype<elem_type>(), dtype)) {
00117     if (is_fixed_len_string && std::is_same<elem_type, std::string>::value) {
00118         throw DataSetException(
00119             "Can't output std::string as fixed-length. "
00120             "Use raw arrays or FixedLenStringArray");
00121     }
00122     // We warn. In case they are really not convertible an exception will rise on read/write
00123     if (dtype.getClass() != data_type.getClass()) {
00124         HIGHFIVE_LOG_WARN(getName() + "\": data and hdf5 dataset have different types: " +
00125             data_type.string() + " -> " + dtype.string());
00126     } else if ((dtype.getClass() & data_type.getClass()) == DataTypeClass::Float) {
00127         HIGHFIVE_LOG_WARN_IF(
00128             (op == read) && (dtype.getSize() > data_type.getSize()),
00129             getName() + "\": hdf5 dataset has higher floating point precision than data on read: " +
00130             dtype.string() + " -> " + data_type.string());
00131     }
00132     HIGHFIVE_LOG_WARN_IF(
00133         (op == write) && (dtype.getSize() < data_type.getSize()),
00134         getName() +
00135         "\": data has higher floating point precision than hdf5 dataset on write: " +
00136         data_type.string() + " -> " + dtype.string());
00137     }
00138 }
00139
00140 } // namespace details
00141
00142 } // namespace HighFive

```

10.42 highfive/bits/H5Reference_misc.hpp File Reference

```

#include <string>
#include <H5Ppublic.h>
#include "H5Utils.hpp"
#include "../H5Object.hpp"

```


Namespaces

- namespace [HighFive](#)

10.43 H5Reference_misc.hpp

[Go to the documentation of this file.](#)

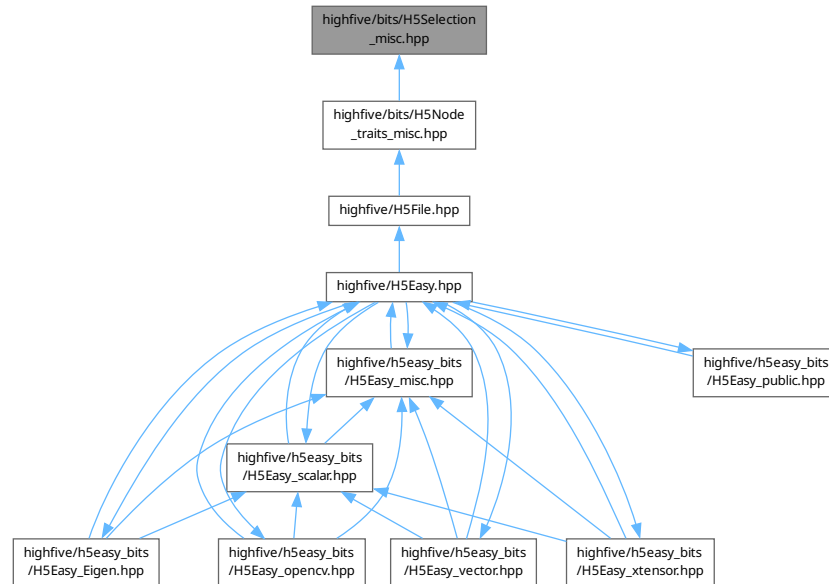
```

00001  /*
00002  *   Copyright (c), 2020, EPFL - Blue Brain Project
00003  *
00004  *   Distributed under the Boost Software License, Version 1.0.
00005  *   (See accompanying file LICENSE_1_0.txt or copy at
00006  *    http://www.boost.org/LICENSE_1_0.txt)
00007  *
00008  */
00009
00010 #pragma once
00011
00012 #include <string>
00013 #include <H5Ppublic.h>
00014
00015 #include "H5Utils.hpp"
00016
00017 #include "../H5Object.hpp"
00018
00019 namespace HighFive {
00020
00021 inline Reference::Reference(const Object& location, const Object& object)
00022     : parent_id(location.getId()) {
00023     obj_name = details::get_name(
00024         [&](char* buffer, size_t length) { return H5Iget_name(object.getId(), buffer, length); });
00025 }
00026
00027 inline void Reference::create_ref(hobj_ref_t* refptr) const {
00028     if (H5Rcreate(refptr, parent_id, obj_name.c_str(), H5R_OBJECT, -1) < 0) {
00029         HDF5ErrMapper::ToException<ReferenceException>(
00030             std::string("Unable to create the reference for \"" + obj_name + "\"");
00031     )
00032 }
00033
00034 inline ObjectType Reference::getType(const Object& location) const {
00035     return get_ref(location).getType();
00036 }
00037
00038 template <typename T>
00039 inline T Reference::dereference(const Object& location) const {
00040     static_assert(std::is_same<DataSet, T::value || std::is_same<Group, T::value,
00041         "We can only (de)reference HighFive::Group or HighFive::DataSet");
00042     auto obj = get_ref(location);
00043     if (obj.getType() != T::type) {
00044         HDF5ErrMapper::ToException<ReferenceException>("Trying to dereference the wrong type");
00045     }
00046 #if defined __GNUC__ && __GNUC__ < 9
00047     return std::move(obj);
00048 #else
00049     return obj;
00050 #endif
00051 }
00052
00053 inline Object Reference::get_ref(const Object& location) const {
00054     hid_t res;
00055     #if (H5Rdereference_vers == 2)
00056     if ((res = H5Rdereference(location.getId(), H5P_DEFAULT, H5R_OBJECT, &href)) < 0) {
00057         HDF5ErrMapper::ToException<ReferenceException>("Unable to dereference.");
00058     }
00059 #else
00060     if ((res = H5Rdereference(location.getId(), H5R_OBJECT, &href)) < 0) {
00061         HDF5ErrMapper::ToException<ReferenceException>("Unable to dereference.");
00062     }
00063 #endif
00064     return Object(res);
00065 }
00066
00067 } // namespace HighFive

```

10.44 highfive/bits/H5Selection_misc.hpp File Reference

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



Namespaces

- namespace [HighFive](#)

10.45 H5Selection_misc.hpp

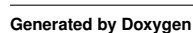
[Go to the documentation of this file.](#)

```

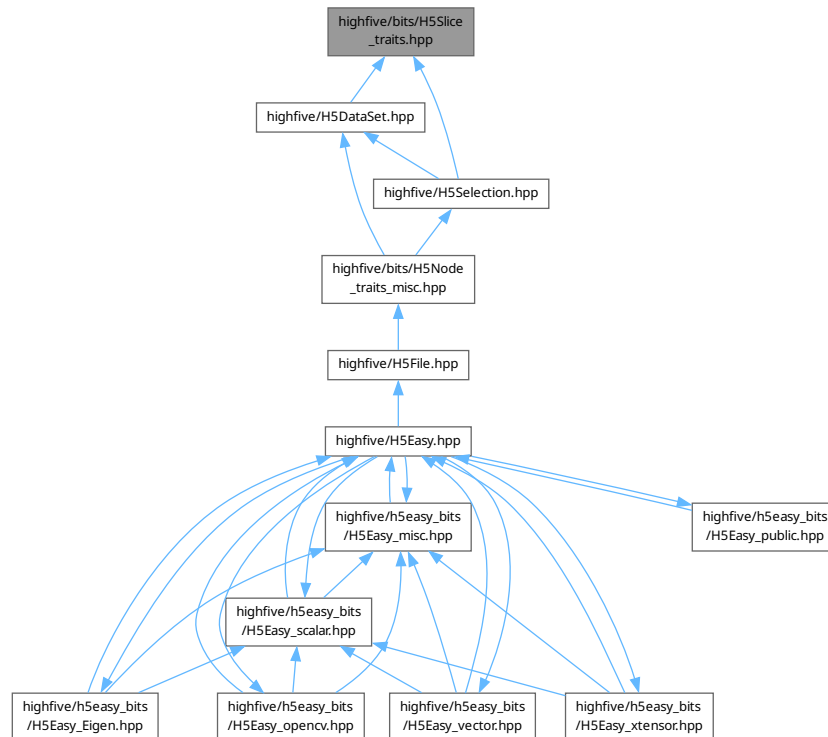
00001 /*
00002  * Copyright (c), 2017, Adrien Devresse <adrien.devresse@epfl.ch>
00003  *
00004  * Distributed under the Boost Software License, Version 1.0.
00005  * (See accompanying file LICENSE_1_0.txt or copy at
00006  *  http://www.boost.org/LICENSE_1_0.txt)
00007  *
00008  */
00009 #pragma once
00010
00011 namespace HighFive {
00012
00013 inline Selection::Selection(const DataSpace& memspace,
00014                             const DataSpace& file_space,
00015                             const DataSet& set)
00016     : _mem_space(memspace)
00017     , _file_space(file_space)
00018     , _set(set) {}
00019
00020 inline DataSpace Selection::getSpace() const noexcept {
00021     return _file_space;
00022 }
00023
00024 inline DataSpace Selection::getMemSpace() const noexcept {
00025     return _mem_space;
00026 }
00027
00028 inline DataSet& Selection::getDataset() noexcept {
00029     return _set;
  
```


10.46 highfive/bits/H5Slice_traits.hpp File Reference

Include dependency graph for H5Slice_traits.hpp:



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



Classes

- class [HighFive::ElementSet](#)
- struct [HighFive::RegularHyperSlab](#)
- class [HighFive::HyperSlab](#)
- class [HighFive::SliceTraits< Derivate >](#)

Namespaces

- namespace [HighFive](#)

Functions

- `std::vector< hsize_t > HighFive::toHDF5SizeVector (const std::vector< size_t > &from)`
- `std::vector< size_t > HighFive::toSTLSizeVector (const std::vector< hsize_t > &from)`

10.47 H5Slice_traits.hpp

[Go to the documentation of this file.](#)

```

00001  /*
00002   * Copyright (c), 2017, Adrien Devresse <adrien.devresse@epfl.ch>
00003   *
00004   * Distributed under the Boost Software License, Version 1.0.
00005   * (See accompanying file LICENSE_1_0.txt or copy at
00006   * http://www.boost.org/LICENSE_1_0.txt)
00007   *
00008   */
00009 #pragma once
00010
00011 #include <cstdlib>
00012 #include <vector>
00013
00014 #include "H5_definitions.hpp"
00015 #include "H5Utils.hpp"
00016
00017 #include "../H5PropertyList.hpp"
00018
00019 namespace HighFive {
00020
00021 class ElementSet {
00022 public:
00023     explicit ElementSet(std::initializer_list<std::size_t> list);
00024     explicit ElementSet(std::initializer_list<std::vector<std::size_t> list);
00025     explicit ElementSet(const std::vector<std::size_t>& element_ids);
00026     explicit ElementSet(const std::vector<std::vector<std::size_t>& element_ids);
00027
00028 private:
00029     std::vector<std::size_t> _ids;
00030
00031     template <typename Derivate>
00032     friend class SliceTraits;
00033 };
00034
00035 namespace detail {
00036
00037 template <class To, class From>
00038 inline std::vector<To> convertSizeVector(const std::vector<From>& from) {
00039     std::vector<To> to(from.size());
00040     std::copy(from.cbegin(), from.cend(), to.begin());
00041
00042     return to;
00043 }
00044
00045 // namespace detail
00046
00047 inline std::vector<hsize_t> toHDF5SizeVector(const std::vector<size_t>& from) {
00048     return detail::convertSizeVector<hsize_t>(from);
00049 }
00050
00051 inline std::vector<size_t> toSTLSizeVector(const std::vector<hsize_t>& from) {
00052     return detail::convertSizeVector<size_t>(from);
00053 }
00054
00055 struct RegularHyperSlab {
00056     RegularHyperSlab() = default;
00057
00058     RegularHyperSlab(std::vector<size_t> offset_,
00059                     std::vector<size_t> count_ = {},
00060                     std::vector<size_t> stride_ = {},
00061                     std::vector<size_t> block_ = {})
00062         : offset(toHDF5SizeVector(offset_))
00063         , count(toHDF5SizeVector(count_))
00064         , stride(toHDF5SizeVector(stride_))
00065         , block(toHDF5SizeVector(block_)) {}
00066
00067     static RegularHyperSlab fromHDF5Sizes(std::vector<hsize_t> offset_,
00068                                           std::vector<hsize_t> count_ = {},
00069                                           std::vector<hsize_t> stride_ = {},
00070                                           std::vector<hsize_t> block_ = {}) {
00071         RegularHyperSlab slab;
00072         slab.offset = offset_;
00073         slab.count = count_;
00074         slab.stride = stride_;
00075         slab.block = block_;
00076
00077         return slab;
00078     }
00079
00080     size_t rank() const {
00081         return std::max(std::max(offset.size(), count.size()),
00082                         std::max(stride.size(), block.size()));
00083     }
00084 }
00085
00086 }

```

```

00101
00102     std::vector<size_t> packedDims() const {
00103         auto n_dims = rank();
00104         auto dims = std::vector<size_t>(n_dims, 0);
00105
00106         for (size_t i = 0; i < n_dims; ++i) {
00107             dims[i] = count[i] * (block.empty() ? 1 : block[i]);
00108         }
00109
00110         return dims;
00111     }
00112
00113     std::vector<hsize_t> offset;
00114     std::vector<hsize_t> count;
00115     std::vector<hsize_t> stride;
00116     std::vector<hsize_t> block;
00117 };
00118
00119 class HyperSlab {
00120 public:
00121     HyperSlab() {
00122         selects.emplace_back(RegularHyperSlab{}, Op::None);
00123     };
00124
00125     explicit HyperSlab(const RegularHyperSlab& sel) {
00126         selects.emplace_back(sel, Op::Set);
00127     }
00128
00129     HyperSlab operator|(const RegularHyperSlab& sel) const {
00130         auto ret = *this;
00131         ret |= sel;
00132         return ret;
00133     }
00134
00135     HyperSlab& operator|=(const RegularHyperSlab& sel) {
00136         selects.emplace_back(sel, Op::Or);
00137         return *this;
00138     }
00139
00140     HyperSlab operator&(const RegularHyperSlab& sel) const {
00141         auto ret = *this;
00142         ret &= sel;
00143         return ret;
00144     }
00145
00146     HyperSlab& operator&=(const RegularHyperSlab& sel) {
00147         selects.emplace_back(sel, Op::And);
00148         return *this;
00149     }
00150
00151     HyperSlab operator^(const RegularHyperSlab& sel) const {
00152         auto ret = *this;
00153         ret ^= sel;
00154         return ret;
00155     }
00156
00157     HyperSlab& operator^=(const RegularHyperSlab& sel) {
00158         selects.emplace_back(sel, Op::Xor);
00159         return *this;
00160     }
00161
00162     HyperSlab& notA(const RegularHyperSlab& sel) {
00163         selects.emplace_back(sel, Op::NotA);
00164         return *this;
00165     }
00166
00167     HyperSlab& notB(const RegularHyperSlab& sel) {
00168         selects.emplace_back(sel, Op::NotB);
00169         return *this;
00170     }
00171
00172     DataSpace apply(const DataSpace& space_) const {
00173         auto space = space_.clone();
00174         for (const auto& sel: selects) {
00175             if (sel.op == Op::None) {
00176                 H5Sselect_none(space.getId());
00177             } else {
00178                 auto error_code =
00179                     H5Sselect_hyperslab(space.getId(),
00180                                         convert(sel.op),
00181                                         sel.offset.empty() ? nullptr : sel.offset.data(),
00182                                         sel.stride.empty() ? nullptr : sel.stride.data(),
00183                                         sel.count.empty() ? nullptr : sel.count.data(),
00184                                         sel.block.empty() ? nullptr : sel.block.data());
00185
00186                 if (error_code < 0) {
00187                     HDF5ErrMapper::ToException<DataSpaceException>("Unable to select hyperslab");
00188                 }
00189             }
00190         }
00191     }
00192 };

```

```

00189         }
00190     }
00191 }
00192     return space;
00193 }
00194
00195 private:
00196     enum class Op {
00197         Noop,
00198         Set,
00199         Or,
00200         And,
00201         Xor,
00202         NotB,
00203         NotA,
00204         Append,
00205         Prepend,
00206         Invalid,
00207         None,
00208     };
00209
00210     H5S_seloper_t convert(Op op) const {
00211         switch (op) {
00212             case Op::Noop:
00213                 return H5S_SELECT_NOOP;
00214             case Op::Set:
00215                 return H5S_SELECT_SET;
00216             case Op::Or:
00217                 return H5S_SELECT_OR;
00218             case Op::And:
00219                 return H5S_SELECT_AND;
00220             case Op::Xor:
00221                 return H5S_SELECT_XOR;
00222             case Op::NotB:
00223                 return H5S_SELECT_NOTB;
00224             case Op::NotA:
00225                 return H5S_SELECT_NOTA;
00226             case Op::Append:
00227                 return H5S_SELECT_APPEND;
00228             case Op::Prepend:
00229                 return H5S_SELECT_PREPEND;
00230             case Op::Invalid:
00231                 return H5S_SELECT_INVALID;
00232             default:
00233                 throw DataSpaceException("Invalid HyperSlab operation.");
00234         }
00235     }
00236
00237     struct Select_: public RegularHyperSlab {
00238         Select_(const RegularHyperSlab& sel, Op op_)
00239             : RegularHyperSlab(sel)
00240             , op(op_) {}
00241
00242         Op op;
00243     };
00244
00245     std::vector<Select_> selects;
00246 };
00247
00248 template <typename Derivate>
00249 class SliceTraits {
00250 public:
00251     Selection select(const HyperSlab& hyperslab) const;
00252
00253     Selection select(const std::vector<size_t>& offset,
00254                     const std::vector<size_t>& count,
00255                     const std::vector<size_t>& stride = {},
00256                     const std::vector<size_t>& block = {}) const;
00257
00258     Selection select(const std::vector<size_t>& columns) const;
00259
00260     Selection select(const ElementSet& elements) const;
00261
00262     template <typename T>
00263     T read(const DataTransferProps& xfer_props = DataTransferProps()) const;
00264
00265     template <typename T>
00266     void read(T& array, const DataTransferProps& xfer_props = DataTransferProps()) const;
00267
00268     template <typename T>
00269     void read(T* array,
00270              const DataType& dtype = DataType(),
00271              const DataTransferProps& xfer_props = DataTransferProps()) const;
00272
00273     template <typename T>
00274     void write(const T& buffer, const DataTransferProps& xfer_props = DataTransferProps());
00275
00276 };

```

```

00334     template <typename T>
00335     void write_raw(const T* buffer,
00336                   const DataType& dtype = DataType(),
00337                   const DataTransferProps& xfer_props = DataTransferProps());
00338
00339     protected:
00340     inline Selection select_impl(const HyperSlab& hyperslab, const DataSpace& memspace) const;
00341 };
00342
00343 } // namespace HighFive

```

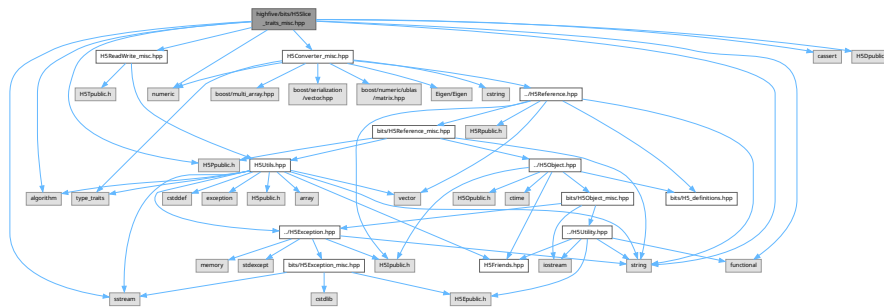
10.48 highfive/bits/H5Slice_traits_misc.hpp File Reference

```

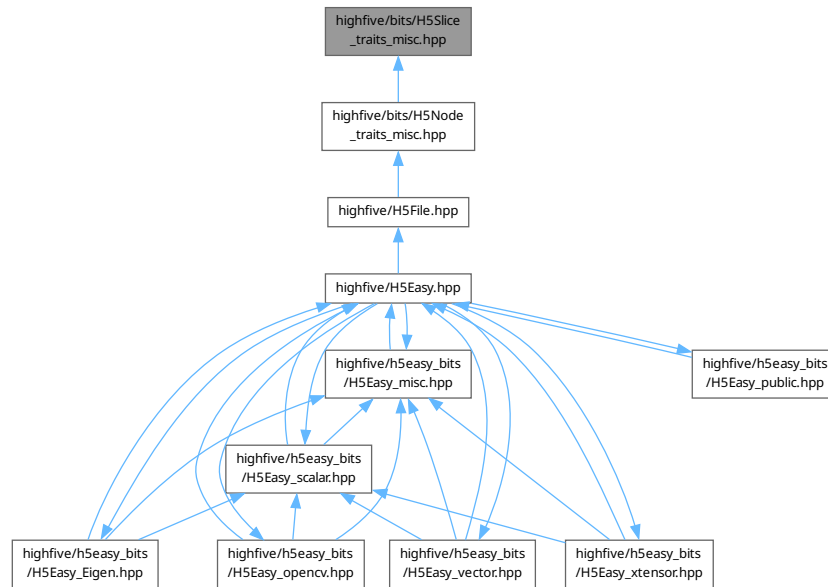
#include <algorithm>
#include <cassert>
#include <functional>
#include <numeric>
#include <sstream>
#include <string>
#include <H5Dpublic.h>
#include <H5Ppublic.h>
#include "H5ReadWrite_misc.hpp"
#include "H5Converter_misc.hpp"

```

Include dependency graph for H5Slice_traits_misc.hpp:



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



Namespaces

- namespace [HighFive](#)

10.49 H5Slice_traits_misc.hpp

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c), 2017, Adrien Devresse <adrien.devresse@epfl.ch>
00003  *
00004  * Distributed under the Boost Software License, Version 1.0.
00005  * (See accompanying file LICENSE_1_0.txt or copy at
00006  * http://www.boost.org/LICENSE_1_0.txt)
00007  *
00008  */
00009 #pragma once
00010
00011 #include <algorithm>
00012 #include <cassert>
00013 #include <functional>
00014 #include <numeric>
00015 #include <sstream>
00016 #include <string>
00017
00018 #include <H5Dpublic.h>
00019 #include <H5Ppublic.h>
00020
00021 #include "H5ReadWrite_misc.hpp"
00022 #include "H5Converter_misc.hpp"
00023
00024 namespace HighFive {
00025
00026 namespace details {
00027
00028 // map the correct reference to the dataset depending of the layout
00029 // dataset -> itself
00030 // subselection -> parent dataset
00031 inline const DataSet& get_dataset(const Selection& sel) {
00032     return sel.getDataset();
00033 }

```

```

00034
00035 inline const DataSet& get_dataset(const DataSet& ds) {
00036     return ds;
00037 }
00038
00039 // map the correct memspace identifier depending of the layout
00040 // dataset -> entire memspace
00041 // selection -> resolve space id
00042 inline hid_t get_memspace_id(const Selection& ptr) {
00043     return ptr.getMemSpace().getId();
00044 }
00045
00046 inline hid_t get_memspace_id(const DataSet&) {
00047     return H5S_ALL;
00048 }
00049 // namespace details
00050
00051 inline ElementSet::ElementSet(std::initializer_list<std::size_t> list)
00052     : _ids(list) {}
00053
00054 inline ElementSet::ElementSet(std::initializer_list<std::vector<std::size_t>> list)
00055     : ElementSet(std::vector<std::vector<std::size_t>>(list)) {}
00056
00057 inline ElementSet::ElementSet(const std::vector<std::size_t>& element_ids)
00058     : _ids(element_ids) {}
00059
00060 inline ElementSet::ElementSet(const std::vector<std::vector<std::size_t>>& element_ids) {
00061     for (const auto& vec: element_ids) {
00062         std::copy(vec.begin(), vec.end(), std::back_inserter(_ids));
00063     }
00064 }
00065
00066 template <typename Derivate>
00067 inline Selection SliceTraits<Derivate>::select_impl(const HyperSlab& hyperslab,
00068                                                     const DataSpace& memspace) const {
00069     // Note: The current limitation are that memspace must describe a
00070     //        packed memspace.
00071     //
00072     //        The reason for this is that we're unable to unpack general
00073     //        hyperslabs when the memory is not contiguous, e.g.
00074     //        `std::vector<std::vector<double>>`.
00075     const auto& slice = static_cast<const Derivate&>(*this);
00076     auto filespace = hyperslab.apply(slice.getSpace());
00077
00078     return detail::make_selection(memspace, filespace, details::get_dataset(slice));
00079 }
00080
00081 template <typename Derivate>
00082 inline Selection SliceTraits<Derivate>::select(const HyperSlab& hyper_slab) const {
00083     const auto& slice = static_cast<const Derivate&>(*this);
00084     auto filespace = slice.getSpace();
00085     filespace = hyper_slab.apply(filespace);
00086
00087     auto n_elements = H5Sget_select_npoints(filespace.getId());
00088     auto memspace = DataSpace(std::array<size_t, 1>{size_t(n_elements)});
00089
00090     return detail::make_selection(memspace, filespace, details::get_dataset(slice));
00091 }
00092
00093
00094 template <typename Derivate>
00095 inline Selection SliceTraits<Derivate>::select(const std::vector<size_t>& offset,
00096                                               const std::vector<size_t>& count,
00097                                               const std::vector<size_t>& stride,
00098                                               const std::vector<size_t>& block) const {
00099     auto slab = HyperSlab(RegularHyperSlab(offset, count, stride, block));
00100     auto memspace = DataSpace(count);
00101     return select_impl(slab, memspace);
00102 }
00103
00104 template <typename Derivate>
00105 inline Selection SliceTraits<Derivate>::select(const std::vector<size_t>& columns) const {
00106     const auto& slice = static_cast<const Derivate&>(*this);
00107     const DataSpace& space = slice.getSpace();
00108     std::vector<size_t> dims = space.getDimensions();
00109
00110     std::vector<size_t> counts = dims;
00111     counts.back() = 1;
00112
00113     std::vector<size_t> offsets(dims.size(), 0);
00114
00115     HyperSlab slab;
00116     for (const auto& column: columns) {
00117         offsets.back() = column;
00118         slab |= RegularHyperSlab(offsets, counts);
00119     }
00120 }

```



```

00121     std::vector<size_t> memdims = dims;
00122     memdims.back() = columns.size();
00123
00124     return select_impl(slab, DataSpace(memdims));
00125 }
00126
00127 template <typename Derivate>
00128 inline Selection SliceTraits<Derivate>::select(const ElementSet& elements) const {
00129     const auto& slice = static_cast<const Derivate&>(*this);
00130     const hsize_t* data = nullptr;
00131     const DataSpace space = slice.getSpace().clone();
00132     const std::size_t length = elements._ids.size();
00133     if (length % space.getNumberDimensions() != 0) {
00134         throw DataSpaceException(
00135             "Number of coordinates in elements picking "
00136             "should be a multiple of the dimensions.");
00137     }
00138     const std::size_t num_elements = length / space.getNumberDimensions();
00139     std::vector<hsize_t> raw_elements;
00140
00141     // optimised at compile time
00142     // switch for data conversion on 32bits platforms
00143     if (std::is_same<std::size_t, hsize_t>::value) {
00144         // `if constexpr` can't be used, thus a reinterpret_cast is needed.
00145         data = reinterpret_cast<const hsize_t*>(&(elements._ids[0]));
00146     } else {
00147         raw_elements.resize(length);
00148         std::copy(elements._ids.begin(), elements._ids.end(), raw_elements.begin());
00149         data = raw_elements.data();
00150     }
00151
00152     if (H5Sselect_elements(space.getId(), H5S_SELECT_SET, num_elements, data) < 0) {
00153         HDF5ErrMapper::ToException<DataSpaceException>("Unable to select elements");
00154     }
00155
00156     return detail::make_selection(DataSpace(num_elements), space, details::get_dataset(slice));
00157 }
00158
00159
00160 template <typename Derivate>
00161 template <typename T>
00162 inline T SliceTraits<Derivate>::read(const DataTransferProps& xfer_props) const {
00163     T array;
00164     read(array, xfer_props);
00165     return array;
00166 }
00167
00168
00169 template <typename Derivate>
00170 template <typename T>
00171 inline void SliceTraits<Derivate>::read(T& array, const DataTransferProps& xfer_props) const {
00172     const auto& slice = static_cast<const Derivate&>(*this);
00173     const DataSpace& mem_space = slice.getMemSpace();
00174
00175     const details::BufferInfo<T> buffer_info(
00176         slice.getDataType(),
00177         [&slice]() -> std::string { return details::get_dataset(slice).getPath(); },
00178         details::BufferInfo<T>::Operation::read);
00179
00180     if (!details::checkDimensions(mem_space, buffer_info.n_dimensions)) {
00181         std::ostringstream ss;
00182         ss << "Impossible to read DataSet of dimensions " << mem_space.getNumberDimensions()
00183             << " into arrays of dimensions " << buffer_info.n_dimensions;
00184         throw DataSpaceException(ss.str());
00185     }
00186     auto dims = mem_space.getDimensions();
00187
00188     if (mem_space.getElementCount() == 0) {
00189         auto effective_dims = details::squeezeDimensions(dims,
00190                                                         details::inspector<T>::recursive_ndim);
00191
00192         details::inspector<T>::prepare(array, effective_dims);
00193         return;
00194     }
00195
00196     auto r = details::data_converter::get_reader<T>(dims, array);
00197     read(r.get_pointer(), buffer_info.data_type, xfer_props);
00198     // re-arrange results
00199     r.unserialize();
00200     auto t = create_datatype<typename details::inspector<T>::base_type>();
00201     auto c = t.getClass();
00202     if (c == DataTypeClass::VarLen || t.isVariableStr()) {
00203 #if H5_VERSION_GE(1, 12, 0)
00204         // This one have been created in 1.12.0
00205         (void) H5Treclaim(t.getId(), mem_space.getId(), xfer_props.getId(), r.get_pointer());
00206 #else
00207         // This one is deprecated since 1.12.0

