

AARE

Generated by Doxygen 1.9.7



<b>1 aare</b>	<b>1</b>
1.1 Folder structure	1
1.2 file_io class diagram	1
1.3 Test the zmq socket with a detector simulator	1
1.4 Test the zmq processing replaying data	2
1.5 generate documentation	2
<b>2 Namespace Index</b>	<b>3</b>
2.1 Namespace List	3
<b>3 Hierarchical Index</b>	<b>5</b>
3.1 Class Hierarchy	5
<b>4 Data Structure Index</b>	<b>7</b>
4.1 Data Structures	7
<b>5 File Index</b>	<b>9</b>
5.1 File List	9
<b>6 Namespace Documentation</b>	<b>11</b>
6.1 aare Namespace Reference	11
6.1.1 Detailed Description	13
6.1.2 Typedef Documentation	13
6.1.2.1 DataTypeVariants	13
6.1.2.2 dynamic_shape	13
6.1.2.3 Shape	13
6.1.2.4 shape_t	13
6.1.3 Enumeration Type Documentation	13
6.1.3.1 DetectorType	13
6.1.3.2 endian	13
6.1.3.3 TimingMode	14
6.1.4 Function Documentation	14
6.1.4.1 c_strides()	14
6.1.4.2 element_offset() [1/2]	14
6.1.4.3 element_offset() [2/2]	14
6.1.4.4 load()	14
6.1.4.5 make_array()	15
6.1.4.6 make_shape()	15
6.1.4.7 save()	15
6.1.4.8 StringTo() [1/5]	15
6.1.4.9 StringTo() [2/5]	15
6.1.4.10 StringTo() [3/5]	15
6.1.4.11 StringTo() [4/5]	15
6.1.4.12 StringTo() [5/5]	16

6.1.4.13 toString() [1/3]	16
6.1.4.14 toString() [2/3]	16
6.1.4.15 toString() [3/3]	16
6.2 aare.File Namespace Reference	16
6.3 aare.Frame Namespace Reference	16
6.4 aare::logger Namespace Reference	16
6.4.1 Enumeration Type Documentation	17
6.4.1.1 LOGGING_LEVEL	17
6.4.2 Function Documentation	17
6.4.2.1 debug()	17
6.4.2.2 error()	18
6.4.2.3 get_logger_instance()	18
6.4.2.4 info()	18
6.4.2.5 log()	18
6.4.2.6 set_output_file()	18
6.4.2.7 set_streams() [1/2]	18
6.4.2.8 set_streams() [2/2]	18
6.4.2.9 set_verbosity()	18
6.4.2.10 warn()	19
6.5 aare::logger::internal Namespace Reference	19
6.5.1 Variable Documentation	19
6.5.1.1 logger_instance	19
6.6 aare::network_io Namespace Reference	19
6.7 aare::NumpyHelpers Namespace Reference	19
6.7.1 Function Documentation	20
6.7.1.1 get_value_from_map()	20
6.7.1.2 in_array()	20
6.7.1.3 is_digits()	20
6.7.1.4 parse_bool()	20
6.7.1.5 parse_descr()	20
6.7.1.6 parse_dict()	20
6.7.1.7 parse_str()	20
6.7.1.8 parse_tuple()	21
6.7.1.9 trim()	21
6.7.1.10 write_boolean()	21
6.7.1.11 write_header() [1/2]	21
6.7.1.12 write_header() [2/2]	21
6.7.1.13 write_header_dict()	21
6.7.1.14 write_magic()	21
6.7.1.15 write_tuple()	22
6.7.2 Variable Documentation	22
6.7.2.1 magic_str	22

6.7.2.2 magic_string_length . . . . .	22
6.8 example Namespace Reference . . . . .	22
6.9 example.read_frame Namespace Reference . . . . .	22
6.9.1 Variable Documentation . . . . .	22
6.9.1.1 arr . . . . .	22
6.9.1.2 data_path . . . . .	22
6.9.1.3 file . . . . .	23
6.9.1.4 frame . . . . .	23
6.9.1.5 root_dir . . . . .	23
6.10 folly Namespace Reference . . . . .	23
6.11 read_first_frame_number Namespace Reference . . . . .	23
6.11.1 Variable Documentation . . . . .	23
6.11.1.1 frame_number . . . . .	23
6.11.1.2 header_dt . . . . .	23
6.12 read_frame Namespace Reference . . . . .	24
6.12.1 Variable Documentation . . . . .	24
6.12.1.1 cols . . . . .	24
6.12.1.2 count . . . . .	24
6.12.1.3 data . . . . .	24
6.12.1.4 dtype . . . . .	24
6.12.1.5 f . . . . .	24
6.12.1.6 file_name . . . . .	24
6.12.1.7 frames . . . . .	24
6.12.1.8 header . . . . .	25
6.12.1.9 header_dt . . . . .	25
6.12.1.10 rows . . . . .	25
6.12.1.11 uint16 . . . . .	25
6.13 read_multiport Namespace Reference . . . . .	25
6.13.1 Variable Documentation . . . . .	26
6.13.1.1 axis . . . . .	26
6.13.1.2 count . . . . .	26
6.13.1.3 data . . . . .	26
6.13.1.4 dtype . . . . .	26
6.13.1.5 f . . . . .	26
6.13.1.6 file_name . . . . .	26
6.13.1.7 frame_cols . . . . .	26
6.13.1.8 frame_rows . . . . .	26
6.13.1.9 frames . . . . .	26
6.13.1.10 header . . . . .	26
6.13.1.11 header_dt . . . . .	27
6.13.1.12 part_cols . . . . .	27
6.13.1.13 part_rows . . . . .	27

6.13.1.14 parts	27
6.13.1.15 parts_data	27
6.13.1.16 uint16	27
6.14 simdjson Namespace Reference	27
6.15 write_test_files Namespace Reference	27
6.15.1 Variable Documentation	28
6.15.1.1 arr	28
6.15.1.2 arr2	28
<b>7 Data Structure Documentation</b>	<b>29</b>
7.1 aare::CircularFifo< ItemType > Class Template Reference	29
7.1.1 Member Typedef Documentation	30
7.1.1.1 value_type	30
7.1.2 Constructor & Destructor Documentation	30
7.1.2.1 CircularFifo() [1/2]	30
7.1.2.2 CircularFifo() [2/2]	30
7.1.2.3 ~CircularFifo()	30
7.1.3 Member Function Documentation	30
7.1.3.1 frontPtr()	30
7.1.3.2 isFull()	30
7.1.3.3 next()	30
7.1.3.4 numFilledSlots()	31
7.1.3.5 numFreeSlots()	31
7.1.3.6 pop_free()	31
7.1.3.7 pop_value() [1/2]	31
7.1.3.8 pop_value() [2/2]	31
7.1.3.9 push_free()	31
7.1.3.10 push_value()	31
7.1.3.11 try_pop_free()	31
7.1.3.12 try_push_free()	32
7.1.3.13 try_push_value()	32
7.1.4 Field Documentation	32
7.1.4.1 fifo_size	32
7.1.4.2 filled_slots	32
7.1.4.3 free_slots	32
7.2 aare::ClusterFinder< T > Class Template Reference	32
7.2.1 Constructor & Destructor Documentation	33
7.2.1.1 ClusterFinder()	33
7.2.2 Member Function Documentation	34
7.2.2.1 add_link()	34
7.2.2.2 check_neighbours()	34
7.2.2.3 clear_hits()	34

7.2.2.4 find_clusters()	34
7.2.2.5 find_clusters_X()	34
7.2.2.6 first_pass()	34
7.2.2.7 labeled()	34
7.2.2.8 print_connections()	35
7.2.2.9 rec_FillHit()	35
7.2.2.10 second_pass()	35
7.2.2.11 set_noiseMap()	35
7.2.2.12 set_peripheralThresholdFactor()	35
7.2.2.13 single_pass()	35
7.2.2.14 steal_hits()	35
7.2.2.15 store_clusters()	35
7.2.2.16 total_clusters()	36
7.2.3 Field Documentation	36
7.2.3.1 binary_	36
7.2.3.2 child	36
7.2.3.3 current_label	36
7.2.3.4 di	36
7.2.3.5 di_	36
7.2.3.6 dj	36
7.2.3.7 dj_	36
7.2.3.8 h_size	37
7.2.3.9 hits	37
7.2.3.10 labeled_	37
7.2.3.11 noiseMap	37
7.2.3.12 original_	37
7.2.3.13 peripheral_labeled_	37
7.2.3.14 peripheralThresholdFactor_	37
7.2.3.15 shape_	37
7.2.3.16 threshold_	37
7.2.3.17 use_noise_map	38
7.3 aare::DType Class Reference	38
7.3.1 Member Enumeration Documentation	38
7.3.1.1 TypeIndex	38
7.3.2 Constructor & Destructor Documentation	39
7.3.2.1 DType() [1/3]	39
7.3.2.2 DType() [2/3]	39
7.3.2.3 DType() [3/3]	39
7.3.3 Member Function Documentation	39
7.3.3.1 bitdepth()	39
7.3.3.2 operator"!="() [1/2]	39
7.3.3.3 operator"!="() [2/2]	39

7.3.3.4 operator==( ) [1/2]	39
7.3.3.5 operator==( ) [2/2]	39
7.3.3.6 str()	40
7.3.4 Field Documentation	40
7.3.4.1 m_type	40
7.4 aare::File Class Reference	40
7.4.1 Detailed Description	41
7.4.2 Constructor & Destructor Documentation	41
7.4.2.1 File() [1/2]	41
7.4.2.2 File() [2/2]	41
7.4.2.3 ~File()	41
7.4.3 Member Function Documentation	42
7.4.3.1 bitdepth()	42
7.4.3.2 bytes_per_frame()	42
7.4.3.3 cols()	42
7.4.3.4 frame_number()	42
7.4.3.5 iread()	42
7.4.3.6 pixels()	42
7.4.3.7 read() [1/2]	42
7.4.3.8 read() [2/2]	42
7.4.3.9 read_into() [1/2]	42
7.4.3.10 read_into() [2/2]	43
7.4.3.11 rows()	43
7.4.3.12 seek()	43
7.4.3.13 tell()	43
7.4.3.14 total_frames()	43
7.4.3.15 write()	43
7.4.4 Field Documentation	43
7.4.4.1 file_impl	43
7.5 aare.File.File Class Reference	43
7.5.1 Detailed Description	44
7.5.2 Constructor & Destructor Documentation	44
7.5.2.1 __init__()	44
7.5.3 Member Function Documentation	44
7.5.3.1 __getattribute__()	44
7.5.4 Field Documentation	44
7.5.4.1 _file	44
7.5.4.2 path	44
7.6 aare::FileConfig Struct Reference	45
7.6.1 Detailed Description	45
7.6.2 Member Function Documentation	45
7.6.2.1 operator!=( )	45



7.6.2.2 operator==( )	45
7.6.3 Field Documentation	45
7.6.3.1 cols	45
7.6.3.2 dtype	45
7.6.3.3 geometry	46
7.6.3.4 rows	46
7.7 aare::FileInterface Class Reference	46
7.7.1 Detailed Description	47
7.7.2 Constructor & Destructor Documentation	47
7.7.2.1 ~FileInterface()	47
7.7.3 Member Function Documentation	48
7.7.3.1 bitdepth()	48
7.7.3.2 bytes_per_frame()	48
7.7.3.3 cols()	48
7.7.3.4 frame_number()	48
7.7.3.5 iread() [1/2]	49
7.7.3.6 iread() [2/2]	49
7.7.3.7 pixels()	49
7.7.3.8 read() [1/2]	50
7.7.3.9 read() [2/2]	50
7.7.3.10 read_into() [1/2]	50
7.7.3.11 read_into() [2/2]	51
7.7.3.12 rows()	51
7.7.3.13 seek()	51
7.7.3.14 tell()	52
7.7.3.15 total_frames()	52
7.7.3.16 write()	52
7.7.4 Field Documentation	53
7.7.4.1 current_frame	53
7.7.4.2 m_base_name	53
7.7.4.3 m_base_path	53
7.7.4.4 m_bitdepth	53
7.7.4.5 m_cols	53
7.7.4.6 m_ext	53
7.7.4.7 m_index	53
7.7.4.8 m_fname	53
7.7.4.9 m_mode	53
7.7.4.10 m_rows	54
7.7.4.11 m_total_frames	54
7.7.4.12 m_type	54
7.7.4.13 max_frames_per_file	54
7.7.4.14 version	54

7.8 aare::Frame Class Reference	54
7.8.1 Constructor & Destructor Documentation	55
7.8.1.1 Frame() [1/4]	55
7.8.1.2 Frame() [2/4]	55
7.8.1.3 Frame() [3/4]	55
7.8.1.4 Frame() [4/4]	55
7.8.1.5 ~Frame()	55
7.8.2 Member Function Documentation	55
7.8.2.1 bitdepth()	55
7.8.2.2 cols()	56
7.8.2.3 data()	56
7.8.2.4 get()	56
7.8.2.5 image()	56
7.8.2.6 operator=()	56
7.8.2.7 rows()	56
7.8.2.8 set()	56
7.8.2.9 size()	56
7.8.2.10 view()	56
7.8.3 Field Documentation	57
7.8.3.1 m_bitdepth	57
7.8.3.2 m_cols	57
7.8.3.3 m_data	57
7.8.3.4 m_rows	57
7.9 aare.Frame.Frame Class Reference	57
7.9.1 Detailed Description	57
7.9.2 Constructor & Destructor Documentation	57
7.9.2.1 __init__()	57
7.9.3 Member Function Documentation	58
7.9.3.1 __getattribute__()	58
7.9.4 Field Documentation	58
7.9.4.1 _frameImpl	58
7.10 aare::ClusterFinder< T >::Hit Struct Reference	58
7.10.1 Field Documentation	58
7.10.1.1 col	58
7.10.1.2 cols	59
7.10.1.3 energy	59
7.10.1.4 enes	59
7.10.1.5 max	59
7.10.1.6 reserved	59
7.10.1.7 row	59
7.10.1.8 rows	59
7.10.1.9 size	59

7.11 aare::logger::Logger Class Reference	60
7.11.1 Constructor & Destructor Documentation	60
7.11.1.1 Logger()	60
7.11.1.2 ~Logger()	60
7.11.2 Member Function Documentation	61
7.11.2.1 debug()	61
7.11.2.2 error()	61
7.11.2.3 info()	61
7.11.2.4 log()	61
7.11.2.5 log_() [1/2]	61
7.11.2.6 log_() [2/2]	61
7.11.2.7 set_output_file()	61
7.11.2.8 set_streams() [1/2]	62
7.11.2.9 set_streams() [2/2]	62
7.11.2.10 set_verbosity()	62
7.11.2.11 warn()	62
7.11.3 Field Documentation	62
7.11.3.1 error_buf	62
7.11.3.2 error_output	62
7.11.3.3 out_file	62
7.11.3.4 standard_buf	62
7.11.3.5 standard_output	62
7.11.3.6 VERBOSITY_LEVEL	63
7.12 MoveOnlyInt Struct Reference	63
7.12.1 Constructor & Destructor Documentation	63
7.12.1.1 MoveOnlyInt() [1/4]	63
7.12.1.2 MoveOnlyInt() [2/4]	63
7.12.1.3 MoveOnlyInt() [3/4]	63
7.12.1.4 MoveOnlyInt() [4/4]	63
7.12.2 Member Function Documentation	64
7.12.2.1 operator=() [1/2]	64
7.12.2.2 operator=() [2/2]	64
7.12.2.3 operator==()	64
7.12.3 Field Documentation	64
7.12.3.1 value	64
7.13 aare::NDArray< T, Ndim > Class Template Reference	64
7.13.1 Member Typedef Documentation	66
7.13.1.1 value_type	66
7.13.2 Constructor & Destructor Documentation	66
7.13.2.1 NDArray() [1/6]	66
7.13.2.2 NDArray() [2/6]	66
7.13.2.3 NDArray() [3/6]	66

7.13.2.4 NDAarray() [4/6]	66
7.13.2.5 NDAarray() [5/6]	66
7.13.2.6 NDAarray() [6/6]	67
7.13.2.7 ~NDAarray()	67
7.13.3 Member Function Documentation	67
7.13.3.1 begin()	67
7.13.3.2 buffer()	67
7.13.3.3 byte_strides()	67
7.13.3.4 data()	67
7.13.3.5 end()	67
7.13.3.6 operator!=()	67
7.13.3.7 operator&=()	68
7.13.3.8 operator()() [1/4]	68
7.13.3.9 operator()() [2/4]	68
7.13.3.10 operator()() [3/4]	68
7.13.3.11 operator()() [4/4]	68
7.13.3.12 operator*() [1/2]	68
7.13.3.13 operator*() [2/2]	68
7.13.3.14 operator*-=() [1/2]	69
7.13.3.15 operator*-=() [2/2]	69
7.13.3.16 operator+() [1/2]	69
7.13.3.17 operator+() [2/2]	69
7.13.3.18 operator++()	69
7.13.3.19 operator+=() [1/2]	69
7.13.3.20 operator+=() [2/2]	69
7.13.3.21 operator-() [1/2]	69
7.13.3.22 operator-() [2/2]	70
7.13.3.23 operator-=() [1/2]	70
7.13.3.24 operator-=() [2/2]	70
7.13.3.25 operator/() [1/2]	70
7.13.3.26 operator/() [2/2]	70
7.13.3.27 operator/=() [1/2]	70
7.13.3.28 operator/=() [2/2]	70
7.13.3.29 operator=() [1/3]	71
7.13.3.30 operator=() [2/3]	71
7.13.3.31 operator=() [3/3]	71
7.13.3.32 operator==()	71
7.13.3.33 operator>()	71
7.13.3.34 Print()	71
7.13.3.35 Print_all()	71
7.13.3.36 Print_some()	71
7.13.3.37 reset()	72

7.13.3.38 shape() [1/2]	72
7.13.3.39 shape() [2/2]	72
7.13.3.40 size()	72
7.13.3.41 span()	72
7.13.3.42 sqrt()	72
7.13.3.43 strides()	72
7.13.3.44 total_bytes()	72
7.13.3.45 value()	73
7.13.4 Field Documentation	73
7.13.4.1 data_	73
7.13.4.2 shape_	73
7.13.4.3 size_	73
7.13.4.4 strides_	73
7.14 aare::NDView< T, Ndim > Class Template Reference	73
7.14.1 Constructor & Destructor Documentation	74
7.14.1.1 NDView() [1/5]	74
7.14.1.2 NDView() [2/5]	75
7.14.1.3 NDView() [3/5]	75
7.14.1.4 NDView() [4/5]	75
7.14.1.5 NDView() [5/5]	75
7.14.2 Member Function Documentation	75
7.14.2.1 begin()	75
7.14.2.2 data()	75
7.14.2.3 elementwise() [1/2]	75
7.14.2.4 elementwise() [2/2]	76
7.14.2.5 end()	76
7.14.2.6 operator>() [1/3]	76
7.14.2.7 operator>() [2/3]	76
7.14.2.8 operator>() [3/3]	76
7.14.2.9 operator*=( )	76
7.14.2.10 operator+=( )	76
7.14.2.11 operator-=( )	77
7.14.2.12 operator/=( ) [1/2]	77
7.14.2.13 operator/=( ) [2/2]	77
7.14.2.14 operator=( ) [1/2]	77
7.14.2.15 operator=( ) [2/2]	77
7.14.2.16 operator==( )	77
7.14.2.17 operator[]()	77
7.14.2.18 shape() [1/2]	77
7.14.2.19 shape() [2/2]	78
7.14.2.20 size()	78
7.14.3 Field Documentation	78

7.14.3.1 buffer_ . . . . .	78
7.14.3.2 shape_ . . . . .	78
7.14.3.3 size_ . . . . .	78
7.14.3.4 strides_ . . . . .	78
7.15 aare::network_io::NetworkError Class Reference . . . . .	78
7.15.1 Detailed Description . . . . .	79
7.15.2 Constructor & Destructor Documentation . . . . .	79
7.15.2.1 NetworkError() [1/2] . . . . .	79
7.15.2.2 NetworkError() [2/2] . . . . .	79
7.15.3 Member Function Documentation . . . . .	79
7.15.3.1 what() . . . . .	79
7.15.4 Field Documentation . . . . .	79
7.15.4.1 m_msg . . . . .	79
7.16 aare::NumpyFile Class Reference . . . . .	80
7.16.1 Detailed Description . . . . .	81
7.16.2 Constructor & Destructor Documentation . . . . .	82
7.16.2.1 NumpyFile() . . . . .	82
7.16.2.2 ~NumpyFile() . . . . .	82
7.16.3 Member Function Documentation . . . . .	82
7.16.3.1 bitdepth() . . . . .	82
7.16.3.2 bytes_per_frame() . . . . .	82
7.16.3.3 cols() . . . . .	83
7.16.3.4 dtype() . . . . .	83
7.16.3.5 frame_number() . . . . .	83
7.16.3.6 get_frame() . . . . .	83
7.16.3.7 get_frame_into() . . . . .	83
7.16.3.8 iread() [1/2] . . . . .	84
7.16.3.9 iread() [2/2] . . . . .	84
7.16.3.10 load() . . . . .	84
7.16.3.11 load_metadata() . . . . .	85
7.16.3.12 pixels() . . . . .	85
7.16.3.13 read() [1/2] . . . . .	85
7.16.3.14 read() [2/2] . . . . .	85
7.16.3.15 read_into() [1/2] . . . . .	86
7.16.3.16 read_into() [2/2] . . . . .	86
7.16.3.17 rows() . . . . .	86
7.16.3.18 seek() . . . . .	87
7.16.3.19 shape() . . . . .	87
7.16.3.20 tell() . . . . .	87
7.16.3.21 total_frames() . . . . .	87
7.16.3.22 write() . . . . .	87
7.16.4 Field Documentation . . . . .	88

7.16.4.1 current_frame . . . . .	88
7.16.4.2 fp . . . . .	88
7.16.4.3 header_len . . . . .	88
7.16.4.4 header_len_size . . . . .	88
7.16.4.5 header_size . . . . .	88
7.16.4.6 initial_header_len . . . . .	88
7.16.4.7 m_base_name . . . . .	88
7.16.4.8 m_base_path . . . . .	89
7.16.4.9 m_bitdepth . . . . .	89
7.16.4.10 m_cols . . . . .	89
7.16.4.11 m_ext . . . . .	89
7.16.4.12 m_findex . . . . .	89
7.16.4.13 m_fname . . . . .	89
7.16.4.14 m_header . . . . .	89
7.16.4.15 m_mode . . . . .	89
7.16.4.16 m_rows . . . . .	89
7.16.4.17 m_total_frames . . . . .	89
7.16.4.18 m_type . . . . .	90
7.16.4.19 major_ver_ . . . . .	90
7.16.4.20 max_frames_per_file . . . . .	90
7.16.4.21 minor_ver_ . . . . .	90
7.16.4.22 version . . . . .	90
7.17 aare::NumpyHeader Struct Reference . . . . .	90
7.17.1 Member Function Documentation . . . . .	90
7.17.1.1 to_string() . . . . .	90
7.17.2 Field Documentation . . . . .	91
7.17.2.1 dtype . . . . .	91
7.17.2.2 fortran_order . . . . .	91
7.17.2.3 shape . . . . .	91
7.18 folly::ProducerConsumerQueue< T > Struct Template Reference . . . . .	91
7.18.1 Member Typedef Documentation . . . . .	92
7.18.1.1 AtomicIndex . . . . .	92
7.18.1.2 value_type . . . . .	92
7.18.2 Constructor & Destructor Documentation . . . . .	92
7.18.2.1 ProducerConsumerQueue() [1/2] . . . . .	92
7.18.2.2 ProducerConsumerQueue() [2/2] . . . . .	92
7.18.2.3 ~ProducerConsumerQueue() . . . . .	92
7.18.3 Member Function Documentation . . . . .	92
7.18.3.1 capacity() . . . . .	92
7.18.3.2 frontPtr() . . . . .	93
7.18.3.3 isEmpty() . . . . .	93
7.18.3.4 isFull() . . . . .	93