```

```

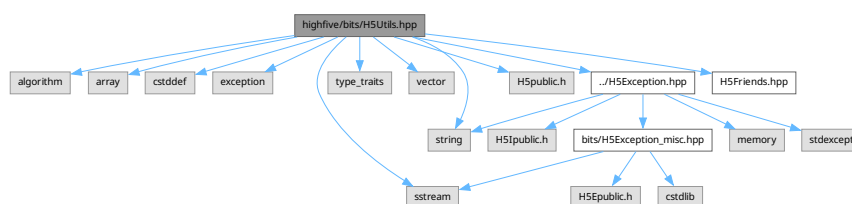
00208         (void) H5Dvlen_reclaim(t.getId(), mem_space.getId(), xfer_props.getId(), r.get_pointer());
00209     #endif
00210     }
00211 }
00212
00213
00214 template <typename Derivate>
00215 template <typename T>
00216 inline void SliceTraits<Derivate>::read(T* array,
00217                                         const DataType& dtype,
00218                                         const DataTransferProps& xfer_props) const {
00219     static_assert(!std::is_const<T>::value,
00220                 "read() requires a non-const structure to read data into");
00221     const auto& slice = static_cast<const Derivate&>(*this);
00222     using element_type = typename details::inspector<T>::base_type;
00223
00224     // Auto-detect mem datatype if not provided
00225     const DataType& mem_datatype = dtype.empty() ? create_and_check_datatype<element_type>()
00226                                                  : dtype;
00227
00228     if (H5Dread(details::get_dataset(slice).getId(),
00229                mem_datatype.getId(),
00230                details::get_memspace_id(slice),
00231                slice.getSpace().getId(),
00232                xfer_props.getId(),
00233                static_cast<void*>(array)) < 0) {
00234         HDF5ErrMapper::ToException<DataSetException>("Error during HDF5 Read.");
00235     }
00236 }
00237
00238
00239 template <typename Derivate>
00240 template <typename T>
00241 inline void SliceTraits<Derivate>::write(const T& buffer, const DataTransferProps& xfer_props) {
00242     const auto& slice = static_cast<const Derivate&>(*this);
00243     const DataSpace& mem_space = slice.getMemSpace();
00244
00245     if (mem_space.getElementCount() == 0) {
00246         return;
00247     }
00248
00249     const details::BufferInfo<T> buffer_info(
00250         slice.getDataType(),
00251         [&slice]() -> std::string { return details::get_dataset(slice).getPath(); },
00252         details::BufferInfo<T>::Operation::write);
00253
00254     if (!details::checkDimensions(mem_space, buffer_info.n_dimensions)) {
00255         std::ostringstream ss;
00256         ss << "Impossible to write buffer of dimensions "
00257             << details::format_vector(mem_space.getDimensions())
00258             << " into dataset with n = " << buffer_info.n_dimensions << " dimensions.";
00259         throw DataSpaceException(ss.str());
00260     }
00261     auto w = details::data_converter::serialize<T>(buffer);
00262     write_raw(w.get_pointer(), buffer_info.data_type, xfer_props);
00263 }
00264
00265
00266 template <typename Derivate>
00267 template <typename T>
00268 inline void SliceTraits<Derivate>::write_raw(const T* buffer,
00269                                              const DataType& dtype,
00270                                              const DataTransferProps& xfer_props) {
00271     using element_type = typename details::inspector<T>::base_type;
00272     const auto& slice = static_cast<const Derivate&>(*this);
00273     const auto& mem_datatype = dtype.empty() ? create_and_check_datatype<element_type>() : dtype;
00274
00275     if (H5Dwrite(details::get_dataset(slice).getId(),
00276                 mem_datatype.getId(),
00277                 details::get_memspace_id(slice),
00278                 slice.getSpace().getId(),
00279                 xfer_props.getId(),
00280                 static_cast<const void*>(buffer)) < 0) {
00281         HDF5ErrMapper::ToException<DataSetException>("Error during HDF5 Write: ");
00282     }
00283 }
00284
00285
00286 } // namespace HighFive

```

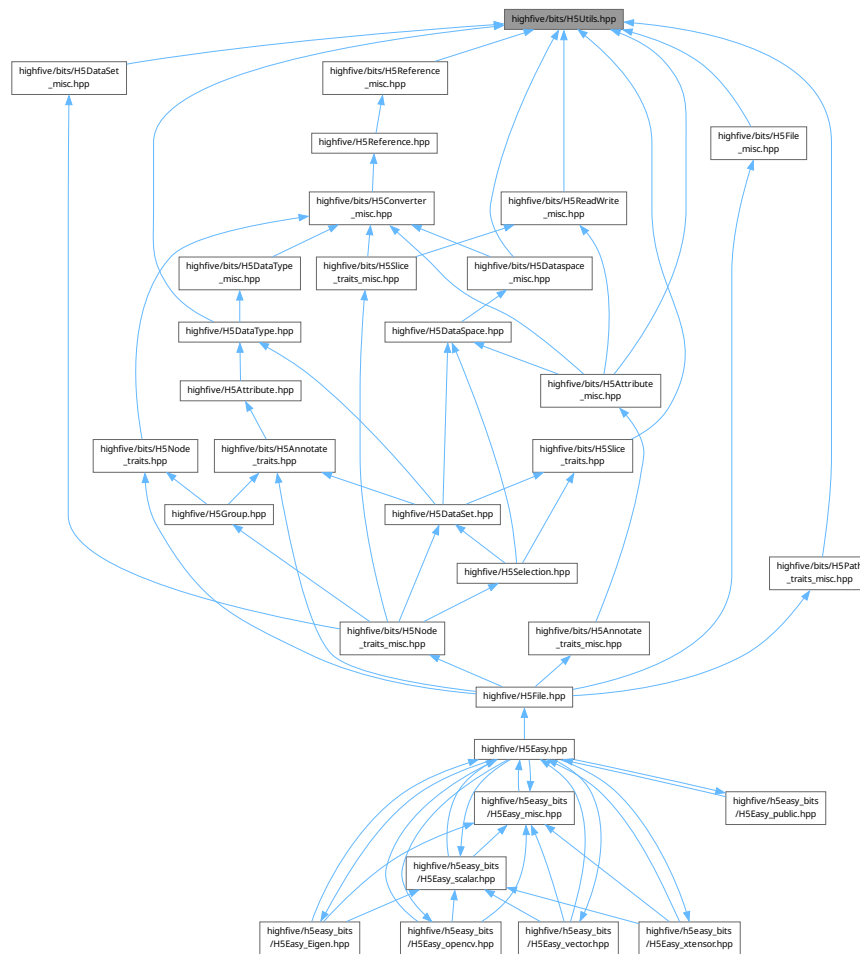
10.50 highfive/bits/H5Utils.hpp File Reference

```
#include <algorithm>
#include <array>
#include <cstdint>
#include <exception>
#include <string>
#include <type_traits>
#include <vector>
#include <sstream>
#include <H5public.h>
#include "../H5Exception.hpp"
#include "H5Friends.hpp"
```

Include dependency graph for H5Utils.hpp:



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



Namespaces

- namespace [HighFive](#)

10.51 H5Utils.hpp

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c), 2017, Adrien Devresse <adrien.devresse@epfl.ch>
00003  *
00004  * Distributed under the Boost Software License, Version 1.0.
00005  * (See accompanying file LICENSE_1_0.txt or copy at
00006  * http://www.boost.org/LICENSE_1_0.txt)
00007  *
00008  */
00009 #pragma once
00010
00011 // internal utilities functions
00012 #include <algorithm>
00013 #include <array>
00014 #include <cstdint> // __GLIBCXX__
00015 #include <exception>
00016 #include <string>
00017 #include <type_traits>

```

```

00018 #include <vector>
00019 #include <sstream>
00020
00021 #include <H5public.h>
00022
00023 #include "../H5Exception.hpp"
00024 #include "H5Friends.hpp"
00025
00026 namespace HighFive {
00027
00028 // If ever used, recognize dimensions of FixedLenStringArray
00029 template <std::size_t N>
00030 class FixedLenStringArray;
00031
00032 namespace details {
00033 // converter function for hsize_t -> size_t when hsize_t != size_t
00034 template <typename Size>
00035 inline std::vector<std::size_t> to_vector_size_t(const std::vector<Size>& vec) {
00036     static_assert(std::is_same<Size, std::size_t>::value == false,
00037         " hsize_t != size_t mandatory here");
00038     std::vector<size_t> res(vec.size());
00039     std::transform(vec.cbegin(), vec.cend(), res.begin(), [](Size e) {
00040         return static_cast<size_t>(e);
00041     });
00042     return res;
00043 }
00044
00045 // converter function for hsize_t -> size_t when size_t == hsize_t
00046 inline std::vector<std::size_t> to_vector_size_t(const std::vector<std::size_t>& vec) {
00047     return vec;
00048 }
00049
00050 // read name from a H5 object using the specified function
00051 template <typename T>
00052 inline std::string get_name(T fct) {
00053     const size_t maxLength = 255;
00054     char buffer[maxLength + 1];
00055     ssize_t retcode = fct(buffer, static_cast<hsize_t>(maxLength) + 1);
00056     if (retcode < 0) {
00057         HDF5ErrMapper::ToException<GroupException>("Error accessing object name");
00058     }
00059     const size_t length = static_cast<std::size_t>(retcode);
00060     if (length <= maxLength) {
00061         return std::string(buffer, length);
00062     }
00063     std::vector<char> bigBuffer(length + 1, 0);
00064     fct(bigBuffer.data(), length + 1);
00065     return std::string(bigBuffer.data(), length);
00066 }
00067
00068 template <class Container>
00069 inline std::string format_vector(const Container& container) {
00070     auto sout = std::stringstream{};
00071
00072     sout << "[ ";
00073     for (size_t i = 0; i < container.size(); ++i) {
00074         sout << container[i] << (i == container.size() - 1 ? "" : ", ");
00075     }
00076     sout << "]";
00077
00078     return sout.str();
00079 }
00080
00081 } // namespace details
00082 } // namespace HighFive

```

10.52 highfive/H5Attribute.hpp File Reference

```

#include <vector>
#include <H5Apublic.h>
#include "H5DataType.hpp"
#include "H5Object.hpp"
#include "bits/H5Friends.hpp"
#include "bits/H5Path_traits.hpp"

```


Namespaces

- namespace [HighFive](#)

10.53 H5Attribute.hpp

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c), 2017, Ali Can Demiralp <ali.demiralp@rwth-aachen.de>
00003  *
00004  * Distributed under the Boost Software License, Version 1.0.
00005  * (See accompanying file LICENSE_1_0.txt or copy at
00006  * http://www.boost.org/LICENSE_1_0.txt)
00007  *
00008  */
00009 #pragma once
00010
00011 #include <vector>
00012
00013 #include <H5Apublic.h>
00014
00015 #include "H5DataType.hpp"
00016 #include "H5Object.hpp"
00017 #include "bits/H5Friends.hpp"
00018 #include "bits/H5Path_traits.hpp"
00019
00020 namespace HighFive {
00021     class DataSpace;
00022
00023     namespace detail {
00024
00040     Attribute make_attribute(hid_t hid);
00041     } // namespace detail
00042
00046     class Attribute: public Object, public PathTraits<Attribute> {
00047     public:
00048         const static ObjectType type = ObjectType::Attribute;
00049
00053         std::string getName() const;
00054
00055         size_t getStorageSize() const;
00056
00061         DataType getDataType() const;
00062
00067         DataSpace getSpace() const;
00068
00074         DataSpace getMemSpace() const;
00075
00077         template <typename T>
00078         T read() const;
00079
00087         template <typename T>
00088         void read(T& array) const;
00089
00091         template <typename T>
00092         void read(T* array, const DataType& dtype = {}) const;
00093
00101         template <typename T>
00102         void write(const T& buffer);
00103
00105         template <typename T>
00106         void write_raw(const T* buffer, const DataType& dtype = {});
00107
00109         AttributeCreateProps getCreatePropertyList() const {
00110             return details::get_plist<AttributeCreateProps>(*this, H5Aget_create_plist);
00111         }
00112
00113         // No empty attributes
00114         Attribute() = delete;
00115
00116     protected:
00117         using Object::Object;
00118
00119     private:
00120     #if HIGHFIVE_HAS_FRIEND_DECLARATIONS
00121         template <typename Derivate>
00122         friend class ::HighFive::AnnotateTraits;
00123     #endif
00124
00125         friend Attribute detail::make_attribute(hid_t);

```

```

00126 };
00127
00128 namespace detail {
00129 inline Attribute make_attribute(hid_t hid) {
00130     return Attribute(hid);
00131 }
00132 } // namespace detail
00133
00134 } // namespace HighFive

```

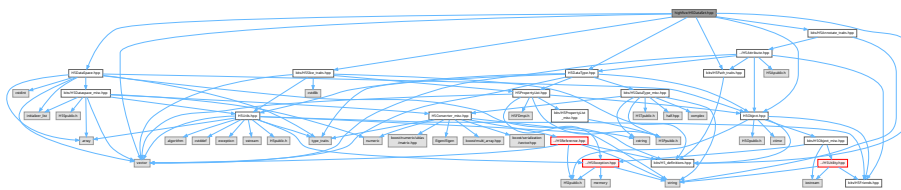
10.54 highfive/H5DataSet.hpp File Reference

```

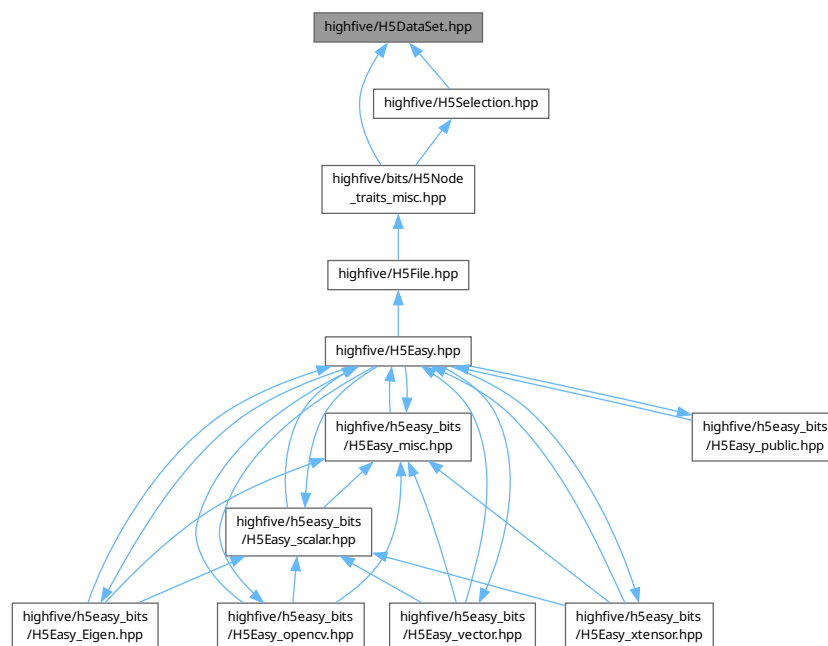
#include <vector>
#include "H5DataSpace.hpp"
#include "H5DataType.hpp"
#include "H5Object.hpp"
#include "bits/H5_definitions.hpp"
#include "bits/H5Annotate_traits.hpp"
#include "bits/H5Slice_traits.hpp"
#include "bits/H5Path_traits.hpp"

```

Include dependency graph for H5DataSet.hpp:



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



Classes

- class [HighFive::DataSet](#)
Class representing a dataset.

Namespaces

- namespace [HighFive](#)

10.55 H5DataSet.hpp

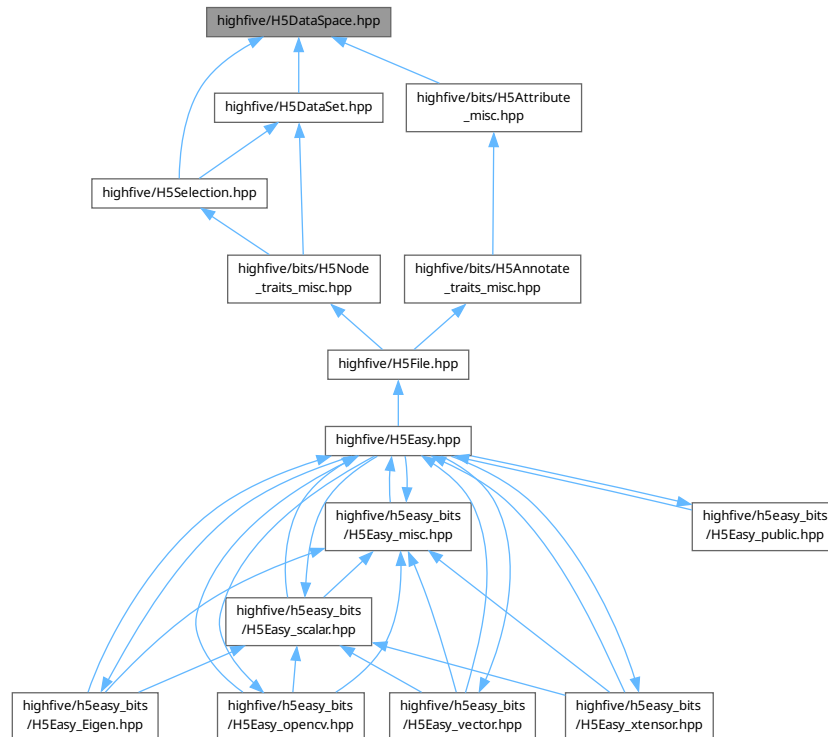
[Go to the documentation of this file.](#)

```

00001  /*
00002  *   Copyright (c), 2017, Adrien Devresse <adrien.devresse@epfl.ch>
00003  *
00004  *   Distributed under the Boost Software License, Version 1.0.
00005  *   (See accompanying file LICENSE_1_0.txt or copy at
00006  *    http://www.boost.org/LICENSE_1_0.txt)
00007  *
00008  */
00009  #pragma once
00010
00011  #include <vector>
00012
00013  #include "H5DataSpace.hpp"
00014  #include "H5DataType.hpp"
00015  #include "H5Object.hpp"
00016  #include "bits/H5_definitions.hpp"
00017  #include "bits/H5Annotate_traits.hpp"
00018  #include "bits/H5Slice_traits.hpp"
00019  #include "bits/H5Path_traits.hpp"
00020  #include "bits/H5_definitions.hpp"
00021
00022  namespace HighFive {
00023
00027  class DataSet: public Object,
00028                public SliceTraits<DataSet>,
00029                public AnnotateTraits<DataSet>,
00030                public PathTraits<DataSet> {
00031  public:
00032      const static ObjectType type = ObjectType::Dataset;
00033
00038      uint64_t getStorageSize() const;
00039
00044      uint64_t getOffset() const;
00045
00050      DataType getDataType() const;
00051
00056      DataSpace getSpace() const;
00057
00063      DataSpace getMemSpace() const;
00064
00065
00071      void resize(const std::vector<size_t>& dims);
00072
00073
00078      inline std::vector<size_t> getDimensions() const {
00079          return getSpace().getDimensions();
00080      }
00081
00087      inline size_t getElementCount() const {
00088          return getSpace().getElementCount();
00089      }
00090
00092      DataSetCreateProps getCreatePropertyList() const {
00093          return details::get_plist<DataSetCreateProps>(*this, H5Dget_create_plist);
00094      }
00095
00097      DataSetAccessProps getAccessPropertyList() const {
00098          return details::get_plist<DataSetAccessProps>(*this, H5Dget_access_plist);
00099      }
00100
00102      H5_DEPRECATED("Default constructor creates unsafe uninitialized objects")
00103      DataSet() = default;
00104
00105  protected:

```


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



Classes

- class [HighFive::DataSpace](#)
Class representing the space (dimensions) of a dataset.

Namespaces

- namespace [HighFive](#)

10.57 H5DataSpace.hpp

[Go to the documentation of this file.](#)

```
00001 /*
00002  * Copyright (c), 2017, Adrien Devresse <adrien.devresse@epfl.ch>
00003  *
00004  * Distributed under the Boost Software License, Version 1.0.
00005  * (See accompanying file LICENSE_1_0.txt or copy at
00006  * http://www.boost.org/LICENSE_1_0.txt)
00007  *
00008  */
00009 #pragma once
00010
00011 #include <vector>
00012 #include <array>
00013 #include <cstdint>
00014 #include <type_traits>
00015 #include <initializer_list>
00016
```

```

00017 #include "H5Object.hpp"
00018 #include "bits/H5_definitions.hpp"
00019
00020 namespace HighFive {
00021
00022 class DataSpace: public Object {
00023 public:
00024     const static ObjectType type = ObjectType::DataSpace;
00025
00026     static const size_t UNLIMITED = SIZE_MAX;
00027
00028     enum DataspaceType {
00029         dataspace_scalar,
00030         dataspace_null
00031         // simple dataspace are handle directly from their dimensions
00032     };
00033
00034     explicit DataSpace(const std::vector<size_t>& dims);
00035
00036     // create a dataspace of N-dimensions
00037     template <size_t N>
00038     explicit DataSpace(const std::array<size_t, N>& dims);
00039
00040     DataSpace(const std::initializer_list<size_t>& items);
00041
00042     template <typename... Args>
00043     explicit DataSpace(size_t dim1, Args... dims);
00044
00045     template <typename IT,
00046               typename = typename std::enable_if<!std::is_integral<IT>::value, IT>::type>
00047     DataSpace(const IT begin, const IT end);
00048
00049     explicit DataSpace(const std::vector<size_t>& dims, const std::vector<size_t>& maxdims);
00050
00051     explicit DataSpace(DataspaceType dtype);
00052
00053     DataSpace clone() const;
00054
00055     size_t getNumberDimensions() const;
00056
00057     std::vector<size_t> getDimensions() const;
00058
00059     size_t getElementCount() const;
00060
00061     std::vector<size_t> getMaxDimensions() const;
00062
00063     template <typename T>
00064     static DataSpace From(const T& value);
00065
00066     template <std::size_t N, std::size_t Width>
00067     static DataSpace FromCharArrayStrings(const char (&)[N][Width]);
00068
00069 protected:
00070     DataSpace() = default;
00071
00072     friend class Attribute;
00073     friend class File;
00074     friend class DataSet;
00075 };
00076
00077 } // namespace HighFive
00078
00079 // We include bits right away since DataSpace is user-constructible
00080 #include "bits/H5Dataspace_misc.hpp"

```

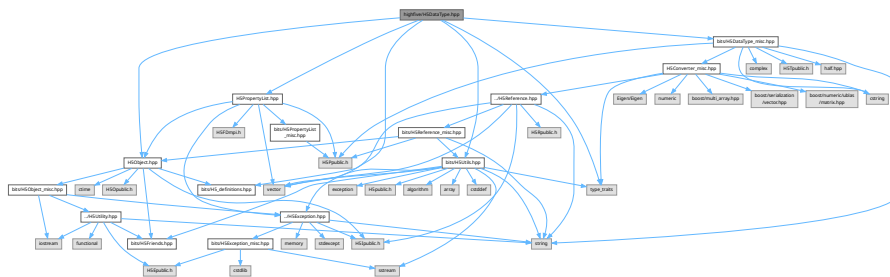
10.58 highfive/H5DataType.hpp File Reference

```

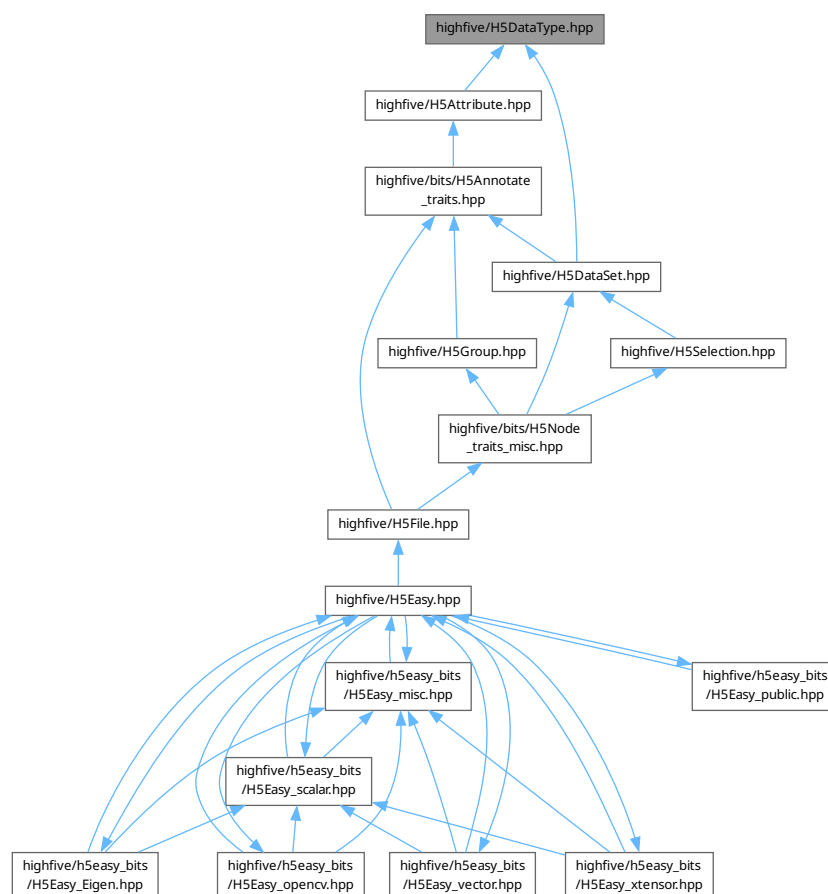
#include <type_traits>
#include <vector>
#include "H5Object.hpp"
#include "bits/H5Utils.hpp"
#include "H5PropertyList.hpp"
#include "bits/H5DataType_misc.hpp"

```

Include dependency graph for H5DataType.hpp:



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



Classes

- class [HighFive::DataType](#)
HDF5 Data Type.
- class [HighFive::AtomicType< T >](#)
create an HDF5 [DataType](#) from a C++ type
- class [HighFive::CompoundType](#)

- *Create a compound HDF5 datatype.*
- struct [HighFive::CompoundType::member_def](#)
Use for defining a sub-type of compound type.
- class [HighFive::EnumType< T >](#)
Create a enum HDF5 datatype.
- struct [HighFive::EnumType< T >::member_def](#)
Use for defining a member of enum type.
- class [HighFive::FixedLenStringArray< N >](#)
A structure representing a set of fixed-length strings.

Namespaces

- namespace [HighFive](#)

Macros

- #define [HIGHFIVE_REGISTER_TYPE](#)(type, function)
Macro to extend datatype of [HighFive](#).

Enumerations

- enum class [HighFive::DataTypeClass](#) {
[HighFive::Time](#) = 1 << 1 , [HighFive::Integer](#) = 1 << 2 , [HighFive::Float](#) = 1 << 3 , [HighFive::String](#) = 1 << 4 ,
[HighFive::BitField](#) = 1 << 5 , [HighFive::Opaque](#) = 1 << 6 , [HighFive::Compound](#) = 1 << 7 ,
[HighFive::Reference](#) = 1 << 8 ,
[HighFive::Enum](#) = 1 << 9 , [HighFive::VarLen](#) = 1 << 10 , [HighFive::Array](#) = 1 << 11 , [HighFive::Invalid](#) = 0
 }
Enum of Fundamental data classes.

Functions

- [DataTypeClass](#) [HighFive::operator|](#) ([DataTypeClass](#) lhs, [DataTypeClass](#) rhs)
- [DataTypeClass](#) [HighFive::operator&](#) ([DataTypeClass](#) lhs, [DataTypeClass](#) rhs)
- template<typename T >
[DataType](#) [HighFive::create_datatype](#) ()
Create a [DataType](#) instance representing type T.
- template<typename T >
[DataType](#) [HighFive::create_and_check_datatype](#) ()
Create a [DataType](#) instance representing type T and perform a sanity check on its size.

10.58.1 Macro Definition Documentation

10.58.1.1 HIGHFIVE_REGISTER_TYPE

```
#define HIGHFIVE_REGISTER_TYPE(  
    type,  
    function )
```

Value:

```
template <>  
inline HighFive::DataType HighFive::create_datatype<type>() { \  
    return function(); \  
}
```

Macro to extend datatype of [HighFive](#).

This macro has to be called outside of any namespace.

```
enum FooBar { FOO = 1, BAR = 2 };  
EnumType create_enum_foobar() {  
    return EnumType<FooBar>({{"FOO", FooBar::FOO},  
                             {"BAR", FooBar::BAR}});  
}  
HIGHFIVE_REGISTER_TYPE(FooBar, create_enum_foobar)
```

10.59 H5DataType.hpp

[Go to the documentation of this file.](#)

```
00001 /*  
00002  * Copyright (c), 2017, Adrien Devresse <adrien.devresse@epfl.ch>  
00003  *  
00004  * Distributed under the Boost Software License, Version 1.0.  
00005  * (See accompanying file LICENSE_1_0.txt or copy at  
00006  *      http://www.boost.org/LICENSE_1_0.txt)  
00007  *  
00008  */  
00009 #pragma once  
00010  
00011 #include <type_traits>  
00012 #include <vector>  
00013  
00014 #include "H5Object.hpp"  
00015 #include "bits/H5Utils.hpp"  
00016  
00017 #include "H5PropertyList.hpp"  
00018  
00019 namespace HighFive {  
00020  
00021  
00025 enum class DataTypeClass {  
00026     Time = 1 << 1,  
00027     Integer = 1 << 2,  
00028     Float = 1 << 3,  
00029     String = 1 << 4,  
00030     BitField = 1 << 5,  
00031     Opaque = 1 << 6,  
00032     Compound = 1 << 7,  
00033     Reference = 1 << 8,  
00034     Enum = 1 << 9,  
00035     VarLen = 1 << 10,  
00036     Array = 1 << 11,  
00037     Invalid = 0  
00038 };  
00039  
00040 inline DataTypeClass operator|(DataTypeClass lhs, DataTypeClass rhs) {  
00041     using T = std::underlying_type<DataTypeClass>::type;  
00042     return static_cast<DataTypeClass>(static_cast<T>(lhs) | static_cast<T>(rhs));  
00043 }  
00044  
00045 inline DataTypeClass operator&(DataTypeClass lhs, DataTypeClass rhs) {  
00046     using T = std::underlying_type<DataTypeClass>::type;  
00047     return static_cast<DataTypeClass>(static_cast<T>(lhs) & static_cast<T>(rhs));  
00048 }  
00049  
00050
```

```

00054 class DataType: public Object {
00055     public:
00056         bool operator==(const DataType& other) const;
00057
00058         bool operator!=(const DataType& other) const;
00059
00063         DataTypeClass getClass() const;
00064
00071         size_t getSize() const;
00072
00076         std::string string() const;
00077
00081         bool isVariableStr() const;
00082
00086         bool isFixedLenStr() const;
00087
00092         bool empty() const noexcept;
00093
00095         bool isReference() const;
00096
00098         DataTypeCreateProps getCreatePropertyList() const {
00099             return details::get_plist<DataTypeCreateProps>(*this, H5Tget_create_plist);
00100         }
00101
00102     protected:
00103         using Object::Object;
00104
00105         friend class Attribute;
00106         friend class File;
00107         friend class DataSet;
00108         friend class CompoundType;
00109 };
00110
00116 template <typename T>
00117 class AtomicType: public DataType {
00118     public:
00119         AtomicType();
00120
00121         using basic_type = T;
00122 };
00123
00124
00128 class CompoundType: public DataType {
00129     public:
00132         struct member_def {
00133             member_def(std::string t_name, DataType t_base_type, size_t t_offset = 0)
00134                 : name(std::move(t_name))
00135                 , base_type(std::move(t_base_type))
00136                 , offset(t_offset) {}
00137             std::string name;
00138             DataType base_type;
00139             size_t offset;
00140         };
00141
00142         CompoundType(const CompoundType& other) = default;
00143
00148         inline CompoundType(const std::vector<member_def>& t_members, size_t size = 0)
00149             : members(t_members) {
00150                 create(size);
00151             }
00152         inline CompoundType(std::vector<member_def>&& t_members, size_t size = 0)
00153             : members(std::move(t_members)) {
00154                 create(size);
00155             }
00156         inline CompoundType(const std::initializer_list<member_def>& t_members, size_t size = 0)
00157             : members(t_members) {
00158                 create(size);
00159             }
00160
00164         inline CompoundType(DataType&& type)
00165             : DataType(type) {
00166             if (getClass() != DataTypeClass::Compound) {
00167                 std::ostringstream ss;
00168                 ss << "hid " << _hid << " does not refer to a compound data type";
00169                 throw DataTypeException(ss.str());
00170             }
00171             int result = H5Tget_nmembers(_hid);
00172             if (result < 0) {
00173                 throw DataTypeException("Could not get members of compound datatype");
00174             }
00175             size_t n_members = static_cast<size_t>(result);
00176             members.reserve(n_members);
00177             for (unsigned i = 0; i < n_members; i++) {
00178                 const char* name = H5Tget_member_name(_hid, i);
00179                 size_t offset = H5Tget_member_offset(_hid, i);
00180                 hid_t member_hid = H5Tget_member_type(_hid, i);
00181                 DataType member_type(member_hid);

```



```

00182         members.emplace_back(name, member_type, offset);
00183     }
00184 }
00185
00186 inline void commit(const Object& object, const std::string& name) const;
00187
00188 inline const std::vector<member_def>& getMembers() const noexcept {
00189     return members;
00190 }
00191
00192 private:
00193     std::vector<member_def> members;
00194
00195     void create(size_t size = 0);
00196 };
00197
00198 template <typename T>
00199 class EnumType: public DataType {
00200 public:
00201     struct member_def {
00202         member_def(const std::string& t_name, T t_value)
00203             : name(t_name)
00204             , value(std::move(t_value)) {}
00205         std::string name;
00206         T value;
00207     };
00208
00209     EnumType(const EnumType& other) = default;
00210
00211     EnumType(const std::vector<member_def>& t_members)
00212         : members(t_members) {
00213         static_assert(std::is_enum<T>::value, "EnumType<T>::create takes only enum");
00214         if (members.empty()) {
00215             HDF5ErrMapper::ToException<DataTypeException>(
00216                 "Could not create an enum without members");
00217         }
00218         create();
00219     }
00220
00221     EnumType(std::initializer_list<member_def> t_members)
00222         : EnumType(std::vector<member_def>{{t_members}}) {}
00223
00224     void commit(const Object& object, const std::string& name) const;
00225
00226 private:
00227     std::vector<member_def> members;
00228
00229     void create();
00230 };
00231
00232 template <typename T>
00233 DataType create_datatype();
00234
00235 template <typename T>
00236 DataType create_and_check_datatype();
00237
00238 template <std::size_t N>
00239 class FixedLenStringArray {
00240 public:
00241     FixedLenStringArray() = default;
00242
00243     FixedLenStringArray(const char array[][N], std::size_t length);
00244
00245     explicit FixedLenStringArray(const std::vector<std::string>& vec);
00246
00247     FixedLenStringArray(const std::string* iter_begin, const std::string* iter_end);
00248
00249     FixedLenStringArray(const std::initializer_list<std::string>&);
00250
00251     void push_back(const std::string&);
00252
00253     void push_back(const std::array<char, N>&);
00254
00255     std::string getString(std::size_t index) const;
00256
00257     // Container interface
00258     inline const char* operator[](std::size_t i) const noexcept {
00259         return datavec[i].data();
00260     }
00261     inline const char* at(std::size_t i) const {
00262         return datavec.at(i).data();
00263     }
00264     inline bool empty() const noexcept {
00265         return datavec.empty();
00266     }
00267 };

```

```

00325     }
00326     inline std::size_t size() const noexcept {
00327         return datavec.size();
00328     }
00329     inline void resize(std::size_t n) {
00330         datavec.resize(n);
00331     }
00332     inline const char* front() const {
00333         return datavec.front().data();
00334     }
00335     inline const char* back() const {
00336         return datavec.back().data();
00337     }
00338     inline char* data() noexcept {
00339         return datavec[0].data();
00340     }
00341     inline const char* data() const noexcept {
00342         return datavec[0].data();
00343     }
00344
00345 private:
00346     using vector_t = typename std::vector<std::array<char, N>;
00347
00348 public:
00349     // Use the underlying iterator
00350     using iterator = typename vector_t::iterator;
00351     using const_iterator = typename vector_t::const_iterator;
00352     using reverse_iterator = typename vector_t::reverse_iterator;
00353     using const_reverse_iterator = typename vector_t::const_reverse_iterator;
00354     using value_type = typename vector_t::value_type;
00355
00356     inline iterator begin() noexcept {
00357         return datavec.begin();
00358     }
00359     inline iterator end() noexcept {
00360         return datavec.end();
00361     }
00362     inline const_iterator begin() const noexcept {
00363         return datavec.begin();
00364     }
00365     inline const_iterator cbegin() const noexcept {
00366         return datavec.cbegin();
00367     }
00368     inline const_iterator end() const noexcept {
00369         return datavec.end();
00370     }
00371     inline const_iterator cend() const noexcept {
00372         return datavec.cend();
00373     }
00374     inline reverse_iterator rbegin() noexcept {
00375         return datavec.rbegin();
00376     }
00377     inline reverse_iterator rend() noexcept {
00378         return datavec.rend();
00379     }
00380     inline const_reverse_iterator rbegin() const noexcept {
00381         return datavec.rbegin();
00382     }
00383     inline const_reverse_iterator rend() const noexcept {
00384         return datavec.rend();
00385     }
00386
00387 private:
00388     vector_t datavec;
00389 };
00390
00391 } // namespace HighFive
00392
00393
00406 #define HIGHFIVE_REGISTER_TYPE(type, function) \
00407     template <> \
00408     inline HighFive::DataType HighFive::create_datatype<type>() { \
00409         return function(); \
00410     } \
00411
00412 #include "bits/H5DataType_misc.hpp"

```

10.60 highfive/H5Easy.hpp File Reference

```

#include <string>
#include <vector>

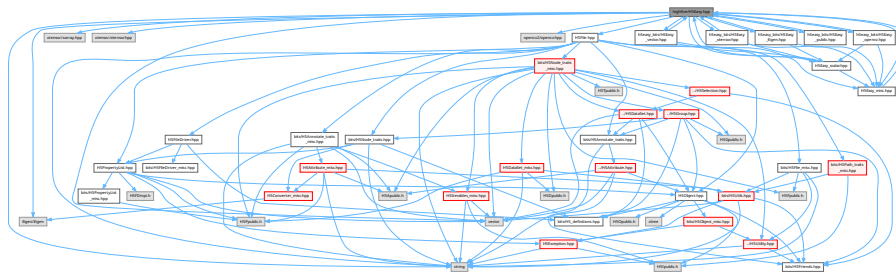
```

```

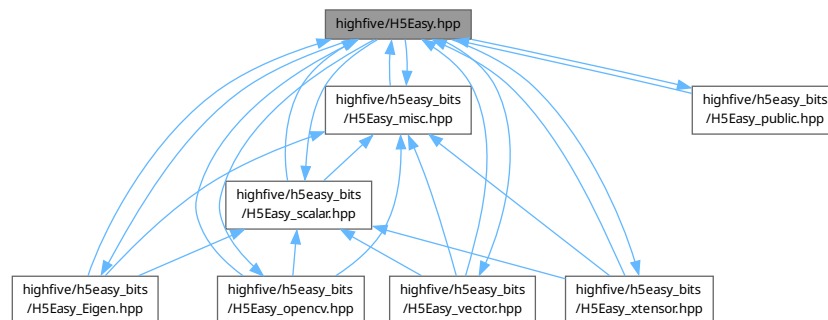
#include <xtensor/xarray.hpp>
#include <xtensor/xtensor.hpp>
#include <Eigen/Eigen>
#include <opencv2/opencv.hpp>
#include "H5File.hpp"
#include "h5easy_bits/H5Easy_Eigen.hpp"
#include "h5easy_bits/H5Easy_misc.hpp"
#include "h5easy_bits/H5Easy_opencv.hpp"
#include "h5easy_bits/H5Easy_public.hpp"
#include "h5easy_bits/H5Easy_scalar.hpp"
#include "h5easy_bits/H5Easy_vector.hpp"
#include "h5easy_bits/H5Easy_xtensor.hpp"

```

Include dependency graph for H5Easy.hpp:



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



Classes

- class [H5Easy::Compression](#)
Signal to set compression level for written DataSets.
- class [H5Easy::DumpOptions](#)
Define options for dumping data.

Namespaces

- namespace [H5Easy](#)
Read/dump DataSets or Attribute using a minimalistic syntax. To this end, the functions are templated, and accept:

Enumerations

- enum class [H5Easy::DumpMode](#) { [H5Easy::Create](#) = 0 , [H5Easy::Overwrite](#) = 1 }
Write mode for DataSets.
- enum class [H5Easy::Flush](#) { [H5Easy::False](#) = 0 , [H5Easy::True](#) = 1 }
Signal to enable/disable automatic flushing after write operations.

Functions

- [size_t H5Easy::getSize](#) (const [File](#) &file, const std::string &path)
Get the size of an existing DataSet in an open HDF5 file.
- [std::vector< size_t > H5Easy::getShape](#) (const [File](#) &file, const std::string &path)
Get the shape of an existing DataSet in an readable file.
- [template<class T > DataSet H5Easy::dump](#) ([File](#) &file, const std::string &path, const T &data, [DumpMode](#) mode=[DumpMode::Create](#))
Write object (templated) to a (new) DataSet in an open HDF5 file.
- [template<class T > DataSet H5Easy::dump](#) ([File](#) &file, const std::string &path, const T &data, const [DumpOptions](#) &options)
Write object (templated) to a (new) DataSet in an open HDF5 file.
- [template<class T > DataSet H5Easy::dump](#) ([File](#) &file, const std::string &path, const T &data, const std::vector< size_t > &idx)
Write a scalar to a (new, extendible) DataSet in an open HDF5 file.
- [template<class T > DataSet H5Easy::dump](#) ([File](#) &file, const std::string &path, const T &data, const std::initializer_list< size_t > &idx)
Write a scalar to a (new, extendible) DataSet in an open HDF5 file.
- [template<class T > DataSet H5Easy::dump](#) ([File](#) &file, const std::string &path, const T &data, const std::vector< size_t > &idx, const [DumpOptions](#) &options)
Write a scalar to a (new, extendible) DataSet in an open HDF5 file.
- [template<class T > DataSet H5Easy::dump](#) ([File](#) &file, const std::string &path, const T &data, const std::initializer_list< size_t > &idx, const [DumpOptions](#) &options)
Write a scalar to a (new, extendible) DataSet in an open HDF5 file.
- [template<class T > T H5Easy::load](#) (const [File](#) &file, const std::string &path, const std::vector< size_t > &idx)
Load entry {i, j, ...} from a DataSet in an open HDF5 file to a scalar.
- [template<class T > T H5Easy::load](#) (const [File](#) &file, const std::string &path)
Load a DataSet in an open HDF5 file to an object (templated).
- [template<class T > Attribute H5Easy::dumpAttribute](#) ([File](#) &file, const std::string &path, const std::string &key, const T &data, [DumpMode](#) mode=[DumpMode::Create](#))
Write object (templated) to a (new) Attribute in an open HDF5 file.
- [template<class T > Attribute H5Easy::dumpAttribute](#) ([File](#) &file, const std::string &path, const std::string &key, const T &data, const [DumpOptions](#) &options)
Write object (templated) to a (new) Attribute in an open HDF5 file.
- [template<class T > T H5Easy::loadAttribute](#) (const [File](#) &file, const std::string &path, const std::string &key)
Load a Attribute in an open HDF5 file to an object (templated).

10.61 H5Easy.hpp

[Go to the documentation of this file.](#)

```

00001  /*
00002   *   Copyright (c), 2017, Adrien Devresse <adrien.devresse@epfl.ch>
00003   *
00004   *   Distributed under the Boost Software License, Version 1.0.
00005   *   (See accompanying file LICENSE_1_0.txt or copy at
00006   *    http://www.boost.org/LICENSE_1_0.txt)
00007   *
00008   */
00009
00010 #pragma once
00011
00012 #include <string>
00013 #include <vector>
00014
00015 // optionally enable xtensor plug-in and load the library
00016 #ifndef XTENSOR_VERSION_MAJOR
00017 #define XTENSOR_VERSION_MAJOR 0
00018 #endif
00019 #ifndef H5_USE_XTENSOR
00020 #define H5_USE_XTENSOR 0
00021 #endif
00022 #endif
00023
00024 #ifndef H5_USE_XTENSOR
00025 #include <xtensor/xarray.hpp>
00026 #include <xtensor/xtensor.hpp>
00027 #endif
00028
00029 // optionally enable Eigen plug-in and load the library
00030 #ifndef EIGEN_WORLD_VERSION
00031 #define EIGEN_WORLD_VERSION 0
00032 #endif
00033 #ifndef H5_USE_EIGEN
00034 #define H5_USE_EIGEN 0
00035 #endif
00036 #endif
00037
00038 #ifndef H5_USE_EIGEN
00039 #include <Eigen/Eigen>
00040 #endif
00041
00042 // optionally enable OpenCV plug-in and load the library
00043 #ifndef CV_MAJOR_VERSION
00044 #define CV_MAJOR_VERSION 0
00045 #endif
00046 #ifndef H5_USE_OPENCV
00047 #define H5_USE_OPENCV 0
00048 #endif
00049 #endif
00050
00051 #ifndef H5_USE_OPENCV
00052 #include <opencv2/opencv.hpp>
00053 #endif
00054
00055 #include "H5File.hpp"
00056
00057 namespace H5Easy {
00058
00059 using HighFive::AtomicType;
00060 using HighFive::Attribute;
00061 using HighFive::Chunking;
00062 using HighFive::DataSet;
00063 using HighFive::DataSetCreateProps;
00064 using HighFive::DataSpace;
00065 using HighFive::Deflate;
00066 using HighFive::Exception;
00067 using HighFive::File;
00068 using HighFive::ObjectType;
00069 using HighFive::Shuffle;
00070
00071 enum class DumpMode {
00072     Create = 0,
00073     Overwrite = 1
00074 };
00075
00076 enum class Flush {
00077     False = 0,
00078     True = 1
00079 };
00080
00081 class Compression {
00082 public:
00083     explicit Compression(bool enable = true);
00084
00085     template <class T>
00086     Compression(T level);
00087
00088 };
00089
00090 }
00091
00092 
```

```

00107     inline unsigned get() const;
00108
00109 private:
00110     unsigned m_compression_level;
00111 };
00112
00121 class DumpOptions {
00122 public:
00123     DumpOptions() = default;
00124
00125     template <class... Args>
00126     DumpOptions(Args... args) {
00127         set(args...);
00128     }
00129
00130     inline void set(DumpMode mode);
00131
00132     inline void set(Flush mode);
00133
00134     inline void set(const Compression& level);
00135
00136     template <class T, class... Args>
00137     inline void set(T arg, Args... args);
00138
00139     template <class T>
00140     inline void setChunkSize(const std::vector<T>& shape);
00141
00142     inline void setChunkSize(std::initializer_list<size_t> shape);
00143
00144     inline bool overwrite() const;
00145
00146     inline bool flush() const;
00147
00148     inline bool compress() const;
00149
00150     inline unsigned getCompressionLevel() const;
00151
00152     inline bool isChunked() const;
00153
00154     inline std::vector<hsize_t> getChunkSize() const;
00155 private:
00156     bool m_overwrite = false;
00157     bool m_flush = true;
00158     unsigned m_compression_level = 0;
00159     std::vector<hsize_t> m_chunk_size = {};
00160 };
00161
00162 inline size_t getSize(const File& file, const std::string& path);
00163
00164 inline std::vector<size_t> getShape(const File& file, const std::string& path);
00165
00166 template <class T>
00167 inline DataSet dump(File& file,
00168     const std::string& path,
00169     const T& data,
00170     DumpMode mode = DumpMode::Create);
00171
00172 template <class T>
00173 inline DataSet dump(File& file, const std::string& path, const T& data, const DumpOptions& options);
00174
00175 template <class T>
00176 inline DataSet dump(File& file,
00177     const std::string& path,
00178     const T& data,
00179     const std::vector<size_t>& idx);
00180
00181 template <class T>
00182 inline DataSet dump(File& file,
00183     const std::string& path,
00184     const T& data,
00185     const std::initializer_list<size_t>& idx);
00186
00187 template <class T>
00188 inline DataSet dump(File& file,
00189     const std::string& path,
00190     const T& data,
00191     const std::vector<size_t>& idx,
00192     const DumpOptions& options);
00193
00194 template <class T>
00195 inline DataSet dump(File& file,
00196     const std::string& path,
00197     const T& data,
00198     const std::initializer_list<size_t>& idx,
00199     const DumpOptions& options);
00200
00201

```

```

00330 template <class T>
00331 inline T load(const File& file, const std::string& path, const std::vector<size_t>& idx);
00332
00341 template <class T>
00342 inline T load(const File& file, const std::string& path);
00343
00355 template <class T>
00356 inline Attribute dumpAttribute(File& file,
00357                               const std::string& path,
00358                               const std::string& key,
00359                               const T& data,
00360                               DumpMode mode = DumpMode::Create);
00361
00373 template <class T>
00374 inline Attribute dumpAttribute(File& file,
00375                               const std::string& path,
00376                               const std::string& key,
00377                               const T& data,
00378                               const DumpOptions& options);
00379
00389 template <class T>
00390 inline T loadAttribute(const File& file, const std::string& path, const std::string& key);
00391
00392 } // namespace H5Easy
00393
00394 #include "h5easy_bits/H5Easy_Eigen.hpp"
00395 #include "h5easy_bits/H5Easy_misc.hpp"
00396 #include "h5easy_bits/H5Easy_opencv.hpp"
00397 #include "h5easy_bits/H5Easy_public.hpp"
00398 #include "h5easy_bits/H5Easy_scalar.hpp"
00399 #include "h5easy_bits/H5Easy_vector.hpp"
00400 #include "h5easy_bits/H5Easy_xtensor.hpp"

```

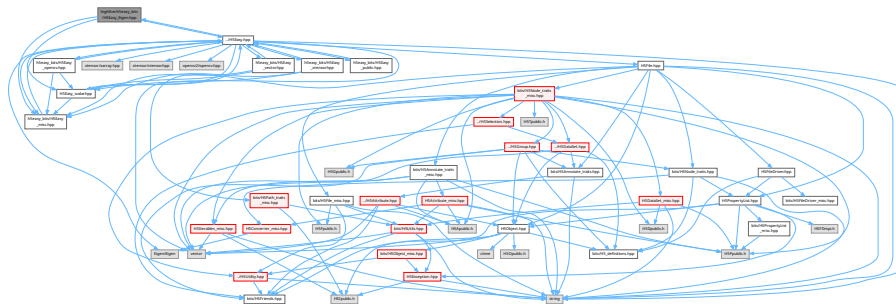
10.62 highfive/h5easy_bits/H5Easy_Eigen.hpp File Reference

```

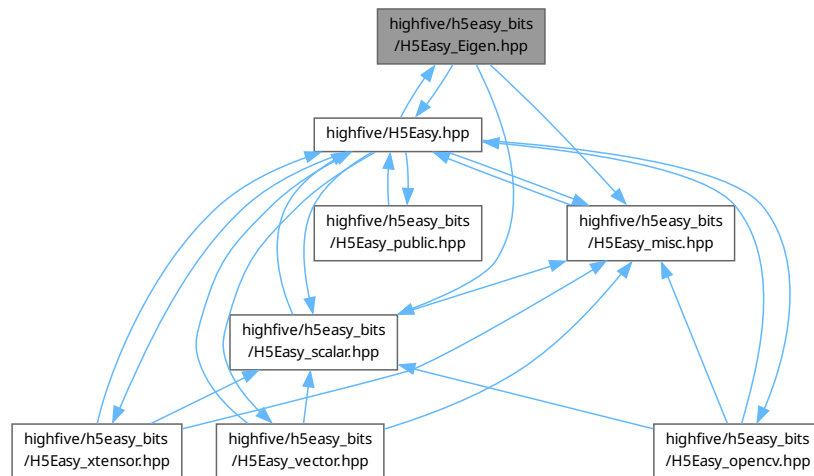
#include "../H5Easy.hpp"
#include "H5Easy_misc.hpp"
#include "H5Easy_scalar.hpp"

```

Include dependency graph for H5Easy_Eigen.hpp:



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



Namespaces

- namespace [H5Easy](#)

Read/dump DataSets or Attribute using a minimalistic syntax. To this end, the functions are templated, and accept:

10.63 H5Easy_Eigen.hpp

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c), 2017, Adrien Devresse <adrien.devresse@epfl.ch>
00003  *
00004  * Distributed under the Boost Software License, Version 1.0.
00005  * (See accompanying file LICENSE_1_0.txt or copy at
00006  * http://www.boost.org/LICENSE_1_0.txt)
00007  *
00008  */
00009 #pragma once
00010
00011 #include "../H5Easy.hpp"
00012 #include "H5Easy_misc.hpp"
00013 #include "H5Easy_scalar.hpp"
00014
00015 #ifndef H5_USE_EIGEN
00016
00017 namespace H5Easy {
00018
00019 namespace detail {
00020
00021 template <typename T>
00022 struct io_impl<T, typename std::enable_if<std::is_base_of<Eigen::DenseBase<T>, T>::value>::type> {
00023     // abbreviate row-major <-> col-major conversions
00024     template <typename S>
00025     struct types {
00026         using row_major = Eigen::Ref<
00027             const Eigen::Array<typename std::decay<T>::type::Scalar,
00028                 std::decay<T>::type::RowsAtCompileTime,
00029                 std::decay<T>::type::ColsAtCompileTime,
00030                 std::decay<T>::type::ColsAtCompileTime == 1 ? Eigen::ColMajor
00031                     : Eigen::RowMajor,
00032                 std::decay<T>::type::MaxRowsAtCompileTime,
00033                 std::decay<T>::type::MaxColsAtCompileTime>,
00034                 0,
00035                 Eigen::InnerStride<1>;
00036     };

```



```

00037         using col_major =
00038             Eigen::Map<Eigen::Array<typename std::decay<T>::type::Scalar,
00039                 std::decay<T>::type::RowsAtCompileTime,
00040                 std::decay<T>::type::ColsAtCompileTime,
00041                 std::decay<T>::type::ColsAtCompileTime == 1 ? Eigen::ColMajor
00042                                     : Eigen::RowMajor,
00043                 std::decay<T>::type::MaxRowsAtCompileTime,
00044                 std::decay<T>::type::MaxColsAtCompileTime>;
00045     };
00046
00047     // return the shape of Eigen::DenseBase<T> object as size 1 or 2 "std::vector<size_t>"
00048     inline static std::vector<size_t> shape(const T& data) {
00049         if (std::decay<T>::type::RowsAtCompileTime == 1) {
00050             return {static_cast<size_t>(data.cols())};
00051         }
00052         if (std::decay<T>::type::ColsAtCompileTime == 1) {
00053             return {static_cast<size_t>(data.rows())};
00054         }
00055         return {static_cast<size_t>(data.rows()), static_cast<size_t>(data.cols())};
00056     }
00057
00058     using EigenIndex = Eigen::DenseIndex;
00059
00060     // get the shape of a "DataSet" as size 2 "std::vector<Eigen::Index>"
00061     template <class D>
00062     inline static std::vector<EigenIndex> shape(const File& file,
00063         const std::string& path,
00064         const D& dataset,
00065         int RowsAtCompileTime) {
00066         std::vector<size_t> dims = dataset.getDimensions();
00067
00068         if (dims.size() == 1 && RowsAtCompileTime == 1) {
00069             return std::vector<EigenIndex>{1u, static_cast<EigenIndex>(dims[0])};
00070         }
00071         if (dims.size() == 1) {
00072             return std::vector<EigenIndex>{static_cast<EigenIndex>(dims[0]), 1u};
00073         }
00074         if (dims.size() == 2) {
00075             return std::vector<EigenIndex>{static_cast<EigenIndex>(dims[0]),
00076                 static_cast<EigenIndex>(dims[1])};
00077         }
00078
00079         throw detail::error(file, path, "H5Easy::load: Inconsistent rank");
00080     }
00081
00082     inline static DataSet dump(File& file,
00083         const std::string& path,
00084         const T& data,
00085         const DumpOptions& options) {
00086         using row_major_type = typename types<T>::row_major;
00087         using value_type = typename std::decay<T>::type::Scalar;
00088         row_major_type row_major(data);
00089         DataSet dataset = initDataSet<value_type>(file, path, shape(data), options);
00090         dataset.write_raw(row_major.data());
00091         if (options.flush()) {
00092             file.flush();
00093         }
00094         return dataset;
00095     }
00096
00097     inline static T load(const File& file, const std::string& path) {
00098         DataSet dataset = file.getDataSet(path);
00099         std::vector<typename T::Index> dims = shape(file, path, dataset, T::RowsAtCompileTime);
00100         T data(dims[0], dims[1]);
00101         dataset.read(data.data());
00102         if (data.IsVectorAtCompileTime || data.IsRowMajor) {
00103             return data;
00104         }
00105         using col_major = typename types<T>::col_major;
00106         return col_major(data.data(), dims[0], dims[1]);
00107     }
00108
00109     inline static Attribute dumpAttribute(File& file,
00110         const std::string& path,
00111         const std::string& key,
00112         const T& data,
00113         const DumpOptions& options) {
00114         using row_major_type = typename types<T>::row_major;
00115         using value_type = typename std::decay<T>::type::Scalar;
00116         row_major_type row_major(data);
00117         Attribute attribute = initAttribute<value_type>(file, path, key, shape(data), options);
00118         attribute.write_raw(row_major.data());
00119         if (options.flush()) {
00120             file.flush();
00121         }
00122         return attribute;
00123     }

```

```

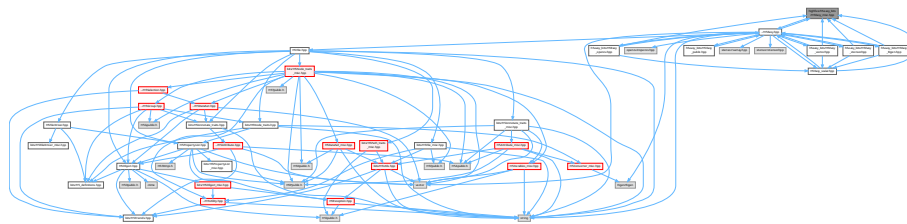
00124
00125     inline static T loadAttribute(const File& file,
00126                                 const std::string& path,
00127                                 const std::string& key) {
00128         DataSet dataset = file.getDataSet(path);
00129         Attribute attribute = dataset.getAttribute(key);
00130         DataSpace dataspace = attribute.getSpace();
00131         std::vector<typename T::Index> dims = shape(file, path, dataspace, T::RowsAtCompileTime);
00132         T data(dims[0], dims[1]);
00133         attribute.read(data.data());
00134         if (data.IsVectorAtCompileTime || data.IsRowMajor) {
00135             return data;
00136         }
00137         using col_major = typename types<T>::col_major;
00138         return col_major(data.data(), dims[0], dims[1]);
00139     }
00140 };
00141
00142 } // namespace detail
00143 } // namespace H5Easy
00144
00145 #endif // H5_USE_EIGEN

```

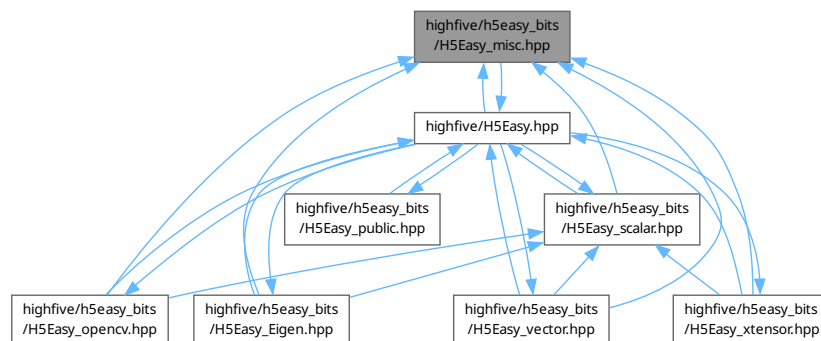
10.64 highfive/h5easy_bits/H5Easy_misc.hpp File Reference

#include `"../H5Easy.hpp"`

Include dependency graph for H5Easy_misc.hpp:



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



Namespaces

- namespace **H5Easy**

Read/dump DataSets or Attribute using a minimalistic syntax. To this end, the functions are templated, and accept:

10.65 H5Easy_misc.hpp

[Go to the documentation of this file.](#)