7.18.3.5 operator=()	93
7.18.3.6 popFront()	93
7.18.3.7 read()	93
7.18.3.8 sizeGuess()	93
7.18.3.9 write()	93
7.18.4 Field Documentation	94
7.18.4.1 pad0_	94
7.18.4.2 pad1_	94
7.18.4.3 readIndex_	94
7.18.4.4 records_	94
7.18.4.5 size_	94
7.18.4.6 writeIndex_	94
7.19 aare::RawFile Class Reference	94
7.19.1 Detailed Description	97
7.19.2 Constructor & Destructor Documentation	97
7.19.2.1 RawFile()	97
7.19.2.2 ~RawFile()	97
7.19.3 Member Function Documentation	97
7.19.3.1 bitdepth()	97
7.19.3.2 bytes_per_frame()	98
7.19.3.3 cols()	98
7.19.3.4 data_fname()	98
7.19.3.5 find_geometry()	98
7.19.3.6 find_number_of_subfiles()	99
7.19.3.7 frame_number()	99
7.19.3.8 get_frame()	99
7.19.3.9 get_frame_into()	99
7.19.3.10 iread() [1/2]	100
7.19.3.11 iread() [2/2]	100
7.19.3.12 is_master_file()	100
7.19.3.13 master_fname()	101
7.19.3.14 open_subfiles()	101
7.19.3.15 parse_fname()	101
7.19.3.16 parse_json_metadata()	101
7.19.3.17 parse_metadata()	101
7.19.3.18 parse_raw_metadata()	101
7.19.3.19 pixels()	101
7.19.3.20 read() [1/2]	101
7.19.3.21 read() [2/2]	102
7.19.3.22 read_header()	102
7.19.3.23 read_into() [1/2]	103
7.19.3.24 read_into() [2/2]	103



7.19.3.25 rows()	103
7.19.3.26 seek()	103
7.19.3.27 set_config()	104
7.19.3.28 tell()	104
7.19.3.29 total_frames()	104
7.19.3.30 write()	104
7.19.4 Field Documentation	105
7.19.4.1 cfg	105
7.19.4.2 current_frame	105
7.19.4.3 geometry	105
7.19.4.4 m_base_name	105
7.19.4.5 m_base_path	105
7.19.4.6 m_bitdepth	105
7.19.4.7 m_cols	105
7.19.4.8 m_ext	105
7.19.4.9 m_index	106
7.19.4.10 m_fname	106
7.19.4.11 m_mode	106
7.19.4.12 m_rows	106
7.19.4.13 m_total_frames	106
7.19.4.14 m_type	106
7.19.4.15 max_frames_per_file	106
7.19.4.16 n_subfile_parts	106
7.19.4.17 n_subfiles	106
7.19.4.18 positions	106
7.19.4.19 quad	107
7.19.4.20 subfile_cols	107
7.19.4.21 subfile_rows	107
7.19.4.22 subfiles	107
7.19.4.23 timing_mode	107
7.19.4.24 version	107
7.20 aare::RawFileConfig Struct Reference	107
7.20.1 Member Function Documentation	108
7.20.1.1 operator==()	108
7.20.2 Field Documentation	108
7.20.2.1 module_gap_col	108
7.20.2.2 module_gap_row	108
7.21 aare::sls_detector_header Struct Reference	108
7.21.1 Field Documentation	108
7.21.1.1 bunchId	108
7.21.1.2 column	109
7.21.1.3 debug	109

7.21.1.4 detType . . . . .	109
7.21.1.5 expLength . . . . .	109
7.21.1.6 frameNumber . . . . .	109
7.21.1.7 modId . . . . .	109
7.21.1.8 packetMask . . . . .	109
7.21.1.9 packetNumber . . . . .	109
7.21.1.10 reserved . . . . .	109
7.21.1.11 roundRNumber . . . . .	109
7.21.1.12 row . . . . .	110
7.21.1.13 timestamp . . . . .	110
7.21.1.14 version . . . . .	110
7.22 aare::SubFile Class Reference . . . . .	110
7.22.1 Detailed Description . . . . .	111
7.22.2 Member Typedef Documentation . . . . .	111
7.22.2.1 pfunc . . . . .	111
7.22.3 Constructor & Destructor Documentation . . . . .	111
7.22.3.1 SubFile() . . . . .	111
7.22.4 Member Function Documentation . . . . .	112
7.22.4.1 bytes_per_part() . . . . .	112
7.22.4.2 frame_number() . . . . .	112
7.22.4.3 get_part() . . . . .	112
7.22.4.4 pixels_per_part() . . . . .	112
7.22.4.5 read_impl_flip() . . . . .	113
7.22.4.6 read_impl_normal() . . . . .	113
7.22.4.7 read_impl_reorder() . . . . .	113
7.22.5 Field Documentation . . . . .	114
7.22.5.1 fp . . . . .	114
7.22.5.2 m_bitdepth . . . . .	114
7.22.5.3 m_cols . . . . .	114
7.22.5.4 m_fname . . . . .	114
7.22.5.5 m_rows . . . . .	114
7.22.5.6 m_sub_file_index_ . . . . .	114
7.22.5.7 n_frames . . . . .	114
7.22.5.8 read_impl . . . . .	114
7.22.5.9 read_impl_map . . . . .	115
7.23 aare::xy Struct Reference . . . . .	115
7.23.1 Member Function Documentation . . . . .	115
7.23.1.1 operator!=(()) . . . . .	115
7.23.1.2 operator==(()) . . . . .	116
7.23.2 Field Documentation . . . . .	116
7.23.2.1 col . . . . .	116
7.23.2.2 row . . . . .	116

7.24 aare::ZmqFrame Struct Reference . . . . .	116
7.24.1 Detailed Description . . . . .	116
7.24.2 Field Documentation . . . . .	116
7.24.2.1 frame . . . . .	116
7.24.2.2 header . . . . .	117
7.25 aare::ZmqHeader Struct Reference . . . . .	117
7.25.1 Detailed Description . . . . .	118
7.25.2 Member Function Documentation . . . . .	118
7.25.2.1 from_string() . . . . .	118
7.25.2.2 operator==() . . . . .	118
7.25.2.3 to_string() . . . . .	118
7.25.3 Field Documentation . . . . .	118
7.25.3.1 acqIndex . . . . .	118
7.25.3.2 addJsonHeader . . . . .	118
7.25.3.3 column . . . . .	118
7.25.3.4 completeImage . . . . .	118
7.25.3.5 data . . . . .	119
7.25.3.6 detSpec1 . . . . .	119
7.25.3.7 detSpec2 . . . . .	119
7.25.3.8 detSpec3 . . . . .	119
7.25.3.9 detSpec4 . . . . .	119
7.25.3.10 detType . . . . .	119
7.25.3.11 dynamicRange . . . . .	119
7.25.3.12 expLength . . . . .	119
7.25.3.13 fileIndex . . . . .	119
7.25.3.14 flipRows . . . . .	119
7.25.3.15 fname . . . . .	120
7.25.3.16 frameIndex . . . . .	120
7.25.3.17 frameNumber . . . . .	120
7.25.3.18 imageSize . . . . .	120
7.25.3.19 jsonversion . . . . .	120
7.25.3.20 modId . . . . .	120
7.25.3.21 ndetx . . . . .	120
7.25.3.22 ndety . . . . .	120
7.25.3.23 npixelsx . . . . .	121
7.25.3.24 npixelsy . . . . .	121
7.25.3.25 packetNumber . . . . .	121
7.25.3.26 progress . . . . .	121
7.25.3.27 quad . . . . .	121
7.25.3.28 row . . . . .	121
7.25.3.29 rx_roi . . . . .	121
7.25.3.30 timestamp . . . . .	121

7.25.3.31 version	122
7.26 aare::ZmqSocket Class Reference	122
7.26.1 Constructor & Destructor Documentation	123
7.26.1.1 ZmqSocket() [1/3]	123
7.26.1.2 ~ZmqSocket()	123
7.26.1.3 ZmqSocket() [2/3]	123
7.26.1.4 ZmqSocket() [3/3]	123
7.26.2 Member Function Documentation	123
7.26.2.1 disconnect()	123
7.26.2.2 operator=()	123
7.26.2.3 set_potential_frame_size()	123
7.26.2.4 set_timeout_ms()	123
7.26.2.5 set_zmq_hwm()	124
7.26.3 Field Documentation	124
7.26.3.1 m_context	124
7.26.3.2 m_endpoint	124
7.26.3.3 m_header_buffer	124
7.26.3.4 m_max_header_size	124
7.26.3.5 m_potential_frame_size	124
7.26.3.6 m_socket	124
7.26.3.7 m_timeout_ms	124
7.26.3.8 m_zmq_hwm	124
7.27 aare::ZmqSocketReceiver Class Reference	125
7.27.1 Constructor & Destructor Documentation	126
7.27.1.1 ZmqSocketReceiver()	126
7.27.2 Member Function Documentation	126
7.27.2.1 connect()	126
7.27.2.2 disconnect()	126
7.27.2.3 receive_data()	126
7.27.2.4 receive_header()	126
7.27.2.5 receive_n()	127
7.27.2.6 receive_zmqframe()	127
7.27.2.7 set_potential_frame_size()	127
7.27.2.8 set_timeout_ms()	127
7.27.2.9 set_zmq_hwm()	127
7.27.3 Field Documentation	127
7.27.3.1 m_context	127
7.27.3.2 m_endpoint	127
7.27.3.3 m_header_buffer	127
7.27.3.4 m_max_header_size	127
7.27.3.5 m_potential_frame_size	128
7.27.3.6 m_socket	128

7.27.3.7 m_timeout_ms	128
7.27.3.8 m_zmq_hwm	128
7.28 aare::ZmqSocketSender Class Reference	128
7.28.1 Constructor & Destructor Documentation	129
7.28.1.1 ZmqSocketSender()	129
7.28.2 Member Function Documentation	129
7.28.2.1 bind()	129
7.28.2.2 disconnect()	129
7.28.2.3 send() [1/3]	129
7.28.2.4 send() [2/3]	130
7.28.2.5 send() [3/3]	130
7.28.2.6 set_potential_frame_size()	130
7.28.2.7 set_timeout_ms()	131
7.28.2.8 set_zmq_hwm()	131
7.28.3 Field Documentation	131
7.28.3.1 m_context	131
7.28.3.2 m_endpoint	131
7.28.3.3 m_header_buffer	131
7.28.3.4 m_max_header_size	131
7.28.3.5 m_potential_frame_size	131
7.28.3.6 m_socket	131
7.28.3.7 m_timeout_ms	131
7.28.3.8 m_zmq_hwm	131
<b>8 File Documentation</b>	<b>133</b>
8.1 core/include/aare/core/CircularFifo.hpp File Reference	133
8.2 CircularFifo.hpp	133
8.3 core/include/aare/core/defs.hpp File Reference	135
8.4 defs.hpp	136
8.5 network_io/include/aare/network_io/defs.hpp File Reference	136
8.6 defs.hpp	137
8.7 core/include/aare/core/DType.hpp File Reference	137
8.8 DType.hpp	138
8.9 core/include/aare/core/Frame.hpp File Reference	138
8.10 Frame.hpp	139
8.11 core/include/aare/core/NDArray.hpp File Reference	140
8.12 NDArray.hpp	141
8.13 core/include/aare/core/NDView.hpp File Reference	145
8.14 NDView.hpp	146
8.15 core/include/aare/core/ProducerConsumerQueue.hpp File Reference	148
8.15.1 Variable Documentation	148
8.15.1.1 hardware_destructive_interference_size	148

8.16 ProducerConsumerQueue.hpp . . . . .	149
8.17 core/include/aare/core/VariableSizeClusterFinder.hpp File Reference . . . . .	151
8.17.1 Variable Documentation . . . . .	151
8.17.1.1 MAX_CLUSTER_SIZE . . . . .	151
8.18 VariableSizeClusterFinder.hpp . . . . .	152
8.19 core/src/defs.cpp File Reference . . . . .	155
8.20 core/src/DType.cpp File Reference . . . . .	156
8.21 core/src/Frame.cpp File Reference . . . . .	156
8.22 core/test/CircularFifo.test.cpp File Reference . . . . .	156
8.22.1 Function Documentation . . . . .	157
8.22.1.1 TEST_CASE() [1/8] . . . . .	157
8.22.1.2 TEST_CASE() [2/8] . . . . .	157
8.22.1.3 TEST_CASE() [3/8] . . . . .	157
8.22.1.4 TEST_CASE() [4/8] . . . . .	157
8.22.1.5 TEST_CASE() [5/8] . . . . .	157
8.22.1.6 TEST_CASE() [6/8] . . . . .	157
8.22.1.7 TEST_CASE() [7/8] . . . . .	157
8.22.1.8 TEST_CASE() [8/8] . . . . .	157
8.23 core/test/defs.test.cpp File Reference . . . . .	158
8.23.1 Function Documentation . . . . .	158
8.23.1.1 TEST_CASE() . . . . .	158
8.24 core/test/DType.test.cpp File Reference . . . . .	158
8.24.1 Function Documentation . . . . .	158
8.24.1.1 TEST_CASE() [1/4] . . . . .	158
8.24.1.2 TEST_CASE() [2/4] . . . . .	158
8.24.1.3 TEST_CASE() [3/4] . . . . .	159
8.24.1.4 TEST_CASE() [4/4] . . . . .	159
8.25 core/test/Frame.test.cpp File Reference . . . . .	159
8.25.1 Function Documentation . . . . .	159
8.25.1.1 TEST_CASE() [1/4] . . . . .	159
8.25.1.2 TEST_CASE() [2/4] . . . . .	159
8.25.1.3 TEST_CASE() [3/4] . . . . .	159
8.25.1.4 TEST_CASE() [4/4] . . . . .	159
8.26 core/test/NDArry.test.cpp File Reference . . . . .	160
8.26.1 Function Documentation . . . . .	160
8.26.1.1 TEST_CASE() [1/14] . . . . .	160
8.26.1.2 TEST_CASE() [2/14] . . . . .	160
8.26.1.3 TEST_CASE() [3/14] . . . . .	160
8.26.1.4 TEST_CASE() [4/14] . . . . .	160
8.26.1.5 TEST_CASE() [5/14] . . . . .	161
8.26.1.6 TEST_CASE() [6/14] . . . . .	161
8.26.1.7 TEST_CASE() [7/14] . . . . .	161

8.26.1.8 TEST_CASE() [8/14]	161
8.26.1.9 TEST_CASE() [9/14]	161
8.26.1.10 TEST_CASE() [10/14]	161
8.26.1.11 TEST_CASE() [11/14]	161
8.26.1.12 TEST_CASE() [12/14]	161
8.26.1.13 TEST_CASE() [13/14]	161
8.26.1.14 TEST_CASE() [14/14]	162
8.27 core/test/NDView.test.cpp File Reference	162
8.27.1 Function Documentation	162
8.27.1.1 TEST_CASE() [1/11]	162
8.27.1.2 TEST_CASE() [2/11]	162
8.27.1.3 TEST_CASE() [3/11]	162
8.27.1.4 TEST_CASE() [4/11]	163
8.27.1.5 TEST_CASE() [5/11]	163
8.27.1.6 TEST_CASE() [6/11]	163
8.27.1.7 TEST_CASE() [7/11]	163
8.27.1.8 TEST_CASE() [8/11]	163
8.27.1.9 TEST_CASE() [9/11]	163
8.27.1.10 TEST_CASE() [10/11]	163
8.27.1.11 TEST_CASE() [11/11]	163
8.28 core/test/ProducerConsumerQueue.test.cpp File Reference	163
8.28.1 Function Documentation	164
8.28.1.1 TEST_CASE() [1/3]	164
8.28.1.2 TEST_CASE() [2/3]	164
8.28.1.3 TEST_CASE() [3/3]	164
8.29 core/test/wrappers.test.cpp File Reference	164
8.29.1 Function Documentation	164
8.29.1.1 TEST_CASE() [1/3]	164
8.29.1.2 TEST_CASE() [2/3]	165
8.29.1.3 TEST_CASE() [3/3]	165
8.30 data/jungfrau/read_frame.py File Reference	165
8.31 python/example/read_frame.py File Reference	165
8.32 data/numpy/write_test_files.py File Reference	166
8.33 data/scripts/read_first_frame_number.py File Reference	166
8.34 data/jungfrau/read_multiport.py File Reference	166
8.35 data/scripts/read_multiport.py File Reference	167
8.36 examples/json_example.cpp File Reference	167
8.36.1 Macro Definition Documentation	167
8.36.1.1 AARE_ROOT_DIR_VAR	167
8.36.2 Function Documentation	167
8.36.2.1 main()	167
8.36.2.2 test()	167

8.37 examples/logger_example.cpp File Reference . . . . .	167
8.37.1 Function Documentation . . . . .	168
8.37.1.1 main() . . . . .	168
8.38 examples/multiport_example.cpp File Reference . . . . .	168
8.38.1 Macro Definition Documentation . . . . .	168
8.38.1.1 AARE_ROOT_DIR_VAR . . . . .	168
8.38.2 Function Documentation . . . . .	168
8.38.2.1 main() . . . . .	168
8.38.2.2 test() . . . . .	168
8.39 examples/mythen_example.cpp File Reference . . . . .	169
8.39.1 Macro Definition Documentation . . . . .	169
8.39.1.1 AARE_ROOT_DIR_VAR . . . . .	169
8.39.2 Function Documentation . . . . .	169
8.39.2.1 main() . . . . .	169
8.39.2.2 test1() . . . . .	169
8.39.2.3 test2() . . . . .	169
8.40 examples/numpy_read_example.cpp File Reference . . . . .	169
8.40.1 Macro Definition Documentation . . . . .	170
8.40.1.1 AARE_ROOT_DIR_VAR . . . . .	170
8.40.2 Function Documentation . . . . .	170
8.40.2.1 main() . . . . .	170
8.40.2.2 test() . . . . .	170
8.41 examples/numpy_write_example.cpp File Reference . . . . .	170
8.41.1 Macro Definition Documentation . . . . .	171
8.41.1.1 AARE_ROOT_DIR_VAR . . . . .	171
8.41.2 Function Documentation . . . . .	171
8.41.2.1 main() . . . . .	171
8.42 examples/raw_example.cpp File Reference . . . . .	171
8.42.1 Macro Definition Documentation . . . . .	171
8.42.1.1 AARE_ROOT_DIR_VAR . . . . .	171
8.42.2 Function Documentation . . . . .	171
8.42.2.1 main() . . . . .	171
8.42.2.2 test() . . . . .	171
8.43 examples/zmq_receiver_example.cpp File Reference . . . . .	172
8.43.1 Function Documentation . . . . .	172
8.43.1.1 main() . . . . .	172
8.44 examples/zmq_restream_example.cpp File Reference . . . . .	172
8.44.1 Function Documentation . . . . .	172
8.44.1.1 main() . . . . .	172
8.45 examples/zmq_sender_example.cpp File Reference . . . . .	173
8.45.1 Function Documentation . . . . .	173
8.45.1.1 main() . . . . .	173



8.46 file_io/include/aare/file_io/File.hpp File Reference	173
8.47 File.hpp	174
8.48 file_io/include/aare/file_io/FileInterface.hpp File Reference	174
8.49 FileInterface.hpp	175
8.50 file_io/include/aare/file_io/NumpyFile.hpp File Reference	176
8.51 NumpyFile.hpp	176
8.52 file_io/include/aare/file_io/NumpyHelpers.hpp File Reference	177
8.53 NumpyHelpers.hpp	178
8.54 file_io/include/aare/file_io/RawFile.hpp File Reference	179
8.55 RawFile.hpp	179
8.56 file_io/include/aare/file_io/SubFile.hpp File Reference	180
8.57 SubFile.hpp	181
8.58 file_io/src/File.cpp File Reference	181
8.59 file_io/src/NumpyFile.cpp File Reference	182
8.60 file_io/src/NumpyHelpers.cpp File Reference	182
8.61 file_io/src/RawFile.cpp File Reference	183
8.61.1 Typedef Documentation	183
8.61.1.1 json	183
8.62 file_io/src/SubFile.cpp File Reference	183
8.63 file_io/test/NumpyFile.test.cpp File Reference	183
8.63.1 Function Documentation	184
8.63.1.1 TEST_CASE() [1/2]	184
8.63.1.2 TEST_CASE() [2/2]	184
8.64 file_io/test/NumpyHelpers.test.cpp File Reference	184
8.64.1 Function Documentation	184
8.64.1.1 TEST_CASE() [1/7]	184
8.64.1.2 TEST_CASE() [2/7]	184
8.64.1.3 TEST_CASE() [3/7]	185
8.64.1.4 TEST_CASE() [4/7]	185
8.64.1.5 TEST_CASE() [5/7]	185
8.64.1.6 TEST_CASE() [6/7]	185
8.64.1.7 TEST_CASE() [7/7]	185
8.65 file_io/test/RawFile.test.cpp File Reference	185
8.65.1 Function Documentation	186
8.65.1.1 TEST_CASE() [1/5]	186
8.65.1.2 TEST_CASE() [2/5]	186
8.65.1.3 TEST_CASE() [3/5]	186
8.65.1.4 TEST_CASE() [4/5]	186
8.65.1.5 TEST_CASE() [5/5]	186
8.66 include/aare/aare.hpp File Reference	186
8.67 aare.hpp	186
8.68 network_io/include/aare/network_io/ZmqHeader.hpp File Reference	186

8.69 ZmqHeader.hpp	187
8.70 network_io/include/aare/network_io/ZmqSocket.hpp File Reference	188
8.71 ZmqSocket.hpp	189
8.72 network_io/include/aare/network_io/ZmqSocketReceiver.hpp File Reference	189
8.73 ZmqSocketReceiver.hpp	190
8.74 network_io/include/aare/network_io/ZmqSocketSender.hpp File Reference	190
8.75 ZmqSocketSender.hpp	191
8.76 network_io/src/ZmqHeader.cpp File Reference	191
8.76.1 Function Documentation	191
8.76.1.1 write_array()	191
8.76.1.2 write_digit()	191
8.76.1.3 write_map()	192
8.76.1.4 write_str()	192
8.77 network_io/src/ZmqSocket.cpp File Reference	192
8.78 network_io/src/ZmqSocketReceiver.cpp File Reference	193
8.79 network_io/src/ZmqSocketSender.cpp File Reference	193
8.80 network_io/test/ZmqHeader.test.cpp File Reference	193
8.80.1 Function Documentation	193
8.80.1.1 TEST_CASE()	193
8.81 python/aare/File.py File Reference	194
8.82 python/aare/Frame.py File Reference	194
8.83 python/aare/__init__.py File Reference	194
8.84 python/example/__init__.py File Reference	194
8.85 python/src/bindings.cpp File Reference	194
8.85.1 Function Documentation	195
8.85.1.1 PYBIND11_MODULE()	195
8.86 README.md File Reference	195
8.87 tests/test.cpp File Reference	195
8.87.1 Function Documentation	195
8.87.1.1 TEST_CASE() [1/2]	195
8.87.1.2 TEST_CASE() [2/2]	195
8.88 utils/include/aare/utils/logger.hpp File Reference	195
8.88.1 Macro Definition Documentation	197
8.88.1.1 LOCATION	197
8.88.2 Function Documentation	197
8.88.2.1 operator<<() [1/3]	197
8.88.2.2 operator<<() [2/3]	197
8.88.2.3 operator<<() [3/3]	197
8.89 logger.hpp	197
8.90 utils/src/logger.cpp File Reference	199
8.91 utils/test/logger.test.cpp File Reference	200





# Chapter 1

## aare

Data analysis library for PSI hybrid detectors

### 1.1 Folder structure

Folder	subfolder	Content
include/	aare/	top level header/s
core/	include/	public headers for core
	src/	source files and non public headers

### 1.2 file\_io class diagram

### 1.3 Test the zmq socket with a detector simulator

#### 1. Download and build the slsDetectorPackage

```
git clone https://github.com/slsdetectorgroup/slsDetectorPackage.git --branch=8.0.1 #or the desired branch
cd slsDetectorPackage
mkdir build && cd build
cmake .. -DSLS_USE_SIMULATOR=ON
make -j8 #or your number of cores
```

#### 2. Launch the slsReceiver

```
bin/slsReceiver
```

#### 3. Launch the virtual server

```
bin/jungfrauDetectorServer_virtual
```

#### 4 Configure the detector simulator

```
#sample config file is in etc/ in the aare repo
sls_detector_put config etc/virtual_jf.config
```

```
#Now you can take images using sls_detector_acquire
sls_detector_acquire
```

#### 5. Run the zmq example

```
examples/zmq_example
```

```
#Will print the headers fof the frames received
```

## **1.4 Test the zmq processing replaying data**

To be implemented

## **1.5 generate documentation**

## Chapter 2

# Namespace Index

### 2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">aare</a>	<a href="#">Frame</a> class to represent a single frame of data model class should be able to work with streams coming from files or network . . . . .	11
<a href="#">aare.File</a>	. . . . .	16
<a href="#">aare.Frame</a>	. . . . .	16
<a href="#">aare::logger</a>	. . . . .	16
<a href="#">aare::logger::internal</a>	. . . . .	19
<a href="#">aare::network_io</a>	. . . . .	19
<a href="#">aare::NumpyHelpers</a>	. . . . .	19
<a href="#">example</a>	. . . . .	22
<a href="#">example.read_frame</a>	. . . . .	22
<a href="#">folly</a>	. . . . .	23
<a href="#">read_first_frame_number</a>	. . . . .	23
<a href="#">read_frame</a>	. . . . .	24
<a href="#">read_multiport</a>	. . . . .	25
<a href="#">simdjson</a>	. . . . .	27
<a href="#">write_test_files</a>	. . . . .	27