```

00001  /*
00002  *   Copyright (c), 2017, Adrien Devresse <adrien.devresse@epfl.ch>
00003  *
00004  *   Distributed under the Boost Software License, Version 1.0.
00005  *   (See accompanying file LICENSE_1_0.txt or copy at
00006  *    http://www.boost.org/LICENSE_1_0.txt)
00007  *
00008  */
00009  #pragma once
00010
00011  #include "../H5Easy.hpp"
00012
00013  namespace H5Easy {
00014
00015  namespace detail {
00016
00017  // Generate error-stream and return "Exception" (not yet thrown).
00018  inline Exception error(const File& file, const std::string& path, const std::string& message) {
00019      std::ostringstream ss;
00020      ss << message << std::endl
00021          << "Path: " << path << std::endl
00022          << "Filename: " << file.getName() << std::endl;
00023      return Exception(ss.str());
00024  }
00025
00026  // Generate specific dump error
00027  inline Exception dump_error(File& file, const std::string& path) {
00028      if (file.getObjectType(path) == ObjectType::Dataset) {
00029          return error(file,
00030                      path,
00031                      "H5Easy: Dataset already exists, dump with H5Easy::DumpMode::Overwrite "
00032                      "to overwrite (with an array of the same shape).");
00033      } else {
00034          return error(
00035              file,
00036              path,
00037              "H5Easy: path exists, but does not correspond to a Dataset. Dump not possible.");
00038      }
00039  }
00040
00041  // get a opened DataSet: nd-array
00042  template <class T>
00043  inline DataSet initDataset(File& file,
00044                           const std::string& path,
00045                           const std::vector<size_t>& shape,
00046                           const DumpOptions& options) {
00047      if (!file.exist(path)) {
00048          if (!options.compress() && !options.isChunked()) {
00049              return file.createDataSet<T>(path, DataSpace(shape), {}, {}, true);
00050          } else {
00051              std::vector<hsize_t> chunks(shape.begin(), shape.end());
00052              if (options.isChunked()) {
00053                  chunks = options.getChunkSize();
00054                  if (chunks.size() != shape.size()) {
00055                      throw error(file, path, "H5Easy::dump: Incorrect rank ChunkSize");
00056                  }
00057              }
00058              DataSetCreateProps props;
00059              props.add(Chunking(chunks));
00060              if (options.compress()) {
00061                  props.add(Shuffle());
00062                  props.add(Deflate(options.getCompressionLevel()));
00063              }
00064              return file.createDataSet<T>(path, DataSpace(shape), props, {}, true);
00065          }
00066      } else if (options.overwrite() && file.getObjectType(path) == ObjectType::Dataset) {
00067          DataSet dataset = file.getDataSet(path);
00068          if (dataset.getDimensions() != shape) {
00069              throw error(file, path, "H5Easy::dump: Inconsistent dimensions");
00070          }
00071          return dataset;
00072      }
00073      throw dump_error(file, path);
00074  }
00075
00076  // get a opened DataSet: scalar
00077  template <class T>
00078  inline DataSet initScalarDataSet(File& file,
00079                                  const std::string& path,
00080                                  const T& data,
00081                                  const DumpOptions& options) {
00082      if (!file.exist(path)) {

```

```

00083         return file.createDataSet<T>(path, DataSpace::From(data), {}, {}, true);
00084     } else if (options.overwrite() && file.getObjectType(path) == ObjectType::Dataset) {
00085         DataSet dataset = file.getDataSet(path);
00086         if (dataset.getElementCount() != 1) {
00087             throw error(file, path, "H5Easy::dump: Existing field not a scalar");
00088         }
00089         return dataset;
00090     }
00091     throw dump_error(file, path);
00092 }
00093
00094 // get a opened Attribute: nd-array
00095 template <class T>
00096 inline Attribute initAttribute(File& file,
00097                               const std::string& path,
00098                               const std::string& key,
00099                               const std::vector<size_t>& shape,
00100                               const DumpOptions& options) {
00101     if (!file.exist(path)) {
00102         throw error(file, path, "H5Easy::dumpAttribute: DataSet does not exist");
00103     }
00104     if (file.getObjectType(path) != ObjectType::Dataset) {
00105         throw error(file, path, "H5Easy::dumpAttribute: path not a DataSet");
00106     }
00107     DataSet dataset = file.getDataSet(path);
00108     if (!dataset.hasAttribute(key)) {
00109         return dataset.createAttribute<T>(key, DataSpace(shape));
00110     } else if (options.overwrite()) {
00111         Attribute attribute = dataset.getAttribute(key);
00112         DataSpace dataspace = attribute.getSpace();
00113         if (dataspace.getDimensions() != shape) {
00114             throw error(file, path, "H5Easy::dumpAttribute: Inconsistent dimensions");
00115         }
00116         return attribute;
00117     }
00118     throw error(file,
00119                 path,
00120                 "H5Easy: Attribute exists, overwrite with H5Easy::DumpMode::Overwrite.");
00121 }
00122
00123 // get a opened Attribute: scalar
00124 template <class T>
00125 inline Attribute initScalarAttribute(File& file,
00126                                     const std::string& path,
00127                                     const std::string& key,
00128                                     const T& data,
00129                                     const DumpOptions& options) {
00130     if (!file.exist(path)) {
00131         throw error(file, path, "H5Easy::dumpAttribute: DataSet does not exist");
00132     }
00133     if (file.getObjectType(path) != ObjectType::Dataset) {
00134         throw error(file, path, "H5Easy::dumpAttribute: path not a DataSet");
00135     }
00136     DataSet dataset = file.getDataSet(path);
00137     if (!dataset.hasAttribute(key)) {
00138         return dataset.createAttribute<T>(key, DataSpace::From(data));
00139     } else if (options.overwrite()) {
00140         Attribute attribute = dataset.getAttribute(key);
00141         DataSpace dataspace = attribute.getSpace();
00142         if (dataspace.getElementCount() != 1) {
00143             throw error(file, path, "H5Easy::dumpAttribute: Existing field not a scalar");
00144         }
00145         return attribute;
00146     }
00147     throw error(file,
00148                 path,
00149                 "H5Easy: Attribute exists, overwrite with H5Easy::DumpMode::Overwrite.");
00150 }
00151
00152 } // namespace detail
00153 } // namespace H5Easy

```

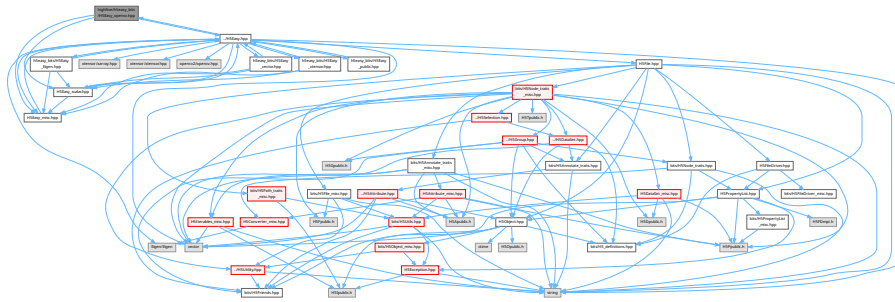
10.66 highfive/h5easy_bits/H5Easy_opencv.hpp File Reference

```

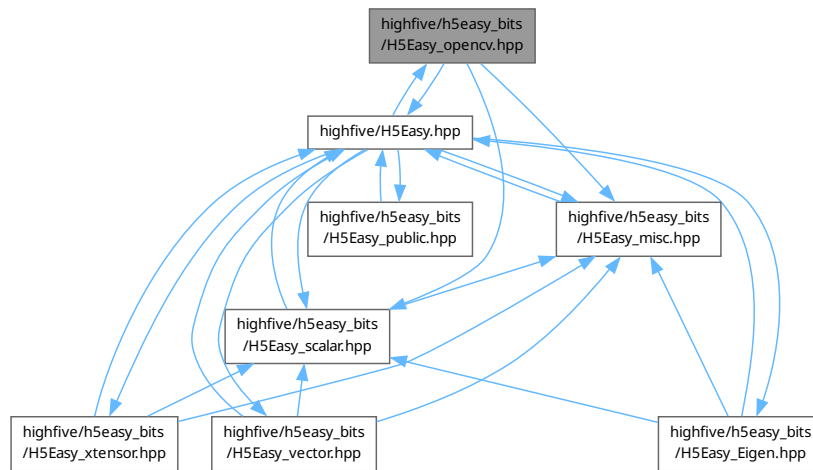
#include "../H5Easy.hpp"
#include "H5Easy_misc.hpp"
#include "H5Easy_scalar.hpp"

```

Include dependency graph for H5Easy_opencv.hpp:



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



Namespaces

- namespace **H5Easy**

Read/dump DataSets or Attribute using a minimalistic syntax. To this end, the functions are templated, and accept:

10.67 H5Easy_opencv.hpp

[Go to the documentation of this file.](#)

```
00001 /*
00002  * Copyright (c), 2017, Adrien Devresse <adrien.devresse@epfl.ch>
00003  *
00004  * Distributed under the Boost Software License, Version 1.0.
00005  * (See accompanying file LICENSE_1_0.txt or copy at
00006  * http://www.boost.org/LICENSE_1_0.txt)
00007  *
00008 */
00009 #pragma once
00010
00011 #include "../H5Easy.hpp"
00012 #include "H5Easy_misc.hpp"
00013 #include "H5Easy_scalar.hpp"
00014
```

```

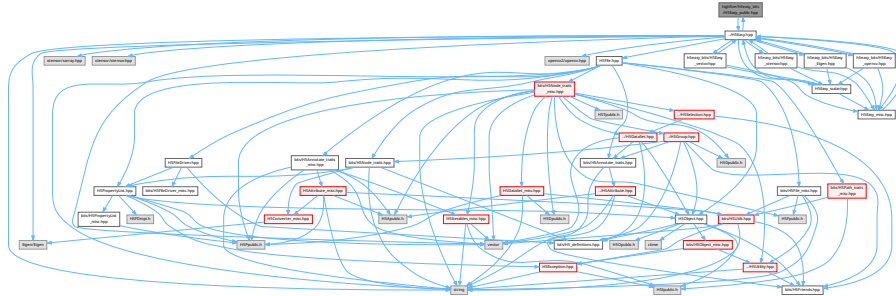
00015 #ifdef H5_USE_OPENCV
00016
00017 namespace H5Easy {
00018
00019 namespace detail {
00020
00021 template <class T>
00022 struct is_opencv: std::false_type {};
00023 template <class T>
00024 struct is_opencv<cv::Mat_<T>: std::true_type {};
00025
00026 template <typename T>
00027 struct io_impl<T, typename std::enable_if<is_opencv<T>::value>::type> {
00028     inline static std::vector<size_t> shape(const T& data) {
00029         return std::vector<size_t>{static_cast<size_t>(data.rows), static_cast<size_t>(data.cols)};
00030     }
00031
00032     inline static std::vector<int> shape(const File& file,
00033         const std::string& path,
00034         const std::vector<size_t> dims) {
00035         if (dims.size() == 1) {
00036             return std::vector<int>{static_cast<int>(dims[0]), 1ul};
00037         }
00038         if (dims.size() == 2) {
00039             return std::vector<int>{static_cast<int>(dims[0]), static_cast<int>(dims[1])};
00040         }
00041
00042         throw detail::error(file, path, "H5Easy::load: Inconsistent rank");
00043     }
00044
00045     inline static DataSet dump(File& file,
00046         const std::string& path,
00047         const T& data,
00048         const DumpOptions& options) {
00049         using value_type = typename T::value_type;
00050         DataSet dataset = initDataSet<value_type>(file, path, shape(data), options);
00051         std::vector<value_type> v(data.begin(), data.end());
00052         dataset.write_raw(v.data());
00053         if (options.flush()) {
00054             file.flush();
00055         }
00056         return dataset;
00057     }
00058
00059     inline static T load(const File& file, const std::string& path) {
00060         using value_type = typename T::value_type;
00061         DataSet dataset = file.getDataSet(path);
00062         std::vector<int> dims = shape(file, path, dataset.getDimensions());
00063         T data(dims[0], dims[1]);
00064         dataset.read(reinterpret_cast<value_type*>(data.data()));
00065         return data;
00066     }
00067
00068     inline static Attribute dumpAttribute(File& file,
00069         const std::string& path,
00070         const std::string& key,
00071         const T& data,
00072         const DumpOptions& options) {
00073         using value_type = typename T::value_type;
00074         Attribute attribute = initAttribute<value_type>(file, path, key, shape(data), options);
00075         std::vector<value_type> v(data.begin(), data.end());
00076         attribute.write_raw(v.data());
00077         if (options.flush()) {
00078             file.flush();
00079         }
00080         return attribute;
00081     }
00082
00083     inline static T loadAttribute(const File& file,
00084         const std::string& path,
00085         const std::string& key) {
00086         using value_type = typename T::value_type;
00087         DataSet dataset = file.getDataSet(path);
00088         Attribute attribute = dataset.getAttribute(key);
00089         DataSpace dataspace = attribute.getSpace();
00090         std::vector<int> dims = shape(file, path, dataspace.getDimensions());
00091         T data(dims[0], dims[1]);
00092         attribute.read(reinterpret_cast<value_type*>(data.data()));
00093         return data;
00094     }
00095 };
00096
00097 } // namespace detail
00098 } // namespace H5Easy
00099
00100 #endif // H5_USE_OPENCV

```

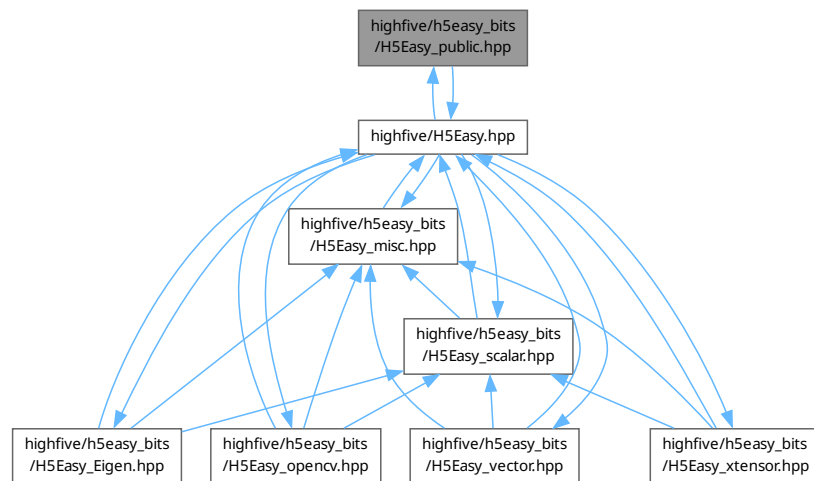
10.68 highfive/h5easy_bits/H5Easy_public.hpp File Reference

```
#include "../H5Easy.hpp"
```

Include dependency graph for H5Easy_public.hpp:



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



Namespaces

- namespace [H5Easy](#)

Read/dump DataSets or Attribute using a minimalistic syntax. To this end, the functions are templated, and accept:

Functions

- size_t [H5Easy::getSize](#) (const [File](#) &file, const std::string &path)
Get the size of an existing DataSet in an open HDF5 file.
- std::vector< size_t > [H5Easy::getShape](#) (const [File](#) &file, const std::string &path)
Get the shape of an existing DataSet in an readable file.
- template<class T >
[DataSet H5Easy::dump](#) ([File](#) &file, const std::string &path, const T &data, const [DumpOptions](#) &options)

- Write object (templated) to a (new) DataSet in an open HDF5 file.*

 - `template<class T >`
`DataSet H5Easy::dump (File &file, const std::string &path, const T &data, DumpMode mode=DumpMode::Create)`
Write object (templated) to a (new) DataSet in an open HDF5 file.
 - `template<class T >`
`DataSet H5Easy::dump (File &file, const std::string &path, const T &data, const std::vector< size_t > &idx, const DumpOptions &options)`
Write a scalar to a (new, extendible) DataSet in an open HDF5 file.
 - `template<class T >`
`DataSet H5Easy::dump (File &file, const std::string &path, const T &data, const std::initializer_list< size_t > &idx, const DumpOptions &options)`
Write a scalar to a (new, extendible) DataSet in an open HDF5 file.
 - `template<class T >`
`DataSet H5Easy::dump (File &file, const std::string &path, const T &data, const std::vector< size_t > &idx)`
Write a scalar to a (new, extendible) DataSet in an open HDF5 file.
 - `template<class T >`
`DataSet H5Easy::dump (File &file, const std::string &path, const T &data, const std::initializer_list< size_t > &idx)`
Write a scalar to a (new, extendible) DataSet in an open HDF5 file.
 - `template<class T >`
`T H5Easy::load (const File &file, const std::string &path, const std::vector< size_t > &idx)`
Load entry {i, j, ...} from a DataSet in an open HDF5 file to a scalar.
 - `template<class T >`
`T H5Easy::load (const File &file, const std::string &path)`
Load a DataSet in an open HDF5 file to an object (templated).
 - `template<class T >`
`Attribute H5Easy::dumpAttribute (File &file, const std::string &path, const std::string &key, const T &data, DumpMode mode=DumpMode::Create)`
Write object (templated) to a (new) Attribute in an open HDF5 file.
 - `template<class T >`
`Attribute H5Easy::dumpAttribute (File &file, const std::string &path, const std::string &key, const T &data, const DumpOptions &options)`
Write object (templated) to a (new) Attribute in an open HDF5 file.
 - `template<class T >`
`T H5Easy::loadAttribute (const File &file, const std::string &path, const std::string &key)`
Load a Attribute in an open HDF5 file to an object (templated).

10.69 H5Easy_public.hpp

Go to the documentation of this file.

```

00001 /*
00002  * Copyright (c), 2017, Adrien Devresse <adrien.devresse@epfl.ch>
00003  *
00004  * Distributed under the Boost Software License, Version 1.0.
00005  * (See accompanying file LICENSE_1_0.txt or copy at
00006  * http://www.boost.org/LICENSE_1_0.txt)
00007  *
00008  */
00009 #pragma once
00010
00011 #include "../H5Easy.hpp"
00012
00013 namespace H5Easy {
00014
00015 inline Compression::Compression(bool enable) {
00016     if (enable) {
00017         m_compression_level = 9;
00018     } else {
00019         m_compression_level = 0;
00020     }
00021 }
00022
00023 }

```



```

00020     }
00021 }
00022
00023 template <class T>
00024 inline Compression::Compression(T level)
00025     : m_compression_level(static_cast<unsigned>(level)) {}
00026
00027 inline unsigned Compression::get() const {
00028     return m_compression_level;
00029 }
00030
00031 inline void DumpOptions::set(DumpMode mode) {
00032     m_overwrite = static_cast<bool>(mode);
00033 }
00034
00035 inline void DumpOptions::set(Flush mode) {
00036     m_flush = static_cast<bool>(mode);
00037 }
00038
00039 inline void DumpOptions::set(const Compression& level) {
00040     m_compression_level = level.get();
00041 }
00042
00043 template <class T, class... Args>
00044 inline void DumpOptions::set(T arg, Args... args) {
00045     set(arg);
00046     set(args...);
00047 }
00048
00049 template <class T>
00050 inline void DumpOptions::setChunkSize(const std::vector<T>& shape) {
00051     m_chunk_size = std::vector<hsize_t>(shape.begin(), shape.end());
00052 }
00053
00054 inline void DumpOptions::setChunkSize(std::initializer_list<size_t> shape) {
00055     m_chunk_size = std::vector<hsize_t>(shape.begin(), shape.end());
00056 }
00057
00058 inline bool DumpOptions::overwrite() const {
00059     return m_overwrite;
00060 }
00061
00062 inline bool DumpOptions::flush() const {
00063     return m_flush;
00064 }
00065
00066 inline bool DumpOptions::compress() const {
00067     return m_compression_level > 0;
00068 }
00069
00070 inline unsigned DumpOptions::getCompressionLevel() const {
00071     return m_compression_level;
00072 }
00073
00074 inline bool DumpOptions::isChunked() const {
00075     return m_chunk_size.size() > 0;
00076 }
00077
00078 inline std::vector<hsize_t> DumpOptions::getChunkSize() const {
00079     return m_chunk_size;
00080 }
00081
00082 inline size_t getSize(const File& file, const std::string& path) {
00083     return file.getDataSet(path).getElementCount();
00084 }
00085
00086 inline std::vector<size_t> getShape(const File& file, const std::string& path) {
00087     return file.getDataSet(path).getDimensions();
00088 }
00089
00090 template <class T>
00091 inline DataSet dump(File& file,
00092     const std::string& path,
00093     const T& data,
00094     const DumpOptions& options) {
00095     return detail::io_impl<T>::dump(file, path, data, options);
00096 }
00097
00098 template <class T>
00099 inline DataSet dump(File& file, const std::string& path, const T& data, DumpMode mode) {
00100     return detail::io_impl<T>::dump(file, path, data, DumpOptions(mode));
00101 }
00102
00103 template <class T>
00104 inline DataSet dump(File& file,
00105     const std::string& path,
00106     const T& data,

```

```

00107         const std::vector<size_t>& idx,
00108         const DumpOptions& options) {
00109     return detail::io_impl<T>::dump_extend(file, path, data, idx, options);
00110 }
00111
00112 template <class T>
00113 inline DataSet dump(File& file,
00114     const std::string& path,
00115     const T& data,
00116     const std::initializer_list<size_t>& idx,
00117     const DumpOptions& options) {
00118     return detail::io_impl<T>::dump_extend(file, path, data, idx, options);
00119 }
00120
00121 template <class T>
00122 inline DataSet dump(File& file,
00123     const std::string& path,
00124     const T& data,
00125     const std::vector<size_t>& idx) {
00126     return detail::io_impl<T>::dump_extend(file, path, data, idx, DumpOptions());
00127 }
00128
00129 template <class T>
00130 inline DataSet dump(File& file,
00131     const std::string& path,
00132     const T& data,
00133     const std::initializer_list<size_t>& idx) {
00134     return detail::io_impl<T>::dump_extend(file, path, data, idx, DumpOptions());
00135 }
00136
00137 template <class T>
00138 inline T load(const File& file, const std::string& path, const std::vector<size_t>& idx) {
00139     return detail::io_impl<T>::load_part(file, path, idx);
00140 }
00141
00142 template <class T>
00143 inline T load(const File& file, const std::string& path) {
00144     return detail::io_impl<T>::load(file, path);
00145 }
00146
00147 template <class T>
00148 inline Attribute dumpAttribute(File& file,
00149     const std::string& path,
00150     const std::string& key,
00151     const T& data,
00152     DumpMode mode) {
00153     return detail::io_impl<T>::dumpAttribute(file, path, key, data, DumpOptions(mode));
00154 }
00155
00156 template <class T>
00157 inline Attribute dumpAttribute(File& file,
00158     const std::string& path,
00159     const std::string& key,
00160     const T& data,
00161     const DumpOptions& options) {
00162     return detail::io_impl<T>::dumpAttribute(file, path, key, data, options);
00163 }
00164
00165 template <class T>
00166 inline T loadAttribute(const File& file, const std::string& path, const std::string& key) {
00167     return detail::io_impl<T>::loadAttribute(file, path, key);
00168 }
00169
00170 } // namespace H5Easy

```

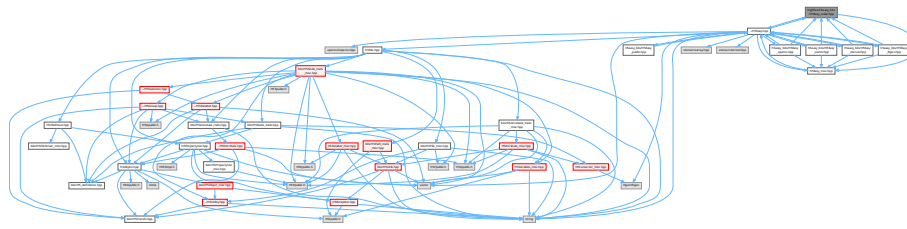
10.70 highfive/h5easy_bits/H5Easy_scalar.hpp File Reference

```

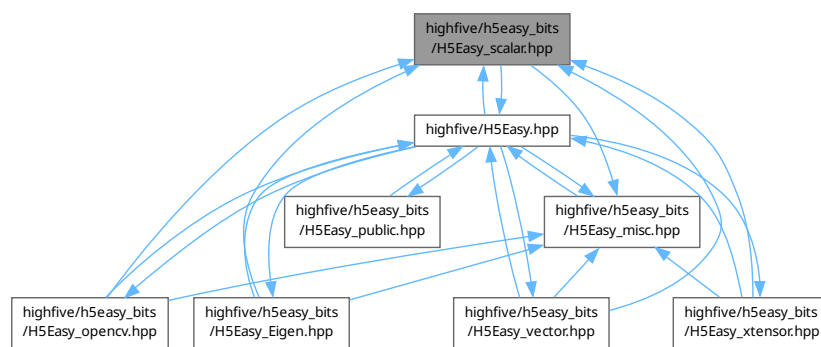
#include "../H5Easy.hpp"
#include "H5Easy_misc.hpp"

```

Include dependency graph for H5Easy_scalar.hpp:



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



Namespaces

- namespace **H5Easy**

Read/dump DataSets or Attribute using a minimalistic syntax. To this end, the functions are templated, and accept:

10.71 H5Easy_scalar.hpp

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c), 2017, Adrien Devresse <adrien.devresse@epfl.ch>
00003  *
00004  * Distributed under the Boost Software License, Version 1.0.
00005  * (See accompanying file LICENSE_1_0.txt or copy at
00006  * http://www.boost.org/LICENSE_1_0.txt)
00007  *
00008  */
00009 #pragma once
00010
00011 #include "../H5Easy.hpp"
00012 #include "H5Easy_misc.hpp"
00013
00014 namespace H5Easy {
00015
00016     namespace detail {
00017
00018         /*
00019         Base template for partial specialization: the fallback if specialized templates don't match.
00020         Used e.g. for scalars.
00021         */
00022         template <typename T, typename = void>
00023         struct io_impl {
00024             inline static DataSet dump(File& file,

```

```

00025         const std::string& path,
00026         const T& data,
00027         const DumpOptions& options) {
00028     DataSet dataset = initScalarDataSet(file, path, data, options);
00029     dataset.write(data);
00030     if (options.flush()) {
00031         file.flush();
00032     }
00033     return dataset;
00034 }
00035
00036 inline static T load(const File& file, const std::string& path) {
00037     DataSet dataset = file.getDataSet(path);
00038     T data;
00039     dataset.read(data);
00040     return data;
00041 }
00042
00043 inline static Attribute dumpAttribute(File& file,
00044         const std::string& path,
00045         const std::string& key,
00046         const T& data,
00047         const DumpOptions& options) {
00048     Attribute attribute = initScalarAttribute(file, path, key, data, options);
00049     attribute.write(data);
00050     if (options.flush()) {
00051         file.flush();
00052     }
00053     return attribute;
00054 }
00055
00056 inline static T loadAttribute(const File& file,
00057         const std::string& path,
00058         const std::string& key) {
00059     DataSet dataset = file.getDataSet(path);
00060     Attribute attribute = dataset.getAttribute(key);
00061     T data;
00062     attribute.read(data);
00063     return data;
00064 }
00065
00066 inline static DataSet dump_extend(File& file,
00067         const std::string& path,
00068         const T& data,
00069         const std::vector<size_t>& idx,
00070         const DumpOptions& options) {
00071     std::vector<size_t> ones(idx.size(), 1);
00072
00073     if (file.exist(path)) {
00074         DataSet dataset = file.getDataSet(path);
00075         std::vector<size_t> dims = dataset.getDimensions();
00076         std::vector<size_t> shape = dims;
00077         if (dims.size() != idx.size()) {
00078             throw detail::error(
00079                 file,
00080                 path,
00081                 "H5Easy::dump: Dimension of the index and the existing field do not match");
00082         }
00083         for (size_t i = 0; i < dims.size(); ++i) {
00084             shape[i] = std::max(dims[i], idx[i] + 1);
00085         }
00086         if (shape != dims) {
00087             dataset.resize(shape);
00088         }
00089         dataset.select(idx, ones).write(data);
00090         if (options.flush()) {
00091             file.flush();
00092         }
00093         return dataset;
00094     }
00095
00096     std::vector<size_t> shape = idx;
00097     const size_t unlim = DataSpace::UNLIMITED;
00098     std::vector<size_t> unlim_shape(idx.size(), unlim);
00099     std::vector<size_t> chunks(idx.size(), 10);
00100     if (options.isChunked()) {
00101         chunks = options.getChunkSize();
00102         if (chunks.size() != idx.size()) {
00103             throw error(file, path, "H5Easy::dump: Incorrect dimension ChunkSize");
00104         }
00105     }
00106     for (size_t& i: shape) {
00107         i++;
00108     }
00109     DataSpace dataspace = DataSpace(shape, unlim_shape);
00110     DataSetCreateProps props;
00111     props.add(Chunking(chunks));

```

```

00112     DataSet dataset = file.createDataSet(path, dataspace, AtomicType<T>(), props, {}, true);
00113     dataset.select(idx, ones).write(data);
00114     if (options.flush()) {
00115         file.flush();
00116     }
00117     return dataset;
00118 }
00119
00120 inline static T load_part(const File& file,
00121                          const std::string& path,
00122                          const std::vector<size_t>& idx) {
00123     std::vector<size_t> ones(idx.size(), 1);
00124     DataSet dataset = file.getDataSet(path);
00125     T data;
00126     dataset.select(idx, ones).read(data);
00127     return data;
00128 }
00129 };
00130
00131 } // namespace detail
00132 } // namespace H5Easy