## Chapter 3

# Hierarchical Index

### 3.1 Class Hierarchy

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

aare::CircularFifo< ItemType > . . . . .	29
aare::ClusterFinder< T > . . . . .	32
aare::DType . . . . .	38
aare::File . . . . .	40
aare.File.File . . . . .	43
aare::FileConfig . . . . .	45
aare::FileInterface . . . . .	46
aare::NumpyFile . . . . .	80
aare::RawFile . . . . .	94
aare::Frame . . . . .	54
aare.Frame.Frame . . . . .	57
aare::ClusterFinder< T >::Hit . . . . .	58
aare::logger::Logger . . . . .	60
MoveOnlyInt . . . . .	63
aare::NDArray< T, Ndim > . . . . .	64
aare::NDArray< bool, 2 > . . . . .	64
aare::NDArray< int, 2 > . . . . .	64
aare::NDView< T, Ndim > . . . . .	73
aare::NDView< T, 2 > . . . . .	73
aare::NumpyHeader . . . . .	90
folly::ProducerConsumerQueue< T > . . . . .	91
folly::ProducerConsumerQueue< ItemType > . . . . .	91
aare::RawFileConfig . . . . .	107
std::runtime_error . . . . .	
aare::network_io::NetworkError . . . . .	78
aare::sls_detector_header . . . . .	108
aare::SubFile . . . . .	110
aare::xy . . . . .	115
aare::ZmqFrame . . . . .	116
aare::ZmqHeader . . . . .	117
aare::ZmqSocket . . . . .	122
aare::ZmqSocketReceiver . . . . .	125
aare::ZmqSocketSender . . . . .	128



## Chapter 4

# Data Structure Index

### 4.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">aare::CircularFifo&lt; ItemType &gt;</a>	29
<a href="#">aare::ClusterFinder&lt; T &gt;</a>	32
<a href="#">aare::DType</a>	38
<a href="#">aare::File</a>	
RAII <a href="#">File</a> class for reading and writing image files in various formats wrapper on a <a href="#">FileInterface</a> to abstract the underlying file format	40
<a href="#">aare.File.File</a>	43
<a href="#">aare::FileConfig</a>	
<a href="#">FileConfig</a> structure to store the configuration of a file dtype: data type of the file rows: number of rows in the file cols: number of columns in the file geometry: geometry of the file	45
<a href="#">aare::FileInterface</a>	
<a href="#">FileInterface</a> class to define the interface for file operations	46
<a href="#">aare::Frame</a>	54
<a href="#">aare.Frame.Frame</a>	57
<a href="#">aare::ClusterFinder&lt; T &gt;::Hit</a>	58
<a href="#">aare::logger::Logger</a>	60
<a href="#">MoveOnlyInt</a>	63
<a href="#">aare::NDArray&lt; T, Ndim &gt;</a>	64
<a href="#">aare::NDView&lt; T, Ndim &gt;</a>	73
<a href="#">aare::network_io::NetworkError</a>	
<a href="#">NetworkError</a> exception class	78
<a href="#">aare::NumpyFile</a>	
<a href="#">NumpyFile</a> class to read and write numpy files	80
<a href="#">aare::NumpyHeader</a>	90
<a href="#">folly::ProducerConsumerQueue&lt; T &gt;</a>	91
<a href="#">aare::RawFile</a>	
<a href="#">RawFile</a> class to read .raw and .json files	94
<a href="#">aare::RawFileConfig</a>	107
<a href="#">aare::sls_detector_header</a>	108
<a href="#">aare::SubFile</a>	
Class to read a subfile from a <a href="#">RawFile</a>	110
<a href="#">aare::xy</a>	115
<a href="#">aare::ZmqFrame</a>	
<a href="#">ZmqFrame</a> structure wrapper class to contain a <a href="#">ZmqHeader</a> and a <a href="#">Frame</a>	116
<a href="#">aare::ZmqHeader</a>	117
<a href="#">aare::ZmqSocket</a>	122
<a href="#">aare::ZmqSocketReceiver</a>	125
<a href="#">aare::ZmqSocketSender</a>	128



# Chapter 5

## File Index

### 5.1 File List

Here is a list of all files with brief descriptions:

core/include/aare/core/CircularFifo.hpp	133
core/include/aare/core/defs.hpp	135
core/include/aare/core/DType.hpp	137
core/include/aare/core/Frame.hpp	138
core/include/aare/core/NDArray.hpp	140
core/include/aare/core/NDView.hpp	145
core/include/aare/core/ProducerConsumerQueue.hpp	148
core/include/aare/core/VariableSizeClusterFinder.hpp	151
core/src/defs.cpp	155
core/src/DType.cpp	156
core/src/Frame.cpp	156
core/test/CircularFifo.test.cpp	156
core/test/defs.test.cpp	158
core/test/DType.test.cpp	158
core/test/Frame.test.cpp	159
core/test/NDArray.test.cpp	160
core/test/NDView.test.cpp	162
core/test/ProducerConsumerQueue.test.cpp	163
core/test/wrappers.test.cpp	164
data/jungfrau/read_frame.py	165
data/jungfrau/read_multiport.py	166
data/numpy/write_test_files.py	166
data/scripts/read_first_frame_number.py	166
data/scripts/read_multiport.py	167
examples/json_example.cpp	167
examples/logger_example.cpp	167
examples/multiport_example.cpp	168
examples/mythen_example.cpp	169
examples/numpy_read_example.cpp	169
examples/numpy_write_example.cpp	170
examples/raw_example.cpp	171
examples/zmq_receiver_example.cpp	172
examples/zmq_restream_example.cpp	172
examples/zmq_sender_example.cpp	173
file_io/include/aare/file_io/File.hpp	173

file_io/include/aare/file_io/FileInterface.hpp	174
file_io/include/aare/file_io/NumpyFile.hpp	176
file_io/include/aare/file_io/NumpyHelpers.hpp	177
file_io/include/aare/file_io/RawFile.hpp	179
file_io/include/aare/file_io/SubFile.hpp	180
file_io/src/File.cpp	181
file_io/src/NumpyFile.cpp	182
file_io/src/NumpyHelpers.cpp	182
file_io/src/RawFile.cpp	183
file_io/src/SubFile.cpp	183
file_io/test/NumpyFile.test.cpp	183
file_io/test/NumpyHelpers.test.cpp	184
file_io/test/RawFile.test.cpp	185
include/aare/aare.hpp	186
network_io/include/aare/network_io/defs.hpp	136
network_io/include/aare/network_io/ZmqHeader.hpp	186
network_io/include/aare/network_io/ZmqSocket.hpp	188
network_io/include/aare/network_io/ZmqSocketReceiver.hpp	189
network_io/include/aare/network_io/ZmqSocketSender.hpp	190
network_io/src/ZmqHeader.cpp	191
network_io/src/ZmqSocket.cpp	192
network_io/src/ZmqSocketReceiver.cpp	193
network_io/src/ZmqSocketSender.cpp	193
network_io/test/ZmqHeader.test.cpp	193
python/aare/__init__.py	194
python/aare/File.py	194
python/aare/Frame.py	194
python/example/__init__.py	194
python/example/read_frame.py	165
python/src/bindings.cpp	194
tests/test.cpp	195
utils/include/aare/utils/logger.hpp	195
utils/src/logger.cpp	199
utils/test/logger.test.cpp	200

## Chapter 6

# Namespace Documentation

### 6.1 aare Namespace Reference

[Frame](#) class to represent a single frame of data model class should be able to work with streams coming from files or network.

#### Namespaces

- namespace [File](#)
- namespace [Frame](#)
- namespace [logger](#)
- namespace [network\\_io](#)
- namespace [NumpyHelpers](#)

#### Data Structures

- class [CircularFifo](#)
- class [ClusterFinder](#)
- class [DType](#)
- class [File](#)
  - [File](#) class for reading and writing image files in various formats wrapper on a [FileInterface](#) to abstract the underlying file format.*
- struct [FileConfig](#)
  - [FileConfig](#) structure to store the configuration of a file dtype: data type of the file rows: number of rows in the file cols: number of columns in the file geometry: geometry of the file.*
- class [FileInterface](#)
  - [FileInterface](#) class to define the interface for file operations.*
- class [Frame](#)
- class [NDArray](#)
- class [NDView](#)
- class [NumpyFile](#)
  - [NumpyFile](#) class to read and write numpy files.*
- struct [NumpyHeader](#)
- class [RawFile](#)
  - [RawFile](#) class to read .raw and .json files.*
- struct [RawFileConfig](#)

- struct [sls\\_detector\\_header](#)
- class [SubFile](#)  
*Class to read a subfile from a [RawFile](#).*
- struct [xy](#)
- struct [ZmqFrame](#)  
*[ZmqFrame](#) structure wrapper class to contain a [ZmqHeader](#) and a [Frame](#).*
- struct [ZmqHeader](#)
- class [ZmqSocket](#)
- class [ZmqSocketReceiver](#)
- class [ZmqSocketSender](#)

## Typedefs

- using [dynamic\\_shape](#) = std::vector< ssize\_t >
- using [DataTypeVariants](#) = std::variant< uint16\_t, uint32\_t >
- template<ssize\_t Ndim>  
using [Shape](#) = std::array< ssize\_t, Ndim >
- using [shape\\_t](#) = std::vector< size\_t >

## Enumerations

- enum class [DetectorType](#) {  
[Jungfrau](#) , [Eiger](#) , [Mythen3](#) , [Moench](#) ,  
[ChipTestBoard](#) }
- enum class [TimingMode](#) { [Auto](#) , [Trigger](#) }
- enum class [endian](#) { [little](#) = \_\_ORDER\_LITTLE\_ENDIAN\_\_ , [big](#) = \_\_ORDER\_BIG\_ENDIAN\_\_ , [native](#) = \_\_BYTE\_ORDER\_\_ }

## Functions

- template<class T >  
T [StringTo](#) (std::string sv)
- template<class T >  
std::string [toString](#) (T sv)
- template<> [DetectorType](#) [StringTo](#) (std::string)
- template<> std::string [toString](#) ([DetectorType](#) type)
- template<> [TimingMode](#) [StringTo](#) (std::string)
- template<typename T , ssize\_t Ndim>  
void [save](#) ([NDArray](#)< T, Ndim > &img, std::string pathname)
- template<typename T , ssize\_t Ndim>  
[NDArray](#)< T, Ndim > [load](#) (const std::string &pathname, std::array< ssize\_t, Ndim > shape)
- template<ssize\_t Ndim>  
[Shape](#)< Ndim > [make\\_shape](#) (const std::vector< size\_t > &shape)
- template<ssize\_t Dim = 0, typename Strides >  
ssize\_t [element\\_offset](#) (const Strides &)
- template<ssize\_t Dim = 0, typename Strides , typename... lx>  
ssize\_t [element\\_offset](#) (const Strides &strides, ssize\_t i, lx... index)
- template<ssize\_t Ndim>  
std::array< ssize\_t, Ndim > [c\\_strides](#) (const std::array< ssize\_t, Ndim > &shape)
- template<ssize\_t Ndim>  
std::array< ssize\_t, Ndim > [make\\_array](#) (const std::vector< ssize\_t > &vec)
- template<> std::string [toString](#) ([DetectorType](#) type)
- template<> [DetectorType](#) [StringTo](#) (std::string name)
- template<> [TimingMode](#) [StringTo](#) (std::string mode)



### 6.1.1 Detailed Description

`Frame` class to represent a single frame of data model class should be able to work with streams coming from files or network.

### 6.1.2 Typedef Documentation

#### 6.1.2.1 DataTypeVariants

```
using aare::DataTypeVariants = typedef std::variant<uint16_t, uint32_t>
```

#### 6.1.2.2 dynamic\_shape

```
using aare::dynamic_shape = typedef std::vector<ssize_t>
```

#### 6.1.2.3 Shape

```
template<ssize_t Ndim>
using aare::Shape = typedef std::array<ssize_t, Ndim>
```

#### 6.1.2.4 shape\_t

```
using aare::shape_t = typedef std::vector<size_t>
```

### 6.1.3 Enumeration Type Documentation

#### 6.1.3.1 DetectorType

```
enum class aare::DetectorType [strong]
```

Enumerator

Jungfrau	
Eiger	
Mythen3	
Moench	
ChipTestBoard	

#### 6.1.3.2 endian

```
enum class aare::endian [strong]
```

**Enumerator**

little	
big	
native	

**6.1.3.3 TimingMode**

```
enum class aare::TimingMode [strong]
```

**Enumerator**

Auto	
Trigger	

**6.1.4 Function Documentation****6.1.4.1 c\_strides()**

```
template<ssize_t Ndim>
std::array< ssize_t, Ndim > aare::c_strides (
    const std::array< ssize_t, Ndim > & shape )
```

**6.1.4.2 element\_offset() [1/2]**

```
template<ssize_t Dim = 0, typename Strides >
ssize_t aare::element_offset (
    const Strides & )
```

**6.1.4.3 element\_offset() [2/2]**

```
template<ssize_t Dim = 0, typename Strides , typename... Ix>
ssize_t aare::element_offset (
    const Strides & strides,
    ssize_t i,
    Ix... index )
```

**6.1.4.4 load()**

```
template<typename T , ssize_t Ndim>
NDArray< T, Ndim > aare::load (
    const std::string & pathname,
    std::array< ssize_t, Ndim > shape )
```

#### 6.1.4.5 make\_array()

```
template<ssize_t Ndim>
std::array< ssize_t, Ndim > aare::make_array (
    const std::vector< ssize_t > & vec )
```

#### 6.1.4.6 make\_shape()

```
template<ssize_t Ndim>
Shape< Ndim > aare::make_shape (
    const std::vector< size_t > & shape )
```

#### 6.1.4.7 save()

```
template<typename T , ssize_t Ndim>
void aare::save (
    NArray< T, Ndim > & img,
    std::string pathname )
```

#### 6.1.4.8 StringTo() [1/5]

```
template<>
TimingMode aare::StringTo (
    std::string mode )
```

#### 6.1.4.9 StringTo() [2/5]

```
template<>
DetectorType aare::StringTo (
    std::string name )
```

#### 6.1.4.10 StringTo() [3/5]

```
template<class T >
T aare::StringTo (
    std::string sv )
```

#### 6.1.4.11 StringTo() [4/5]

```
template<>
DetectorType aare::StringTo (
    std::string name )
```

#### 6.1.4.12 StringTo() [5/5]

```
template<>
TimingMode aare::StringTo (
    std::string name )
```

#### 6.1.4.13 toString() [1/3]

```
template<>
std::string aare::toString (
    DetectorType type )
```

#### 6.1.4.14 toString() [2/3]

```
template<>
std::string aare::toString (
    DetectorType type )
```

#### 6.1.4.15 toString() [3/3]

```
template<class T >
std::string aare::toString (
    T sv )
```

## 6.2 aare.File Namespace Reference

### Data Structures

- class [File](#)

## 6.3 aare.Frame Namespace Reference

### Data Structures

- class [Frame](#)

## 6.4 aare::logger Namespace Reference

### Namespaces

- namespace [internal](#)

## Data Structures

- class [Logger](#)

## Enumerations

- enum [LOGGING\\_LEVEL](#) { [DEBUG](#) = 0 , [INFO](#) = 1 , [WARNING](#) = 2 , [ERROR](#) = 3 }

## Functions

- template<[LOGGING\\_LEVEL](#) level, typename... Strings>  
void [log](#) (const Strings... s)
- template<typename... Strings>  
void [debug](#) (const Strings... s)
- template<typename... Strings>  
void [info](#) (const Strings... s)
- template<typename... Strings>  
void [warn](#) (const Strings... s)
- template<typename... Strings>  
void [error](#) (const Strings... s)
- void [set\\_streams](#) (std::streambuf \*out, std::streambuf \*err)
- void [set\\_streams](#) (std::streambuf \*out)
- void [set\\_verbosity](#) ([LOGGING\\_LEVEL](#) level)
- void [set\\_output\\_file](#) (std::string filename)
- [Logger](#) & [get\\_logger\\_instance](#) ()

## 6.4.1 Enumeration Type Documentation

### 6.4.1.1 LOGGING\_LEVEL

```
enum aare::logger::LOGGING_LEVEL
```

#### Enumerator

DEBUG	
INFO	
WARNING	
ERROR	

## 6.4.2 Function Documentation

### 6.4.2.1 debug()

```
template<typename... Strings>
void aare::logger::debug (
    const Strings... s )
```

#### 6.4.2.2 error()

```
template<typename... Strings>
void aare::logger::error (
    const Strings... s )
```

#### 6.4.2.3 get\_logger\_instance()

```
Logger & aare::logger::get_logger_instance ( )
```

#### 6.4.2.4 info()

```
template<typename... Strings>
void aare::logger::info (
    const Strings... s )
```

#### 6.4.2.5 log()

```
template<LOGGING_LEVEL level, typename... Strings>
void aare::logger::log (
    const Strings... s )
```

#### 6.4.2.6 set\_output\_file()

```
void aare::logger::set_output_file (
    std::string filename )
```

#### 6.4.2.7 set\_streams() [1/2]

```
void aare::logger::set_streams (
    std::streambuf * out )
```

#### 6.4.2.8 set\_streams() [2/2]

```
void aare::logger::set_streams (
    std::streambuf * out,
    std::streambuf * err )
```

#### 6.4.2.9 set\_verbosity()

```
void aare::logger::set_verbosity (
    LOGGING_LEVEL level )
```

#### 6.4.2.10 warn()

```
template<typename... Strings>
void aare::logger::warn (
    const Strings... s )
```

## 6.5 aare::logger::internal Namespace Reference

### Variables

- [aare::logger::Logger logger\\_instance](#) = [aare::logger::Logger\(\)](#)

### 6.5.1 Variable Documentation

#### 6.5.1.1 logger\_instance

```
aare::logger::Logger aare::logger::internal::logger_instance = aare::logger::Logger\(\)
```

## 6.6 aare::network\_io Namespace Reference

### Data Structures

- class [NetworkError](#)  
*[NetworkError](#) exception class.*

## 6.7 aare::NumpyHelpers Namespace Reference

### Functions

- `std::string parse\_str (const std::string &in)`
- `std::string trim (const std::string &str)`
- `std::vector< std::string > parse\_tuple (std::string in)`
- `bool parse\_bool (const std::string &in)`
- `std::string get\_value\_from\_map (const std::string &mapstr)`
- `std::unordered_map< std::string, std::string > parse\_dict (std::string in, const std::vector< std::string > &keys)`
- `template<typename T, size_t N>  
bool in\_array (T val, const std::array< T, N > &arr)`
- `bool is\_digits (const std::string &str)`
- `aare::DType parse\_descr (std::string typestring)`
- `size_t write\_header (std::filesystem::path fname, const NumpyHeader &header)`
- `size_t write\_header (std::ostream &out, const NumpyHeader &header)`
- `void write\_magic (std::ostream &ostream, int version_major, int version_minor)`
- `template<typename T >  
std::string write\_tuple (const std::vector< T > &v)`
- `std::string write\_boolean (bool b)`
- `std::string write\_header\_dict (const std::string &descr, bool fortran_order, const shape\_t &shape)`

## Variables

- `constexpr std::array< char, 6 > magic_str {"\x93", 'N', 'U', 'M', 'P', 'Y'}`
- `constexpr uint8_t magic_string_length {6}`

## 6.7.1 Function Documentation

### 6.7.1.1 `get_value_from_map()`

```
std::string aare::NumpyHelpers::get_value_from_map (
    const std::string & mapstr )
```

### 6.7.1.2 `in_array()`

```
template<typename T , size_t N>
bool aare::NumpyHelpers::in_array (
    T val,
    const std::array< T, N > & arr )
```

### 6.7.1.3 `is_digits()`

```
bool aare::NumpyHelpers::is_digits (
    const std::string & str )
```

### 6.7.1.4 `parse_bool()`

```
bool aare::NumpyHelpers::parse_bool (
    const std::string & in )
```

### 6.7.1.5 `parse_descr()`

```
aare::DType aare::NumpyHelpers::parse_descr (
    std::string typestring )
```

### 6.7.1.6 `parse_dict()`

```
std::unordered_map< std::string, std::string > aare::NumpyHelpers::parse_dict (
    std::string in,
    const std::vector< std::string > & keys )
```

### 6.7.1.7 `parse_str()`

```
std::string aare::NumpyHelpers::parse_str (
    const std::string & in )
```



#### 6.7.1.8 parse\_tuple()

```
std::vector< std::string > aare::NumpyHelpers::parse_tuple (
    std::string in )
```

#### 6.7.1.9 trim()

```
std::string aare::NumpyHelpers::trim (
    const std::string & str )
```

Removes leading and trailing whitespaces

#### 6.7.1.10 write\_boolean()

```
std::string aare::NumpyHelpers::write_boolean (
    bool b ) [inline]
```

#### 6.7.1.11 write\_header() [1/2]

```
size_t aare::NumpyHelpers::write_header (
    std::filesystem::path fname,
    const NumpyHeader & header )
```

#### 6.7.1.12 write\_header() [2/2]

```
size_t aare::NumpyHelpers::write_header (
    std::ostream & out,
    const NumpyHeader & header )
```

#### 6.7.1.13 write\_header\_dict()

```
std::string aare::NumpyHelpers::write_header_dict (
    const std::string & descr,
    bool fortran_order,
    const shape_t & shape ) [inline]
```

#### 6.7.1.14 write\_magic()

```
void aare::NumpyHelpers::write_magic (
    std::ostream & ostream,
    int version_major,
    int version_minor )
```

### 6.7.1.15 write\_tuple()

```
template<typename T >
std::string aare::NumpyHelpers::write_tuple (
    const std::vector< T > & v ) [inline]
```

## 6.7.2 Variable Documentation

### 6.7.2.1 magic\_str

```
const constexpr std::array<char, 6> aare::NumpyHelpers::magic_str {'\x93', 'N', 'U', 'M', 'P',
'Y'} [constexpr]
```

### 6.7.2.2 magic\_string\_length

```
const uint8_t aare::NumpyHelpers::magic_string_length {6}
```

## 6.8 example Namespace Reference

### Namespaces

- namespace [read\\_frame](#)

## 6.9 example.read\_frame Namespace Reference

### Variables

- [root\\_dir](#) = Path(os.environ.get("PROJECT\_ROOT\_DIR"))
- [data\\_path](#) = str([root\\_dir](#) / "data"/"jungfrau\_single\_master\_0.json")
- [file](#) = File([data\\_path](#))
- [frame](#) = file.get\_frame(0)
- [arr](#) = np.array(frame.get\_array())

### 6.9.1 Variable Documentation

#### 6.9.1.1 arr

```
example.read_frame.arr = np.array(frame.get_array())
```

#### 6.9.1.2 data\_path

```
example.read_frame.data_path = str(root\_dir / "data"/"jungfrau_single_master_0.json")
```

### 6.9.1.3 file

```
example.read_frame.file = File(data_path)
```

### 6.9.1.4 frame

```
example.read_frame.frame = file.get_frame(0)
```

### 6.9.1.5 root\_dir

```
example.read_frame.root_dir = Path(os.environ.get("PROJECT_ROOT_DIR"))
```

## 6.10 folly Namespace Reference

### Data Structures

- struct [ProducerConsumerQueue](#)

## 6.11 read\_first\_frame\_number Namespace Reference

### Variables

- [header\\_dt](#)
- [frame\\_number](#) = np.fromfile(f, dtype=[header\\_dt](#), count=1)[["Frame Number"](#)][0]

### 6.11.1 Variable Documentation

#### 6.11.1.1 frame\_number

```
read_first_frame_number.frame_number = np.fromfile(f, dtype=header\_dt, count=1)["Frame Number"][0]
```

#### 6.11.1.2 header\_dt

```
read_first_frame_number.header_dt
```

#### Initial value:

```
00001 = np.dtype(
00002     [
00003         ("Frame Number", "u8"),
00004         ("SubFrame Number/ExpLength", "u4"),
00005         ("Packet Number", "u4"),
00006         ("Bunch ID", "u8"),
00007         ("Timestamp", "u8"),
00008         ("Module Id", "u2"),
00009         ("Row", "u2"),
00010         ("Column", "u2"),
00011         ("Reserved", "u2"),
00012         ("Debug", "u4"),
00013         ("Round Robin Number", "u2"),
00014         ("Detector Type", "u1"),
00015         ("Header Version", "u1"),
00016         ("Packets caught mask", "8u8")
00017     ]
00018 )
```

## 6.12 read\_frame Namespace Reference

### Variables

- `header_dt`
- `int rows = 512`
- `int cols = 1024`
- `int frames = 10`
- `data = np.zeros((frames,rows,cols), dtype = np.uint16)`
- `header = np.zeros(frames, dtype = header_dt)`
- `str file_name = 'jungfrau_single_d0_f{}_0.raw'.format(file_id)`
- `f`
- `dtype`
- `count`
- `uint16`