```

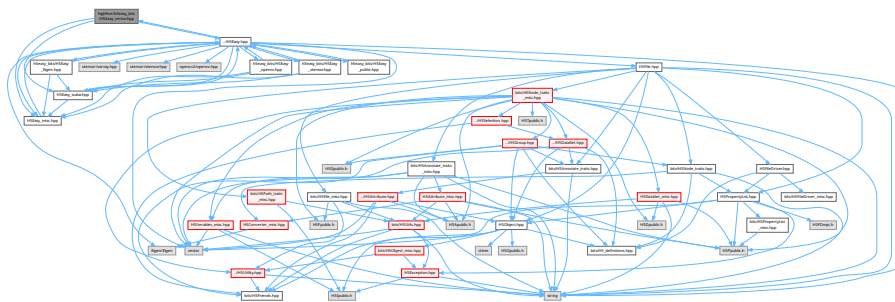
10.72 highfive/h5easy_bits/H5Easy_vector.hpp File Reference

```

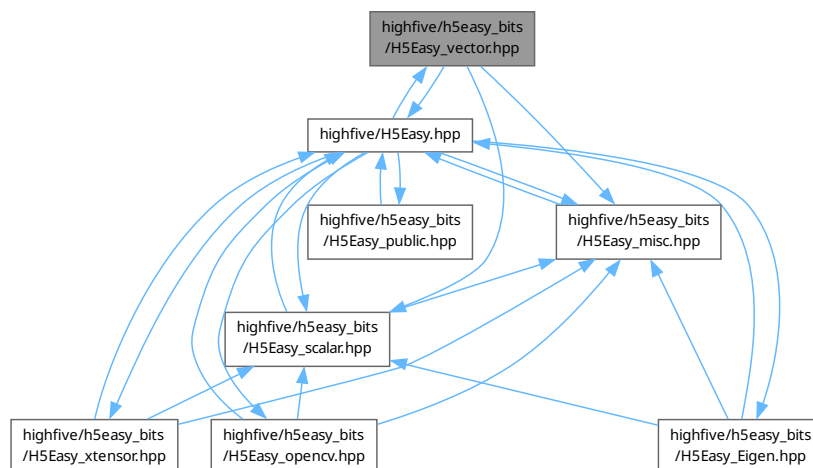
#include "../H5Easy.hpp"
#include "H5Easy_misc.hpp"
#include "H5Easy_scalar.hpp"

```

Include dependency graph for H5Easy_vector.hpp:



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



Namespaces

- namespace [H5Easy](#)

Read/dump DataSets or Attribute using a minimalistic syntax. To this end, the functions are templated, and accept:

10.73 H5Easy_vector.hpp

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c), 2017, Adrien Devresse <adrien.devresse@epfl.ch>
00003  *
00004  * Distributed under the Boost Software License, Version 1.0.
00005  * (See accompanying file LICENSE_1_0.txt or copy at
00006  * http://www.boost.org/LICENSE_1_0.txt)
00007  *
00008  */
00009 #pragma once
00010
00011 #include "../H5Easy.hpp"
00012 #include "H5Easy_misc.hpp"
00013 #include "H5Easy_scalar.hpp"
00014
00015 namespace H5Easy {
00016
00017 namespace detail {
00018
00019 template <class T>
00020 struct is_vector: std::false_type {};
00021 template <class T>
00022 struct is_vector<std::vector<T>: std::true_type {};
00023
00024 using HighFive::details::inspector;
00025
00026 template <typename T>
00027 struct io_impl<T, typename std::enable_if<is_vector<T>::value>::type> {
00028     inline static DataSet dump(File& file,
00029                               const std::string& path,
00030                               const T& data,
00031                               const DumpOptions& options) {
00032         using value_type = typename inspector<T>::base_type;
00033         auto dims = inspector<T>::getDimensions(data);
00034         DataSet dataset = initDataSet<value_type>(file,
00035                                                  path,
00036                                                  std::vector<size_t>(dims.begin(), dims.end()),
00037                                                  options);
00038         dataset.write(data);
00039         if (options.flush()) {
00040             file.flush();
00041         }
00042         return dataset;
00043     }
00044
00045     inline static T load(const File& file, const std::string& path) {
00046         DataSet dataset = file.getDataSet(path);
00047         T data;
00048         dataset.read(data);
00049         return data;
00050     }
00051
00052     inline static Attribute dumpAttribute(File& file,
00053                                          const std::string& path,
00054                                          const std::string& key,
00055                                          const T& data,
00056                                          const DumpOptions& options) {
00057         using value_type = typename inspector<T>::base_type;
00058         auto dims = inspector<T>::getDimensions(data);
00059         std::vector<size_t> shape(dims.begin(), dims.end());
00060         Attribute attribute = initAttribute<value_type>(file, path, key, shape, options);
00061         attribute.write(data);
00062         if (options.flush()) {
00063             file.flush();
00064         }
00065         return attribute;
00066     }
00067
00068     inline static T loadAttribute(const File& file,
00069                                  const std::string& path,
00070                                  const std::string& key) {
00071         DataSet dataset = file.getDataSet(path);

```

```

00072         Attribute attribute = dataset.getAttribute(key);
00073         T data;
00074         attribute.read(data);
00075         return data;
00076     }
00077 };
00078
00079 } // namespace detail
00080 } // namespace H5Easy

```

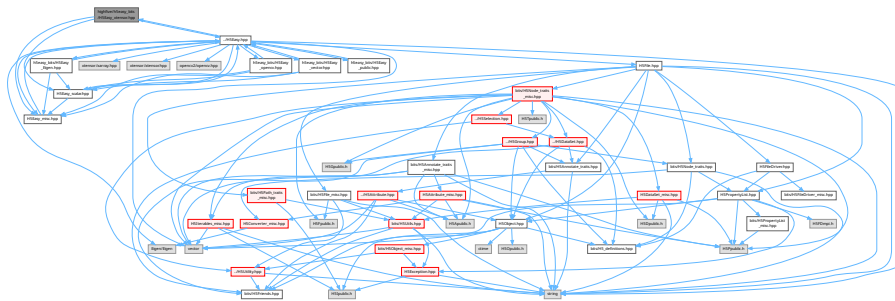
10.74 highfive/h5easy_bits/H5Easy_xtensor.hpp File Reference

```

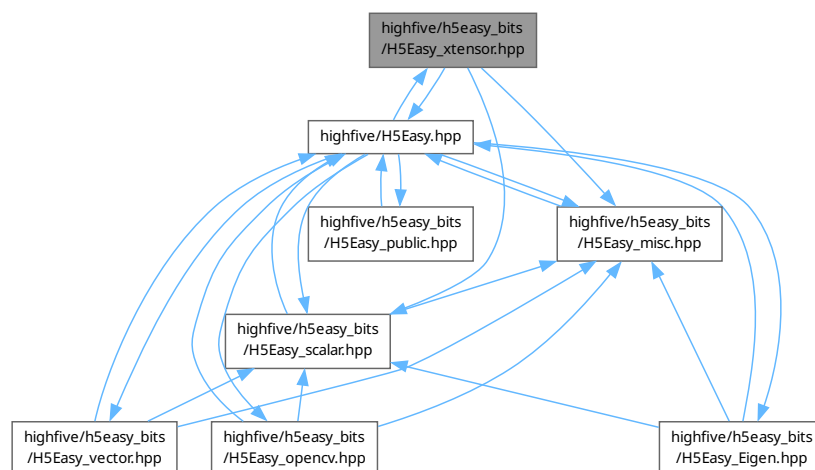
#include "../H5Easy.hpp"
#include "H5Easy_misc.hpp"
#include "H5Easy_scalar.hpp"

```

Include dependency graph for H5Easy_xtensor.hpp:



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



Namespaces

- namespace [H5Easy](#)

Read/dump DataSets or Attribute using a minimalistic syntax. To this end, the functions are templated, and accept:

10.75 H5Easy_xtensor.hpp

[Go to the documentation of this file.](#)

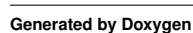
```

00001  /*
00002  *   Copyright (c), 2017, Adrien Devresse <adrien.devresse@epfl.ch>
00003  *
00004  *   Distributed under the Boost Software License, Version 1.0.
00005  *   (See accompanying file LICENSE_1_0.txt or copy at
00006  *    http://www.boost.org/LICENSE_1_0.txt)
00007  *
00008  */
00009  #pragma once
00010
00011  #include "../H5Easy.hpp"
00012  #include "H5Easy_misc.hpp"
00013  #include "H5Easy_scalar.hpp"
00014
00015  #ifdef H5_USE_XTENSOR
00016
00017  namespace H5Easy {
00018
00019  namespace detail {
00020
00021  template <typename T>
00022  struct io_impl<T, typename std::enable_if<xt::is_xexpression<T>::value>::type> {
00023      inline static std::vector<size_t> shape(const T& data) {
00024          return std::vector<size_t>(data.shape().cbegin(), data.shape().cend());
00025      }
00026
00027      inline static DataSet dump(File& file,
00028                                const std::string& path,
00029                                const T& data,
00030                                const DumpOptions& options) {
00031          using value_type = typename std::decay_t<T>::value_type;
00032          DataSet dataset = initDataSet<value_type>(file, path, shape(data), options);
00033          dataset.write_raw(data.data());
00034          if (options.flush()) {
00035              file.flush();
00036          }
00037          return dataset;
00038      }
00039
00040      inline static T load(const File& file, const std::string& path) {
00041          static_assert(
00042              xt::has_data_interface<T>::value,
00043              "Cannot load to xt::xfunction or xt::xgenerator, use e.g. xt::xtensor or xt::xarray");
00044          DataSet dataset = file.getDataSet(path);
00045          std::vector<size_t> dims = dataset.getDimensions();
00046          T data = T::from_shape(dims);
00047          dataset.read(data.data());
00048          return data;
00049      }
00050
00051      inline static Attribute dumpAttribute(File& file,
00052                                           const std::string& path,
00053                                           const std::string& key,
00054                                           const T& data,
00055                                           const DumpOptions& options) {
00056          using value_type = typename std::decay_t<T>::value_type;
00057          Attribute attribute = initAttribute<value_type>(file, path, key, shape(data), options);
00058          attribute.write_raw(data.data());
00059          if (options.flush()) {
00060              file.flush();
00061          }
00062          return attribute;
00063      }
00064
00065      inline static T loadAttribute(const File& file,
00066                                   const std::string& path,
00067                                   const std::string& key) {
00068          static_assert(
00069              xt::has_data_interface<T>::value,
00070              "Cannot load to xt::xfunction or xt::xgenerator, use e.g. xt::xtensor or xt::xarray");
00071          DataSet dataset = file.getDataSet(path);
00072          Attribute attribute = dataset.getAttribute(key);
00073          DataSpace dataspace = attribute.getSpace();
00074          std::vector<size_t> dims = dataspace.getDimensions();
00075          T data = T::from_shape(dims);
00076          attribute.read(data.data());
00077          return data;
00078      }
00079  };
00080
00081  } // namespace detail
00082  } // namespace H5Easy

```


10.76 highfive/H5Exception.hpp File Reference

Include dependency graph for H5Exception.hpp:



Classes

- class [HighFive::Exception](#)
Basic [HighFive Exception](#) class.
- class [HighFive::ObjectException](#)
Exception specific to [HighFive Object](#) interface.
- class [HighFive::DataTypeException](#)
Exception specific to [HighFive DataType](#) interface.
- class [HighFive::FileException](#)
Exception specific to [HighFive File](#) interface.
- class [HighFive::DataSpaceException](#)
Exception specific to [HighFive DataSpace](#) interface.
- class [HighFive::AttributeException](#)
Exception specific to [HighFive Attribute](#) interface.
- class [HighFive::DataSetException](#)
Exception specific to [HighFive DataSet](#) interface.
- class [HighFive::GroupException](#)
Exception specific to [HighFive Group](#) interface.
- class [HighFive::PropertyException](#)
Exception specific to [HighFive Property](#) interface.
- class [HighFive::ReferenceException](#)
Exception specific to [HighFive Reference](#) interface.

Namespaces

- namespace [HighFive](#)

10.77 H5Exception.hpp

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c), 2017, Adrien Devresse <adrien.devresse@epfl.ch>
00003  *
00004  * Distributed under the Boost Software License, Version 1.0.
00005  * (See accompanying file LICENSE_1_0.txt or copy at
00006  *      http://www.boost.org/LICENSE_1_0.txt)
00007  *
00008  */
00009 #pragma once
00010
00011 #include <memory>
00012 #include <stdexcept>
00013 #include <string>
00014
00015 #include <H5Ipublic.h>
00016
00017 namespace HighFive {
00018
00023 class Exception: public std::exception {
00024     public:
00025         Exception(const std::string& err_msg)
00026             : _errmsg(err_msg)
00027             , _next()
00028             , _err_major(0)
00029             , _err_minor(0) {}
00030
00031         virtual ~Exception() throw() {}
00032
00037         inline const char* what() const throw() override {
00038             return _errmsg.c_str();
00039         }
00040

```

```

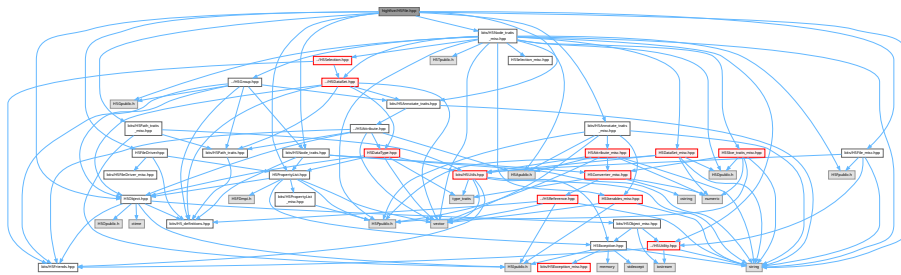
00045     inline virtual void setErrorMsg(const std::string& errmsg) {
00046         _errmsg = errmsg;
00047     }
00048
00049     inline Exception* nextException() const {
00050         return _next.get();
00051     }
00052
00053     inline hid_t getErrMajor() const {
00054         return _err_major;
00055     }
00056
00057     inline hid_t getErrMinor() const {
00058         return _err_minor;
00059     }
00060
00061 protected:
00062     std::string _errmsg;
00063     std::shared_ptr<Exception> _next;
00064     hid_t _err_major, _err_minor;
00065
00066     friend struct HDF5ErrMapper;
00067 };
00068
00069 class ObjectException: public Exception {
00070 public:
00071     ObjectException(const std::string& err_msg)
00072         : Exception(err_msg) {}
00073 };
00074
00075 class DataTypeException: public Exception {
00076 public:
00077     DataTypeException(const std::string& err_msg)
00078         : Exception(err_msg) {}
00079 };
00080
00081 class FileException: public Exception {
00082 public:
00083     FileException(const std::string& err_msg)
00084         : Exception(err_msg) {}
00085 };
00086
00087 class DataSpaceException: public Exception {
00088 public:
00089     DataSpaceException(const std::string& err_msg)
00090         : Exception(err_msg) {}
00091 };
00092
00093 class AttributeException: public Exception {
00094 public:
00095     AttributeException(const std::string& err_msg)
00096         : Exception(err_msg) {}
00097 };
00098
00099 class DataSetException: public Exception {
00100 public:
00101     DataSetException(const std::string& err_msg)
00102         : Exception(err_msg) {}
00103 };
00104
00105 class GroupException: public Exception {
00106 public:
00107     GroupException(const std::string& err_msg)
00108         : Exception(err_msg) {}
00109 };
00110
00111 class PropertyException: public Exception {
00112 public:
00113     PropertyException(const std::string& err_msg)
00114         : Exception(err_msg) {}
00115 };
00116
00117 class ReferenceException: public Exception {
00118 public:
00119     ReferenceException(const std::string& err_msg)
00120         : Exception(err_msg) {}
00121 };
00122 } // namespace HighFive
00123
00124 #include "bits/H5Exception_misc.hpp"

```

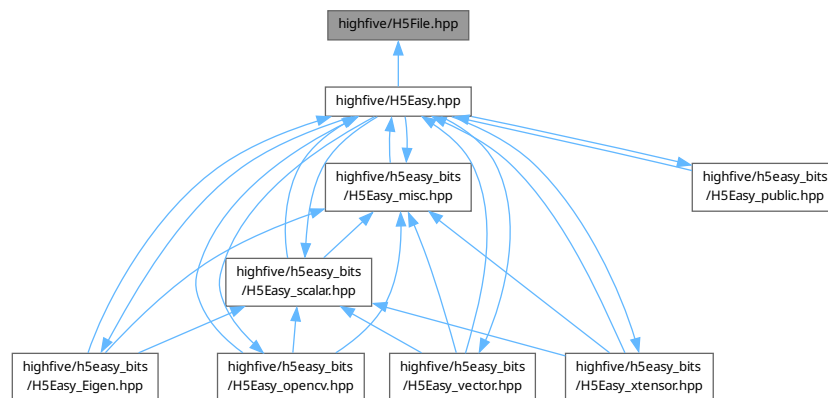
10.78 highfive/H5File.hpp File Reference

```
#include <string>
#include "H5FileDriver.hpp"
#include "H5Object.hpp"
#include "H5PropertyList.hpp"
#include "bits/H5Annotate_traits.hpp"
#include "bits/H5Node_traits.hpp"
#include "bits/H5Annotate_traits_misc.hpp"
#include "bits/H5File_misc.hpp"
#include "bits/H5Node_traits_misc.hpp"
#include "bits/H5Path_traits_misc.hpp"
```

Include dependency graph for H5File.hpp:



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



Classes

- class [HighFive::File](#)
File class.

Namespaces

- namespace [HighFive](#)

10.79 H5File.hpp

[Go to the documentation of this file.](#)

```

00001  /*
00002  *   Copyright (c), 2017, Adrien Devresse <adrien.devresse@epfl.ch>
00003  *
00004  *   Distributed under the Boost Software License, Version 1.0.
00005  *   (See accompanying file LICENSE_1_0.txt or copy at
00006  *    http://www.boost.org/LICENSE_1_0.txt)
00007  *
00008  */
00009  #pragma once
00010
00011  #include <string>
00012
00013  #include "H5FileDriver.hpp"
00014  #include "H5Object.hpp"
00015  #include "H5PropertyList.hpp"
00016  #include "bits/H5Annotate_traits.hpp"
00017  #include "bits/H5Node_traits.hpp"
00018
00019  namespace HighFive {
00020
00021  class File: public Object, public NodeTraits<File>, public AnnotateTraits<File> {
00022  public:
00023      const static ObjectType type = ObjectType::File;
00024
00025      enum : unsigned {
00026          ReadOnly = 0x00u,
00027          ReadWrite = 0x01u,
00028          Truncate = 0x02u,
00029          Excl = 0x04u,
00030          Debug = 0x08u,
00031          Create = 0x10u,
00032          Overwrite = Truncate,
00033          OpenOrCreate = ReadWrite | Create
00034      };
00035
00036      explicit File(const std::string& filename,
00037                  unsigned openFlags = ReadOnly,
00038                  const FileAccessProps& fileAccessProps = FileAccessProps::Default());
00039
00040      File(const std::string& filename,
00041          unsigned openFlags,
00042          const FileCreateProps& fileCreateProps,
00043          const FileAccessProps& fileAccessProps = FileAccessProps::Default());
00044
00045      const std::string& getName() const noexcept;
00046
00047      std::string getPath() const noexcept {
00048          return "/";
00049      }
00050
00051      hsize_t getMetadataBlockSize() const;
00052
00053      std::pair<H5F_libver_t, H5F_libver_t> getVersionBounds() const;
00054
00055      #if H5_VERSION_GE(1, 10, 1)
00056      H5F_fspace_strategy_t getFileSpaceStrategy() const;
00057
00058      hsize_t getFileSpacePageSize() const;
00059      #endif
00060
00061      void flush();
00062
00063      FileCreateProps getCreatePropertyList() const {
00064          return details::get_plist<FileCreateProps>(*this, H5Fget_create_plist);
00065      }
00066
00067      FileAccessProps getAccessPropertyList() const {
00068          return details::get_plist<FileAccessProps>(*this, H5Fget_access_plist);
00069      }
00070
00071  protected:
00072      File() = default;
00073      using Object::Object;
00074
00075  private:
00076      mutable std::string _filename{};
00077
00078      template <typename>
00079      friend class PathTraits;
00080  };
00081
00082
00083

```

```

00124 } // namespace HighFive
00125
00126 // H5File is the main user constructible -> bring in implementation headers
00127 #include "bits/H5Annotate_traits_misc.hpp"
00128 #include "bits/H5File_misc.hpp"
00129 #include "bits/H5Node_traits_misc.hpp"
00130 #include "bits/H5Path_traits_misc.hpp"

```

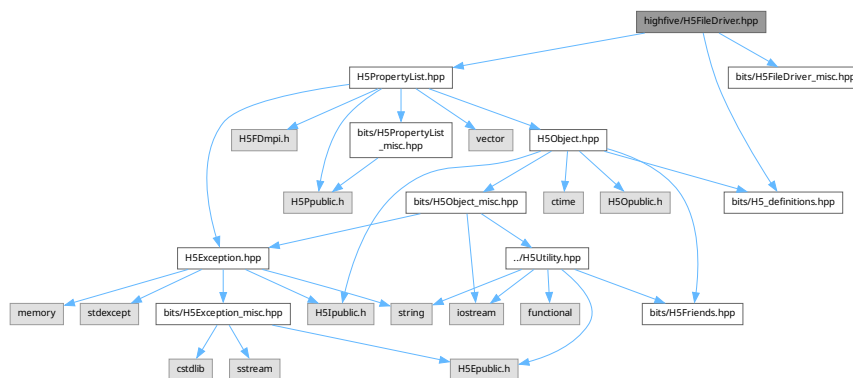
10.80 highfive/H5FileDriver.hpp File Reference

```

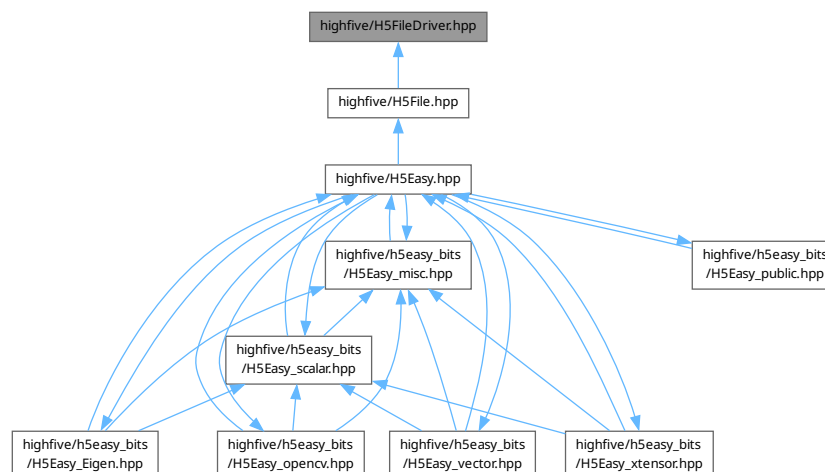
#include "H5PropertyList.hpp"
#include "bits/H5_definitions.hpp"
#include "bits/H5FileDriver_misc.hpp"

```

Include dependency graph for H5FileDriver.hpp:



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



Classes

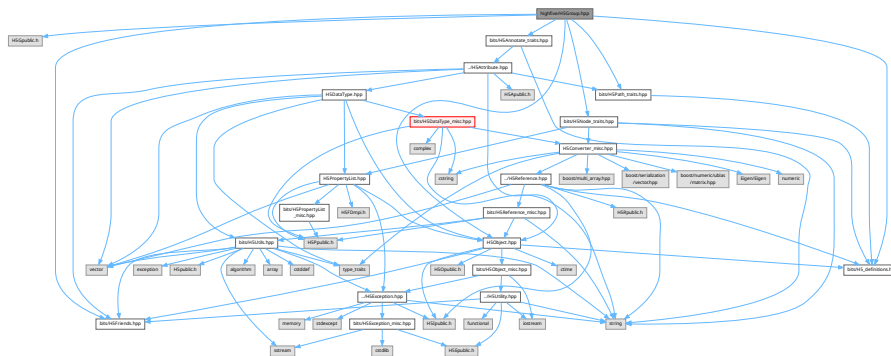
- class [HighFive::FileDriver](#)
file driver base concept
- class [HighFive::MPIOFileDriver](#)
MPIO Driver for Parallel HDF5.

- namespace **HighFive**

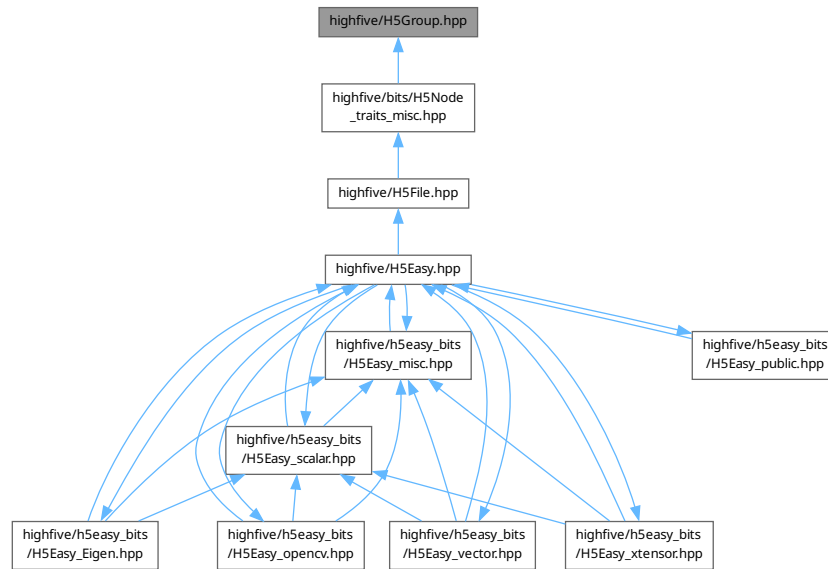
[Go to the documentation of this file.](#)

10.82 highfive/H5Group.hpp File Reference

Include dependency graph for H5Group.hpp:



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



Classes

- class [HighFive::Group](#)
Represents an hdf5 group.

Namespaces

- namespace [HighFive](#)

10.83 H5Group.hpp

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c), 2017, Adrien Devresse <adrien.devresse@epfl.ch>
00003  *
00004  * Distributed under the Boost Software License, Version 1.0.
00005  * (See accompanying file LICENSE_1_0.txt or copy at
00006  * http://www.boost.org/LICENSE_1_0.txt)
00007  *
00008  */
00009 #pragma once
00010
00011 #include <H5Gpublic.h>
00012
00013 #include "H5Object.hpp"
00014 #include "bits/H5Friends.hpp"
00015 #include "bits/H5_definitions.hpp"
00016 #include "bits/H5Annotate_traits.hpp"
00017 #include "bits/H5Node_traits.hpp"
00018 #include "bits/H5Path_traits.hpp"
00019
00020 namespace HighFive {
00021
00022 namespace detail {
00038 Group make_group(hid_t);
00039 } // namespace detail

```



```

00040
00043 class Group: public Object,
00044             public NodeTraits<Group>,
00045             public AnnotateTraits<Group>,
00046             public PathTraits<Group> {
00047     public:
00048         const static ObjectType type = ObjectType::Group;
00049
00051         H5_DEPRECATED("Default constructor creates unsafe uninitialized objects")
00052         Group() = default;
00053
00054         std::pair<unsigned int, unsigned int> getEstimatedLinkInfo() const;
00055
00057         GroupCreateProps getCreatePropertyList() const {
00058             return details::get_plist<GroupCreateProps>(*this, H5Gget_create_plist);
00059         }
00060
00061         Group(Object&& o) noexcept
00062             : Object(std::move(o)){};
00063
00064     protected:
00065         using Object::Object;
00066
00067         friend Group detail::make_group(hid_t);
00068         friend class File;
00069         friend class Reference;
00070 #if HIGHFIVE_HAS_FRIEND_DECLARATIONS
00071         template <typename Derivate>
00072         friend class ::HighFive::NodeTraits;
00073 #endif
00074 };
00075
00076 inline std::pair<unsigned int, unsigned int> Group::getEstimatedLinkInfo() const {
00077     auto gcpl = getCreatePropertyList();
00078     auto eli = EstimatedLinkInfo(gcpl);
00079     return std::make_pair(eli.getEntries(), eli.getNameLength());
00080 }
00081
00082 namespace detail {
00083 inline Group make_group(hid_t hid) {
00084     return Group(hid);
00085 }
00086 } // namespace detail
00087
00088 } // namespace HighFive

```

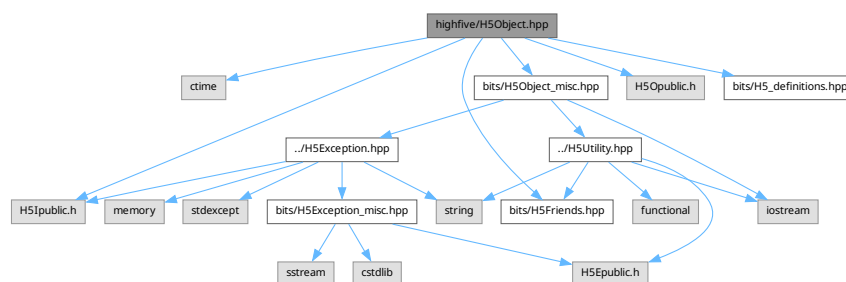
10.84 highfive/H5Object.hpp File Reference

```

#include <ctime>
#include <H5Ipublic.h>
#include <H5Opublic.h>
#include "bits/H5_definitions.hpp"
#include "bits/H5Friends.hpp"
#include "bits/H5Object_misc.hpp"

```

Include dependency graph for H5Object.hpp:



10.85 H5Object.hpp

[Go to the documentation of this file.](#)

```

00001  /*
00002   * Copyright (c), 2017, Adrien Devresse <adrien.devresse@epfl.ch>
00003   *
00004   * Distributed under the Boost Software License, Version 1.0.
00005   * (See accompanying file LICENSE_1_0.txt or copy at
00006   * http://www.boost.org/LICENSE_1_0.txt)
00007   *
00008  */
00009 #pragma once
00010
00011 #include <ctime>
00012
00013 #include <H5Ipublic.h>
00014 #include <H5Opublic.h>
00015
00016 #include "bits/H5_definitions.hpp"
00017 #include "bits/H5Friends.hpp"
00018
00019 namespace HighFive {
00020
00021     enum class ObjectType {
00022         File,
00023         Group,
00024         UserDataType,
00025         DataSpace,
00026         Dataset,
00027         Attribute,
00028         Other // Internal/custom object type
00029     };
00030
00031     namespace detail {
00032         Object make_object(hid_t hid);
00033     } // namespace detail
00034
00035     class Object {
00036     public:
00037         // move constructor, reuse hid
00038         Object(Object&& other) noexcept;
00039
00040         // decrease reference counter
00041         ~Object();
00042
00043         bool isValid() const noexcept;
00044
00045         hid_t getId() const noexcept;
00046
00047         ObjectInfo getInfo() const;
00048
00049         ObjectType getType() const;
00050
00051         // Check if refer to same object
00052         bool operator==(const Object& other) const noexcept {
00053             return _hid == other._hid;
00054         }
00055
00056     protected:
00057         // empty constructor
00058         Object();
00059
00060         // copy constructor, increase reference counter
00061         Object(const Object& other);
00062
00063         // Init with an low-level object id
00064         explicit Object(hid_t);
00065
00066         // Copy-Assignment operator
00067         Object& operator=(const Object& other);
00068
00069         hid_t _hid;
00070
00071     private:
00072         friend Object detail::make_object(hid_t);
00073         friend class Reference;
00074         friend class CompoundType;
00075
00076 #if HIGHFIVE_HAS_FRIEND_DECLARATIONS
00077         template <typename Derivate>
00078         friend class NodeTraits;
00079         template <typename Derivate>
00080         friend class AnnotateTraits;
00081         template <typename Derivate>

```

```

00118     friend class PathTraits;
00119 #endif
00120 };
00121
00122
00126 class ObjectInfo {
00127 public:
00130     H5_DEPRECATED("Deprecated since HighFive 2.2. Soon supporting VOL tokens")
00131     haddr_t getAddress() const noexcept;
00132
00133     size_t getRefCount() const noexcept;
00134
00135     time_t getCreationTime() const noexcept;
00136
00137     time_t getModificationTime() const noexcept;
00138
00139 protected:
00140 #if (H5Oget_info_vers < 3)
00141     H5O_info_t raw_info;
00142 #else
00143     // Use compat H5O_info1_t while getAddress() is supported (deprecated)
00144     H5O_info1_t raw_info;
00145 #endif
00146
00147     friend class Object;
00148 };
00149
00150 // namespace HighFive
00151 }
00152
00153 #include "bits/H5Object_misc.hpp"

```

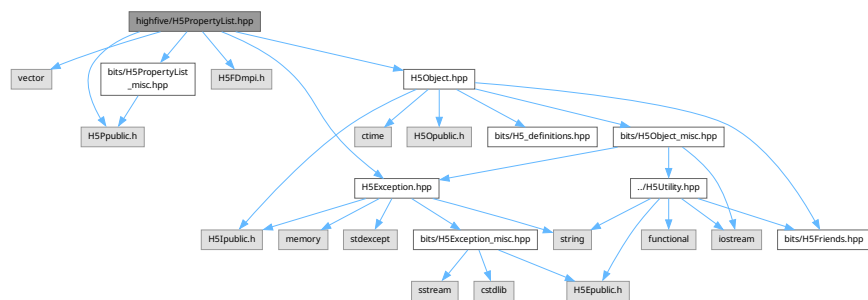
10.86 highfive/H5PropertyList.hpp File Reference

```

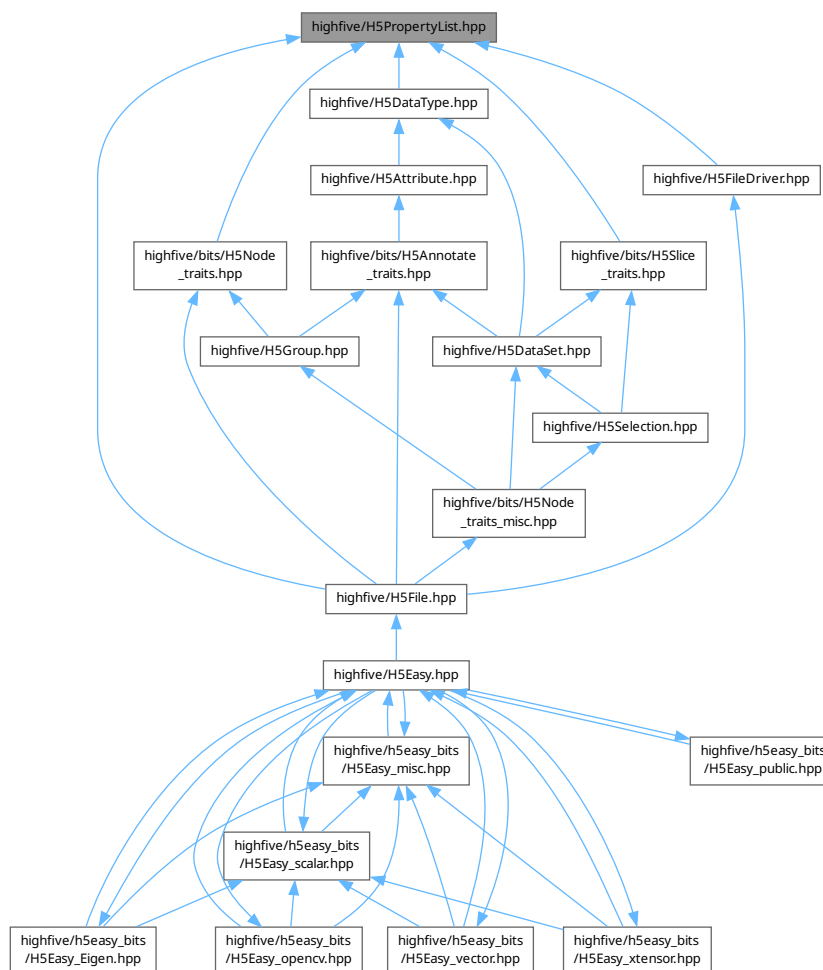
#include <vector>
#include <H5Ppublic.h>
#include <H5FDmpi.h>
#include "H5Exception.hpp"
#include "H5Object.hpp"
#include "bits/H5PropertyList_misc.hpp"

```

Include dependency graph for H5PropertyList.hpp:



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



Classes

- class `HighFive::PropertyListBase`
Base Class for Property lists, providing global default.
- class `HighFive::PropertyList< T >`
HDF5 property Lists.
- class `HighFive::RawPropertyList< T >`
- class `HighFive::MPIOFileAccess`
Configure MPI access for the file.
- class `HighFive::MPIOCollectiveMetadata`
Use collective MPI-IO for metadata read and write.
- class `HighFive::MPIOCollectiveMetadataRead`
Use collective MPI-IO for metadata read?
- class `HighFive::MPIOCollectiveMetadataWrite`
Use collective MPI-IO for metadata write?
- class `HighFive::FileVersionBounds`
Configure the version bounds for the file.

- class [HighFive::MetadataBlockSize](#)
Configure the metadata block size to use writing to files.
- class [HighFive::EstimatedLinkInfo](#)
Set hints as to how many links to expect and their average length.
- class [HighFive::Chunking](#)
- class [HighFive::Deflate](#)
- class [HighFive::Szip](#)
- class [HighFive::Shuffle](#)
- class [HighFive::AllocationTime](#)
When are datasets allocated?
- class [HighFive::Caching](#)
- class [HighFive::CreateIntermediateGroup](#)
- class [HighFive::UseCollectiveIO](#)
- class [HighFive::MpioNoCollectiveCause](#)
The cause for non-collective I/O.
- struct [HighFive::CreationOrder](#)
- class [HighFive::LinkCreationOrder](#)
Track and index creation order time.

Namespaces

- namespace [HighFive](#)

Typedefs

- using [HighFive::ObjectCreateProps](#) = [PropertyList](#)< [PropertyType::OBJECT_CREATE](#) >
- using [HighFive::FileCreateProps](#) = [PropertyList](#)< [PropertyType::FILE_CREATE](#) >
- using [HighFive::FileAccessProps](#) = [PropertyList](#)< [PropertyType::FILE_ACCESS](#) >
- using [HighFive::DataSetCreateProps](#) = [PropertyList](#)< [PropertyType::DATASET_CREATE](#) >
- using [HighFive::DataSetAccessProps](#) = [PropertyList](#)< [PropertyType::DATASET_ACCESS](#) >
- using [HighFive::DataTransferProps](#) = [PropertyList](#)< [PropertyType::DATASET_XFER](#) >
- using [HighFive::GroupCreateProps](#) = [PropertyList](#)< [PropertyType::GROUP_CREATE](#) >
- using [HighFive::GroupAccessProps](#) = [PropertyList](#)< [PropertyType::GROUP_ACCESS](#) >
- using [HighFive::DataTypeCreateProps](#) = [PropertyList](#)< [PropertyType::DATATYPE_CREATE](#) >
- using [HighFive::DataTypeAccessProps](#) = [PropertyList](#)< [PropertyType::DATATYPE_ACCESS](#) >
- using [HighFive::StringCreateProps](#) = [PropertyList](#)< [PropertyType::STRING_CREATE](#) >
- using [HighFive::AttributeCreateProps](#) = [PropertyList](#)< [PropertyType::ATTRIBUTE_CREATE](#) >
- using [HighFive::ObjectCopyProps](#) = [PropertyList](#)< [PropertyType::OBJECT_COPY](#) >
- using [HighFive::LinkCreateProps](#) = [PropertyList](#)< [PropertyType::LINK_CREATE](#) >
- using [HighFive::LinkAccessProps](#) = [PropertyList](#)< [PropertyType::LINK_ACCESS](#) >

Enumerations

- enum class [HighFive::PropertyType](#) : int {
[HighFive::OBJECT_CREATE](#) , [HighFive::FILE_CREATE](#) , [HighFive::FILE_ACCESS](#) , [HighFive::DATASET_CREATE](#)
 ,
[HighFive::DATASET_ACCESS](#) , [HighFive::DATASET_XFER](#) , [HighFive::GROUP_CREATE](#) , [HighFive::GROUP_ACCESS](#)
 ,
[HighFive::DATATYPE_CREATE](#) , [HighFive::DATATYPE_ACCESS](#) , [HighFive::STRING_CREATE](#) ,
[HighFive::ATTRIBUTE_CREATE](#) ,
[HighFive::OBJECT_COPY](#) , [HighFive::LINK_CREATE](#) , [HighFive::LINK_ACCESS](#) }
Types of property lists.

10.87 H5PropertyList.hpp

[Go to the documentation of this file.](#)

```

00001  /*
00002   * Copyright (c), 2017-2018, Adrien Devresse <adrien.devresse@epfl.ch>
00003   *                               Juan Hernando <juan.hernando@epfl.ch>
00004   * Distributed under the Boost Software License, Version 1.0.
00005   * (See accompanying file LICENSE_1_0.txt or copy at
00006   *      http://www.boost.org/LICENSE_1_0.txt)
00007   *
00008   */
00009  #pragma once
00010
00011  #include <vector>
00012
00013  #include <H5Ppublic.h>
00014
00015  // Required by MPIIOFileAccess
00016  #ifdef H5_HAVE_PARALLEL
00017  #include <H5FDmpi.h>
00018  #endif
00019
00020  #include "H5Exception.hpp"
00021  #include "H5Object.hpp"
00022
00023  namespace HighFive {
00024
00025  enum class PropertyType : int {
00026      OBJECT_CREATE,
00027      FILE_CREATE,
00028      FILE_ACCESS,
00029      DATASET_CREATE,
00030      DATASET_ACCESS,
00031      DATASET_XFER,
00032      GROUP_CREATE,
00033      GROUP_ACCESS,
00034      DATATYPE_CREATE,
00035      DATATYPE_ACCESS,
00036      STRING_CREATE,
00037      ATTRIBUTE_CREATE,
00038      OBJECT_COPY,
00039      LINK_CREATE,
00040      LINK_ACCESS,
00041  };
00042
00043  namespace details {
00044      template <typename T, typename U>
00045      T get_plist(const U& obj, hid_t (*f)(hid_t)) {
00046          auto hid = f(obj.getId());
00047          if (hid < 0) {
00048              HDF5ErrMapper::ToException<PropertyException>("Unable to get property list");
00049          }
00050          T t{};
00051          t._hid = hid;
00052          return t;
00053      }
00054  } // namespace details
00055
00056  class PropertyListBase: public Object {
00057  public:
00058      PropertyListBase() noexcept;
00059
00060      static const PropertyListBase& Default() noexcept {
00061          static const PropertyListBase plist{};
00062          return plist;
00063      }
00064
00065  private:
00066      template <typename T, typename U>
00067      friend T details::get_plist(const U&, hid_t (*f)(hid_t));
00068  };
00069
00070  template <PropertyType T>
00071  class PropertyList: public PropertyListBase {
00072  public:
00073      constexpr PropertyType getType() const noexcept {
00074          return T;
00075      }
00076
00077      template <typename P>
00078      void add(const P& property);
00079
00080      static const PropertyList<T>& Default() noexcept {
00081          return static_cast<const PropertyList<T>&>(PropertyListBase::Default());
00082      }
00083  };

```

```

00100
00101     protected:
00102         void _initializeIfNeeded();
00103     };
00104
00105     using ObjectCreateProps = PropertyList<PropertyType::OBJECT_CREATE>;
00106     using FileCreateProps = PropertyList<PropertyType::FILE_CREATE>;
00107     using FileAccessProps = PropertyList<PropertyType::FILE_ACCESS>;
00108     using DataSetCreateProps = PropertyList<PropertyType::DATASET_CREATE>;
00109     using DataSetAccessProps = PropertyList<PropertyType::DATASET_ACCESS>;
00110     using DataTransferProps = PropertyList<PropertyType::DATASET_XFER>;
00111     using GroupCreateProps = PropertyList<PropertyType::GROUP_CREATE>;
00112     using GroupAccessProps = PropertyList<PropertyType::GROUP_ACCESS>;
00113     using DataTypeCreateProps = PropertyList<PropertyType::DATATYPE_CREATE>;
00114     using DataTypeAccessProps = PropertyList<PropertyType::DATATYPE_ACCESS>;
00115     using StringCreateProps = PropertyList<PropertyType::STRING_CREATE>;
00116     using AttributeCreateProps = PropertyList<PropertyType::ATTRIBUTE_CREATE>;
00117     using ObjectCopyProps = PropertyList<PropertyType::OBJECT_COPY>;
00118     using LinkCreateProps = PropertyList<PropertyType::LINK_CREATE>;
00119     using LinkAccessProps = PropertyList<PropertyType::LINK_ACCESS>;
00120
00125     template <PropertyType T>
00126     class RawPropertyList: public PropertyList<T> {
00127     public:
00128         template <typename F, typename... Args>
00129         void add(const F& funct, const Args&... args);
00130     };
00131
00132     #ifdef H5_HAVE_PARALLEL
00133     class MPIOFileAccess {
00134     public:
00135         MPIOFileAccess(MPI_Comm comm, MPI_Info info);
00136
00137     private:
00138         friend FileAccessProps;
00139         void apply(const hid_t list) const;
00140
00141         MPI_Comm _comm;
00142         MPI_Info _info;
00143     };
00144
00145     class MPIOCollectiveMetadata {
00146     public:
00147         explicit MPIOCollectiveMetadata(bool collective = true);
00148         explicit MPIOCollectiveMetadata(const FileAccessProps& plist);
00149
00150         bool isCollectiveRead() const;
00151         bool isCollectiveWrite() const;
00152
00153     private:
00154         friend FileAccessProps;
00155         void apply(hid_t plist) const;
00156
00157         bool collective_read_;
00158         bool collective_write_;
00159     };
00160
00161     class MPIOCollectiveMetadataRead {
00162     public:
00163         explicit MPIOCollectiveMetadataRead(bool collective = true);
00164         explicit MPIOCollectiveMetadataRead(const FileAccessProps& plist);
00165
00166         bool isCollective() const;
00167
00168     private:
00169         friend FileAccessProps;
00170         friend MPIOCollectiveMetadata;
00171         void apply(hid_t plist) const;
00172
00173         bool collective_;
00174     };
00175
00176     class MPIOCollectiveMetadataWrite {
00177     public:
00178         explicit MPIOCollectiveMetadataWrite(bool collective = true);
00179         explicit MPIOCollectiveMetadataWrite(const FileAccessProps& plist);
00180
00181         bool isCollective() const;
00182
00183     private:
00184         friend FileAccessProps;
00185         friend MPIOCollectiveMetadata;
00186         void apply(hid_t plist) const;
00187
00188         bool collective_;
00189     };
00190
00191     class MPIOCollectiveMetadataReadWrite {
00192     public:
00193         explicit MPIOCollectiveMetadataReadWrite(bool collective = true);
00194         explicit MPIOCollectiveMetadataReadWrite(const FileAccessProps& plist);
00195
00196         bool isCollective() const;
00197
00198     private:
00199         friend FileAccessProps;
00200         friend MPIOCollectiveMetadata;
00201         void apply(hid_t plist) const;
00202
00203         bool collective_;
00204     };

```



```

00225     bool collective_;
00226 };
00227
00228 #endif
00229
00246 class FileVersionBounds {
00247 public:
00248     FileVersionBounds(H5F_libver_t low, H5F_libver_t high);
00249     explicit FileVersionBounds(const FileAccessProps& fapl);
00250
00251     std::pair<H5F_libver_t, H5F_libver_t> getVersion() const;
00252
00253 private:
00254     friend FileAccessProps;
00255     void apply(const hid_t list) const;
00256
00257     H5F_libver_t _low;
00258     H5F_libver_t _high;
00259 };
00260
00266 class MetadataBlockSize {
00267 public:
00268     explicit MetadataBlockSize(hsize_t size);
00269     explicit MetadataBlockSize(const FileAccessProps& fapl);
00270
00271     hsize_t getSize() const;
00272
00273 private:
00274     friend FileAccessProps;
00275     void apply(const hid_t list) const;
00276     hsize_t _size;
00277 };
00278
00279 #if H5_VERSION_GE(1, 10, 1)
00286 class FileSpaceStrategy {
00287 public:
00288     FileSpaceStrategy(H5F_fspace_strategy_t strategy, hbool_t persist, hsize_t threshold);
00289     explicit FileSpaceStrategy(const FileCreateProps& fcpl);
00290
00291     H5F_fspace_strategy_t getStrategy() const;
00292     hbool_t getPersist() const;
00293     hsize_t getThreshold() const;
00300
00301 private:
00302     friend FileCreateProps;
00303
00304     void apply(const hid_t list) const;
00305
00306     H5F_fspace_strategy_t _strategy;
00307     hbool_t _persist;
00308     hsize_t _threshold;
00309 };
00310
00320 class FileSpacePageSize {
00321 public:
00322     explicit FileSpacePageSize(hsize_t page_size);
00323     explicit FileSpacePageSize(const FileCreateProps& fcpl);
00324
00325     hsize_t getPageSize() const;
00330
00331 private:
00332     friend FileCreateProps;
00333     void apply(const hid_t list) const;
00334
00335     hsize_t _page_size;
00336 };
00337
00338 #ifndef H5_HAVE_PARALLEL
00350 class PageBufferSize {
00351 public:
00352     explicit PageBufferSize(size_t page_buffer_size,
00353                             unsigned min_meta_percent = 0,
00354                             unsigned min_raw_percent = 0);
00355
00356     explicit PageBufferSize(const FileAccessProps& fapl);
00362
00363     size_t getPageBufferSize() const;
00364     unsigned getMinMetaPercent() const;
00365     unsigned getMinRawPercent() const;
00366
00367 private:
00368     friend FileAccessProps;
00369
00370     void apply(hid_t list) const;
00371
00372     size_t _page_buffer_size;
00373     unsigned _min_meta;

```

```

00374     unsigned _min_raw;
00375 };
00376 #endif
00377 #endif
00378
00381 class EstimatedLinkInfo {
00382 public:
00387     explicit EstimatedLinkInfo(unsigned entries, unsigned length);
00388
00389     explicit EstimatedLinkInfo(const GroupCreateProps& gcpl);
00390
00392     unsigned getEntries() const;
00393
00395     unsigned getNameLength() const;
00396
00397 private:
00398     friend GroupCreateProps;
00399     void apply(hid_t hid) const;
00400     unsigned _entries;
00401     unsigned _length;
00402 };
00403
00404
00405 class Chunking {
00406 public:
00407     explicit Chunking(const std::vector<hsize_t>& dims);
00408     Chunking(const std::initializer_list<hsize_t>& items);
00409
00410     template <typename... Args>
00411     explicit Chunking(hsize_t item, Args... args);
00412
00413     explicit Chunking(DataSetCreateProps& plist, size_t max_dims = 32);
00414
00415     const std::vector<hsize_t>& getDimensions() const noexcept;
00416
00417 private:
00418     friend DataSetCreateProps;
00419     void apply(hid_t hid) const;
00420     std::vector<hsize_t> _dims;
00421 };
00422
00423 class Deflate {
00424 public:
00425     explicit Deflate(unsigned level);
00426
00427 private:
00428     friend DataSetCreateProps;
00429     friend GroupCreateProps;
00430     void apply(hid_t hid) const;
00431     const unsigned _level;
00432 };
00433
00434 class Szip {
00435 public:
00436     explicit Szip(unsigned options_mask = H5_SZIP_EC_OPTION_MASK,
00437                  unsigned pixels_per_block = H5_SZIP_MAX_PIXELS_PER_BLOCK);
00438
00439     unsigned getOptionsMask() const;
00440     unsigned getPixelsPerBlock() const;
00441
00442 private:
00443     friend DataSetCreateProps;
00444     void apply(hid_t hid) const;
00445     const unsigned _options_mask;
00446     const unsigned _pixels_per_block;
00447 };
00448
00449 class Shuffle {
00450 public:
00451     Shuffle() = default;
00452
00453 private:
00454     friend DataSetCreateProps;
00455     void apply(hid_t hid) const;
00456 };
00457
00463 class AllocationTime {
00464 public:
00465     explicit AllocationTime(H5D_alloc_time_t alloc_time);
00466     explicit AllocationTime(const DataSetCreateProps& dcpl);
00467
00468     H5D_alloc_time_t getAllocationTime();
00469
00470 private:
00471     friend DataSetCreateProps;
00472     void apply(hid_t dcpl) const;
00473

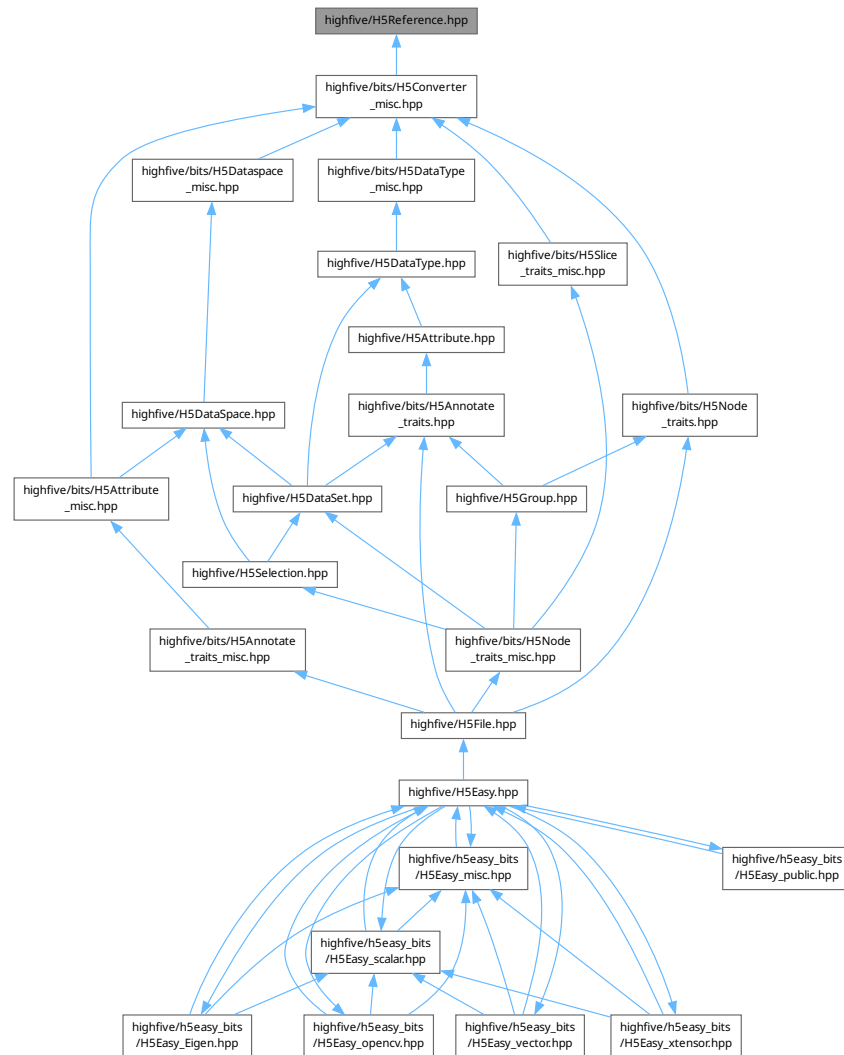
```

```

00474     H5D_alloc_time_t _alloc_time;
00475 };
00476
00477 class Caching {
00478 public:
00483     Caching(const size_t numSlots,
00484             const size_t cacheSize,
00485             const double w0 = static_cast<double>(H5D_CHUNK_CACHE_W0_DEFAULT));
00486
00487     explicit Caching(const DataSetCreateProps& dcpl);
00488
00489     size_t getNumSlots() const;
00490     size_t getCacheSize() const;
00491     double getW0() const;
00492
00493 private:
00494     friend DataSetAccessProps;
00495     void apply(hid_t hid) const;
00496     size_t _numSlots;
00497     size_t _cacheSize;
00498     double _w0;
00499 };
00500
00501 class CreateIntermediateGroup {
00502 public:
00503     explicit CreateIntermediateGroup(bool create = true);
00504
00505     explicit CreateIntermediateGroup(const ObjectCreateProps& ocpl);
00506     explicit CreateIntermediateGroup(const LinkCreateProps& lcpl);
00507
00508     bool isSet() const;
00509
00510 protected:
00511     void fromPropertyList(hid_t hid);
00512
00513 private:
00514     friend ObjectCreateProps;
00515     friend LinkCreateProps;
00516     void apply(hid_t hid) const;
00517     bool _create;
00518 };
00519
00520 #ifdef H5_HAVE_PARALLEL
00521 class UseCollectiveIO {
00522 public:
00523     explicit UseCollectiveIO(bool enable = true);
00524
00525     explicit UseCollectiveIO(const DataTransferProps& dxpl);
00526
00527     bool isCollective() const;
00528
00529 private:
00530     friend DataTransferProps;
00531     void apply(hid_t hid) const;
00532     bool _enable;
00533 };
00534 #endif
00535
00536 class MpioNoCollectiveCause {
00537 public:
00538     explicit MpioNoCollectiveCause(const DataTransferProps& dxpl);
00539
00540     bool wasCollective() const;
00541
00542     uint32_t getLocalCause() const;
00543
00544     uint32_t getGlobalCause() const;
00545
00546     std::pair<uint32_t, uint32_t> getCause() const;
00547
00548 private:
00549     friend DataTransferProps;
00550     uint32_t _local_cause;
00551     uint32_t _global_cause;
00552 };
00553 #endif
00554
00555 struct CreationOrder {
00556     enum _CreationOrder {
00557         Tracked = H5P_CRT_ORDER_TRACKED,
00558         Indexed = H5P_CRT_ORDER_INDEXED,
00559     };
00560 };
00561
00562 class LinkCreationOrder {
00563 public:
00564     explicit LinkCreationOrder(unsigned flags)

```


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



Classes

- class [HighFive::Reference](#)
An HDF5 (object) reference type.

Namespaces

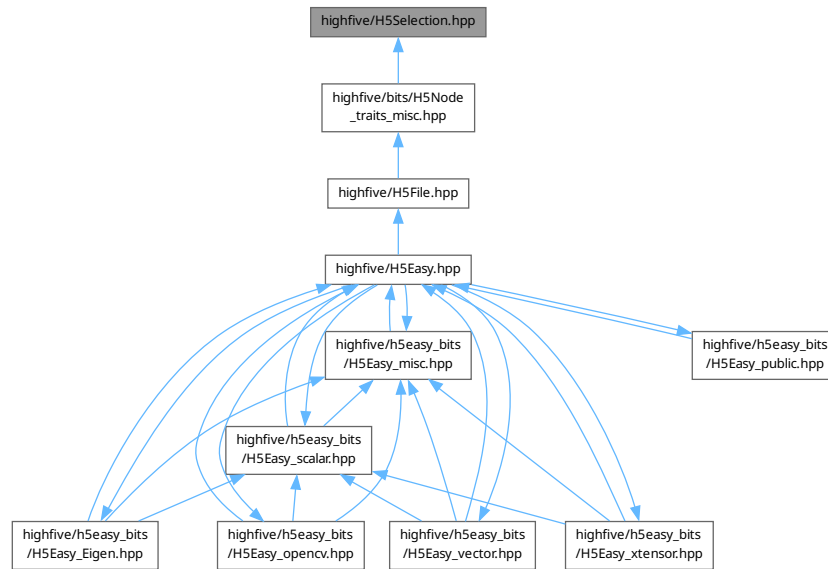
- namespace [HighFive](#)

10.89 H5Reference.hpp

[Go to the documentation of this file.](#)