### 6.12.1 Variable Documentation

#### 6.12.1.1 cols

```
int read_frame.cols = 1024
```

#### 6.12.1.2 count

```
read_frame.count
```

#### 6.12.1.3 data

```
read_frame.data = np.zeros((frames,rows,cols), dtype = np.uint16)
```

#### 6.12.1.4 dtype

```
read_frame.dtype
```

#### 6.12.1.5 f

```
read_frame.f
```

#### 6.12.1.6 file\_name

```
str read_frame.file_name = 'jungfrau_single_d0_f{}_0.raw'.format(file_id)
```

#### 6.12.1.7 frames

```
int read_frame.frames = 10
```

### 6.12.1.8 header

```
read_frame.header = np.zeros(frames, dtype = header_dt)
```

### 6.12.1.9 header\_dt

```
read_frame.header_dt
```

#### Initial value:

```
00001 = np.dtype(
00002     [
00003         ("Frame Number", "u8"),
00004         ("SubFrame Number/ExpLength", "u4"),
00005         ("Packet Number", "u4"),
00006         ("Bunch ID", "u8"),
00007         ("Timestamp", "u8"),
00008         ("Module Id", "u2"),
00009         ("Row", "u2"),
00010         ("Column", "u2"),
00011         ("Reserved", "u2"),
00012         ("Debug", "u4"),
00013         ("Round Robin Number", "u2"),
00014         ("Detector Type", "u1"),
00015         ("Header Version", "u1"),
00016         ("Packets caught mask", "8u8")
00017     ]
00018 )
```

### 6.12.1.10 rows

```
int read_frame.rows = 512
```

### 6.12.1.11 uint16

```
read_frame.uint16
```

## 6.13 read\_multiport Namespace Reference

### Variables

- [header\\_dt](#)
- int [frames](#) = 1
- int [parts](#) = 2
- int [frame\\_cols](#) = 1024
- int [frame\\_rows](#) = 512
- int [part\\_cols](#) = 1024
- int [part\\_rows](#) = 256
- [parts\\_data](#) = np.zeros((frames,parts,part\_rows,part\_cols), dtype = np.uint16)
- [data](#) = np.zeros((frames,frame\_rows,frame\_cols), dtype = np.uint16)
- [header](#) = np.zeros((frames,parts), dtype = [header\\_dt](#))
- str [file\\_name](#) = f'jungfrau\_double\_d{part}\_f{frame}\_{0}.raw'
- [f](#)
- [dtype](#)
- [count](#)
- [uint16](#)
- [axis](#)

### 6.13.1 Variable Documentation

#### 6.13.1.1 axis

```
read_multiport.axis
```

#### 6.13.1.2 count

```
read_multiport.count
```

#### 6.13.1.3 data

```
read_multiport.data = np.zeros((frames, frame_rows, frame_cols), dtype = np.uint16)
```

#### 6.13.1.4 dtype

```
read_multiport.dtype
```

#### 6.13.1.5 f

```
read_multiport.f
```

#### 6.13.1.6 file\_name

```
str read_multiport.file_name = f'jungfrau_double_d{part}_f{frame}_{0}.raw'
```

#### 6.13.1.7 frame\_cols

```
int read_multiport.frame_cols = 1024
```

#### 6.13.1.8 frame\_rows

```
int read_multiport.frame_rows = 512
```

#### 6.13.1.9 frames

```
int read_multiport.frames = 1
```

#### 6.13.1.10 header

```
read_multiport.header = np.zeros((frames, parts), dtype = header_dt)
```

### 6.13.1.11 header\_dt

read\_multiport.header\_dt

#### Initial value:

```

00001 = np.dtype(
00002     [
00003         ("Frame Number", "u8"),
00004         ("SubFrame Number/ExpLength", "u4"),
00005         ("Packet Number", "u4"),
00006         ("Bunch ID", "u8"),
00007         ("Timestamp", "u8"),
00008         ("Module Id", "u2"),
00009         ("Row", "u2"),
00010         ("Column", "u2"),
00011         ("Reserved", "u2"),
00012         ("Debug", "u4"),
00013         ("Round Robin Number", "u2"),
00014         ("Detector Type", "u1"),
00015         ("Header Version", "u1"),
00016         ("Packets caught mask", "8u8")
00017     ]
00018 )

```

### 6.13.1.12 part\_cols

int read\_multiport.part\_cols = 1024

### 6.13.1.13 part\_rows

int read\_multiport.part\_rows = 256

### 6.13.1.14 parts

int read\_multiport.parts = 2

### 6.13.1.15 parts\_data

read\_multiport.parts\_data = np.zeros((frames,parts,part\_rows,part\_cols), dtype = np.uint16)

### 6.13.1.16 uint16

read\_multiport.uint16

## 6.14 simdjson Namespace Reference

## 6.15 write\_test\_files Namespace Reference

### Variables

- `arr` = `np.arange(10, dtype = np.int32)`
- `arr2` = `np.zeros((3,2,5), dtype = np.float64)`

## 6.15.1 Variable Documentation

### 6.15.1.1 arr

```
write_test_files.arr = np.arange(10, dtype = np.int32)
```

### 6.15.1.2 arr2

```
write_test_files.arr2 = np.zeros((3,2,5), dtype = np.float64)
```



## Chapter 7

# Data Structure Documentation

### 7.1 aare::CircularFifo< ItemType > Class Template Reference

```
#include <CircularFifo.hpp>
```

#### Public Types

- using [value\\_type](#) = ItemType

#### Public Member Functions

- [CircularFifo](#) ()
- [CircularFifo](#) (uint32\_t size)
- bool [next](#) ()
- [~CircularFifo](#) ()
- auto [numFilledSlots](#) () const noexcept
- auto [numFreeSlots](#) () const noexcept
- auto [isFull](#) () const noexcept
- ItemType [pop\\_free](#) ()
- bool [try\\_pop\\_free](#) (ItemType &v)
- ItemType [pop\\_value](#) (std::chrono::nanoseconds wait, std::atomic< bool > &stopped)
- ItemType [pop\\_value](#) ()
- ItemType \* [frontPtr](#) ()
- template<class... Args>  
void [push\\_value](#) (Args &&...recordArgs)
- template<class... Args>  
bool [try\\_push\\_value](#) (Args &&...recordArgs)
- template<class... Args>  
void [push\\_free](#) (Args &&...recordArgs)
- template<class... Args>  
bool [try\\_push\\_free](#) (Args &&...recordArgs)

#### Private Attributes

- uint32\_t [fifo\\_size](#)
- [folly::ProducerConsumerQueue](#)< ItemType > [free\\_slots](#)
- [folly::ProducerConsumerQueue](#)< ItemType > [filled\\_slots](#)

## 7.1.1 Member Typedef Documentation

### 7.1.1.1 value\_type

```
template<class ItemType >
using aare::CircularFifo< ItemType >::value_type = ItemType
```

## 7.1.2 Constructor & Destructor Documentation

### 7.1.2.1 CircularFifo() [1/2]

```
template<class ItemType >
aare::CircularFifo< ItemType >::CircularFifo ( ) [inline]
```

### 7.1.2.2 CircularFifo() [2/2]

```
template<class ItemType >
aare::CircularFifo< ItemType >::CircularFifo (
    uint32_t size ) [inline]
```

### 7.1.2.3 ~CircularFifo()

```
template<class ItemType >
aare::CircularFifo< ItemType >::~~CircularFifo ( ) [inline]
```

## 7.1.3 Member Function Documentation

### 7.1.3.1 frontPtr()

```
template<class ItemType >
ItemType * aare::CircularFifo< ItemType >::frontPtr ( ) [inline]
```

### 7.1.3.2 isFull()

```
template<class ItemType >
auto aare::CircularFifo< ItemType >::isFull ( ) const [inline], [noexcept]
```

### 7.1.3.3 next()

```
template<class ItemType >
bool aare::CircularFifo< ItemType >::next ( ) [inline]
```

#### 7.1.3.4 numFilledSlots()

```
template<class ItemType >
auto aare::CircularFifo< ItemType >::numFilledSlots ( ) const [inline], [noexcept]
```

#### 7.1.3.5 numFreeSlots()

```
template<class ItemType >
auto aare::CircularFifo< ItemType >::numFreeSlots ( ) const [inline], [noexcept]
```

#### 7.1.3.6 pop\_free()

```
template<class ItemType >
ItemType aare::CircularFifo< ItemType >::pop_free ( ) [inline]
```

#### 7.1.3.7 pop\_value() [1/2]

```
template<class ItemType >
ItemType aare::CircularFifo< ItemType >::pop_value ( ) [inline]
```

#### 7.1.3.8 pop\_value() [2/2]

```
template<class ItemType >
ItemType aare::CircularFifo< ItemType >::pop_value (
    std::chrono::nanoseconds wait,
    std::atomic< bool > & stopped ) [inline]
```

#### 7.1.3.9 push\_free()

```
template<class ItemType >
template<class... Args>
void aare::CircularFifo< ItemType >::push_free (
    Args &&... recordArgs ) [inline]
```

#### 7.1.3.10 push\_value()

```
template<class ItemType >
template<class... Args>
void aare::CircularFifo< ItemType >::push_value (
    Args &&... recordArgs ) [inline]
```

#### 7.1.3.11 try\_pop\_free()

```
template<class ItemType >
bool aare::CircularFifo< ItemType >::try_pop_free (
    ItemType & v ) [inline]
```

### 7.1.3.12 try\_push\_free()

```
template<class ItemType >
template<class... Args>
bool aare::CircularFifo< ItemType >::try_push_free (
    Args &&... recordArgs ) [inline]
```

### 7.1.3.13 try\_push\_value()

```
template<class ItemType >
template<class... Args>
bool aare::CircularFifo< ItemType >::try_push_value (
    Args &&... recordArgs ) [inline]
```

## 7.1.4 Field Documentation

### 7.1.4.1 fifo\_size

```
template<class ItemType >
uint32_t aare::CircularFifo< ItemType >::fifo_size [private]
```

### 7.1.4.2 filled\_slots

```
template<class ItemType >
 folly::ProducerConsumerQueue<ItemType> aare::CircularFifo< ItemType >::filled_slots [private]
```

### 7.1.4.3 free\_slots

```
template<class ItemType >
 folly::ProducerConsumerQueue<ItemType> aare::CircularFifo< ItemType >::free_slots [private]
```

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

- [core/include/aare/core/CircularFifo.hpp](#)

## 7.2 aare::ClusterFinder< T > Class Template Reference

```
#include <VariableSizeClusterFinder.hpp>
```

### Data Structures

- struct [Hit](#)

## Public Member Functions

- [ClusterFinder](#) (image\_shape shape, T threshold)
- [NDArray](#)< int, 2 > [labeled](#) ()
- void [set\\_noiseMap](#) ([NDView](#)< T, 2 > noise\_map)
- void [set\\_peripheralThresholdFactor](#) (int factor)
- void [find\\_clusters](#) ([NDView](#)< T, 2 > img)
- void [find\\_clusters\\_X](#) ([NDView](#)< T, 2 > img)
- void [rec\\_FillHit](#) (int clusterIndex, int i, int j)
- void [single\\_pass](#) ([NDView](#)< T, 2 > img)
- void [first\\_pass](#) ()
- void [second\\_pass](#) ()
- void [store\\_clusters](#) ()
- std::vector< [Hit](#) > [steal\\_hits](#) ()
- void [clear\\_hits](#) ()
- void [print\\_connections](#) ()
- size\_t [total\\_clusters](#) () const

## Private Member Functions

- int [check\\_neighbours](#) (int i, int j)
- void [add\\_link](#) (int from, int to)

## Private Attributes

- const std::array< ssize\_t, 2 > [shape\\_](#)
- [NDView](#)< T, 2 > [original\\_](#)
- [NDArray](#)< int, 2 > [labeled\\_](#)
- [NDArray](#)< int, 2 > [peripheral\\_labeled\\_](#)
- [NDArray](#)< bool, 2 > [binary\\_](#)
- T [threshold\\_](#)
- [NDView](#)< T, 2 > [noiseMap](#)
- bool [use\\_noise\\_map](#) = false
- int [peripheralThresholdFactor\\_](#) = 5
- int [current\\_label](#)
- const std::array< int, 4 > [di](#) {{0, -1, -1, -1}}
- const std::array< int, 4 > [dj](#) {{-1, -1, 0, 1}}
- const std::array< int, 8 > [di\\_](#) {{0, 0, -1, 1, -1, 1, -1, 1}}
- const std::array< int, 8 > [dj\\_](#) {{-1, 1, 0, 0, 1, -1, -1, 1}}
- std::map< int, int > [child](#)
- std::unordered\_map< int, [Hit](#) > [h\\_size](#)
- std::vector< [Hit](#) > [hits](#)

## 7.2.1 Constructor & Destructor Documentation

### 7.2.1.1 ClusterFinder()

```
template<typename T >
aare::ClusterFinder< T >::ClusterFinder (
    image_shape shape,
    T threshold ) [inline]
```

## 7.2.2 Member Function Documentation

### 7.2.2.1 add\_link()

```
template<typename T >
void aare::ClusterFinder< T >::add_link (
    int from,
    int to ) [inline], [private]
```

### 7.2.2.2 check\_neighbours()

```
template<typename T >
int aare::ClusterFinder< T >::check_neighbours (
    int i,
    int j ) [private]
```

### 7.2.2.3 clear\_hits()

```
template<typename T >
void aare::ClusterFinder< T >::clear_hits ( ) [inline]
```

### 7.2.2.4 find\_clusters()

```
template<typename T >
void aare::ClusterFinder< T >::find_clusters (
    NDView< T, 2 > img )
```

### 7.2.2.5 find\_clusters\_X()

```
template<typename T >
void aare::ClusterFinder< T >::find_clusters_X (
    NDView< T, 2 > img )
```

### 7.2.2.6 first\_pass()

```
template<typename T >
void aare::ClusterFinder< T >::first_pass
```

### 7.2.2.7 labeled()

```
template<typename T >
NDArray< int, 2 > aare::ClusterFinder< T >::labeled ( ) [inline]
```

### 7.2.2.8 print\_connections()

```
template<typename T >
void aare::ClusterFinder< T >::print_connections ( ) [inline]
```

### 7.2.2.9 rec\_FillHit()

```
template<typename T >
void aare::ClusterFinder< T >::rec_FillHit (
    int clusterIndex,
    int i,
    int j )
```

### 7.2.2.10 second\_pass()

```
template<typename T >
void aare::ClusterFinder< T >::second_pass
```

### 7.2.2.11 set\_noiseMap()

```
template<typename T >
void aare::ClusterFinder< T >::set_noiseMap (
    NDView< T, 2 > noise_map ) [inline]
```

### 7.2.2.12 set\_peripheralThresholdFactor()

```
template<typename T >
void aare::ClusterFinder< T >::set_peripheralThresholdFactor (
    int factor ) [inline]
```

### 7.2.2.13 single\_pass()

```
template<typename T >
void aare::ClusterFinder< T >::single_pass (
    NDView< T, 2 > img )
```

### 7.2.2.14 steal\_hits()

```
template<typename T >
std::vector< Hit > aare::ClusterFinder< T >::steal_hits ( ) [inline]
```

### 7.2.2.15 store\_clusters()

```
template<typename T >
void aare::ClusterFinder< T >::store_clusters
```

### 7.2.2.16 total\_clusters()

```
template<typename T >
size_t aare::ClusterFinder< T >::total_clusters ( ) const [inline]
```

## 7.2.3 Field Documentation

### 7.2.3.1 binary\_

```
template<typename T >
NDArray<bool, 2> aare::ClusterFinder< T >::binary_ [private]
```

### 7.2.3.2 child

```
template<typename T >
std::map<int, int> aare::ClusterFinder< T >::child [private]
```

### 7.2.3.3 current\_label

```
template<typename T >
int aare::ClusterFinder< T >::current_label [private]
```

### 7.2.3.4 di

```
template<typename T >
const std::array<int, 4> aare::ClusterFinder< T >::di {{0, -1, -1, -1}} [private]
```

### 7.2.3.5 di\_

```
template<typename T >
const std::array<int, 8> aare::ClusterFinder< T >::di_ {{0, 0, -1, 1, -1, 1, -1, 1}} [private]
```

### 7.2.3.6 dj

```
template<typename T >
const std::array<int, 4> aare::ClusterFinder< T >::dj {{-1, -1, 0, 1}} [private]
```

### 7.2.3.7 dj\_

```
template<typename T >
const std::array<int, 8> aare::ClusterFinder< T >::dj_ {{-1, 1, 0, 0, 1, -1, -1, 1}} [private]
```



**7.2.3.8 h\_size**

```
template<typename T >
std::unordered_map<int, Hit> aare::ClusterFinder< T >::h_size [private]
```

**7.2.3.9 hits**

```
template<typename T >
std::vector<Hit> aare::ClusterFinder< T >::hits [private]
```

**7.2.3.10 labeled\_**

```
template<typename T >
NDArray<int, 2> aare::ClusterFinder< T >::labeled_ [private]
```

**7.2.3.11 noiseMap**

```
template<typename T >
NDView<T, 2> aare::ClusterFinder< T >::noiseMap [private]
```

**7.2.3.12 original\_**

```
template<typename T >
NDView<T, 2> aare::ClusterFinder< T >::original_ [private]
```

**7.2.3.13 peripheral\_labeled\_**

```
template<typename T >
NDArray<int, 2> aare::ClusterFinder< T >::peripheral_labeled_ [private]
```

**7.2.3.14 peripheralThresholdFactor\_**

```
template<typename T >
int aare::ClusterFinder< T >::peripheralThresholdFactor_ = 5 [private]
```

**7.2.3.15 shape\_**

```
template<typename T >
const std::array<ssize_t, 2> aare::ClusterFinder< T >::shape_ [private]
```

**7.2.3.16 threshold\_**

```
template<typename T >
T aare::ClusterFinder< T >::threshold_ [private]
```

### 7.2.3.17 use\_noise\_map

```
template<typename T >
bool aare::ClusterFinder< T >::use_noise_map = false [private]
```

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

- [core/include/aare/core/VariableSizeClusterFinder.hpp](#)

## 7.3 aare::DType Class Reference

```
#include <DType.hpp>
```

### Public Types

- enum [TypeIndex](#) {  
[INT8](#) , [UINT8](#) , [INT16](#) , [UINT16](#) ,  
[INT32](#) , [UINT32](#) , [INT64](#) , [UINT64](#) ,  
[FLOAT](#) , [DOUBLE](#) , [ERROR](#) }

### Public Member Functions

- [uint8\\_t](#) [bitdepth](#) () const
- [DType](#) (const std::type\_info &t)
- [DType](#) (std::string\_view sv)
- [DType](#) ([DType::TypeIndex](#) ti)
- bool [operator==](#) (const [DType](#) &other) const noexcept
- bool [operator!=](#) (const [DType](#) &other) const noexcept
- bool [operator==](#) (const std::type\_info &t) const
- bool [operator!=](#) (const std::type\_info &t) const
- std::string [str](#) () const

### Private Attributes

- [TypeIndex](#) [m\\_type](#) {[TypeIndex::ERROR](#)}

## 7.3.1 Member Enumeration Documentation

### 7.3.1.1 TypeIndex

```
enum aare::DType::TypeIndex
```

#### Enumerator

INT8	
UINT8	
INT16	
UINT16	
INT32	
UINT32	
INT64	
UINT64	

## 7.3.2 Constructor & Destructor Documentation

### 7.3.2.1 DType() [1/3]

```
aare::DType::DType (
    const std::type_info & t ) [explicit]
```

### 7.3.2.2 DType() [2/3]

```
aare::DType::DType (
    std::string_view sv ) [explicit]
```

### 7.3.2.3 DType() [3/3]

```
aare::DType::DType (
    DType::TypeIndex ti )
```

## 7.3.3 Member Function Documentation

### 7.3.3.1 bitdepth()

```
uint8_t aare::DType::bitdepth ( ) const
```

### 7.3.3.2 operator!=( ) [1/2]

```
bool aare::DType::operator!= (
    const DType & other ) const [noexcept]
```

### 7.3.3.3 operator!=( ) [2/2]

```
bool aare::DType::operator!= (
    const std::type_info & t ) const
```

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

```
bool aare::DType::operator== (
    const DType & other ) const [noexcept]
```

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

```
bool aare::DType::operator== (
    const std::type_info & t ) const
```

### 7.3.3.6 str()

```
std::string aare::DType::str ( ) const
```

## 7.3.4 Field Documentation

### 7.3.4.1 m\_type

```
TypeIndex aare::DType::m_type {TypeIndex::ERROR} [private]
```

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

- [core/include/aare/core/DType.hpp](#)
- [core/src/DType.cpp](#)

## 7.4 aare::File Class Reference

RAII [File](#) class for reading and writing image files in various formats wrapper on a [FileInterface](#) to abstract the underlying file format.

```
#include <File.hpp>
```

### Public Member Functions

- [File](#) (std::filesystem::path fname, std::string mode, [FileConfig](#) cfg={})  
*Construct a new [File](#) object.*
- void [write](#) ([Frame](#) &frame)
- [Frame](#) [read](#) ()
- [Frame](#) [iread](#) (size\_t frame\_number)
- std::vector< [Frame](#) > [read](#) (size\_t n\_frames)
- void [read\\_into](#) (std::byte \*image\_buf)
- void [read\\_into](#) (std::byte \*image\_buf, size\_t n\_frames)
- size\_t [frame\\_number](#) (size\_t frame\_index)
- size\_t [bytes\\_per\\_frame](#) ()
- size\_t [pixels](#) ()
- void [seek](#) (size\_t frame\_number)
- size\_t [tell](#) () const
- size\_t [total\\_frames](#) () const
- ssize\_t [rows](#) () const
- ssize\_t [cols](#) () const
- ssize\_t [bitdepth](#) () const
- [File](#) ([File](#) &&other)  
*Move constructor.*
- [~File](#) ()  
*destructor: will only delete the [FileInterface](#) object*

### Private Attributes

- [FileInterface](#) \* [file\\_impl](#)

### 7.4.1 Detailed Description

RAII [File](#) class for reading and writing image files in various formats wrapper on a [FileInterface](#) to abstract the underlying file format.

#### Note

documentation for each function is in the [FileInterface](#) class

### 7.4.2 Constructor & Destructor Documentation

#### 7.4.2.1 [File\(\)](#) [1/2]

```
aare::File::File (
    std::filesystem::path fname,
    std::string mode,
    FileConfig cfg = {} )
```

Construct a new [File](#) object.

#### Parameters

<i>fname</i>	path to the file
<i>mode</i>	file mode (r, w, a)
<i>cfg</i>	file configuration

#### Exceptions

<i>std::runtime_error</i>	if the file cannot be opened
<i>std::invalid_argument</i>	if the file mode is not supported

#### 7.4.2.2 [File\(\)](#) [2/2]

```
aare::File::File (
    File && other )
```

Move constructor.

#### Parameters

<i>other</i>	<a href="#">File</a> object to move from
--------------	--

#### 7.4.2.3 [~File\(\)](#)

```
aare::File::~~File ( )
```

destructor: will only delete the [FileInterface](#) object

## 7.4.3 Member Function Documentation

### 7.4.3.1 bitdepth()

```
ssize_t aare::File::bitdepth ( ) const
```

### 7.4.3.2 bytes\_per\_frame()

```
size_t aare::File::bytes_per_frame ( )
```

### 7.4.3.3 cols()

```
ssize_t aare::File::cols ( ) const
```

### 7.4.3.4 frame\_number()

```
size_t aare::File::frame_number (
    size_t frame_index )
```

### 7.4.3.5 iread()

```
Frame aare::File::iread (
    size_t frame_number )
```

### 7.4.3.6 pixels()

```
size_t aare::File::pixels ( )
```

### 7.4.3.7 read() [1/2]

```
Frame aare::File::read ( )
```

### 7.4.3.8 read() [2/2]

```
std::vector< Frame > aare::File::read (
    size_t n_frames )
```

### 7.4.3.9 read\_into() [1/2]

```
void aare::File::read_into (
    std::byte * image_buf )
```

**7.4.3.10 read\_into()** [2/2]

```
void aare::File::read_into (
    std::byte * image_buf,
    size_t n_frames )
```

**7.4.3.11 rows()**

```
ssize_t aare::File::rows ( ) const
```

**7.4.3.12 seek()**

```
void aare::File::seek (
    size_t frame_number )
```

**7.4.3.13 tell()**

```
size_t aare::File::tell ( ) const
```

**7.4.3.14 total\_frames()**

```
size_t aare::File::total_frames ( ) const
```

**7.4.3.15 write()**

```
void aare::File::write (
    Frame & frame )
```

**7.4.4 Field Documentation****7.4.4.1 file\_impl**

```
FileInterface* aare::File::file_impl [private]
```

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

- [file\\_io/include/aare/file\\_io/File.hpp](#)
- [file\\_io/src/File.cpp](#)

**7.5 aare.File.File Class Reference****Public Member Functions**

- [\\_\\_init\\_\\_](#) (self, [path](#))
- Any [\\_\\_getattr\\_\\_](#) (self, str [\\_\\_name](#))

## Data Fields

- [path](#)

## Protected Attributes

- [\\_file](#)

### 7.5.1 Detailed Description

File class. uses proxy pattern to wrap around the pybinding class  
abstracts the python binding class that is requires type and detector information  
(e.g. `_FileHandler_Jungfrau_16`)

### 7.5.2 Constructor & Destructor Documentation

#### 7.5.2.1 `__init__()`

```
aare.File.File.__init__ (
    self,
    path )
```

opens the master file and checks the dynamic range and detector

### 7.5.3 Member Function Documentation

#### 7.5.3.1 `__getattr__()`

```
Any aare.File.File.__getattr__ (
    self,
    str __name )
```

Proxy pattern to call the methods of the `_file`

### 7.5.4 Field Documentation

#### 7.5.4.1 `_file`

```
aare.File.File._file [protected]
```

#### 7.5.4.2 `path`

```
aare.File.File.path
```

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

- `python/aare/File.py`



## 7.6 aare::FileConfig Struct Reference

[FileConfig](#) structure to store the configuration of a file dtype: data type of the file rows: number of rows in the file cols: number of columns in the file geometry: geometry of the file.

```
#include <FileInterface.hpp>
```

### Public Member Functions

- bool [operator==](#) (const [FileConfig](#) &other) const
- bool [operator!=](#) (const [FileConfig](#) &other) const

### Data Fields

- [aare::DType](#) dtype = [aare::DType](#)(typeid(uint16\_t))
- uint64\_t rows
- uint64\_t cols
- [xy](#) geometry {1, 1}

### 7.6.1 Detailed Description

[FileConfig](#) structure to store the configuration of a file dtype: data type of the file rows: number of rows in the file cols: number of columns in the file geometry: geometry of the file.