```
00001 /*
00002  * Copyright (c), 2020, EPFL - Blue Brain Project
```


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



Classes

- class [HighFive::Selection](#)
Selection: represent a view on a slice/part of a dataset.

Namespaces

- namespace [HighFive](#)

10.91 H5Selection.hpp

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c), 2017, Adrien Devresse <adrien.devresse@epfl.ch>
00003  *
00004  * Distributed under the Boost Software License, Version 1.0.
00005  * (See accompanying file LICENSE_1_0.txt or copy at
00006  * http://www.boost.org/LICENSE_1_0.txt)
00007  *
00008  */
00009 #pragma once
00010
00011 #include "H5DataSet.hpp"
00012 #include "H5DataSpace.hpp"
00013 #include "bits/H5Slice_traits.hpp"
00014 #include "bits/H5Friends.hpp"
00015
00016 namespace HighFive {
00017
00018 namespace detail {
00019 Selection make_selection(const DataSpace&, const DataSpace&, const DataSet&);
00020 }
00021
00022 class Selection: public SliceTraits<Selection> {
00023 public:
00024     DataSpace getSpace() const noexcept;
00025
00026 };
  
```

```

00034
00040     DataSpace getMemSpace() const noexcept;
00041
00046     DataSet& getDataset() noexcept;
00047     const DataSet& getDataset() const noexcept;
00048
00052     const DataType getDataType() const;
00053
00054 protected:
00055     Selection(const DataSpace& memspace, const DataSpace& file_space, const DataSet& set);
00056
00057 private:
00058     DataSpace _mem_space, _file_space;
00059     DataSet _set;
00060
00061 #if HIGHFIVE_HAS_FRIEND_DECLARATIONS
00062     template <typename Derivate>
00063     friend class ::HighFive::SliceTraits;
00064 #endif
00065     friend Selection detail::make_selection(const DataSpace&, const DataSpace&, const DataSet&);
00066 };
00067
00068 } // namespace HighFive

```

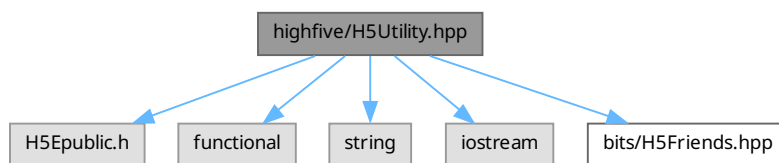
10.92 highfive/H5Utility.hpp File Reference

```

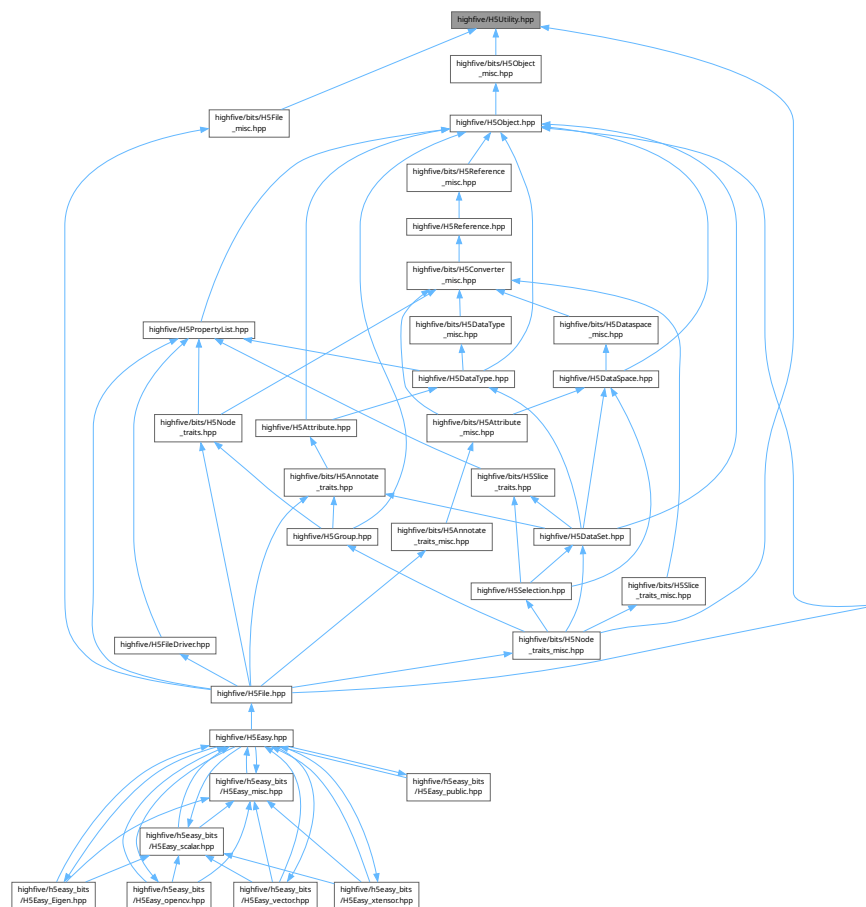
#include <H5Epublic.h>
#include <functional>
#include <string>
#include <iostream>
#include "bits/H5Friends.hpp"

```

Include dependency graph for H5Utility.hpp:



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



Classes

- class `HighFive::SilenceHDF5`
Utility class to disable HDF5 stack printing inside a scope.
- class `HighFive::Logger`
A logger with supporting basic functionality.

Namespaces

- namespace `HighFive`

Macros

- #define HIGHFIVE_LOG_LEVEL_DEBUG 10
- #define HIGHFIVE_LOG_LEVEL_INFO 20
- #define HIGHFIVE_LOG_LEVEL_WARN 30
- #define HIGHFIVE_LOG_LEVEL_ERROR 40
- #define HIGHFIVE_LOG_LEVEL HIGHFIVE_LOG_LEVEL_WARN
- #define HIGHFIVE_LOG_DEBUG(message) ::HighFive::detail::log(::HighFive::LogSeverity::Debug, (message), __FILE__, __LINE__);

- `#define HIGHFIVE_LOG_DEBUG_IF(cond, message)`
- `#define HIGHFIVE_LOG_INFO(message) ::HighFive::detail::log(::HighFive::LogSeverity::Info, (message), __FILE__, __LINE__);`
- `#define HIGHFIVE_LOG_INFO_IF(cond, message)`
- `#define HIGHFIVE_LOG_WARN(message) ::HighFive::detail::log(::HighFive::LogSeverity::Warn, (message), __FILE__, __LINE__);`
- `#define HIGHFIVE_LOG_WARN_IF(cond, message)`
- `#define HIGHFIVE_LOG_ERROR(message) ::HighFive::detail::log(::HighFive::LogSeverity::Error, (message), __FILE__, __LINE__);`
- `#define HIGHFIVE_LOG_ERROR_IF(cond, message)`

Enumerations

- enum class `HighFive::LogSeverity` { `HighFive::Debug` = `HIGHFIVE_LOG_LEVEL_DEBUG` , `HighFive::Info` = `HIGHFIVE_LOG_LEVEL_INFO` , `HighFive::Warn` = `HIGHFIVE_LOG_LEVEL_WARN` , `HighFive::Error` = `HIGHFIVE_LOG_LEVEL_ERROR` }

Functions

- `std::string HighFive::to_string(LogSeverity severity)`
- `void HighFive::default_logging_callback(LogSeverity severity, const std::string &message, const std::string &file, int line)`
- `Logger & HighFive::get_global_logger()`
Obtain a reference to the logger used by HighFive.
- `void HighFive::register_logging_callback(Logger::callback_type cb)`
Sets the callback that's used by the logger.

10.92.1 Macro Definition Documentation

10.92.1.1 HIGHFIVE_LOG_DEBUG

```
#define HIGHFIVE_LOG_DEBUG(  
    message ) ::HighFive::detail::log(::HighFive::LogSeverity::Debug, (message), __  
__FILE__, __LINE__);
```

10.92.1.2 HIGHFIVE_LOG_DEBUG_IF

```
#define HIGHFIVE_LOG_DEBUG_IF(  
    cond,  
    message )
```

Value:

```
if ((cond)) {  
    HIGHFIVE_LOG_DEBUG (message);  
}
```

10.92.1.3 HIGHFIVE_LOG_ERROR

```
#define HIGHFIVE_LOG_ERROR(  
    message ) ::HighFive::detail::log(::HighFive::LogSeverity::Error, (message), __  
__FILE__, __LINE__);
```

10.92.1.4 HIGHFIVE_LOG_ERROR_IF

```
#define HIGHFIVE_LOG_ERROR_IF(  
    cond,  
    message )
```

Value:

```
if ((cond)) {  
    HIGHFIVE_LOG_ERROR (message);  
}
```

10.92.1.5 HIGHFIVE_LOG_INFO

```
#define HIGHFIVE_LOG_INFO(  
    message ) ::HighFive::detail::log(::HighFive::LogSeverity::Info, (message), __FILE__, __LINE__);
```

10.92.1.6 HIGHFIVE_LOG_INFO_IF

```
#define HIGHFIVE_LOG_INFO_IF(  
    cond,  
    message )
```

Value:

```
if ((cond)) {  
    HIGHFIVE_LOG_INFO (message);  
}
```

10.92.1.7 HIGHFIVE_LOG_LEVEL

```
#define HIGHFIVE_LOG_LEVEL HIGHFIVE_LOG_LEVEL_WARN
```

10.92.1.8 HIGHFIVE_LOG_LEVEL_DEBUG

```
#define HIGHFIVE_LOG_LEVEL_DEBUG 10
```

10.92.1.9 HIGHFIVE_LOG_LEVEL_ERROR

```
#define HIGHFIVE_LOG_LEVEL_ERROR 40
```

10.92.1.10 HIGHFIVE_LOG_LEVEL_INFO

```
#define HIGHFIVE_LOG_LEVEL_INFO 20
```

10.92.1.11 HIGHFIVE_LOG_LEVEL_WARN

```
#define HIGHFIVE_LOG_LEVEL_WARN 30
```

10.92.1.12 HIGHFIVE_LOG_WARN

```
#define HIGHFIVE_LOG_WARN(
    message )    ::HighFive::detail::log(::HighFive::LogSeverity::Warn, (message), __FILE__, __LINE__);
```

10.92.1.13 HIGHFIVE_LOG_WARN_IF

```
#define HIGHFIVE_LOG_WARN_IF(
    cond,
    message )
```

Value:

```
if ((cond)) {
    HIGHFIVE_LOG_WARN((message));
}
```

10.93 H5Utility.hpp

[Go to the documentation of this file.](#)

```
00001 /*
00002  * Copyright (c), 2017, Blue Brain Project - EPFL (CH)
00003  *
00004  * Distributed under the Boost Software License, Version 1.0.
00005  * (See accompanying file LICENSE_1_0.txt or copy at
00006  * http://www.boost.org/LICENSE_1_0.txt)
00007  *
00008  */
00009
00010 #pragma once
00011
00012 #include <H5Epublic.h>
00013 #include <functional>
00014 #include <string>
00015 #include <iostream>
00016
00017 #include "bits/H5Friends.hpp"
00018
00019 namespace HighFive {
00020
00021 class SilenceHDF5 {
00022 public:
00023     inline SilenceHDF5(bool enable = true)
00024         : _client_data(nullptr) {
00025         H5Eget_auto2(H5E_DEFAULT, &_func, &_client_data);
00026         if (enable)
00027             H5Eset_auto2(H5E_DEFAULT, NULL, NULL);
00028     }
00029
00030     inline ~SilenceHDF5() {
00031         H5Eset_auto2(H5E_DEFAULT, _func, _client_data);
00032     }
00033
00034 private:
00035     H5E_auto2_t _func;
00036     void* _client_data;
00037 };
00038
00039 #define HIGHFIVE_LOG_LEVEL_DEBUG 10
00040 #define HIGHFIVE_LOG_LEVEL_INFO 20
00041 #define HIGHFIVE_LOG_LEVEL_WARN 30
00042 #define HIGHFIVE_LOG_LEVEL_ERROR 40
00043
00044 #ifndef HIGHFIVE_LOG_LEVEL
00045 #define HIGHFIVE_LOG_LEVEL HIGHFIVE_LOG_LEVEL_WARN
00046 #endif
00047
00048 enum class LogSeverity {
00049     Debug = HIGHFIVE_LOG_LEVEL_DEBUG,
00050     Info = HIGHFIVE_LOG_LEVEL_INFO,
00051     Warn = HIGHFIVE_LOG_LEVEL_WARN,
00052     Error = HIGHFIVE_LOG_LEVEL_ERROR
00053 }
```

```

00056 };
00057
00058 inline std::string to_string(LogSeverity severity) {
00059     switch (severity) {
00060         case LogSeverity::Debug:
00061             return "DEBUG";
00062         case LogSeverity::Info:
00063             return "INFO";
00064         case LogSeverity::Warn:
00065             return "WARN";
00066         case LogSeverity::Error:
00067             return "ERROR";
00068         default:
00069             return "??";
00070     }
00071 }
00072
00073 class Logger {
00074 public:
00075     using callback_type =
00076         std::function<void(LogSeverity, const std::string&, const std::string&, int)>;
00077
00078 public:
00079     Logger() = delete;
00080     Logger(const Logger&) = delete;
00081     Logger(Logger&&) = delete;
00082
00083     explicit Logger(callback_type cb)
00084         : _cb(std::move(cb)) {}
00085
00086     Logger& operator=(const Logger&) = delete;
00087     Logger& operator=(Logger&&) = delete;
00088
00089     inline void log(LogSeverity severity,
00090                     const std::string& message,
00091                     const std::string& file,
00092                     int line) {
00093         _cb(severity, message, file, line);
00094     }
00095
00096     inline void set_logging_callback(callback_type cb) {
00097         _cb = std::move(cb);
00098     }
00099
00100 private:
00101     callback_type _cb;
00102 };
00103
00104 inline void default_logging_callback(LogSeverity severity,
00105                                     const std::string& message,
00106                                     const std::string& file,
00107                                     int line) {
00108     std::clog << file << ": " << line << " :: " << to_string(severity) << message << std::endl;
00109 }
00110
00111 inline Logger& get_global_logger() {
00112     static Logger logger(&default_logging_callback);
00113     return logger;
00114 }
00115
00116 inline void register_logging_callback(Logger::callback_type cb) {
00117     auto& logger = get_global_logger();
00118     logger.set_logging_callback(std::move(cb));
00119 }
00120
00121 namespace detail {
00122     inline void log(LogSeverity severity,
00123                     const std::string& message,
00124                     const std::string& file,
00125                     int line) {
00126         auto& logger = get_global_logger();
00127         logger.log(severity, message, file, line);
00128     }
00129 } // namespace detail
00130
00131 #if HIGHFIVE_LOG_LEVEL <= HIGHFIVE_LOG_LEVEL_DEBUG
00132 #define HIGHFIVE_LOG_DEBUG(message) \
00133     ::HighFive::detail::log(::HighFive::LogSeverity::Debug, (message), __FILE__, __LINE__);
00134
00135 // Useful, for the common pattern: if ...; then log something.
00136 #define HIGHFIVE_LOG_DEBUG_IF(cond, message) \
00137     if ((cond)) { \
00138         HIGHFIVE_LOG_DEBUG(message); \
00139     }
00140 #else
00141 #define HIGHFIVE_LOG_DEBUG(message) ;

```

```

00166 #define HIGHFIVE_LOG_DEBUG_IF(cond, message) ;
00167 #endif
00168
00169 #if HIGHFIVE_LOG_LEVEL <= HIGHFIVE_LOG_LEVEL_INFO
00170 #define HIGHFIVE_LOG_INFO(message) \
00171     ::HighFive::detail::log(::HighFive::LogSeverity::Info, (message), __FILE__, __LINE__);
00172
00173 // Useful, for the common pattern: if ...; then log something.
00174 #define HIGHFIVE_LOG_INFO_IF(cond, message) \
00175     if ((cond)) { \
00176         HIGHFIVE_LOG_INFO((message)); \
00177     }
00178
00179 #else
00180 #define HIGHFIVE_LOG_INFO(message) ;
00181 #define HIGHFIVE_LOG_INFO_IF(cond, message) ;
00182 #endif
00183
00184
00185 #if HIGHFIVE_LOG_LEVEL <= HIGHFIVE_LOG_LEVEL_WARN
00186 #define HIGHFIVE_LOG_WARN(message) \
00187     ::HighFive::detail::log(::HighFive::LogSeverity::Warn, (message), __FILE__, __LINE__);
00188
00189 // Useful, for the common pattern: if ...; then log something.
00190 #define HIGHFIVE_LOG_WARN_IF(cond, message) \
00191     if ((cond)) { \
00192         HIGHFIVE_LOG_WARN((message)); \
00193     }
00194
00195 #else
00196 #define HIGHFIVE_LOG_WARN(message) ;
00197 #define HIGHFIVE_LOG_WARN_IF(cond, message) ;
00198 #endif
00199
00200 #if HIGHFIVE_LOG_LEVEL <= HIGHFIVE_LOG_LEVEL_ERROR
00201 #define HIGHFIVE_LOG_ERROR(message) \
00202     ::HighFive::detail::log(::HighFive::LogSeverity::Error, (message), __FILE__, __LINE__);
00203
00204 // Useful, for the common pattern: if ...; then log something.
00205 #define HIGHFIVE_LOG_ERROR_IF(cond, message) \
00206     if ((cond)) { \
00207         HIGHFIVE_LOG_ERROR((message)); \
00208     }
00209
00210 #else
00211 #define HIGHFIVE_LOG_ERROR(message) ;
00212 #define HIGHFIVE_LOG_ERROR_IF(cond, message) ;
00213 #endif
00214
00215 } // namespace HighFive

```

10.94 /builddir/build/BUILD/HighFive-2.7.1/README.md File Reference

Index

- [/builddir/build/BUILD/HighFive-2.7.1/CHANGELOG.md, 203](#)
- [/builddir/build/BUILD/HighFive-2.7.1/README.md, 352](#)
- [_CreationOrder](#)
 - [HighFive::CreationOrder, 78](#)
- [_H5_STRUCT_PADDING](#)
 - [H5DataType_misc.hpp, 231](#)
- [_err_major](#)
 - [HighFive::Exception, 118](#)
- [_err_minor](#)
 - [HighFive::Exception, 118](#)
- [_errmsg](#)
 - [HighFive::Exception, 118](#)
- [_file_obj](#)
 - [HighFive::PathTraits< Derivate >, 177](#)
- [_hid](#)
 - [HighFive::Object, 172](#)
- [_initializeIfNeeded](#)
 - [HighFive::PropertyList< T >, 182](#)
- [_next](#)
 - [HighFive::Exception, 118](#)
- [~Exception](#)
 - [HighFive::Exception, 116](#)
- [~Object](#)
 - [HighFive::Object, 171](#)
- [~SilenceHDF5](#)
 - [HighFive::SilenceHDF5, 197](#)
- [add](#)
 - [HighFive::PropertyList< T >, 182](#)
 - [HighFive::RawPropertyList< T >, 187](#)
- [AllocationTime](#)
 - [HighFive::AllocationTime, 45](#)
- [apply](#)
 - [HighFive::HyperSlab, 145](#)
- [Array](#)
 - [HighFive, 40](#)
- [at](#)
 - [HighFive::FixedLenStringArray< N >, 133](#)
- [AtomicType](#)
 - [HighFive::AtomicType< char\[StrLen\]>, 56](#)
 - [HighFive::AtomicType< FixedLenStringArray< StrLen >, 59](#)
 - [HighFive::AtomicType< std::complex< T >, 61](#)
 - [HighFive::AtomicType< T >, 52–54](#)
- [Attribute](#)
 - [HighFive, 41](#)
 - [HighFive::Attribute, 64](#)
 - [HighFive::DataSpace, 93](#)
 - [HighFive::DataType, 100](#)
- [ATTRIBUTE_CREATE](#)
 - [HighFive, 41](#)
- [AttributeCreateProps](#)
 - [HighFive, 38](#)
- [AttributeException](#)
 - [HighFive::AttributeException, 69](#)
- [back](#)
 - [HighFive::FixedLenStringArray< N >, 133](#)
- [base_type](#)
 - [HighFive::CompoundType::member_def, 151](#)
- [basic_type](#)
 - [HighFive::AtomicType< T >, 52](#)
- [begin](#)
 - [HighFive::FixedLenStringArray< N >, 133, 134](#)
- [BitField](#)
 - [HighFive, 40](#)
- [block](#)
 - [HighFive::RegularHyperSlab, 193](#)
- [Caching](#)
 - [HighFive::Caching, 69](#)
- [callback_type](#)
 - [HighFive::Logger, 148](#)
- [cbegin](#)
 - [HighFive::FixedLenStringArray< N >, 134](#)
- [cend](#)
 - [HighFive::FixedLenStringArray< N >, 134](#)
- [Chunking](#)
 - [HighFive::Chunking, 70, 71](#)
- [clone](#)
 - [HighFive::DataSpace, 92](#)
- [commit](#)
 - [HighFive::CompoundType, 75](#)
 - [HighFive::EnumType< T >, 113](#)
- [Compound](#)
 - [HighFive, 40](#)
- [CompoundType](#)
 - [HighFive::CompoundType, 74](#)
 - [HighFive::DataType, 100](#)
 - [HighFive::Object, 172](#)
- [compress](#)
 - [H5Easy::DumpOptions, 104](#)
- [Compression](#)
 - [H5Easy::Compression, 76, 77](#)
- [compute_total_size](#)
 - [HighFive, 41](#)
- [const_iterator](#)
 - [HighFive::FixedLenStringArray< N >, 132](#)
- [const_reverse_iterator](#)

- HighFive::FixedLenStringArray< N >, 132
- count
 - HighFive::RegularHyperSlab, 193
- Create
 - H5Easy, 28
 - HighFive::File, 122
- create_and_check_datatype
 - HighFive, 41
- create_datatype
 - HighFive, 42
- create_enum_boolean
 - HighFive, 42
- create_ref
 - HighFive::Reference, 188
- createAttribute
 - HighFive::AnnotateTraits< Derivate >, 46, 47
- createDataSet
 - HighFive::NodeTraits< Derivate >, 163, 164
- createExternalLink
 - HighFive::NodeTraits< Derivate >, 164
- createGroup
 - HighFive::NodeTraits< Derivate >, 165
- CreateIntermediateGroup
 - HighFive::CreateIntermediateGroup, 78
- createSoftLink
 - HighFive::NodeTraits< Derivate >, 165, 166
- CRT_ORDER
 - HighFive, 40
- data
 - HighFive::FixedLenStringArray< N >, 134
- DataSet
 - HighFive::DataSet, 83
 - HighFive::DataSpace, 93
 - HighFive::DataType, 100
- Dataset
 - HighFive, 41
- DATASET_ACCESS
 - HighFive, 41
- DATASET_CREATE
 - HighFive, 41
- DATASET_XFER
 - HighFive, 41
- DataSetAccessProps
 - HighFive, 38
- DataSetCreateProps
 - HighFive, 38
- DataSetException
 - HighFive::DataSetException, 88
- DataSpace
 - HighFive, 41
 - HighFive::DataSpace, 91, 92
- dataspace_null
 - HighFive::DataSpace, 90
- dataspace_scalar
 - HighFive::DataSpace, 90
- DataSpaceException
 - HighFive::DataSpaceException, 96
- DataspaceType
 - HighFive::DataSpace, 90
- DataSetAccessProps
 - HighFive, 38
- DataTypeAccessProps
 - HighFive, 38
- DataTypeClass
 - HighFive, 39
- DataTypeCreateProps
 - HighFive, 38
- DataTypeException
 - HighFive::DataTypeException, 102
- Debug
 - HighFive, 40
 - HighFive::File, 122
- Default
 - HighFive::PropertyList< T >, 182
 - HighFive::PropertyListBase, 184
- default_logging_callback
 - HighFive, 42
- Deflate
 - HighFive::Deflate, 103
- deleteAttribute
 - HighFive::AnnotateTraits< Derivate >, 47
- Deprecated List, 15
- dereference
 - HighFive::Reference, 189
- dump
 - H5Easy, 29–31
- dumpAttribute
 - H5Easy, 31, 32
- DumpMode
 - H5Easy, 28
- DumpOptions
 - H5Easy::DumpOptions, 104
- ElementSet
 - HighFive::ElementSet, 108, 109
- empty
 - HighFive::DataType, 98
 - HighFive::FixedLenStringArray< N >, 134
- end
 - HighFive::FixedLenStringArray< N >, 134
- Enum
 - HighFive, 40
- EnumType
 - HighFive::EnumType< T >, 112, 113
- Error
 - HighFive, 40
- EstimatedLinkInfo
 - HighFive::EstimatedLinkInfo, 114
- Exception
 - HighFive::Exception, 116
- Excl
 - HighFive::File, 122
- exist

- HighFive::NodeTraits< Derivate >, 166
- External
 - HighFive, 40
- False
 - H5Easy, 29
- File
 - HighFive, 41
 - HighFive::DataSpace, 93
 - HighFive::DataType, 100
 - HighFive::File, 122, 123
 - HighFive::Group, 141
- FILE_ACCESS
 - HighFive, 41
- FILE_CREATE
 - HighFive, 41
- FileAccessProps
 - HighFive, 38
- FileCreateProps
 - HighFive, 38
- FileException
 - HighFive::FileException, 129
- FileVersionBounds
 - HighFive::FileVersionBounds, 130
- find_first_atomic_member_size
 - HighFive, 42
- FixedLenStringArray
 - HighFive::FixedLenStringArray< N >, 132, 133
- Float
 - HighFive, 40
- float16_t
 - HighFive, 38
- Flush
 - H5Easy, 29
- flush
 - H5Easy::DumpOptions, 104
 - HighFive::File, 123
- From
 - HighFive::DataSpace, 92
- FromCharArrayStrings
 - HighFive::DataSpace, 92
- fromHDF5Sizes
 - HighFive::RegularHyperSlab, 192
- fromPropertyList
 - HighFive::CreateIntermediateGroup, 78
 - HighFive::LinkCreationOrder, 147
- front
 - HighFive::FixedLenStringArray< N >, 134
- get
 - H5Easy::Compression, 77
- get_global_logger
 - HighFive, 42
- getAccessPropertyList
 - HighFive::DataSet, 83
 - HighFive::File, 123
- getAddress
 - HighFive::ObjectInfo, 175
- getAllocationTime
 - HighFive::AllocationTime, 46
- getAttribute
 - HighFive::AnnotateTraits< Derivate >, 48
- getCacheSize
 - HighFive::Caching, 70
- getCause
 - HighFive::MpioNoCollectiveCause, 160
- getChunkSize
 - H5Easy::DumpOptions, 105
- getClass
 - HighFive::DataType, 98
- getCompressionLevel
 - H5Easy::DumpOptions, 105
- getCreatePropertyList
 - HighFive::Attribute, 64
 - HighFive::DataSet, 83
 - HighFive::DataType, 98
 - HighFive::File, 123
 - HighFive::Group, 140
- getCreationTime
 - HighFive::ObjectInfo, 175
- getDataSet
 - HighFive::NodeTraits< Derivate >, 166
- getDataset
 - HighFive::Selection, 195
- getDataType
 - HighFive::Attribute, 64
 - HighFive::DataSet, 83
 - HighFive::Selection, 196
- getDimensions
 - HighFive::Chunking, 71
 - HighFive::DataSet, 83
 - HighFive::DataSpace, 92
- getElementCount
 - HighFive::DataSet, 84
 - HighFive::DataSpace, 92
- getEntries
 - HighFive::EstimatedLinkInfo, 114
- getErrMajor
 - HighFive::Exception, 116
- getErrMinor
 - HighFive::Exception, 116
- getEstimatedLinkInfo
 - HighFive::Group, 140
- getFile
 - HighFive::PathTraits< Derivate >, 177
- getFlags
 - HighFive::LinkCreationOrder, 147
- getGlobalCause
 - HighFive::MpioNoCollectiveCause, 160
- getGroup
 - HighFive::NodeTraits< Derivate >, 167
- getId
 - HighFive::Object, 171
- getInfo
 - HighFive::Object, 171
- getLinkType
 - HighFive::NodeTraits< Derivate >, 167

- getLocalCause
 - HighFive::MpioNoCollectiveCause, 160
- getMaxDimensions
 - HighFive::DataSpace, 93
- getMembers
 - HighFive::CompoundType, 75
- getMemSpace
 - HighFive::Attribute, 64
 - HighFive::DataSet, 84
 - HighFive::Selection, 196
- getMetadataBlockSize
 - HighFive::File, 123
- getModificationTime
 - HighFive::ObjectInfo, 175
- getName
 - HighFive::Attribute, 65
 - HighFive::File, 123
- getNameLength
 - HighFive::EstimatedLinkInfo, 114
- getNumberAttributes
 - HighFive::AnnotateTraits< Derivate >, 48
- getNumberDimensions
 - HighFive::DataSpace, 93
- getNumberObjects
 - HighFive::NodeTraits< Derivate >, 167
- getNumSlots
 - HighFive::Caching, 70
- getObjectName
 - HighFive::NodeTraits< Derivate >, 168
- getObjectType
 - HighFive::NodeTraits< Derivate >, 168
- getOffset
 - HighFive::DataSet, 84
- getOptionsMask
 - HighFive::Szip, 201
- getPath
 - HighFive::File, 124
 - HighFive::PathTraits< Derivate >, 177
- getPixelsPerBlock
 - HighFive::Szip, 201
- getRefCount
 - HighFive::ObjectInfo, 175
- getShape
 - H5Easy, 32
- getSize
 - H5Easy, 33
 - HighFive::DataType, 99
 - HighFive::MetadataBlockSize, 153
- getSpace
 - HighFive::Attribute, 65
 - HighFive::DataSet, 84
 - HighFive::Selection, 196
- getStorageSize
 - HighFive::Attribute, 65
 - HighFive::DataSet, 84
- getString
 - HighFive::FixedLenStringArray< N >, 135
- getType
 - HighFive::Object, 171
 - HighFive::PropertyList< T >, 182
 - HighFive::Reference, 189
- getVersion
 - HighFive::FileVersionBounds, 130
- getVersionBounds
 - HighFive::File, 124
- getW0
 - HighFive::Caching, 70
- Group
 - HighFive, 41
 - HighFive::Group, 140
- GROUP_ACCESS
 - HighFive, 41
- GROUP_CREATE
 - HighFive, 41
- GroupAccessProps
 - HighFive, 39
- GroupCreateProps
 - HighFive, 39
- GroupException
 - HighFive::GroupException, 143
- H5_definitions.hpp
 - H5_DEPRECATED, 204
- H5_DEPRECATED
 - H5_definitions.hpp, 204
- H5DataType.hpp
 - HIGHFIVE_REGISTER_TYPE, 297
- H5DataType_misc.hpp
 - _H5_STRUCT_PADDING, 231
- H5Easy, 27
 - Create, 28
 - dump, 29–31
 - dumpAttribute, 31, 32
 - DumpMode, 28
 - False, 29
 - Flush, 29
 - getShape, 32
 - getSize, 33
 - load, 33
 - loadAttribute, 34
 - Overwrite, 28
 - True, 29
- H5Easy::Compression, 75
 - Compression, 76, 77
 - get, 77
- H5Easy::DumpOptions, 103
 - compress, 104
 - DumpOptions, 104
 - flush, 104
 - getChunkSize, 105
 - getCompressionLevel, 105
 - isChunked, 105
 - overwrite, 105
 - set, 105, 106
 - setChunkSize, 106, 108
- H5Utility.hpp
 - HIGHFIVE_LOG_DEBUG, 348