### 7.6.2 Member Function Documentation

#### 7.6.2.1 operator"!=()

```
bool aare::FileConfig::operator!= (
    const FileConfig & other ) const [inline]
```

#### 7.6.2.2 operator==(

```
bool aare::FileConfig::operator== (
    const FileConfig & other ) const [inline]
```

### 7.6.3 Field Documentation

#### 7.6.3.1 cols

```
uint64_t aare::FileConfig::cols
```

#### 7.6.3.2 dtype

```
aare::DType aare::FileConfig::dtype = aare::DType(typeid(uint16_t))
```

### 7.6.3.3 geometry

```
xy aare::FileConfig::geometry {1, 1}
```

### 7.6.3.4 rows

```
uint64_t aare::FileConfig::rows
```

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

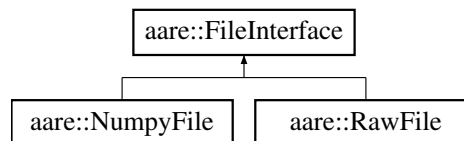
- [file\\_io/include/aare/file\\_io/FileInterface.hpp](#)

## 7.7 aare::FileInterface Class Reference

[FileInterface](#) class to define the interface for file operations.

```
#include <FileInterface.hpp>
```

Inheritance diagram for aare::FileInterface:



### Public Member Functions

- virtual void [write](#) ([Frame](#) &frame)=0  
*write a frame to the file*
- virtual [Frame](#) [read](#) ()=0  
*write a vector of frames to the file*
- virtual std::vector< [Frame](#) > [read](#) (size\_t n\_frames)=0  
*read n\_frames from the file at the current position*
- virtual void [read\\_into](#) (std::byte \*image\_buf)=0  
*read one frame from the file at the current position and store it in the provided buffer*
- virtual void [read\\_into](#) (std::byte \*image\_buf, size\_t n\_frames)=0  
*read n\_frames from the file at the current position and store them in the provided buffer*
- virtual size\_t [frame\\_number](#) (size\_t frame\_index)=0  
*get the frame number at the given frame index*
- virtual size\_t [bytes\\_per\\_frame](#) ()=0  
*get the size of one frame in bytes*
- virtual size\_t [pixels](#) ()=0  
*get the number of pixels in one frame*
- virtual void [seek](#) (size\_t [frame\\_number](#))=0  
*seek to the given frame number*
- virtual size\_t [tell](#) ()=0  
*get the current position of the file pointer*

- virtual `size_t total_frames ()` const =0  
*get the total number of frames in the file*
- virtual `ssize_t rows ()` const =0  
*get the number of rows in the file*
- virtual `ssize_t cols ()` const =0  
*get the number of columns in the file*
- virtual `ssize_t bitdepth ()` const =0  
*get the bitdepth of the file*
- `Frame` `iread (size_t frame_number)`  
*read one frame from the file at the given frame number*
- `std::vector< Frame >` `iread (size_t frame_number, size_t n_frames)`  
*read n\_frames from the file starting at the given frame number*
- virtual `~FileInterface ()`

## Data Fields

- `std::string` `m_mode`
- `std::filesystem::path` `m_fname`
- `std::filesystem::path` `m_base_path`
- `std::string` `m_base_name`
- `std::string` `m_ext`
- `int` `m_findex`
- `size_t` `m_total_frames {}`
- `size_t` `max_frames_per_file {}`
- `std::string` `version`
- `DetectorType` `m_type`
- `ssize_t` `m_rows {}`
- `ssize_t` `m_cols {}`
- `ssize_t` `m_bitdepth {}`
- `size_t` `current_frame {}`

## 7.7.1 Detailed Description

`FileInterface` class to define the interface for file operations.

### Note

parent class for `NumpyFile` and `RawFile`

all functions are pure virtual and must be implemented by the derived classes

## 7.7.2 Constructor & Destructor Documentation

### 7.7.2.1 ~FileInterface()

```
virtual aare::FileInterface::~~FileInterface ( ) [inline], [virtual]
```

## 7.7.3 Member Function Documentation

### 7.7.3.1 bitdepth()

```
virtual ssize_t aare::FileInterface::bitdepth ( ) const [pure virtual]
```

get the bitdepth of the file

#### Returns

bitdepth of the file

Implemented in [aare::NumpyFile](#), and [aare::RawFile](#).

### 7.7.3.2 bytes\_per\_frame()

```
virtual size_t aare::FileInterface::bytes_per_frame ( ) [pure virtual]
```

get the size of one frame in bytes

#### Returns

size of one frame

Implemented in [aare::NumpyFile](#), and [aare::RawFile](#).

### 7.7.3.3 cols()

```
virtual ssize_t aare::FileInterface::cols ( ) const [pure virtual]
```

get the number of columns in the file

#### Returns

number of columns in the file

Implemented in [aare::NumpyFile](#), and [aare::RawFile](#).

### 7.7.3.4 frame\_number()

```
virtual size_t aare::FileInterface::frame_number (
    size_t frame_index ) [pure virtual]
```

get the frame number at the given frame index

#### Parameters

<i>frame_index</i>	index of the frame
--------------------	--------------------

**Returns**

frame number

Implemented in [aare::NumpyFile](#), and [aare::RawFile](#).

**7.7.3.5 `iread()` [1/2]**

```
Frame aare::FileInterface::iread (
    size_t frame_number ) [inline]
```

read one frame from the file at the given frame number

**Parameters**

<i>frame_number</i>	frame number to read
---------------------	----------------------

**Returns**

frame

**7.7.3.6 `iread()` [2/2]**

```
std::vector< Frame > aare::FileInterface::iread (
    size_t frame_number,
    size_t n_frames ) [inline]
```

read *n\_frames* from the file starting at the given frame number

**Parameters**

<i>frame_number</i>	frame number to start reading from
<i>n_frames</i>	number of frames to read

**Returns**

vector of frames

**7.7.3.7 `pixels()`**

```
virtual size_t aare::FileInterface::pixels ( ) [pure virtual]
```

get the number of pixels in one frame

**Returns**

number of pixels in one frame

Implemented in [aare::NumpyFile](#), and [aare::RawFile](#).

**7.7.3.8 read()** [1/2]

```
virtual Frame aare::FileInterface::read ( ) [pure virtual]
```

write a vector of frames to the file

**Parameters**

<i>frames</i>	vector of frames to write
---------------	---------------------------

**Returns**

void

read one frame from the file at the current position

**Returns**

Frame

Implemented in [aare::NumpyFile](#), and [aare::RawFile](#).

**7.7.3.9 read()** [2/2]

```
virtual std::vector< Frame > aare::FileInterface::read (
    size_t n_frames ) [pure virtual]
```

read n\_frames from the file at the current position

**Parameters**

<i>n_frames</i>	number of frames to read
-----------------	--------------------------

**Returns**

vector of frames

Implemented in [aare::NumpyFile](#), and [aare::RawFile](#).

**7.7.3.10 read\_into()** [1/2]

```
virtual void aare::FileInterface::read_into (
    std::byte * image_buf ) [pure virtual]
```

read one frame from the file at the current position and store it in the provided buffer

**Parameters**

<i>image_buf</i>	buffer to store the frame
------------------	---------------------------

**Returns**

void

Implemented in [aare::NumpyFile](#), and [aare::RawFile](#).

**7.7.3.11 read\_into() [2/2]**

```
virtual void aare::FileInterface::read_into (
    std::byte * image_buf,
    size_t n_frames ) [pure virtual]
```

read `n_frames` from the file at the current position and store them in the provided buffer

**Parameters**

<i>image_buf</i>	buffer to store the frames
<i>n_frames</i>	number of frames to read

**Returns**

void

Implemented in [aare::NumpyFile](#), and [aare::RawFile](#).

**7.7.3.12 rows()**

```
virtual ssize_t aare::FileInterface::rows ( ) const [pure virtual]
```

get the number of rows in the file

**Returns**

number of rows in the file

Implemented in [aare::NumpyFile](#), and [aare::RawFile](#).

**7.7.3.13 seek()**

```
virtual void aare::FileInterface::seek (
    size_t frame_number ) [pure virtual]
```

seek to the given frame number

**Parameters**

<i>frame_number</i>	frame number to seek to
---------------------	-------------------------

**Returns**

void

Implemented in [aare::NumpyFile](#), and [aare::RawFile](#).

**7.7.3.14 tell()**

```
virtual size_t aare::FileInterface::tell ( ) [pure virtual]
```

get the current position of the file pointer

**Returns**

current position of the file pointer

Implemented in [aare::NumpyFile](#), and [aare::RawFile](#).

**7.7.3.15 total\_frames()**

```
virtual size_t aare::FileInterface::total_frames ( ) const [pure virtual]
```

get the total number of frames in the file

**Returns**

total number of frames in the file

Implemented in [aare::NumpyFile](#), and [aare::RawFile](#).

**7.7.3.16 write()**

```
virtual void aare::FileInterface::write (
    Frame & frame ) [pure virtual]
```

write a frame to the file

**Parameters**

<i>frame</i>	frame to write
--------------	----------------

**Returns**

void

**Exceptions**

<i>std::runtime_error</i>	if the function is not implemented
---------------------------	------------------------------------



Implemented in [aare::NumpyFile](#), and [aare::RawFile](#).

## 7.7.4 Field Documentation

### 7.7.4.1 current\_frame

```
size_t aare::FileInterface::current_frame {}
```

### 7.7.4.2 m\_base\_name

```
std::string aare::FileInterface::m_base_name
```

### 7.7.4.3 m\_base\_path

```
std::filesystem::path aare::FileInterface::m_base_path
```

### 7.7.4.4 m\_bitdepth

```
ssize_t aare::FileInterface::m_bitdepth {}
```

### 7.7.4.5 m\_cols

```
ssize_t aare::FileInterface::m_cols {}
```

### 7.7.4.6 m\_ext

```
std::string aare::FileInterface::m_ext
```

### 7.7.4.7 m\_findex

```
int aare::FileInterface::m_findex
```

### 7.7.4.8 m\_fname

```
std::filesystem::path aare::FileInterface::m_fname
```

### 7.7.4.9 m\_mode

```
std::string aare::FileInterface::m_mode
```

#### 7.7.4.10 m\_rows

```
ssize_t aare::FileInterface::m_rows {}
```

#### 7.7.4.11 m\_total\_frames

```
size_t aare::FileInterface::m_total_frames {}
```

#### 7.7.4.12 m\_type

```
DetectorType aare::FileInterface::m_type
```

#### 7.7.4.13 max\_frames\_per\_file

```
size_t aare::FileInterface::max_frames_per_file {}
```

#### 7.7.4.14 version

```
std::string aare::FileInterface::version
```

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

- file\_io/include/aare/file\_io/[FileInterface.hpp](#)

## 7.8 aare::Frame Class Reference

```
#include <Frame.hpp>
```

### Public Member Functions

- [Frame](#) (ssize\_t rows, ssize\_t cols, ssize\_t m\_bitdepth)
- [Frame](#) (std::byte \*fp, ssize\_t rows, ssize\_t cols, ssize\_t m\_bitdepth)
- std::byte \* [get](#) (int row, int col)
- template<typename T >  
void [set](#) (int row, int col, T data)
- ssize\_t [rows](#) () const
- ssize\_t [cols](#) () const
- ssize\_t [bitdepth](#) () const
- ssize\_t [size](#) () const
- std::byte \* [data](#) () const
- [Frame](#) & [operator=](#) ([Frame](#) &other)
- [Frame](#) ([Frame](#) &&other)
- [Frame](#) (const [Frame](#) &other)
- template<typename T >  
[NDView](#)< T > [view](#) ()
- template<typename T >  
[NDArray](#)< T > [image](#) ()
- [~Frame](#) ()

## Private Attributes

- `ssize_t m_rows`
- `ssize_t m_cols`
- `ssize_t m_bitdepth`
- `std::byte * m_data`

## 7.8.1 Constructor & Destructor Documentation

### 7.8.1.1 `Frame()` [1/4]

```
aare::Frame::Frame (
    ssize_t rows,
    ssize_t cols,
    ssize_t m_bitdepth )
```

### 7.8.1.2 `Frame()` [2/4]

```
aare::Frame::Frame (
    std::byte * fp,
    ssize_t rows,
    ssize_t cols,
    ssize_t m_bitdepth )
```

### 7.8.1.3 `Frame()` [3/4]

```
aare::Frame::Frame (
    Frame && other ) [inline]
```

### 7.8.1.4 `Frame()` [4/4]

```
aare::Frame::Frame (
    const Frame & other ) [inline]
```

### 7.8.1.5 `~Frame()`

```
aare::Frame::~~Frame ( ) [inline]
```

## 7.8.2 Member Function Documentation

### 7.8.2.1 `bitdepth()`

```
ssize_t aare::Frame::bitdepth ( ) const [inline]
```

#### 7.8.2.2 cols()

```
ssize_t aare::Frame::cols ( ) const [inline]
```

#### 7.8.2.3 data()

```
std::byte * aare::Frame::data ( ) const [inline]
```

#### 7.8.2.4 get()

```
std::byte * aare::Frame::get (
    int row,
    int col )
```

#### 7.8.2.5 image()

```
template<typename T >
NDArray< T > aare::Frame::image ( ) [inline]
```

#### 7.8.2.6 operator=()

```
Frame & aare::Frame::operator= (
    Frame & other ) [inline]
```

#### 7.8.2.7 rows()

```
ssize_t aare::Frame::rows ( ) const [inline]
```

#### 7.8.2.8 set()

```
template<typename T >
void aare::Frame::set (
    int row,
    int col,
    T data ) [inline]
```

#### 7.8.2.9 size()

```
ssize_t aare::Frame::size ( ) const [inline]
```

#### 7.8.2.10 view()

```
template<typename T >
NDView< T > aare::Frame::view ( ) [inline]
```

### 7.8.3 Field Documentation

#### 7.8.3.1 m\_bitdepth

```
ssize_t aare::Frame::m_bitdepth [private]
```

#### 7.8.3.2 m\_cols

```
ssize_t aare::Frame::m_cols [private]
```

#### 7.8.3.3 m\_data

```
std::byte* aare::Frame::m_data [private]
```

#### 7.8.3.4 m\_rows

```
ssize_t aare::Frame::m_rows [private]
```

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

- [core/include/aare/core/Frame.hpp](#)
- [core/src/Frame.cpp](#)

## 7.9 aare.Frame.Frame Class Reference

### Public Member Functions

- [\\_\\_init\\_\\_](#) (self, frameImpl)
- Any [\\_\\_getattr\\_\\_](#) (self, str \_\_name)

### Protected Attributes

- [\\_frameImpl](#)

### 7.9.1 Detailed Description

Frame class. uses proxy pattern to wrap around the pybinding class  
the intention behind it is to only use one class for frames in python (not Frame\_8, Frame\_16, etc)

### 7.9.2 Constructor & Destructor Documentation

#### 7.9.2.1 \_\_init\_\_()

```
aare.Frame.Frame.__init__ (
    self,
    frameImpl )
```

### 7.9.3 Member Function Documentation

#### 7.9.3.1 `__getattr__()`

```
Any aare.Frame.Frame.__getattr__ (
    self,
    str __name )
```

Proxy pattern to call the methods of the frameImpl

### 7.9.4 Field Documentation

#### 7.9.4.1 `_frameImpl`

```
aare.Frame.Frame._frameImpl [protected]
```

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

- [python/aare/Frame.py](#)

## 7.10 `aare::ClusterFinder< T >::Hit` Struct Reference

```
#include <VariableSizeClusterFinder.hpp>
```

### Data Fields

- `int16_t size` {}
- `int16_t row` {}
- `int16_t col` {}
- `uint16_t reserved` {}
- `T energy` {}
- `T max` {}
- `int16_t rows` [`MAX_CLUSTER_SIZE`] = {0}
- `int16_t cols` [`MAX_CLUSTER_SIZE`] = {0}
- `double enes` [`MAX_CLUSTER_SIZE`] = {0}

### 7.10.1 Field Documentation

#### 7.10.1.1 `col`

```
template<typename T >
int16_t aare::ClusterFinder< T >::Hit::col {}
```

### 7.10.1.2 cols

```
template<typename T >
int16_t aare::ClusterFinder< T >::Hit::cols[MAX_CLUSTER_SIZE] = {0}
```

### 7.10.1.3 energy

```
template<typename T >
T aare::ClusterFinder< T >::Hit::energy {}
```

### 7.10.1.4 enes

```
template<typename T >
double aare::ClusterFinder< T >::Hit::enes[MAX_CLUSTER_SIZE] = {0}
```

### 7.10.1.5 max

```
template<typename T >
T aare::ClusterFinder< T >::Hit::max {}
```

### 7.10.1.6 reserved

```
template<typename T >
uint16_t aare::ClusterFinder< T >::Hit::reserved {}
```

### 7.10.1.7 row

```
template<typename T >
int16_t aare::ClusterFinder< T >::Hit::row {}
```

### 7.10.1.8 rows

```
template<typename T >
int16_t aare::ClusterFinder< T >::Hit::rows[MAX_CLUSTER_SIZE] = {0}
```

### 7.10.1.9 size

```
template<typename T >
int16_t aare::ClusterFinder< T >::Hit::size {}
```

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

- [core/include/aare/core/VariableSizeClusterFinder.hpp](#)

## 7.11 aare::logger::Logger Class Reference

```
#include <logger.hpp>
```

### Public Member Functions

- void [set\\_output\\_file](#) (std::string filename)
- void [set\\_streams](#) (std::streambuf \*out, std::streambuf \*err)
- void [set\\_streams](#) (std::streambuf \*out)
- void [set\\_verbosity](#) (LOGGING\_LEVEL level)
- [Logger](#) ()
- [~Logger](#) ()
- template<LOGGING\_LEVEL level, typename... Strings>  
void [log](#) (const Strings... s)
- template<typename... Strings>  
void [debug](#) (const Strings... s)
- template<typename... Strings>  
void [info](#) (const Strings... s)
- template<typename... Strings>  
void [warn](#) (const Strings... s)
- template<typename... Strings>  
void [error](#) (const Strings... s)

### Private Member Functions

- template<LOGGING\_LEVEL level>  
void [log\\_](#) ()
- template<LOGGING\_LEVEL level, typename First, typename... Strings>  
void [log\\_](#) (First arg, const Strings... s)

### Private Attributes

- std::streambuf \* [standard\\_buf](#) = std::cout.rdbuf()
- std::streambuf \* [error\\_buf](#) = std::cerr.rdbuf()
- std::ostream \* [standard\\_output](#)
- std::ostream \* [error\\_output](#)
- LOGGING\_LEVEL VERBOSITY\_LEVEL = LOGGING\_LEVEL::INFO
- std::ofstream [out\\_file](#)

### 7.11.1 Constructor & Destructor Documentation

#### 7.11.1.1 [Logger\(\)](#)

```
aare::logger::Logger::Logger ( ) [inline]
```

#### 7.11.1.2 [~Logger\(\)](#)

```
aare::logger::Logger::~Logger ( ) [inline]
```



## 7.11.2 Member Function Documentation

### 7.11.2.1 debug()

```
template<typename... Strings>
void aare::logger::Logger::debug (
    const Strings... s ) [inline]
```

### 7.11.2.2 error()

```
template<typename... Strings>
void aare::logger::Logger::error (
    const Strings... s ) [inline]
```

### 7.11.2.3 info()

```
template<typename... Strings>
void aare::logger::Logger::info (
    const Strings... s ) [inline]
```

### 7.11.2.4 log()

```
template<LOGGING_LEVEL level, typename... Strings>
void aare::logger::Logger::log (
    const Strings... s ) [inline]
```

### 7.11.2.5 log\_() [1/2]

```
template<LOGGING_LEVEL level>
void aare::logger::Logger::log_ ( ) [inline], [private]
```

### 7.11.2.6 log\_() [2/2]

```
template<LOGGING_LEVEL level, typename First , typename... Strings>
void aare::logger::Logger::log_ (
    First arg,
    const Strings... s ) [inline], [private]
```

### 7.11.2.7 set\_output\_file()

```
void aare::logger::Logger::set_output_file (
    std::string filename ) [inline]
```

#### 7.11.2.8 set\_streams() [1/2]

```
void aare::logger::Logger::set_streams (
    std::streambuf * out ) [inline]
```

#### 7.11.2.9 set\_streams() [2/2]

```
void aare::logger::Logger::set_streams (
    std::streambuf * out,
    std::streambuf * err ) [inline]
```

#### 7.11.2.10 set\_verbosity()

```
void aare::logger::Logger::set_verbosity (
    LOGGING_LEVEL level ) [inline]
```

#### 7.11.2.11 warn()

```
template<typename... Strings>
void aare::logger::Logger::warn (
    const Strings... s ) [inline]
```

### 7.11.3 Field Documentation

#### 7.11.3.1 error\_buf

```
std::streambuf* aare::logger::Logger::error_buf = std::cerr.rdbuf() [private]
```

#### 7.11.3.2 error\_output

```
std::ostream* aare::logger::Logger::error_output [private]
```

#### 7.11.3.3 out\_file

```
std::ofstream aare::logger::Logger::out_file [private]
```

#### 7.11.3.4 standard\_buf

```
std::streambuf* aare::logger::Logger::standard_buf = std::cout.rdbuf() [private]
```

#### 7.11.3.5 standard\_output

```
std::ostream* aare::logger::Logger::standard_output [private]
```

### 7.11.3.6 VERBOSITY\_LEVEL

```
LOGGING_LEVEL aare::logger::Logger::VERBOSITY_LEVEL = LOGGING_LEVEL::INFO [private]
```

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

- [utils/include/aare/utils/logger.hpp](#)

## 7.12 MoveOnlyInt Struct Reference

### Public Member Functions

- [MoveOnlyInt](#) ()=default
- [MoveOnlyInt](#) (int i)
- [MoveOnlyInt](#) (const [MoveOnlyInt](#) &)=delete
- [MoveOnlyInt](#) & [operator=](#) (const [MoveOnlyInt](#) &)=delete
- [MoveOnlyInt](#) ([MoveOnlyInt](#) &&other)
- [MoveOnlyInt](#) & [operator=](#) ([MoveOnlyInt](#) &&other)
- bool [operator==](#) (int other) const

### Data Fields

- int [value](#) {}

### 7.12.1 Constructor & Destructor Documentation

#### 7.12.1.1 MoveOnlyInt() [1/4]

```
MoveOnlyInt::MoveOnlyInt ( ) [default]
```

#### 7.12.1.2 MoveOnlyInt() [2/4]

```
MoveOnlyInt::MoveOnlyInt (
    int i ) [inline]
```

#### 7.12.1.3 MoveOnlyInt() [3/4]

```
MoveOnlyInt::MoveOnlyInt (
    const MoveOnlyInt & ) [delete]
```

#### 7.12.1.4 MoveOnlyInt() [4/4]

```
MoveOnlyInt::MoveOnlyInt (
    MoveOnlyInt && other ) [inline]
```

## 7.12.2 Member Function Documentation

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

```
MoveOnlyInt & MoveOnlyInt::operator= (
    const MoveOnlyInt & ) [delete]
```

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

```
MoveOnlyInt & MoveOnlyInt::operator= (
    MoveOnlyInt && other ) [inline]
```

### 7.12.2.3 operator==( )

```
bool MoveOnlyInt::operator== (
    int other ) const [inline]
```

## 7.12.3 Field Documentation

### 7.12.3.1 value

```
int MoveOnlyInt::value {}
```

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

- core/test/[CircularFifo.test.cpp](#)

## 7.13 aare::NDArray< T, Ndim > Class Template Reference

```
#include <NDArray.hpp>
```

### Public Types

- using [value\\_type](#) = T

## Public Member Functions

- [NDArray](#) ()
- [NDArray](#) (std::array< ssize\_t, Ndim > [shape](#))
- [NDArray](#) (std::array< ssize\_t, Ndim > [shape](#), T [value](#))
- [NDArray](#) ([NDView](#)< T, Ndim > [span](#))
- [NDArray](#) ([NDArray](#) &&other)
- [NDArray](#) (const [NDArray](#) &other)
- [~NDArray](#) ()
- auto [begin](#) ()
- auto [end](#) ()
- [NDArray](#) & [operator=](#) ([NDArray](#) &&other)
- [NDArray](#) & [operator=](#) (const [NDArray](#) &other)
- [NDArray](#) [operator+](#) (const [NDArray](#) &other)
- [NDArray](#) & [operator+=](#) (const [NDArray](#) &other)
- [NDArray](#) [operator-](#) (const [NDArray](#) &other)
- [NDArray](#) & [operator-=](#) (const [NDArray](#) &other)
- [NDArray](#) [operator\\*](#) (const [NDArray](#) &other)
- [NDArray](#) & [operator\\*=](#) (const [NDArray](#) &other)
- [NDArray](#) [operator/](#) (const [NDArray](#) &other)
- template<typename V >  
  [NDArray](#) & [operator/=](#) (const [NDArray](#)< V, Ndim > &other)
- [NDArray](#)< bool, Ndim > [operator>](#) (const [NDArray](#) &other)
- bool [operator==](#) (const [NDArray](#) &other) const
- bool [operator!=](#) (const [NDArray](#) &other) const
- [NDArray](#) & [operator=](#) (const T &)
- [NDArray](#) & [operator+=](#) (const T &)
- [NDArray](#) [operator+](#) (const T &)
- [NDArray](#) & [operator-=](#) (const T &)
- [NDArray](#) [operator-](#) (const T &)
- [NDArray](#) & [operator\\*=](#) (const T &)
- [NDArray](#) [operator\\*](#) (const T &)
- [NDArray](#) & [operator/=](#) (const T &)
- [NDArray](#) [operator/](#) (const T &)
- [NDArray](#) & [operator&=](#) (const T &)
- void [sqrt](#) ()
- [NDArray](#) & [operator++](#) ()
- template<typename... lx>  
  std::enable\_if< sizeof...(lx)==Ndim, T & >::type [operator\(\)](#) (lx... index)
- template<typename... lx>  
  std::enable\_if< sizeof...(lx)==Ndim, T & >::type [operator\(\)](#) (lx... index) const
- template<typename... lx>  
  std::enable\_if< sizeof...(lx)==Ndim, T >::type [value](#) (lx... index)
- T & [operator\(\)](#) (int i)
- const T & [operator\(\)](#) (int i) const
- T \* [data](#) ()
- std::byte \* [buffer](#) ()
- ssize\_t [size](#) () const
- size\_t [total\\_bytes](#) () const
- std::array< ssize\_t, Ndim > [shape](#) () const noexcept
- ssize\_t [shape](#) (ssize\_t i) const noexcept
- std::array< ssize\_t, Ndim > [strides](#) () const noexcept
- std::array< ssize\_t, Ndim > [byte\\_strides](#) () const noexcept
- [NDView](#)< T, Ndim > [span](#) () const
- void [Print](#) ()
- void [Print\\_all](#) ()
- void [Print\\_some](#) ()
- void [reset](#) ()

## Private Attributes

- `std::array< ssize_t, Ndim >` [shape\\_](#)
- `std::array< ssize_t, Ndim >` [strides\\_](#)
- `ssize_t` [size\\_](#)
- `T *` [data\\_](#)

## 7.13.1 Member Typedef Documentation

### 7.13.1.1 `value_type`

```
template<typename T , ssize_t Ndim = 2>
using aare::NDArray< T, Ndim >::value_type = T
```

## 7.13.2 Constructor & Destructor Documentation

### 7.13.2.1 `NDArray()` [1/6]

```
template<typename T , ssize_t Ndim = 2>
aare::NDArray< T, Ndim >::NDArray ( ) [inline]
```

### 7.13.2.2 `NDArray()` [2/6]

```
template<typename T , ssize_t Ndim = 2>
aare::NDArray< T, Ndim >::NDArray (
    std::array< ssize_t, Ndim > shape ) [inline], [explicit]
```

### 7.13.2.3 `NDArray()` [3/6]

```
template<typename T , ssize_t Ndim = 2>
aare::NDArray< T, Ndim >::NDArray (
    std::array< ssize_t, Ndim > shape,
    T value ) [inline]
```

### 7.13.2.4 `NDArray()` [4/6]

```
template<typename T , ssize_t Ndim = 2>
aare::NDArray< T, Ndim >::NDArray (
    NDView< T, Ndim > span ) [inline]
```

### 7.13.2.5 `NDArray()` [5/6]

```
template<typename T , ssize_t Ndim = 2>
aare::NDArray< T, Ndim >::NDArray (
    NDArray< T, Ndim > && other ) [inline]
```

### 7.13.2.6 NDArray() [6/6]

```
template<typename T , ssize_t Ndim = 2>
aare::NDArray< T, Ndim >::NDArray (
    const NDArray< T, Ndim > & other ) [inline]
```

### 7.13.2.7 ~NDArray()

```
template<typename T , ssize_t Ndim = 2>
aare::NDArray< T, Ndim >::~~NDArray ( ) [inline]
```

## 7.13.3 Member Function Documentation

### 7.13.3.1 begin()

```
template<typename T , ssize_t Ndim = 2>
auto aare::NDArray< T, Ndim >::begin ( ) [inline]
```

### 7.13.3.2 buffer()

```
template<typename T , ssize_t Ndim = 2>
std::byte * aare::NDArray< T, Ndim >::buffer ( ) [inline]
```

### 7.13.3.3 byte\_strides()

```
template<typename T , ssize_t Ndim = 2>
std::array< ssize_t, Ndim > aare::NDArray< T, Ndim >::byte_strides ( ) const [inline], [noexcept]
```

### 7.13.3.4 data()

```
template<typename T , ssize_t Ndim = 2>
T * aare::NDArray< T, Ndim >::data ( ) [inline]
```

### 7.13.3.5 end()

```
template<typename T , ssize_t Ndim = 2>
auto aare::NDArray< T, Ndim >::end ( ) [inline]
```

### 7.13.3.6 operator"!="()

```
template<typename T , ssize_t Ndim>
bool aare::NDArray< T, Ndim >::operator!= (
    const NDArray< T, Ndim > & other ) const
```

**7.13.3.7 operator&=()**

```
template<typename T , ssize_t Ndim>
NDArray< T, Ndim > & aare::NDArray< T, Ndim >::operator&= (
    const T & mask )
```

**7.13.3.8 operator>() [1/4]**

```
template<typename T , ssize_t Ndim = 2>
T & aare::NDArray< T, Ndim >::operator() (
    int i ) [inline]
```

**7.13.3.9 operator>() [2/4]**

```
template<typename T , ssize_t Ndim = 2>
const T & aare::NDArray< T, Ndim >::operator() (
    int i ) const [inline]
```

**7.13.3.10 operator>() [3/4]**

```
template<typename T , ssize_t Ndim = 2>
template<typename... Ix>
std::enable_if< sizeof...(Ix)==Ndim, T & >::type aare::NDArray< T, Ndim >::operator() (
    Ix... index ) [inline]
```

**7.13.3.11 operator>() [4/4]**

```
template<typename T , ssize_t Ndim = 2>
template<typename... Ix>
std::enable_if< sizeof...(Ix)==Ndim, T & >::type aare::NDArray< T, Ndim >::operator() (
    Ix... index ) const [inline]
```

**7.13.3.12 operator\*() [1/2]**

```
template<typename T , ssize_t Ndim>
NDArray< T, Ndim > aare::NDArray< T, Ndim >::operator* (
    const NDArray< T, Ndim > & other )
```

**7.13.3.13 operator\*() [2/2]**

```
template<typename T , ssize_t Ndim>
NDArray< T, Ndim > aare::NDArray< T, Ndim >::operator* (
    const T & value )
```



**7.13.3.14 operator\*=( ) [1/2]**

```
template<typename T , ssize_t Ndim>
NDArray< T, Ndim > & aare::NDArray< T, Ndim >::operator*= (
    const NDArray< T, Ndim > & other )
```

**7.13.3.15 operator\*=( ) [2/2]**

```
template<typename T , ssize_t Ndim>
NDArray< T, Ndim > & aare::NDArray< T, Ndim >::operator*= (
    const T & value )
```

**7.13.3.16 operator+( ) [1/2]**

```
template<typename T , ssize_t Ndim>
NDArray< T, Ndim > aare::NDArray< T, Ndim >::operator+ (
    const NDArray< T, Ndim > & other )
```

**7.13.3.17 operator+( ) [2/2]**

```
template<typename T , ssize_t Ndim>
NDArray< T, Ndim > aare::NDArray< T, Ndim >::operator+ (
    const T & value )
```

**7.13.3.18 operator++( )**

```
template<typename T , ssize_t Ndim>
NDArray< T, Ndim > & aare::NDArray< T, Ndim >::operator++
```

**7.13.3.19 operator+=( ) [1/2]**

```
template<typename T , ssize_t Ndim>
NDArray< T, Ndim > & aare::NDArray< T, Ndim >::operator+= (
    const NDArray< T, Ndim > & other )
```

**7.13.3.20 operator+=( ) [2/2]**

```
template<typename T , ssize_t Ndim>
NDArray< T, Ndim > & aare::NDArray< T, Ndim >::operator+= (
    const T & value )
```

**7.13.3.21 operator-( ) [1/2]**

```
template<typename T , ssize_t Ndim>
NDArray< T, Ndim > aare::NDArray< T, Ndim >::operator- (
    const NDArray< T, Ndim > & other )
```

**7.13.3.22 operator-() [2/2]**

```
template<typename T , ssize_t Ndim>
NDArray< T, Ndim > aare::NDArray< T, Ndim >::operator- (
    const T & value )
```

**7.13.3.23 operator==(1/2]**

```
template<typename T , ssize_t Ndim>
NDArray< T, Ndim > & aare::NDArray< T, Ndim >::operator== (
    const NDArray< T, Ndim > & other )
```

**7.13.3.24 operator==(2/2]**

```
template<typename T , ssize_t Ndim>
NDArray< T, Ndim > & aare::NDArray< T, Ndim >::operator== (
    const T & value )
```

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

```
template<typename T , ssize_t Ndim>
NDArray< T, Ndim > aare::NDArray< T, Ndim >::operator/ (
    const NDArray< T, Ndim > & other )
```

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

```
template<typename T , ssize_t Ndim>
NDArray< T, Ndim > aare::NDArray< T, Ndim >::operator/ (
    const T & value )
```

**7.13.3.27 operator/=(1/2]**

```
template<typename T , ssize_t Ndim = 2>
template<typename V >
NDArray & aare::NDArray< T, Ndim >::operator/= (
    const NDArray< V, Ndim > & other ) [inline]
```

**7.13.3.28 operator/=(2/2]**

```
template<typename T , ssize_t Ndim>
NDArray< T, Ndim > & aare::NDArray< T, Ndim >::operator/= (
    const T & value )
```

**7.13.3.29 operator=()** [1/3]

```
template<typename T , ssize_t Ndim>
NDArray< T, Ndim > & aare::NDArray< T, Ndim >::operator= (
    const NDArray< T, Ndim > & other )
```

**7.13.3.30 operator=()** [2/3]

```
template<typename T , ssize_t Ndim>
NDArray< T, Ndim > & aare::NDArray< T, Ndim >::operator= (
    const T & value )
```

**7.13.3.31 operator=()** [3/3]

```
template<typename T , ssize_t Ndim>
NDArray< T, Ndim > & aare::NDArray< T, Ndim >::operator= (
    NDArray< T, Ndim > && other )
```

**7.13.3.32 operator==()**

```
template<typename T , ssize_t Ndim>
bool aare::NDArray< T, Ndim >::operator== (
    const NDArray< T, Ndim > & other ) const
```

**7.13.3.33 operator>()**

```
template<typename T , ssize_t Ndim>
NDArray< bool, Ndim > aare::NDArray< T, Ndim >::operator> (
    const NDArray< T, Ndim > & other )
```

**7.13.3.34 Print()**

```
template<typename T , ssize_t Ndim>
void aare::NDArray< T, Ndim >::Print
```

**7.13.3.35 Print\_all()**

```
template<typename T , ssize_t Ndim>
void aare::NDArray< T, Ndim >::Print_all
```

**7.13.3.36 Print\_some()**

```
template<typename T , ssize_t Ndim>
void aare::NDArray< T, Ndim >::Print_some
```

**7.13.3.37 reset()**

```
template<typename T , ssize_t Ndim = 2>
void aare::NDArray< T, Ndim >::reset ( ) [inline]
```

**7.13.3.38 shape() [1/2]**

```
template<typename T , ssize_t Ndim = 2>
std::array< ssize_t, Ndim > aare::NDArray< T, Ndim >::shape ( ) const [inline], [noexcept]
```

**7.13.3.39 shape() [2/2]**

```
template<typename T , ssize_t Ndim = 2>
ssize_t aare::NDArray< T, Ndim >::shape (
    ssize_t i ) const [inline], [noexcept]
```

**7.13.3.40 size()**

```
template<typename T , ssize_t Ndim = 2>
ssize_t aare::NDArray< T, Ndim >::size ( ) const [inline]
```

**7.13.3.41 span()**

```
template<typename T , ssize_t Ndim = 2>
NDView< T, Ndim > aare::NDArray< T, Ndim >::span ( ) const [inline]
```

**7.13.3.42 sqrt()**

```
template<typename T , ssize_t Ndim = 2>
void aare::NDArray< T, Ndim >::sqrt ( ) [inline]
```

**7.13.3.43 strides()**

```
template<typename T , ssize_t Ndim = 2>
std::array< ssize_t, Ndim > aare::NDArray< T, Ndim >::strides ( ) const [inline], [noexcept]
```

**7.13.3.44 total\_bytes()**

```
template<typename T , ssize_t Ndim = 2>
size_t aare::NDArray< T, Ndim >::total_bytes ( ) const [inline]
```

### 7.13.3.45 value()

```
template<typename T , ssize_t Ndim = 2>
template<typename... Ix>
std::enable_if< sizeof...(Ix)==Ndim, T >::type aare::NDArray< T, Ndim >::value (
    Ix... index ) [inline]
```

## 7.13.4 Field Documentation

### 7.13.4.1 data\_

```
template<typename T , ssize_t Ndim = 2>
T* aare::NDArray< T, Ndim >::data_ [private]
```

### 7.13.4.2 shape\_

```
template<typename T , ssize_t Ndim = 2>
std::array<ssize_t, Ndim> aare::NDArray< T, Ndim >::shape_ [private]
```

### 7.13.4.3 size\_

```
template<typename T , ssize_t Ndim = 2>
ssize_t aare::NDArray< T, Ndim >::size_ [private]
```

### 7.13.4.4 strides\_

```
template<typename T , ssize_t Ndim = 2>
std::array<ssize_t, Ndim> aare::NDArray< T, Ndim >::strides_ [private]
```

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

- core/include/aare/core/[NDArray.hpp](#)

## 7.14 aare::NDView< T, Ndim > Class Template Reference

```
#include <NDView.hpp>
```

## Public Member Functions

- [NDView](#) ()
- [NDView](#) (T \*buffer, std::array< ssize\_t, Ndim > [shape](#))
- [NDView](#) (T \*buffer, const std::vector< ssize\_t > &[shape](#))
- template<typename... lx>  
std::enable\_if< sizeof...(lx)==Ndim, T & >::type [operator\(\)](#) (lx... index)
- template<typename... lx>  
std::enable\_if< sizeof...(lx)==Ndim, T & >::type [operator\(\)](#) (lx... index) const
- ssize\_t [size](#) () const
- [NDView](#) (const [NDView](#) &)=default
- [NDView](#) ([NDView](#) &&)=default
- T \* [begin](#) ()
- T \* [end](#) ()
- T & [operator\(\)](#) (ssize\_t i)
- T & [operator\[\]](#) (ssize\_t i)
- bool [operator==](#) (const [NDView](#) &other) const
- [NDView](#) & [operator+=](#) (const T val)
- [NDView](#) & [operator-=](#) (const T val)
- [NDView](#) & [operator\\*=](#) (const T val)
- [NDView](#) & [operator/=](#) (const T val)
- [NDView](#) & [operator/=](#) (const [NDView](#) &other)
- [NDView](#) & [operator=](#) (const T val)
- [NDView](#) & [operator=](#) (const [NDView](#) &other)
- auto & [shape](#) ()
- auto [shape](#) (ssize\_t i) const
- T \* [data](#) ()

## Private Member Functions

- template<class BinaryOperation >  
[NDView](#) & [elemenwise](#) (T val, BinaryOperation op)
- template<class BinaryOperation >  
[NDView](#) & [elemenwise](#) (const [NDView](#) &other, BinaryOperation op)

## Private Attributes

- T \* [buffer\\_](#) {nullptr}
- std::array< ssize\_t, Ndim > [strides\\_](#) {}
- std::array< ssize\_t, Ndim > [shape\\_](#) {}
- ssize\_t [size\\_](#) {}

## 7.14.1 Constructor & Destructor Documentation

### 7.14.1.1 NDView() [1/5]

```
template<typename T , ssize_t Ndim = 2>
aare::NDView< T, Ndim >::NDView ( ) [inline]
```

**7.14.1.2 NDView()** [2/5]

```
template<typename T , ssize_t Ndim = 2>
aare::NDView< T, Ndim >::NDView (
    T * buffer,
    std::array< ssize_t, Ndim > shape ) [inline]
```

**7.14.1.3 NDView()** [3/5]

```
template<typename T , ssize_t Ndim = 2>
aare::NDView< T, Ndim >::NDView (
    T * buffer,
    const std::vector< ssize_t > & shape ) [inline]
```

**7.14.1.4 NDView()** [4/5]

```
template<typename T , ssize_t Ndim = 2>
aare::NDView< T, Ndim >::NDView (
    const NDView< T, Ndim > & ) [default]
```

**7.14.1.5 NDView()** [5/5]

```
template<typename T , ssize_t Ndim = 2>
aare::NDView< T, Ndim >::NDView (
    NDView< T, Ndim > && ) [default]
```

**7.14.2 Member Function Documentation****7.14.2.1 begin()**

```
template<typename T , ssize_t Ndim = 2>
T * aare::NDView< T, Ndim >::begin ( ) [inline]
```

**7.14.2.2 data()**

```
template<typename T , ssize_t Ndim = 2>
T * aare::NDView< T, Ndim >::data ( ) [inline]
```

**7.14.2.3 elemenwise()** [1/2]

```
template<typename T , ssize_t Ndim = 2>
template<class BinaryOperation >
NDView & aare::NDView< T, Ndim >::elemenwise (
    const NDView< T, Ndim > & other,
    BinaryOperation op ) [inline], [private]
```

**7.14.2.4 elemenwise() [2/2]**

```
template<typename T , ssize_t Ndim = 2>
template<class BinaryOperation >
NDView & aare::NDView< T, Ndim >::elemenwise (
    T val,
    BinaryOperation op ) [inline], [private]
```

**7.14.2.5 end()**

```
template<typename T , ssize_t Ndim = 2>
T * aare::NDView< T, Ndim >::end ( ) [inline]
```

**7.14.2.6 operator()() [1/3]**

```
template<typename T , ssize_t Ndim = 2>
template<typename... Ix>
std::enable_if< sizeof...(Ix)==Ndim, T & >::type aare::NDView< T, Ndim >::operator() (
    Ix... index ) [inline]
```

**7.14.2.7 operator()() [2/3]**

```
template<typename T , ssize_t Ndim = 2>
template<typename... Ix>
std::enable_if< sizeof...(Ix)==Ndim, T & >::type aare::NDView< T, Ndim >::operator() (
    Ix... index ) const [inline]
```

**7.14.2.8 operator()() [3/3]**

```
template<typename T , ssize_t Ndim = 2>
T & aare::NDView< T, Ndim >::operator() (
    ssize_t i ) [inline]
```

**7.14.2.9 operator\*=( )**

```
template<typename T , ssize_t Ndim = 2>
NDView & aare::NDView< T, Ndim >::operator*= (
    const T val ) [inline]
```

**7.14.2.10 operator+=( )**

```
template<typename T , ssize_t Ndim = 2>
NDView & aare::NDView< T, Ndim >::operator+= (
    const T val ) [inline]
```



**7.14.2.11 operator-=( )**

```
template<typename T , ssize_t Ndim = 2>
NDView & aare::NDView< T, Ndim >::operator-= (
    const T val ) [inline]
```

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

```
template<typename T , ssize_t Ndim = 2>
NDView & aare::NDView< T, Ndim >::operator/= (
    const NDView< T, Ndim > & other ) [inline]
```

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

```
template<typename T , ssize_t Ndim = 2>
NDView & aare::NDView< T, Ndim >::operator/= (
    const T val ) [inline]
```

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

```
template<typename T , ssize_t Ndim = 2>
NDView & aare::NDView< T, Ndim >::operator= (
    const NDView< T, Ndim > & other ) [inline]
```

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

```
template<typename T , ssize_t Ndim = 2>
NDView & aare::NDView< T, Ndim >::operator= (
    const T val ) [inline]
```

**7.14.2.16 operator==( )**

```
template<typename T , ssize_t Ndim = 2>
bool aare::NDView< T, Ndim >::operator== (
    const NDView< T, Ndim > & other ) const [inline]
```

**7.14.2.17 operator[]( )**

```
template<typename T , ssize_t Ndim = 2>
T & aare::NDView< T, Ndim >::operator[] (
    ssize_t i ) [inline]
```

**7.14.2.18 shape() [1/2]**

```
template<typename T , ssize_t Ndim = 2>
auto & aare::NDView< T, Ndim >::shape ( ) [inline]
```

### 7.14.2.19 `shape()` [2/2]

```
template<typename T , ssize_t Ndim = 2>
auto aare::NDView< T, Ndim >::shape (
    ssize_t i ) const [inline]
```

### 7.14.2.20 `size()`

```
template<typename T , ssize_t Ndim = 2>
ssize_t aare::NDView< T, Ndim >::size ( ) const [inline]
```

## 7.14.3 Field Documentation

### 7.14.3.1 `buffer_`

```
template<typename T , ssize_t Ndim = 2>
T* aare::NDView< T, Ndim >::buffer_ {nullptr} [private]
```

### 7.14.3.2 `shape_`

```
template<typename T , ssize_t Ndim = 2>
std::array<ssize_t, Ndim> aare::NDView< T, Ndim >::shape_ {} [private]
```

### 7.14.3.3 `size_`

```
template<typename T , ssize_t Ndim = 2>
ssize_t aare::NDView< T, Ndim >::size_ {} [private]
```

### 7.14.3.4 `strides_`

```
template<typename T , ssize_t Ndim = 2>
std::array<ssize_t, Ndim> aare::NDView< T, Ndim >::strides_ {} [private]
```

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

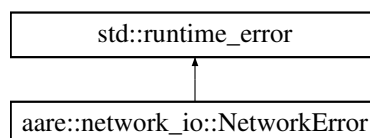
- `core/include/aare/core/NDView.hpp`

## 7.15 `aare::network_io::NetworkError` Class Reference

`NetworkError` exception class.

```
#include <defs.hpp>
```

Inheritance diagram for `aare::network_io::NetworkError`:



## Public Member Functions

- [NetworkError](#) (const char \*msg)
- [NetworkError](#) (const std::string msg)
- virtual const char \* [what](#) () const noexcept override

## Private Attributes

- const char \* [m\\_msg](#)

### 7.15.1 Detailed Description

[NetworkError](#) exception class.