- HIGHFIVE_LOG_DEBUG_IF, [348](#)
- HIGHFIVE_LOG_ERROR, [348](#)
- HIGHFIVE_LOG_ERROR_IF, [348](#)
- HIGHFIVE_LOG_INFO, [349](#)
- HIGHFIVE_LOG_INFO_IF, [349](#)
- HIGHFIVE_LOG_LEVEL, [349](#)
- HIGHFIVE_LOG_LEVEL_DEBUG, [349](#)
- HIGHFIVE_LOG_LEVEL_ERROR, [349](#)
- HIGHFIVE_LOG_LEVEL_INFO, [349](#)
- HIGHFIVE_LOG_LEVEL_WARN, [349](#)
- HIGHFIVE_LOG_WARN, [349](#)
- HIGHFIVE_LOG_WARN_IF, [350](#)
- Hard
 - HighFive, [40](#)
- hasAttribute
 - HighFive::AnnotateTraits< Derivate >, [48](#)
- HDF5ErrMapper
 - HighFive::Exception, [117](#)
- HighFive, [34](#)
 - Array, [40](#)
 - Attribute, [41](#)
 - ATTRIBUTE_CREATE, [41](#)
 - AttributeCreateProps, [38](#)
 - BitField, [40](#)
 - Compound, [40](#)
 - compute_total_size, [41](#)
 - create_and_check_datatype, [41](#)
 - create_datatype, [42](#)
 - create_enum_boolean, [42](#)
 - CRT_ORDER, [40](#)
 - Dataset, [41](#)
 - DATASET_ACCESS, [41](#)
 - DATASET_CREATE, [41](#)
 - DATASET_XFER, [41](#)
 - DataSetAccessProps, [38](#)
 - DataSetCreateProps, [38](#)
 - DataSpace, [41](#)
 - DataTransferProps, [38](#)
 - DATATYPE_ACCESS, [41](#)
 - DATATYPE_CREATE, [41](#)
 - DataTypeAccessProps, [38](#)
 - DataTypeClass, [39](#)
 - DataTypeCreateProps, [38](#)
 - Debug, [40](#)
 - default_logging_callback, [42](#)
 - Enum, [40](#)
 - Error, [40](#)
 - External, [40](#)
 - File, [41](#)
 - FILE_ACCESS, [41](#)
 - FILE_CREATE, [41](#)
 - FileAccessProps, [38](#)
 - FileCreateProps, [38](#)
 - find_first_atomic_member_size, [42](#)
 - Float, [40](#)
 - float16_t, [38](#)
 - get_global_logger, [42](#)
 - Group, [41](#)
 - GROUP_ACCESS, [41](#)
 - GROUP_CREATE, [41](#)
 - GroupAccessProps, [39](#)
 - GroupCreateProps, [39](#)
 - Hard, [40](#)
 - IndexType, [40](#)
 - Info, [40](#)
 - Integer, [40](#)
 - Invalid, [40](#)
 - LINK_ACCESS, [41](#)
 - LINK_CREATE, [41](#)
 - LinkAccessProps, [39](#)
 - LinkCreateProps, [39](#)
 - LinkType, [40](#)
 - LogSeverity, [40](#)
 - NAME, [40](#)
 - OBJECT_COPY, [41](#)
 - OBJECT_CREATE, [41](#)
 - ObjectCopyProps, [39](#)
 - ObjectCreateProps, [39](#)
 - ObjectType, [41](#)
 - Opaque, [40](#)
 - operator&, [42](#)
 - operator|, [42](#)
 - Other, [40](#), [41](#)
 - PropertyType, [41](#)
 - Reference, [40](#)
 - register_logging_callback, [43](#)
 - Soft, [40](#)
 - String, [40](#)
 - STRING_CREATE, [41](#)
 - StringCreateProps, [39](#)
 - Time, [40](#)
 - to_string, [43](#)
 - toHDF5SizeVector, [43](#)
 - toSTLSizeVector, [43](#)
 - unqualified_t, [39](#)
 - UserDataTypes, [41](#)
 - VarLen, [40](#)
 - Warn, [40](#)
- HighFive - HDF5 header-only C++ Library, [1](#)
- highfive/bits/H5_definitions.hpp, [203](#), [204](#)
- highfive/bits/H5Annotate_traits.hpp, [205](#), [206](#)
- highfive/bits/H5Annotate_traits_misc.hpp, [206](#), [207](#)
- highfive/bits/H5Attribute_misc.hpp, [209](#), [210](#)
- highfive/bits/H5Converter_misc.hpp, [212](#), [214](#)
- highfive/bits/H5DataSet_misc.hpp, [224](#), [225](#)
- highfive/bits/H5Dataspace_misc.hpp, [226](#), [227](#)
- highfive/bits/H5DataType_misc.hpp, [229](#), [231](#)
- highfive/bits/H5Exception_misc.hpp, [237](#), [238](#)
- highfive/bits/H5File_misc.hpp, [239](#), [240](#)
- highfive/bits/H5FileDriver_misc.hpp, [242](#)
- highfive/bits/H5Friends.hpp, [243](#)
- highfive/bits/H5Iterables_misc.hpp, [243](#), [244](#)
- highfive/bits/H5Node_traits.hpp, [245](#), [246](#)
- highfive/bits/H5Node_traits_misc.hpp, [248](#), [249](#)
- highfive/bits/H5Object_misc.hpp, [254](#), [255](#)
- highfive/bits/H5Path_traits.hpp, [257](#), [258](#)

- highfive/bits/H5Path_traits_misc.hpp, 259, 260
- highfive/bits/H5PropertyList_misc.hpp, 261, 262
- highfive/bits/H5ReadWrite_misc.hpp, 269, 270
- highfive/bits/H5Reference_misc.hpp, 271, 273
- highfive/bits/H5Selection_misc.hpp, 274
- highfive/bits/H5Slice_traits.hpp, 275, 277
- highfive/bits/H5Slice_traits_misc.hpp, 280, 281
- highfive/bits/H5Utils.hpp, 285, 286
- highfive/H5Attribute.hpp, 287, 289
- highfive/H5DataSet.hpp, 290, 291
- highfive/H5DataSpace.hpp, 292, 293
- highfive/H5DataType.hpp, 294, 297
- highfive/H5Easy.hpp, 300, 303
- highfive/h5easy_bits/H5Easy_Eigen.hpp, 305, 306
- highfive/h5easy_bits/H5Easy_misc.hpp, 308, 309
- highfive/h5easy_bits/H5Easy_opencv.hpp, 310, 311
- highfive/h5easy_bits/H5Easy_public.hpp, 313, 314
- highfive/h5easy_bits/H5Easy_scalar.hpp, 316, 317
- highfive/h5easy_bits/H5Easy_vector.hpp, 319, 320
- highfive/h5easy_bits/H5Easy_xtensor.hpp, 321, 322
- highfive/H5Exception.hpp, 323, 324
- highfive/H5File.hpp, 326, 327
- highfive/H5FileDriver.hpp, 328, 329
- highfive/H5Group.hpp, 329, 330
- highfive/H5Object.hpp, 331, 333
- highfive/H5PropertyList.hpp, 334, 337
- highfive/H5Reference.hpp, 342, 343
- highfive/H5Selection.hpp, 344, 345
- highfive/H5Utility.hpp, 346, 350
- HighFive::AllocationTime, 45
 - AllocationTime, 45
 - getAllocationTime, 46
- HighFive::AnnotateTraits< Derivate >, 46
 - createAttribute, 46, 47
 - deleteAttribute, 47
 - getAttribute, 48
 - getNumberAttributes, 48
 - hasAttribute, 48
 - listAttributeNames, 48
- HighFive::AtomicType< char[StrLen]>, 54
 - AtomicType, 56
- HighFive::AtomicType< FixedLenStringArray< StrLen
> >, 57
 - AtomicType, 59
- HighFive::AtomicType< std::complex< T > >, 59
 - AtomicType, 61
- HighFive::AtomicType< T >, 49
 - AtomicType, 52–54
 - basic_type, 52
- HighFive::Attribute, 62
 - Attribute, 64
 - getCreatePropertyList, 64
 - getDataType, 64
 - getMemSpace, 64
 - getName, 65
 - getSpace, 65
 - getStorageSize, 65
 - Object, 65, 66
 - read, 66
 - type, 67
 - write, 66
 - write_raw, 66
- HighFive::AttributeException, 67
 - AttributeException, 69
- HighFive::Caching, 69
 - Caching, 69
 - getCacheSize, 70
 - getNumSlots, 70
 - getW0, 70
- HighFive::Chunking, 70
 - Chunking, 70, 71
 - getDimensions, 71
- HighFive::CompoundType, 71
 - commit, 75
 - CompoundType, 74
 - getMembers, 75
- HighFive::CompoundType::member_def, 150
 - base_type, 151
 - member_def, 150
 - name, 151
 - offset, 151
- HighFive::CreateIntermediateGroup, 77
 - CreateIntermediateGroup, 78
 - fromPropertyList, 78
 - isSet, 78
- HighFive::CreationOrder, 78
 - _CreationOrder, 78
 - Indexed, 79
 - Tracked, 79
- HighFive::DataSet, 79
 - DataSet, 83
 - getAccessPropertyList, 83
 - getCreatePropertyList, 83
 - getDataType, 83
 - getDimensions, 83
 - getElementCount, 84
 - getMemSpace, 84
 - getOffset, 84
 - getSpace, 84
 - getStorageSize, 84
 - NodeTraits, 86
 - Object, 85
 - Reference, 86
 - resize, 85
 - type, 86
- HighFive::DataSetException, 86
 - DataSetException, 88
- HighFive::DataSpace, 88
 - Attribute, 93
 - clone, 92
 - DataSet, 93
 - DataSpace, 91, 92
 - dataspace_null, 90
 - dataspace_scalar, 90
 - DataspaceType, 90
 - File, 93

- From, [92](#)
- FromArrayStrings, [92](#)
- getDimensions, [92](#)
- getElementCount, [92](#)
- getMaxDimensions, [93](#)
- getNumberDimensions, [93](#)
- type, [94](#)
- UNLIMITED, [94](#)
- HighFive::DataSpaceException, [94](#)
 - DataSpaceException, [96](#)
- HighFive::DataType, [96](#)
 - Attribute, [100](#)
 - CompoundType, [100](#)
 - DataSet, [100](#)
 - empty, [98](#)
 - File, [100](#)
 - getClass, [98](#)
 - getCreatePropertyList, [98](#)
 - getSize, [99](#)
 - isFixedLenStr, [99](#)
 - isReference, [99](#)
 - isVariableStr, [99](#)
 - Object, [99](#)
 - operator!=, [100](#)
 - operator==, [100](#)
 - string, [100](#)
- HighFive::DataTypeException, [101](#)
 - DataTypeException, [102](#)
- HighFive::Deflate, [102](#)
 - Deflate, [103](#)
- HighFive::ElementSet, [108](#)
 - ElementSet, [108](#), [109](#)
 - SliceTraits, [109](#)
- HighFive::EnumType< T >, [110](#)
 - commit, [113](#)
 - EnumType, [112](#), [113](#)
- HighFive::EnumType< T >::member_def, [151](#)
 - member_def, [152](#)
 - name, [152](#)
 - value, [152](#)
- HighFive::EstimatedLinkInfo, [113](#)
 - EstimatedLinkInfo, [114](#)
 - getEntries, [114](#)
 - getNameLength, [114](#)
- HighFive::Exception, [115](#)
 - _err_major, [118](#)
 - _err_minor, [118](#)
 - _errmsg, [118](#)
 - _next, [118](#)
 - ~Exception, [116](#)
 - Exception, [116](#)
 - getErrMajor, [116](#)
 - getErrMinor, [116](#)
 - HDF5ErrMapper, [117](#)
 - nextException, [117](#)
 - setErrorMsg, [117](#)
 - what, [117](#)
- HighFive::File, [118](#)
 - Create, [122](#)
 - Debug, [122](#)
 - Excl, [122](#)
 - File, [122](#), [123](#)
 - flush, [123](#)
 - getAccessPropertyList, [123](#)
 - getCreatePropertyList, [123](#)
 - getMetadataBlockSize, [123](#)
 - getName, [123](#)
 - getPath, [124](#)
 - getVersionBounds, [124](#)
 - Object, [124](#)
 - OpenOrCreate, [122](#)
 - Overwrite, [122](#)
 - PathTraits, [124](#)
 - ReadOnly, [122](#)
 - ReadWrite, [122](#)
 - Truncate, [122](#)
 - type, [125](#)
- HighFive::FileDriver, [125](#)
- HighFive::FileException, [128](#)
 - FileException, [129](#)
- HighFive::FileVersionBounds, [129](#)
 - FileVersionBounds, [130](#)
 - getVersion, [130](#)
- HighFive::FixedLenStringArray< N >, [131](#)
 - at, [133](#)
 - back, [133](#)
 - begin, [133](#), [134](#)
 - cbegin, [134](#)
 - cend, [134](#)
 - const_iterator, [132](#)
 - const_reverse_iterator, [132](#)
 - data, [134](#)
 - empty, [134](#)
 - end, [134](#)
 - FixedLenStringArray, [132](#), [133](#)
 - front, [134](#)
 - getString, [135](#)
 - iterator, [132](#)
 - operator[], [135](#)
 - push_back, [135](#)
 - rbegin, [135](#)
 - rend, [135](#)
 - resize, [136](#)
 - reverse_iterator, [132](#)
 - size, [136](#)
 - value_type, [132](#)
- HighFive::Group, [136](#)
 - File, [141](#)
 - getCreatePropertyList, [140](#)
 - getEstimatedLinkInfo, [140](#)
 - Group, [140](#)
 - Object, [140](#), [141](#)
 - Reference, [141](#)
 - type, [141](#)
- HighFive::GroupException, [142](#)
 - GroupException, [143](#)

- HighFive::HDF5ErrMapper, 143
 - stackWalk, 144
 - ToException, 144
- HighFive::HyperSlab, 144
 - apply, 145
 - HyperSlab, 145
 - notA, 145
 - notB, 145
 - operator&, 145
 - operator&=, 145
 - operator^, 145
 - operator^=, 145
 - operator|, 146
 - operator|=, 146
- HighFive::LinkCreationOrder, 146
 - fromPropertyList, 147
 - getFlags, 147
 - LinkCreationOrder, 146, 147
- HighFive::Logger, 147
 - callback_type, 148
 - log, 149
 - Logger, 148, 149
 - operator=, 149
 - set_logging_callback, 149
- HighFive::MetadataBlockSize, 152
 - getSize, 153
 - MetadataBlockSize, 153
- HighFive::MPIOCollectiveMetadata, 153
 - isCollectiveRead, 154
 - isCollectiveWrite, 154
 - MPIOCollectiveMetadata, 154
- HighFive::MPIOCollectiveMetadataRead, 154
 - isCollective, 155
 - MPIOCollectiveMetadataRead, 155
- HighFive::MPIOCollectiveMetadataWrite, 155
 - isCollective, 156
 - MPIOCollectiveMetadataWrite, 156
- HighFive::MPIOFileAccess, 156
 - MPIOFileAccess, 157
- HighFive::MPIOFileDriver, 157
 - MPIOFileDriver, 159
- HighFive::MpioNoCollectiveCause, 160
 - getCause, 160
 - getGlobalCause, 160
 - getLocalCause, 160
 - MpioNoCollectiveCause, 160
 - wasCollective, 161
- HighFive::NodeTraits< Derivate >, 161
 - createDataSet, 163, 164
 - createExternalLink, 164
 - createGroup, 165
 - createSoftLink, 165, 166
 - exist, 166
 - getDataSet, 166
 - getGroup, 167
 - getLinkType, 167
 - getNumberObjects, 167
 - getObjectName, 168
 - getObjectType, 168
 - listObjectNames, 168
 - rename, 168
 - unlink, 169
- HighFive::Object, 169
 - _hid, 172
 - ~Object, 171
 - CompoundType, 172
 - getId, 171
 - getInfo, 171
 - getType, 171
 - isValid, 172
 - Object, 171
 - operator=, 172
 - operator==, 172
 - Reference, 172
- HighFive::ObjectException, 173
 - ObjectException, 174
- HighFive::ObjectInfo, 174
 - getAddress, 175
 - getCreationTime, 175
 - getModificationTime, 175
 - getRefCount, 175
 - Object, 176
 - raw_info, 176
- HighFive::PathTraits< Derivate >, 176
 - _file_obj, 177
 - getFile, 177
 - getPath, 177
 - PathTraits, 176
- HighFive::PropertyException, 178
 - PropertyException, 179
- HighFive::PropertyList< T >, 180
 - _initializeIfNeeded, 182
 - add, 182
 - Default, 182
 - getType, 182
- HighFive::PropertyListBase, 183
 - Default, 184
 - PropertyListBase, 184
- HighFive::RawPropertyList< T >, 185
 - add, 187
- HighFive::Reference, 187
 - create_ref, 188
 - dereference, 189
 - getType, 189
 - Reference, 188
- HighFive::ReferenceException, 190
 - ReferenceException, 191
- HighFive::RegularHyperSlab, 191
 - block, 193
 - count, 193
 - fromHDF5Sizes, 192
 - offset, 193
 - packedDims, 192
 - rank, 193
 - RegularHyperSlab, 192
 - stride, 193

- HighFive::Selection, [194](#)
 - getDataset, [195](#)
 - getDataType, [196](#)
 - getMemSpace, [196](#)
 - getSpace, [196](#)
 - Selection, [195](#)
- HighFive::Shuffle, [197](#)
 - Shuffle, [197](#)
- HighFive::SilenceHDF5, [197](#)
 - ~SilenceHDF5, [197](#)
 - SilenceHDF5, [197](#)
- HighFive::SliceTraits< Derivate >, [198](#)
 - read, [198](#), [199](#)
 - select, [199](#), [200](#)
 - select_impl, [200](#)
 - write, [200](#)
 - write_raw, [200](#)
- HighFive::Szip, [201](#)
 - getOptionsMask, [201](#)
 - getPixelsPerBlock, [201](#)
 - Szip, [201](#)
- HighFive::UseCollectiveIO, [202](#)
 - isCollective, [202](#)
 - UseCollectiveIO, [202](#)
- HIGHFIVE_LOG_DEBUG
 - H5Utility.hpp, [348](#)
- HIGHFIVE_LOG_DEBUG_IF
 - H5Utility.hpp, [348](#)
- HIGHFIVE_LOG_ERROR
 - H5Utility.hpp, [348](#)
- HIGHFIVE_LOG_ERROR_IF
 - H5Utility.hpp, [348](#)
- HIGHFIVE_LOG_INFO
 - H5Utility.hpp, [349](#)
- HIGHFIVE_LOG_INFO_IF
 - H5Utility.hpp, [349](#)
- HIGHFIVE_LOG_LEVEL
 - H5Utility.hpp, [349](#)
- HIGHFIVE_LOG_LEVEL_DEBUG
 - H5Utility.hpp, [349](#)
- HIGHFIVE_LOG_LEVEL_ERROR
 - H5Utility.hpp, [349](#)
- HIGHFIVE_LOG_LEVEL_INFO
 - H5Utility.hpp, [349](#)
- HIGHFIVE_LOG_LEVEL_WARN
 - H5Utility.hpp, [349](#)
- HIGHFIVE_LOG_WARN
 - H5Utility.hpp, [349](#)
- HIGHFIVE_LOG_WARN_IF
 - H5Utility.hpp, [350](#)
- HIGHFIVE_REGISTER_TYPE
 - H5DataType.hpp, [297](#)
- HyperSlab
 - HighFive::HyperSlab, [145](#)
- Indexed
 - HighFive::CreationOrder, [79](#)
- IndexType
 - HighFive, [40](#)
- Info
 - HighFive, [40](#)
- Integer
 - HighFive, [40](#)
- Invalid
 - HighFive, [40](#)
- isChunked
 - H5Easy::DumpOptions, [105](#)
- isCollective
 - HighFive::MPIOCollectiveMetadataRead, [155](#)
 - HighFive::MPIOCollectiveMetadataWrite, [156](#)
 - HighFive::UseCollectiveIO, [202](#)
- isCollectiveRead
 - HighFive::MPIOCollectiveMetadata, [154](#)
- isCollectiveWrite
 - HighFive::MPIOCollectiveMetadata, [154](#)
- isFixedLenStr
 - HighFive::DataType, [99](#)
- isReference
 - HighFive::DataType, [99](#)
- isSet
 - HighFive::CreateIntermediateGroup, [78](#)
- isValid
 - HighFive::Object, [172](#)
- isVariableStr
 - HighFive::DataType, [99](#)
- iterator
 - HighFive::FixedLenStringArray< N >, [132](#)
- LINK_ACCESS
 - HighFive, [41](#)
- LINK_CREATE
 - HighFive, [41](#)
- LinkAccessProps
 - HighFive, [39](#)
- LinkCreateProps
 - HighFive, [39](#)
- LinkCreationOrder
 - HighFive::LinkCreationOrder, [146](#), [147](#)
- LinkType
 - HighFive, [40](#)
- listAttributeNames
 - HighFive::AnnotateTraits< Derivate >, [48](#)
- listObjectNames
 - HighFive::NodeTraits< Derivate >, [168](#)
- load
 - H5Easy, [33](#)
- loadAttribute
 - H5Easy, [34](#)
- log
 - HighFive::Logger, [149](#)
- Logger
 - HighFive::Logger, [148](#), [149](#)
- LogSeverity
 - HighFive, [40](#)
- member_def
 - HighFive::CompoundType::member_def, [150](#)
 - HighFive::EnumType< T >::member_def, [152](#)

- MetadataBlockSize
 - HighFive::MetadataBlockSize, [153](#)
- MPIOCollectiveMetadata
 - HighFive::MPIOCollectiveMetadata, [154](#)
- MPIOCollectiveMetadataRead
 - HighFive::MPIOCollectiveMetadataRead, [155](#)
- MPIOCollectiveMetadataWrite
 - HighFive::MPIOCollectiveMetadataWrite, [156](#)
- MPIOFileAccess
 - HighFive::MPIOFileAccess, [157](#)
- MPIOFileDriver
 - HighFive::MPIOFileDriver, [159](#)
- MpioNoCollectiveCause
 - HighFive::MpioNoCollectiveCause, [160](#)
- NAME
 - HighFive, [40](#)
- name
 - HighFive::CompoundType::member_def, [151](#)
 - HighFive::EnumType< T >::member_def, [152](#)
- nextException
 - HighFive::Exception, [117](#)
- NodeTraits
 - HighFive::DataSet, [86](#)
- notA
 - HighFive::HyperSlab, [145](#)
- notB
 - HighFive::HyperSlab, [145](#)
- Object
 - HighFive::Attribute, [65](#), [66](#)
 - HighFive::DataSet, [85](#)
 - HighFive::DataType, [99](#)
 - HighFive::File, [124](#)
 - HighFive::Group, [140](#), [141](#)
 - HighFive::Object, [171](#)
 - HighFive::ObjectInfo, [176](#)
- OBJECT_COPY
 - HighFive, [41](#)
- OBJECT_CREATE
 - HighFive, [41](#)
- ObjectCopyProps
 - HighFive, [39](#)
- ObjectCreateProps
 - HighFive, [39](#)
- ObjectException
 - HighFive::ObjectException, [174](#)
- ObjectType
 - HighFive, [41](#)
- offset
 - HighFive::CompoundType::member_def, [151](#)
 - HighFive::RegularHyperSlab, [193](#)
- Opaque
 - HighFive, [40](#)
- OpenOrCreate
 - HighFive::File, [122](#)
- operator!=
 - HighFive::DataType, [100](#)
- operator=
 - HighFive::Logger, [149](#)
 - HighFive::Object, [172](#)
- operator==
 - HighFive::DataType, [100](#)
 - HighFive::Object, [172](#)
- operator&
 - HighFive, [42](#)
 - HighFive::HyperSlab, [145](#)
- operator&=
 - HighFive::HyperSlab, [145](#)
- operator[]
 - HighFive::FixedLenStringArray< N >, [135](#)
- operator^
 - HighFive::HyperSlab, [145](#)
- operator^=
 - HighFive::HyperSlab, [145](#)
- operator|
 - HighFive, [42](#)
 - HighFive::HyperSlab, [146](#)
- operator|=
 - HighFive::HyperSlab, [146](#)
- Other
 - HighFive, [40](#), [41](#)
- Overwrite
 - H5Easy, [28](#)
 - HighFive::File, [122](#)
- overwrite
 - H5Easy::DumpOptions, [105](#)
- packedDims
 - HighFive::RegularHyperSlab, [192](#)
- PathTraits
 - HighFive::File, [124](#)
 - HighFive::PathTraits< Derivate >, [176](#)
- PropertyException
 - HighFive::PropertyException, [179](#)
- PropertyListBase
 - HighFive::PropertyListBase, [184](#)
- PropertyType
 - HighFive, [41](#)
- push_back
 - HighFive::FixedLenStringArray< N >, [135](#)
- rank
 - HighFive::RegularHyperSlab, [193](#)
- raw_info
 - HighFive::ObjectInfo, [176](#)
- rbegin
 - HighFive::FixedLenStringArray< N >, [135](#)
- read
 - HighFive::Attribute, [66](#)
 - HighFive::SliceTraits< Derivate >, [198](#), [199](#)
- ReadOnly
 - HighFive::File, [122](#)
- ReadWrite
 - HighFive::File, [122](#)
- Reference
 - HighFive, [40](#)
 - HighFive::DataSet, [86](#)

- HighFive::Group, [141](#)
- HighFive::Object, [172](#)
- HighFive::Reference, [188](#)
- ReferenceException
 - HighFive::ReferenceException, [191](#)
- register_logging_callback
 - HighFive, [43](#)
- RegularHyperSlab
 - HighFive::RegularHyperSlab, [192](#)
- rename
 - HighFive::NodeTraits< Derivate >, [168](#)
- rend
 - HighFive::FixedLenStringArray< N >, [135](#)
- resize
 - HighFive::DataSet, [85](#)
 - HighFive::FixedLenStringArray< N >, [136](#)
- reverse_iterator
 - HighFive::FixedLenStringArray< N >, [132](#)
- select
 - HighFive::SliceTraits< Derivate >, [199](#), [200](#)
- select_impl
 - HighFive::SliceTraits< Derivate >, [200](#)
- Selection
 - HighFive::Selection, [195](#)
- set
 - H5Easy::DumpOptions, [105](#), [106](#)
- set_logging_callback
 - HighFive::Logger, [149](#)
- setChunkSize
 - H5Easy::DumpOptions, [106](#), [108](#)
- setErrorMsg
 - HighFive::Exception, [117](#)
- Shuffle
 - HighFive::Shuffle, [197](#)
- SilenceHDF5
 - HighFive::SilenceHDF5, [197](#)
- size
 - HighFive::FixedLenStringArray< N >, [136](#)
- SliceTraits
 - HighFive::ElementSet, [109](#)
- Soft
 - HighFive, [40](#)
- stackWalk
 - HighFive::HDF5ErrMapper, [144](#)
- stride
 - HighFive::RegularHyperSlab, [193](#)
- String
 - HighFive, [40](#)
- string
 - HighFive::DataType, [100](#)
- STRING_CREATE
 - HighFive, [41](#)
- StringCreateProps
 - HighFive, [39](#)
- Szip
 - HighFive::Szip, [201](#)
- Time
 - HighFive, [40](#)
- to_string
 - HighFive, [43](#)
- ToException
 - HighFive::HDF5ErrMapper, [144](#)
- toHDF5SizeVector
 - HighFive, [43](#)
- toSTLSizeVector
 - HighFive, [43](#)
- Tracked
 - HighFive::CreationOrder, [79](#)
- True
 - H5Easy, [29](#)
- Truncate
 - HighFive::File, [122](#)
- type
 - HighFive::Attribute, [67](#)
 - HighFive::DataSet, [86](#)
 - HighFive::DataSpace, [94](#)
 - HighFive::File, [125](#)
 - HighFive::Group, [141](#)
- UNLIMITED
 - HighFive::DataSpace, [94](#)
- unlink
 - HighFive::NodeTraits< Derivate >, [169](#)
- unqualified_t
 - HighFive, [39](#)
- UseCollectiveIO
 - HighFive::UseCollectiveIO, [202](#)
- UserDataType
 - HighFive, [41](#)
- value
 - HighFive::EnumType< T >::member_def, [152](#)
- value_type
 - HighFive::FixedLenStringArray< N >, [132](#)
- VarLen
 - HighFive, [40](#)
- Version 2.7.1 - 2023-04-04, [7](#)
- Warn
 - HighFive, [40](#)
- wasCollective
 - HighFive::MpioNoCollectiveCause, [161](#)
- what
 - HighFive::Exception, [117](#)
- write
 - HighFive::Attribute, [66](#)
 - HighFive::SliceTraits< Derivate >, [200](#)
- write_raw
 - HighFive::Attribute, [66](#)
 - HighFive::SliceTraits< Derivate >, [200](#)