### 7.15.2 Constructor & Destructor Documentation

#### 7.15.2.1 NetworkError() [1/2]

```
aare::network_io::NetworkError::NetworkError (  
    const char * msg ) [inline]
```

#### 7.15.2.2 NetworkError() [2/2]

```
aare::network_io::NetworkError::NetworkError (  
    const std::string msg ) [inline]
```

### 7.15.3 Member Function Documentation

#### 7.15.3.1 what()

```
virtual const char * aare::network_io::NetworkError::what ( ) const [inline], [override],  
[virtual], [noexcept]
```

### 7.15.4 Field Documentation

#### 7.15.4.1 m\_msg

```
const char* aare::network_io::NetworkError::m_msg [private]
```

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

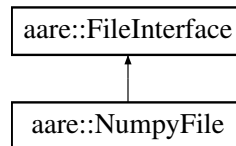
- [network\\_io/include/aare/network\\_io/defs.hpp](#)

## 7.16 aare::NumpyFile Class Reference

[NumpyFile](#) class to read and write numpy files.

```
#include <NumpyFile.hpp>
```

Inheritance diagram for aare::NumpyFile:



### Public Member Functions

- [NumpyFile](#) (const std::filesystem::path &fname, const std::string &mode="r", [FileConfig](#) cfg={})  
*NumpyFile constructor.*
- void [write](#) ([Frame](#) &frame) override  
*write a frame to the file*
- [Frame](#) [read](#) () override  
*write a vector of frames to the file*
- std::vector< [Frame](#) > [read](#) (size\_t n\_frames) override  
*read n\_frames from the file at the current position*
- void [read\\_into](#) (std::byte \*image\_buf) override  
*read one frame from the file at the current position and store it in the provided buffer*
- void [read\\_into](#) (std::byte \*image\_buf, size\_t n\_frames) override  
*read n\_frames from the file at the current position and store them in the provided buffer*
- size\_t [frame\\_number](#) (size\_t frame\_index) override  
*get the frame number at the given frame index*
- size\_t [bytes\\_per\\_frame](#) () override  
*get the size of one frame in bytes*
- size\_t [pixels](#) () override  
*get the number of pixels in one frame*
- void [seek](#) (size\_t [frame\\_number](#)) override  
*seek to the given frame number*
- size\_t [tell](#) () override  
*get the current position of the file pointer*
- size\_t [total\\_frames](#) () const override  
*get the total number of frames in the file*
- ssize\_t [rows](#) () const override  
*get the number of rows in the file*
- ssize\_t [cols](#) () const override  
*get the number of columns in the file*
- ssize\_t [bitdepth](#) () const override  
*get the bitdepth of the file*
- [DType](#) [dtype](#) () const  
*get the data type of the numpy file*
- std::vector< size\_t > [shape](#) () const  
*get the shape of the numpy file*

- `template<typename T, size_t NDim>`  
`NDArray< T, NDim > load ()`  
*load the numpy file into an [NDArray](#)*
- `~NumpyFile ()`
- `Frame iread (size_t frame_number)`  
*read one frame from the file at the given frame number*
- `std::vector< Frame > iread (size_t frame_number, size_t n_frames)`  
*read n\_frames from the file starting at the given frame number*

## Data Fields

- `std::string m_mode`
- `std::filesystem::path m_fname`
- `std::filesystem::path m_base_path`
- `std::string m_base_name`
- `std::string m_ext`
- `int m_findex`
- `size_t m_total_frames {}`
- `size_t max_frames_per_file {}`
- `std::string version`
- `DetectorType m_type`
- `ssize_t m_rows {}`
- `ssize_t m_cols {}`
- `ssize_t m_bitdepth {}`

## Private Member Functions

- `void load_metadata ()`
- `void get_frame_into (size_t, std::byte *)`
- `Frame get_frame (size_t frame_number)`

## Private Attributes

- `FILE * fp = nullptr`
- `size_t initial_header_len = 0`
- `size_t current_frame {}`
- `uint32_t header_len {}`
- `uint8_t header_len_size {}`
- `size_t header_size {}`
- `NumpyHeader m_header`
- `uint8_t major_ver_ {}`
- `uint8_t minor_ver_ {}`

### 7.16.1 Detailed Description

[NumpyFile](#) class to read and write numpy files.

#### Note

derived from [FileInterface](#)

implements all the pure virtual functions from [FileInterface](#)

documentation for the functions can also be found in the [FileInterface](#) class

## 7.16.2 Constructor & Destructor Documentation

### 7.16.2.1 NumpyFile()

```
aare::NumpyFile::NumpyFile (
    const std::filesystem::path & fname,
    const std::string & mode = "r",
    FileConfig cfg = {} )
```

[NumpyFile](#) constructor.

#### Parameters

<i>fname</i>	path to the numpy file
<i>mode</i>	file mode (r, w)
<i>cfg</i>	file configuration

### 7.16.2.2 ~NumpyFile()

```
aare::NumpyFile::~~NumpyFile ( )
```

## 7.16.3 Member Function Documentation

### 7.16.3.1 bitdepth()

```
ssize_t aare::NumpyFile::bitdepth ( ) const [inline], [override], [virtual]
```

get the bitdepth of the file

#### Returns

bitdepth of the file

Implements [aare::FileInterface](#).

### 7.16.3.2 bytes\_per\_frame()

```
size_t aare::NumpyFile::bytes_per_frame ( ) [override], [virtual]
```

get the size of one frame in bytes

#### Returns

size of one frame

Implements [aare::FileInterface](#).

### 7.16.3.3 cols()

```
ssize_t aare::NumpyFile::cols ( ) const [inline], [override], [virtual]
```

get the number of columns in the file

#### Returns

number of columns in the file

Implements [aare::FileInterface](#).

### 7.16.3.4 dtype()

```
DType aare::NumpyFile::dtype ( ) const [inline]
```

get the data type of the numpy file

#### Returns

DType

### 7.16.3.5 frame\_number()

```
size_t aare::NumpyFile::frame_number (
    size_t frame_index ) [inline], [override], [virtual]
```

get the frame number at the given frame index

#### Parameters

<i>frame_index</i>	index of the frame
--------------------	--------------------

#### Returns

frame number

Implements [aare::FileInterface](#).

### 7.16.3.6 get\_frame()

```
Frame aare::NumpyFile::get_frame (
    size_t frame_number ) [private]
```

### 7.16.3.7 get\_frame\_into()

```
void aare::NumpyFile::get_frame_into (
    size_t frame_number,
    std::byte * image_buf ) [private]
```

**7.16.3.8 `iread()`** [1/2]

```
Frame aare::FileInterface::iread (
    size_t frame_number ) [inline], [inherited]
```

read one frame from the file at the given frame number

**Parameters**

<i>frame_number</i>	frame number to read
---------------------	----------------------

**Returns**

frame

**7.16.3.9 `iread()`** [2/2]

```
std::vector< Frame > aare::FileInterface::iread (
    size_t frame_number,
    size_t n_frames ) [inline], [inherited]
```

read *n\_frames* from the file starting at the given frame number

**Parameters**

<i>frame_number</i>	frame number to start reading from
<i>n_frames</i>	number of frames to read

**Returns**

vector of frames

**7.16.3.10 `load()`**

```
template<typename T , size_t NDim>
NDArray< T, NDim > aare::NumpyFile::load ( ) [inline]
```

load the numpy file into an [NDArray](#)

**Template Parameters**

<i>T</i>	data type of the <a href="#">NDArray</a>
<i>NDim</i>	number of dimensions of the <a href="#">NDArray</a>

**Returns**

NDArray<T, NDim>



**7.16.3.11 load\_metadata()**

```
void aare::NumpyFile::load_metadata ( ) [private]
```

**7.16.3.12 pixels()**

```
size_t aare::NumpyFile::pixels ( ) [override], [virtual]
```

get the number of pixels in one frame

**Returns**

number of pixels in one frame

Implements [aare::FileInterface](#).

**7.16.3.13 read() [1/2]**

```
Frame aare::NumpyFile::read ( ) [inline], [override], [virtual]
```

write a vector of frames to the file

**Parameters**

<i>frames</i>	vector of frames to write
---------------	---------------------------

**Returns**

void

read one frame from the file at the current position

**Returns**

[Frame](#)

Implements [aare::FileInterface](#).

**7.16.3.14 read() [2/2]**

```
std::vector< Frame > aare::NumpyFile::read (
    size_t n_frames ) [override], [virtual]
```

read n\_frames from the file at the current position

**Parameters**

<i>n_frames</i>	number of frames to read
-----------------	--------------------------

**Returns**

vector of frames

Implements [aare::FileInterface](#).

**7.16.3.15 read\_into() [1/2]**

```
void aare::NumpyFile::read_into (
    std::byte * image_buf ) [inline], [override], [virtual]
```

read one frame from the file at the current position and store it in the provided buffer

**Parameters**

<i>image_buf</i>	buffer to store the frame
------------------	---------------------------

**Returns**

void

Implements [aare::FileInterface](#).

**7.16.3.16 read\_into() [2/2]**

```
void aare::NumpyFile::read_into (
    std::byte * image_buf,
    size_t n_frames ) [override], [virtual]
```

read n\_frames from the file at the current position and store them in the provided buffer

**Parameters**

<i>image_buf</i>	buffer to store the frames
<i>n_frames</i>	number of frames to read

**Returns**

void

Implements [aare::FileInterface](#).

**7.16.3.17 rows()**

```
ssize_t aare::NumpyFile::rows ( ) const [inline], [override], [virtual]
```

get the number of rows in the file

**Returns**

number of rows in the file

Implements [aare::FileInterface](#).

**7.16.3.18 seek()**

```
void aare::NumpyFile::seek (
    size_t frame_number ) [inline], [override], [virtual]
```

seek to the given frame number

**Parameters**

<i>frame_number</i>	frame number to seek to
---------------------	-------------------------

**Returns**

void

Implements [aare::FileInterface](#).

**7.16.3.19 shape()**

```
std::vector< size_t > aare::NumpyFile::shape ( ) const [inline]
```

get the shape of the numpy file

**Returns**

vector of type size\_t

**7.16.3.20 tell()**

```
size_t aare::NumpyFile::tell ( ) [inline], [override], [virtual]
```

get the current position of the file pointer

**Returns**

current position of the file pointer

Implements [aare::FileInterface](#).

**7.16.3.21 total\_frames()**

```
size_t aare::NumpyFile::total_frames ( ) const [inline], [override], [virtual]
```

get the total number of frames in the file

**Returns**

total number of frames in the file

Implements [aare::FileInterface](#).

**7.16.3.22 write()**

```
void aare::NumpyFile::write (
    Frame & frame ) [override], [virtual]
```

write a frame to the file

**Parameters**

<i>frame</i>	frame to write
--------------	----------------

**Returns**

void

**Exceptions**

<i>std::runtime_error</i>	if the function is not implemented
---------------------------	------------------------------------

Implements [aare::FileInterface](#).

**7.16.4 Field Documentation****7.16.4.1 current\_frame**

```
size_t aare::NumpyFile::current_frame {} [private]
```

**7.16.4.2 fp**

```
FILE* aare::NumpyFile::fp = nullptr [private]
```

**7.16.4.3 header\_len**

```
uint32_t aare::NumpyFile::header_len {} [private]
```

**7.16.4.4 header\_len\_size**

```
uint8_t aare::NumpyFile::header_len_size {} [private]
```

**7.16.4.5 header\_size**

```
size_t aare::NumpyFile::header_size {} [private]
```

**7.16.4.6 initial\_header\_len**

```
size_t aare::NumpyFile::initial_header_len = 0 [private]
```

**7.16.4.7 m\_base\_name**

```
std::string aare::FileInterface::m_base_name [inherited]
```

#### 7.16.4.8 m\_base\_path

`std::filesystem::path aare::FileInterface::m_base_path [inherited]`

#### 7.16.4.9 m\_bitdepth

`ssize_t aare::FileInterface::m_bitdepth {} [inherited]`

#### 7.16.4.10 m\_cols

`ssize_t aare::FileInterface::m_cols {} [inherited]`

#### 7.16.4.11 m\_ext

`std::string aare::FileInterface::m_ext [inherited]`

#### 7.16.4.12 m\_findex

`int aare::FileInterface::m_findex [inherited]`

#### 7.16.4.13 m\_fname

`std::filesystem::path aare::FileInterface::m_fname [inherited]`

#### 7.16.4.14 m\_header

`NumpyHeader aare::NumpyFile::m_header [private]`

#### 7.16.4.15 m\_mode

`std::string aare::FileInterface::m_mode [inherited]`

#### 7.16.4.16 m\_rows

`ssize_t aare::FileInterface::m_rows {} [inherited]`

#### 7.16.4.17 m\_total\_frames

`size_t aare::FileInterface::m_total_frames {} [inherited]`

#### 7.16.4.18 m\_type

```
DetectorType aare::FileInterface::m_type [inherited]
```

#### 7.16.4.19 major\_ver\_

```
uint8_t aare::NumpyFile::major_ver_ {} [private]
```

#### 7.16.4.20 max\_frames\_per\_file

```
size_t aare::FileInterface::max_frames_per_file {} [inherited]
```

#### 7.16.4.21 minor\_ver\_

```
uint8_t aare::NumpyFile::minor_ver_ {} [private]
```

#### 7.16.4.22 version

```
std::string aare::FileInterface::version [inherited]
```

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

- [file\\_io/include/aare/file\\_io/NumpyFile.hpp](#)
- [file\\_io/src/NumpyFile.cpp](#)

## 7.17 aare::NumpyHeader Struct Reference

```
#include <NumpyHelpers.hpp>
```

### Public Member Functions

- `std::string to\_string () const`

### Data Fields

- `DType dtype {aare::DType::ERROR}`
- `bool fortran\_order {false}`
- `shape_t shape {}`

### 7.17.1 Member Function Documentation

#### 7.17.1.1 to\_string()

```
std::string aare::NumpyHeader::to_string ( ) const
```

## 7.17.2 Field Documentation

### 7.17.2.1 dtype

```
DType aare::NumpyHeader::dtype {aare::DType::ERROR}
```

### 7.17.2.2 fortran\_order

```
bool aare::NumpyHeader::fortran_order {false}
```

### 7.17.2.3 shape

```
shape_t aare::NumpyHeader::shape {}
```

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

- [file\\_io/include/aare/file\\_io/NumpyHelpers.hpp](#)
- [file\\_io/src/NumpyHelpers.cpp](#)

## 7.18 folly::ProducerConsumerQueue< T > Struct Template Reference

```
#include <ProducerConsumerQueue.hpp>
```

### Public Types

- typedef T [value\\_type](#)

### Public Member Functions

- [ProducerConsumerQueue](#) (const [ProducerConsumerQueue](#) &)=delete
- [ProducerConsumerQueue](#) & [operator=](#) (const [ProducerConsumerQueue](#) &)=delete
- [ProducerConsumerQueue](#) (uint32\_t size)
- [~ProducerConsumerQueue](#) ()
- template<class... Args>  
  bool [write](#) (Args &&...recordArgs)
- bool [read](#) (T &record)
- T \* [frontPtr](#) ()
- void [popFront](#) ()
- bool [isEmpty](#) () const
- bool [isFull](#) () const
- size\_t [sizeGuess](#) () const
- size\_t [capacity](#) () const

### Private Types

- using [AtomicIndex](#) = std::atomic< unsigned int >

## Private Attributes

- char `pad0_` [`hardware_destructive_interference_size`]
- const uint32\_t `size_`
- T \*const `records_`
- AtomicIndex `readIndex_`
- AtomicIndex `writeIndex_`
- char `pad1_` [`hardware_destructive_interference_size` - sizeof(AtomicIndex)]

## 7.18.1 Member Typedef Documentation

### 7.18.1.1 AtomicIndex

```
template<class T >
using folly::ProducerConsumerQueue< T >::AtomicIndex = std::atomic<unsigned int> [private]
```

### 7.18.1.2 value\_type

```
template<class T >
typedef T folly::ProducerConsumerQueue< T >::value_type
```

## 7.18.2 Constructor & Destructor Documentation

### 7.18.2.1 ProducerConsumerQueue() [1/2]

```
template<class T >
folly::ProducerConsumerQueue< T >::ProducerConsumerQueue (
    const ProducerConsumerQueue< T > & ) [delete]
```

### 7.18.2.2 ProducerConsumerQueue() [2/2]

```
template<class T >
folly::ProducerConsumerQueue< T >::ProducerConsumerQueue (
    uint32_t size ) [inline], [explicit]
```

### 7.18.2.3 ~ProducerConsumerQueue()

```
template<class T >
folly::ProducerConsumerQueue< T >::~~ProducerConsumerQueue ( ) [inline]
```

## 7.18.3 Member Function Documentation

### 7.18.3.1 capacity()

```
template<class T >
size_t folly::ProducerConsumerQueue< T >::capacity ( ) const [inline]
```



### 7.18.3.2 frontPtr()

```
template<class T >
T * folly::ProducerConsumerQueue< T >::frontPtr ( ) [inline]
```

### 7.18.3.3 isEmpty()

```
template<class T >
bool folly::ProducerConsumerQueue< T >::isEmpty ( ) const [inline]
```

### 7.18.3.4 isFull()

```
template<class T >
bool folly::ProducerConsumerQueue< T >::isFull ( ) const [inline]
```

### 7.18.3.5 operator=()

```
template<class T >
ProducerConsumerQueue & folly::ProducerConsumerQueue< T >::operator= (
    const ProducerConsumerQueue< T > & ) [delete]
```

### 7.18.3.6 popFront()

```
template<class T >
void folly::ProducerConsumerQueue< T >::popFront ( ) [inline]
```

### 7.18.3.7 read()

```
template<class T >
bool folly::ProducerConsumerQueue< T >::read (
    T & record ) [inline]
```

### 7.18.3.8 sizeGuess()

```
template<class T >
size_t folly::ProducerConsumerQueue< T >::sizeGuess ( ) const [inline]
```

### 7.18.3.9 write()

```
template<class T >
template<class... Args>
bool folly::ProducerConsumerQueue< T >::write (
    Args &&... recordArgs ) [inline]
```

## 7.18.4 Field Documentation

### 7.18.4.1 pad0\_

```
template<class T >
char folly::ProducerConsumerQueue< T >::pad0_[hardware_destructive_interference_size] [private]
```

### 7.18.4.2 pad1\_

```
template<class T >
char folly::ProducerConsumerQueue< T >::pad1_[hardware_destructive_interference_size - sizeof(AtomicIndex)]
[private]
```

### 7.18.4.3 readIndex\_

```
template<class T >
AtomicIndex folly::ProducerConsumerQueue< T >::readIndex_ [private]
```

### 7.18.4.4 records\_

```
template<class T >
T* const folly::ProducerConsumerQueue< T >::records_ [private]
```

### 7.18.4.5 size\_

```
template<class T >
const uint32_t folly::ProducerConsumerQueue< T >::size_ [private]
```

### 7.18.4.6 writeIndex\_

```
template<class T >
AtomicIndex folly::ProducerConsumerQueue< T >::writeIndex_ [private]
```

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

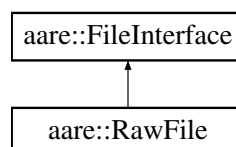
- [core/include/aare/core/ProducerConsumerQueue.hpp](#)

## 7.19 aare::RawFile Class Reference

[RawFile](#) class to read .raw and .json files.

```
#include <RawFile.hpp>
```

Inheritance diagram for aare::RawFile:



## Public Member Functions

- [RawFile](#) (const std::filesystem::path &fname, const std::string &mode="r", const [FileConfig](#) &cfg={})  
*RawFile constructor.*
- void [write](#) ([Frame](#) &frame) override  
*write function is not implemented for RawFile*
- [Frame read](#) () override  
*write a vector of frames to the file*
- std::vector< [Frame](#) > [read](#) (size\_t n\_frames) override  
*read n\_frames from the file at the current position*
- void [read\\_into](#) (std::byte \*image\_buf) override  
*read one frame from the file at the current position and store it in the provided buffer*
- void [read\\_into](#) (std::byte \*image\_buf, size\_t n\_frames) override  
*read n\_frames from the file at the current position and store them in the provided buffer*
- size\_t [frame\\_number](#) (size\_t frame\_index) override  
*get the frame number at the given frame index*
- size\_t [bytes\\_per\\_frame](#) () override  
*get the number of bytess per frame*
- size\_t [pixels](#) () override  
*get the number of pixels in the frame*
- void [seek](#) (size\_t [frame\\_number](#)) override  
*seek to the given frame number*
- size\_t [tell](#) () override  
*get the current position of the file pointer*
- void [set\\_config](#) (int row, int col)  
*set the module gap row and column*
- void [find\\_number\\_of\\_subfiles](#) ()  
*get the number of subfiles for the RawFile*
- std::filesystem::path [master\\_fname](#) ()  
*get the master file name path for the RawFile*
- std::filesystem::path [data\\_fname](#) (int mod\_id, int file\_id)  
*get the data file name path for the RawFile with the given module id and file id*
- [~RawFile](#) ()  
*destructor: will delete the subfiles*
- size\_t [total\\_frames](#) () const override  
*get the total number of frames in the file*
- ssize\_t [rows](#) () const override  
*get the number of rows in the file*
- ssize\_t [cols](#) () const override  
*get the number of columns in the file*
- ssize\_t [bitdepth](#) () const override  
*get the bitdepth of the file*
- [Frame iread](#) (size\_t [frame\\_number](#))  
*read one frame from the file at the given frame number*
- std::vector< [Frame](#) > [iread](#) (size\_t [frame\\_number](#), size\_t n\_frames)  
*read n\_frames from the file starting at the given frame number*

## Static Public Member Functions

- static bool [is\\_master\\_file](#) (std::filesystem::path fpath)  
*check if the file is a master file*

## Data Fields

- `std::string m_mode`
- `std::filesystem::path m_fname`
- `std::filesystem::path m_base_path`
- `std::string m_base_name`
- `std::string m_ext`
- `int m_findex`
- `size_t m_total_frames {}`
- `size_t max_frames_per_file {}`
- `std::string version`
- `DetectorType m_type`
- `ssize_t m_rows {}`
- `ssize_t m_cols {}`
- `ssize_t m_bitdepth {}`
- `size_t current_frame {}`

## Private Member Functions

- `void get_frame_into (size_t frame_number, std::byte *image_buf)`  
*read the frame at the given frame number into the image buffer*
- `Frame get_frame (size_t frame_number)`  
*get the frame at the given frame number*
- `void parse_fname ()`  
*parse the file name to get the extension, base name and index*
- `void parse_metadata ()`  
*parse the metadata from the file*
- `void parse_raw_metadata ()`  
*parse the metadata of a .raw file*
- `void parse_json_metadata ()`  
*parse the metadata of a .json file*
- `void find_geometry ()`  
*finds the geometry of the file*
- `sls_detector_header read_header (const std::filesystem::path &fname)`  
*read the header of the file*
- `void open_subfiles ()`  
*open the subfiles*

## Private Attributes

- `size_t n_subfiles`
- `size_t n_subfile_parts`
- `std::vector< std::vector< SubFile * > > subfiles`
- `int subfile_rows`
- `int subfile_cols`
- `xy geometry`
- `std::vector< xy > positions`
- `RawFileConfig cfg {0, 0}`
- `TimingMode timing_mode`
- `bool quad {false}`

## 7.19.1 Detailed Description

[RawFile](#) class to read .raw and .json files.

### Note

derived from [FileInterface](#)

documentation can also be found in the [FileInterface](#) class

## 7.19.2 Constructor & Destructor Documentation

### 7.19.2.1 RawFile()

```
aare::RawFile::RawFile (
    const std::filesystem::path & fname,
    const std::string & mode = "r",
    const FileConfig & cfg = {} )
```

[RawFile](#) constructor.

#### Parameters

<i>fname</i>	path to the file
<i>mode</i>	file mode (r, w)
<i>cfg</i>	file configuration

### 7.19.2.2 ~RawFile()

```
aare::RawFile::~~RawFile ( )
```

destructor: will delete the subfiles

## 7.19.3 Member Function Documentation

### 7.19.3.1 bitdepth()

```
ssize_t aare::RawFile::bitdepth ( ) const [inline], [override], [virtual]
```

get the bitdepth of the file

#### Returns

bitdepth of the file

Implements [aare::FileInterface](#).

### 7.19.3.2 bytes\_per\_frame()

```
size_t aare::RawFile::bytes_per_frame ( ) [inline], [override], [virtual]
```

get the number of bytes per frame

#### Returns

size of one frame in bytes

Implements [aare::FileInterface](#).

### 7.19.3.3 cols()

```
ssize_t aare::RawFile::cols ( ) const [inline], [override], [virtual]
```

get the number of columns in the file

#### Returns

number of columns in the file

Implements [aare::FileInterface](#).

### 7.19.3.4 data\_fname()

```
std::filesystem::path aare::RawFile::data_fname (
    int mod_id,
    int file_id ) [inline]
```

get the data file name path for the [RawFile](#) with the given module id and file id

#### Parameters

<i>mod_id</i>	module id
<i>file_id</i>	file id

#### Returns

path to the data file

### 7.19.3.5 find\_geometry()

```
void aare::RawFile::find_geometry ( ) [private]
```

finds the geometry of the file

### 7.19.3.6 find\_number\_of\_subfiles()

```
void aare::RawFile::find_number_of_subfiles ( )
```

get the number of subfiles for the [RawFile](#)

#### Returns

number of subfiles

### 7.19.3.7 frame\_number()

```
size_t aare::RawFile::frame_number (
    size_t frame_index ) [override], [virtual]
```

get the frame number at the given frame index

#### Parameters

<i>frame_index</i>	index of the frame
--------------------	--------------------

#### Returns

frame number

Implements [aare::FileInterface](#).

### 7.19.3.8 get\_frame()

```
Frame aare::RawFile::get_frame (
    size_t frame_number ) [private]
```

get the frame at the given frame number

#### Parameters

<i>frame_number</i>	frame number to read
---------------------	----------------------

#### Returns

[Frame](#)

### 7.19.3.9 get\_frame\_into()

```
void aare::RawFile::get_frame_into (
    size_t frame_number,
    std::byte * image_buf ) [private]
```

read the frame at the given frame number into the image buffer

## Parameters

<i>frame_number</i>	frame number to read
<i>image_buf</i>	buffer to store the frame

**7.19.3.10** `iread()` [1/2]

```
Frame aare::FileInterface::iread (  
    size_t frame_number ) [inline], [inherited]
```

read one frame from the file at the given frame number

## Parameters

<i>frame_number</i>	frame number to read
---------------------	----------------------

## Returns

frame

**7.19.3.11** `iread()` [2/2]

```
std::vector< Frame > aare::FileInterface::iread (  
    size_t frame_number,  
    size_t n_frames ) [inline], [inherited]
```

read *n\_frames* from the file starting at the given frame number

## Parameters

<i>frame_number</i>	frame number to start reading from
<i>n_frames</i>	number of frames to read

## Returns

vector of frames

**7.19.3.12** `is_master_file()`

```
bool aare::RawFile::is_master_file (  
    std::filesystem::path fpath ) [static]
```

check if the file is a master file

## Parameters

<i>fpath</i>	path to the file
--------------	------------------



### 7.19.3.13 master\_fname()

```
std::filesystem::path aare::RawFile::master_fname ( ) [inline]
```

get the master file name path for the [RawFile](#)

#### Returns

path to the master file

### 7.19.3.14 open\_subfiles()

```
void aare::RawFile::open_subfiles ( ) [private]
```

open the subfiles

### 7.19.3.15 parse\_fname()

```
void aare::RawFile::parse_fname ( ) [private]
```

parse the file name to get the extension, base name and index

### 7.19.3.16 parse\_json\_metadata()

```
void aare::RawFile::parse_json_metadata ( ) [private]
```

parse the metadata of a .json file

### 7.19.3.17 parse\_metadata()

```
void aare::RawFile::parse_metadata ( ) [private]
```

parse the metadata from the file

### 7.19.3.18 parse\_raw\_metadata()

```
void aare::RawFile::parse_raw_metadata ( ) [private]
```

parse the metadata of a .raw file

### 7.19.3.19 pixels()

```
size_t aare::RawFile::pixels ( ) [inline], [override], [virtual]
```

get the number of pixels in the frame

#### Returns

number of pixels

Implements [aare::FileInterface](#).

### 7.19.3.20 read() [1/2]

```
Frame aare::RawFile::read ( ) [inline], [override], [virtual]
```

write a vector of frames to the file

## Parameters

<i>frames</i>	vector of frames to write
---------------	---------------------------

## Returns

void

read one frame from the file at the current position

## Returns

[Frame](#)

Implements [aare::FileInterface](#).

**7.19.3.21 read()** [2/2]

```
std::vector< Frame > aare::RawFile::read (
    size_t n_frames ) [override], [virtual]
```

read *n\_frames* from the file at the current position

## Parameters

<i>n_frames</i>	number of frames to read
-----------------	--------------------------

## Returns

vector of frames

Implements [aare::FileInterface](#).

**7.19.3.22 read\_header()**

```
sls\_detector\_header aare::RawFile::read_header (
    const std::filesystem::path & fname ) [private]
```

read the header of the file

## Parameters

<i>fname</i>	path to the data subfile
--------------	--------------------------

## Returns

[sls\\_detector\\_header](#)

### 7.19.3.23 read\_into() [1/2]

```
void aare::RawFile::read_into (
    std::byte * image_buf ) [inline], [override], [virtual]
```

read one frame from the file at the current position and store it in the provided buffer

#### Parameters

<i>image_buf</i>	buffer to store the frame
------------------	---------------------------

#### Returns

void

Implements [aare::FileInterface](#).

### 7.19.3.24 read\_into() [2/2]

```
void aare::RawFile::read_into (
    std::byte * image_buf,
    size_t n_frames ) [override], [virtual]
```

read n\_frames from the file at the current position and store them in the provided buffer

#### Parameters

<i>image_buf</i>	buffer to store the frames
<i>n_frames</i>	number of frames to read

#### Returns

void

Implements [aare::FileInterface](#).

### 7.19.3.25 rows()

```
ssize_t aare::RawFile::rows ( ) const [inline], [override], [virtual]
```

get the number of rows in the file

#### Returns

number of rows in the file

Implements [aare::FileInterface](#).

### 7.19.3.26 seek()

```
void aare::RawFile::seek (
    size_t frame_number ) [inline], [override], [virtual]
```

seek to the given frame number

**Parameters**

<i>frame_number</i>	frame number to seek to
---------------------	-------------------------

**Returns**

void

Implements [aare::FileInterface](#).

**7.19.3.27 set\_config()**

```
void aare::RawFile::set_config (
    int row,
    int col ) [inline]
```

set the module gap row and column

**Parameters**

<i>row</i>	gap between rows
<i>col</i>	gap between columns

**7.19.3.28 tell()**

```
size_t aare::RawFile::tell ( ) [inline], [override], [virtual]
```

get the current position of the file pointer

**Returns**

current position of the file pointer

Implements [aare::FileInterface](#).

**7.19.3.29 total\_frames()**

```
size_t aare::RawFile::total_frames ( ) const [inline], [override], [virtual]
```

get the total number of frames in the file

**Returns**

total number of frames in the file

Implements [aare::FileInterface](#).

**7.19.3.30 write()**

```
void aare::RawFile::write (
    Frame & frame ) [inline], [override], [virtual]
```

write function is not implemented for [RawFile](#)

## Parameters

<i>frame</i>	frame to write
--------------	----------------

Implements [aare::FileInterface](#).

## 7.19.4 Field Documentation

### 7.19.4.1 cfg

```
RawFileConfig aare::RawFile::cfg {0, 0} [private]
```

### 7.19.4.2 current\_frame

```
size_t aare::FileInterface::current_frame {} [inherited]
```

### 7.19.4.3 geometry

```
xy aare::RawFile::geometry [private]
```

### 7.19.4.4 m\_base\_name

```
std::string aare::FileInterface::m_base_name [inherited]
```

### 7.19.4.5 m\_base\_path

```
std::filesystem::path aare::FileInterface::m_base_path [inherited]
```

### 7.19.4.6 m\_bitdepth

```
ssize_t aare::FileInterface::m_bitdepth {} [inherited]
```

### 7.19.4.7 m\_cols

```
ssize_t aare::FileInterface::m_cols {} [inherited]
```

### 7.19.4.8 m\_ext

```
std::string aare::FileInterface::m_ext [inherited]
```

#### 7.19.4.9 m\_findex

```
int aare::FileInterface::m_findex [inherited]
```

#### 7.19.4.10 m\_fname

```
std::filesystem::path aare::FileInterface::m_fname [inherited]
```

#### 7.19.4.11 m\_mode

```
std::string aare::FileInterface::m_mode [inherited]
```

#### 7.19.4.12 m\_rows

```
ssize_t aare::FileInterface::m_rows {} [inherited]
```

#### 7.19.4.13 m\_total\_frames

```
size_t aare::FileInterface::m_total_frames {} [inherited]
```

#### 7.19.4.14 m\_type

```
DetectorType aare::FileInterface::m_type [inherited]
```

#### 7.19.4.15 max\_frames\_per\_file

```
size_t aare::FileInterface::max_frames_per_file {} [inherited]
```

#### 7.19.4.16 n\_subfile\_parts

```
size_t aare::RawFile::n_subfile_parts [private]
```

#### 7.19.4.17 n\_subfiles

```
size_t aare::RawFile::n_subfiles [private]
```

#### 7.19.4.18 positions

```
std::vector<xy> aare::RawFile::positions [private]
```

**7.19.4.19 quad**

```
bool aare::RawFile::quad {false} [private]
```

**7.19.4.20 subfile\_cols**

```
int aare::RawFile::subfile_cols [private]
```

**7.19.4.21 subfile\_rows**

```
int aare::RawFile::subfile_rows [private]
```

**7.19.4.22 subfiles**

```
std::vector<std::vector<SubFile *> > aare::RawFile::subfiles [private]
```

**7.19.4.23 timing\_mode**

```
TimingMode aare::RawFile::timing_mode [private]
```

**7.19.4.24 version**

```
std::string aare::FileInterface::version [inherited]
```

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

- file\_io/include/aare/file\_io/[RawFile.hpp](#)
- file\_io/src/[RawFile.cpp](#)

**7.20 aare::RawFileConfig Struct Reference**

```
#include <defs.hpp>
```

**Public Member Functions**

- bool [operator==](#) (const [RawFileConfig](#) &other) const

**Data Fields**

- int [module\\_gap\\_row](#) {}
- int [module\\_gap\\_col](#) {}

## 7.20.1 Member Function Documentation

### 7.20.1.1 operator==()

```
bool aare::RawFileConfig::operator== (
    const RawFileConfig & other ) const [inline]
```

## 7.20.2 Field Documentation

### 7.20.2.1 module\_gap\_col

```
int aare::RawFileConfig::module_gap_col {}
```

### 7.20.2.2 module\_gap\_row

```
int aare::RawFileConfig::module_gap_row {}
```

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

- [core/include/aare/core/defs.hpp](#)

## 7.21 aare::sls\_detector\_header Struct Reference

```
#include <defs.hpp>
```

### Data Fields

- uint64\_t [frameNumber](#)
- uint32\_t [expLength](#)
- uint32\_t [packetNumber](#)
- uint64\_t [bunchId](#)
- uint64\_t [timestamp](#)
- uint16\_t [modId](#)
- uint16\_t [row](#)
- uint16\_t [column](#)
- uint16\_t [reserved](#)
- uint32\_t [debug](#)
- uint16\_t [roundRNumber](#)
- uint8\_t [detType](#)
- uint8\_t [version](#)
- uint8\_t [packetMask](#) [64]

### 7.21.1 Field Documentation

#### 7.21.1.1 bunchId

```
uint64_t aare::sls_detector_header::bunchId
```



### 7.21.1.2 column

uint16\_t aare::sls\_detector\_header::column

### 7.21.1.3 debug

uint32\_t aare::sls\_detector\_header::debug

### 7.21.1.4 detType

uint8\_t aare::sls\_detector\_header::detType

### 7.21.1.5 expLength

uint32\_t aare::sls\_detector\_header::expLength

### 7.21.1.6 frameNumber

uint64\_t aare::sls\_detector\_header::frameNumber

### 7.21.1.7 modId

uint16\_t aare::sls\_detector\_header::modId

### 7.21.1.8 packetMask

uint8\_t aare::sls\_detector\_header::packetMask[64]

### 7.21.1.9 packetNumber

uint32\_t aare::sls\_detector\_header::packetNumber

### 7.21.1.10 reserved

uint16\_t aare::sls\_detector\_header::reserved

### 7.21.1.11 roundRNumber

uint16\_t aare::sls\_detector\_header::roundRNumber

### 7.21.1.12 row

```
uint16_t aare::sls_detector_header::row
```

### 7.21.1.13 timestamp

```
uint64_t aare::sls_detector_header::timestamp
```

### 7.21.1.14 version

```
uint8_t aare::sls_detector_header::version
```

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

- [core/include/aare/core/defs.hpp](#)

## 7.22 aare::SubFile Class Reference

Class to read a subfile from a [RawFile](#).

```
#include <SubFile.hpp>
```

### Public Member Functions

- [SubFile](#) (std::filesystem::path fname, [DetectorType](#) detector, ssize\_t rows, ssize\_t cols, uint16\_t bitdepth)  
*SubFile constructor.*
- size\_t [read\\_impl\\_normal](#) (std::byte \*buffer)  
*read the subfile into a buffer*
- template<typename DataType >  
size\_t [read\\_impl\\_flip](#) (std::byte \*buffer)  
*read the subfile into a buffer with the bytes flipped*
- template<typename DataType >  
size\_t [read\\_impl\\_reorder](#) (std::byte \*buffer)  
*read the subfile into a buffer with the bytes reordered*
- size\_t [get\\_part](#) (std::byte \*buffer, int [frame\\_number](#))  
*read the subfile into a buffer with the bytes reordered and flipped*
- size\_t [frame\\_number](#) (int frame\_index)
- size\_t [bytes\\_per\\_part](#) ()
- size\_t [pixels\\_per\\_part](#) ()

### Protected Types

- using [pfunc](#) = size\_t(SubFile::\*)(std::byte \*)  
*type of the read\_impl function pointer*

## Protected Attributes

- `pfunc read_impl = nullptr`
- `std::map< std::pair< DetectorType, int >, pfunc > read_impl_map`  
*map to store the read\_impl functions for different detectors*
- `FILE * fp = nullptr`
- `ssize_t m_bitdepth`
- `std::filesystem::path m_fname`
- `ssize_t m_rows {}`
- `ssize_t m_cols {}`
- `ssize_t n_frames {}`
- `int m_sub_file_index_ {}`

### 7.22.1 Detailed Description

Class to read a subfile from a [RawFile](#).

### 7.22.2 Member Typedef Documentation

#### 7.22.2.1 pfunc

```
using aare::SubFile::pfunc = size_t (SubFile::*)(std::byte *) [protected]
```

type of the read\_impl function pointer

#### Parameters

<i>buffer</i>	pointer to the buffer to read the data into
---------------	---

#### Returns

number of bytes read

### 7.22.3 Constructor & Destructor Documentation

#### 7.22.3.1 SubFile()

```
aare::SubFile::SubFile (
    std::filesystem::path fname,
    DetectorType detector,
    ssize_t rows,
    ssize_t cols,
    uint16_t bitdepth )
```

[SubFile](#) constructor.

#### Parameters

<i>fname</i>	path to the subfile
--------------	---------------------

## Parameters

<i>detector</i>	detector type
<i>rows</i>	number of rows in the subfile
<i>cols</i>	number of columns in the subfile
<i>bitdepth</i>	bitdepth of the subfile

## Exceptions

<i>std::invalid_argument</i>	if the detector,type pair is not supported
------------------------------	--

## 7.22.4 Member Function Documentation

### 7.22.4.1 bytes\_per\_part()

```
size_t aare::SubFile::bytes_per_part ( ) [inline]
```

### 7.22.4.2 frame\_number()

```
size_t aare::SubFile::frame_number (
    int frame_index )
```

### 7.22.4.3 get\_part()

```
size_t aare::SubFile::get_part (
    std::byte * buffer,
    int frame_number )
```

read the subfile into a buffer with the bytes reordered and flipped

## Parameters

<i>buffer</i>	pointer to the buffer to read the data into
<i>frame_number</i>	frame number to read

## Returns

number of bytes read

### 7.22.4.4 pixels\_per\_part()

```
size_t aare::SubFile::pixels_per_part ( ) [inline]
```

#### 7.22.4.5 read\_impl\_flip()

```
template<typename DataType >
size_t aare::SubFile::read_impl_flip (
    std::byte * buffer )
```

read the subfile into a buffer with the bytes flipped

##### Parameters

<i>buffer</i>	pointer to the buffer to read the data into
---------------	---

##### Returns

number of bytes read

#### 7.22.4.6 read\_impl\_normal()

```
size_t aare::SubFile::read_impl_normal (
    std::byte * buffer )
```

read the subfile into a buffer

##### Parameters

<i>buffer</i>	pointer to the buffer to read the data into
---------------	---

##### Returns

number of bytes read

#### 7.22.4.7 read\_impl\_reorder()

```
template<typename DataType >
size_t aare::SubFile::read_impl_reorder (
    std::byte * buffer )
```

read the subfile into a buffer with the bytes reordered

##### Parameters

<i>buffer</i>	pointer to the buffer to read the data into
---------------	---

##### Returns

number of bytes read

## 7.22.5 Field Documentation

### 7.22.5.1 fp

```
FILE* aare::SubFile::fp = nullptr [protected]
```

### 7.22.5.2 m\_bitdepth

```
ssize_t aare::SubFile::m_bitdepth [protected]
```

### 7.22.5.3 m\_cols

```
ssize_t aare::SubFile::m_cols {} [protected]
```

### 7.22.5.4 m\_fname

```
std::filesystem::path aare::SubFile::m_fname [protected]
```

### 7.22.5.5 m\_rows

```
ssize_t aare::SubFile::m_rows {} [protected]
```

### 7.22.5.6 m\_sub\_file\_index\_

```
int aare::SubFile::m_sub_file_index_ {} [protected]
```

### 7.22.5.7 n\_frames

```
ssize_t aare::SubFile::n_frames {} [protected]
```

### 7.22.5.8 read\_impl

```
pfunc aare::SubFile::read_impl = nullptr [protected]
```

### 7.22.5.9 read\_impl\_map

```
std::map<std::pair<DetectorType, int>, pfunc> aare::SubFile::read_impl_map [protected]
```

#### Initial value:

```
= {
    {{DetectorType::Moench, 16}, &SubFile::read_impl_reorder<uint16_t>},
    {{DetectorType::Jungfrau, 16}, &SubFile::read_impl_normal},
    {{DetectorType::ChipTestBoard, 16}, &SubFile::read_impl_normal},
    {{DetectorType::Mythen3, 32}, &SubFile::read_impl_normal},
    {{DetectorType::Eiger, 32}, &SubFile::read_impl_normal},
    {{DetectorType::Eiger, 16}, &SubFile::read_impl_normal}
}
```

map to store the read\_impl functions for different detectors

#### Note

- the key is a pair of DetectorType and bitdepth
- the value is a pointer to the read\_impl function specific for the detector
- the read\_impl function will be set to the appropriate function in the constructor

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

- file\_io/include/aare/file\_io/SubFile.hpp
- file\_io/src/SubFile.cpp

## 7.23 aare::xy Struct Reference

```
#include <defs.hpp>
```

### Public Member Functions

- bool `operator==` (const `xy` &other) const
- bool `operator!=` (const `xy` &other) const

### Data Fields

- int `row`
- int `col`

## 7.23.1 Member Function Documentation

### 7.23.1.1 operator"!=()

```
bool aare::xy::operator!= (
    const xy & other ) const [inline]
```

### 7.23.1.2 operator==( )

```
bool aare::xy::operator== (
    const xy & other ) const [inline]
```

## 7.23.2 Field Documentation

### 7.23.2.1 col

```
int aare::xy::col
```

### 7.23.2.2 row

```
int aare::xy::row
```

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

- [core/include/aare/core/defs.hpp](#)

## 7.24 aare::ZmqFrame Struct Reference

[ZmqFrame](#) structure wrapper class to contain a [ZmqHeader](#) and a [Frame](#).

```
#include <defs.hpp>
```

### Data Fields

- [ZmqHeader](#) header
- [Frame](#) frame

### 7.24.1 Detailed Description

[ZmqFrame](#) structure wrapper class to contain a [ZmqHeader](#) and a [Frame](#).

## 7.24.2 Field Documentation

### 7.24.2.1 frame

```
Frame aare::ZmqFrame::frame
```



### 7.24.2.2 header

`ZmqHeader` `aare::ZmqFrame::header`

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

- `network_io/include/aare/network_io/defs.hpp`

## 7.25 aare::ZmqHeader Struct Reference

```
#include <ZmqHeader.hpp>
```

### Public Member Functions

- `std::string to_string () const`
- `void from_string (std::string &s)`
- `bool operator== (const ZmqHeader &other) const`

### Data Fields

- `bool data {true}`
- `uint32_t jsonversion {0}`
- `uint32_t dynamicRange {0}`
- `uint64_t fileIndex {0}`
- `uint32_t ndetx {0}`
- `uint32_t ndety {0}`
- `uint32_t npixelsx {0}`
- `uint32_t npixelsy {0}`
- `uint32_t imageSize {0}`
- `uint64_t acqIndex {0}`
- `uint64_t frameIndex {0}`
- `double progress {0}`
- `std::string fname`
- `uint64_t frameNumber {0}`
- `uint32_t expLength {0}`
- `uint32_t packetNumber {0}`
- `uint64_t detSpec1 {0}`
- `uint64_t timestamp {0}`
- `uint16_t modId {0}`
- `uint16_t row {0}`
- `uint16_t column {0}`
- `uint16_t detSpec2 {0}`
- `uint32_t detSpec3 {0}`
- `uint16_t detSpec4 {0}`
- `uint8_t detType {0}`
- `uint8_t version {0}`
- `int flipRows {0}`
- `uint32_t quad {0}`
- `bool completeImage {false}`
- `std::map< std::string, std::string > addJsonHeader`
- `std::array< int, 4 > rx_roi {}`

### 7.25.1 Detailed Description

zmq header structure (from sIsDetectorPackage)

### 7.25.2 Member Function Documentation

#### 7.25.2.1 from\_string()

```
void aare::ZmqHeader::from_string (
    std::string & s )
```

#### 7.25.2.2 operator==()

```
bool aare::ZmqHeader::operator== (
    const ZmqHeader & other ) const
```

#### 7.25.2.3 to\_string()

```
std::string aare::ZmqHeader::to_string ( ) const
```

serialize struct to json string

### 7.25.3 Field Documentation

#### 7.25.3.1 acqIndex

```
uint64_t aare::ZmqHeader::acqIndex {0}
```

frame number from detector

#### 7.25.3.2 addJsonHeader

```
std::map<std::string, std::string> aare::ZmqHeader::addJsonHeader
```

additional json header

#### 7.25.3.3 column

```
uint16_t aare::ZmqHeader::column {0}
```

#### 7.25.3.4 completeImage

```
bool aare::ZmqHeader::completeImage {false}
```

true if complete image, else missing packets

### 7.25.3.5 data

```
bool aare::ZmqHeader::data {true}
```

true if incoming data, false if end of acquisition

### 7.25.3.6 detSpec1

```
uint64_t aare::ZmqHeader::detSpec1 {0}
```

### 7.25.3.7 detSpec2

```
uint16_t aare::ZmqHeader::detSpec2 {0}
```

### 7.25.3.8 detSpec3

```
uint32_t aare::ZmqHeader::detSpec3 {0}
```

### 7.25.3.9 detSpec4

```
uint16_t aare::ZmqHeader::detSpec4 {0}
```

### 7.25.3.10 detType

```
uint8_t aare::ZmqHeader::detType {0}
```

### 7.25.3.11 dynamicRange

```
uint32_t aare::ZmqHeader::dynamicRange {0}
```

### 7.25.3.12 expLength

```
uint32_t aare::ZmqHeader::expLength {0}
```

### 7.25.3.13 fileIndex

```
uint64_t aare::ZmqHeader::fileIndex {0}
```

### 7.25.3.14 flipRows

```
int aare::ZmqHeader::flipRows {0}
```

if rows of image should be flipped

**7.25.3.15 fname**

```
std::string aare::ZmqHeader::fname
```

file name prefix

**7.25.3.16 frameIndex**

```
uint64_t aare::ZmqHeader::frameIndex {0}
```

frame index (starting at 0 for each acquisition)

**7.25.3.17 frameNumber**

```
uint64_t aare::ZmqHeader::frameNumber {0}
```

header from detector

**7.25.3.18 imageSize**

```
uint32_t aare::ZmqHeader::imageSize {0}
```

number of bytes for an image in this socket

**7.25.3.19 jsonversion**

```
uint32_t aare::ZmqHeader::jsonversion {0}
```

**7.25.3.20 modId**

```
uint16_t aare::ZmqHeader::modId {0}
```

**7.25.3.21 ndetx**

```
uint32_t aare::ZmqHeader::ndetx {0}
```

number of detectors/port in x axis

**7.25.3.22 ndety**

```
uint32_t aare::ZmqHeader::ndety {0}
```

number of detectors/port in y axis

### 7.25.3.23 npixelsx

```
uint32_t aare::ZmqHeader::npixelsx {0}
```

number of pixels/channels in x axis for this zmq socket

### 7.25.3.24 npixelsy

```
uint32_t aare::ZmqHeader::npixelsy {0}
```

number of pixels/channels in y axis for this zmq socket

### 7.25.3.25 packetNumber

```
uint32_t aare::ZmqHeader::packetNumber {0}
```

### 7.25.3.26 progress

```
double aare::ZmqHeader::progress {0}
```

progress in percentage

### 7.25.3.27 quad

```
uint32_t aare::ZmqHeader::quad {0}
```

quad type (eiger hardware specific)

### 7.25.3.28 row

```
uint16_t aare::ZmqHeader::row {0}
```

### 7.25.3.29 rx\_roi

```
std::array<int, 4> aare::ZmqHeader::rx_roi {}
```

(xmin, xmax, ymin, ymax) roi only in files written

### 7.25.3.30 timestamp

```
uint64_t aare::ZmqHeader::timestamp {0}
```

### 7.25.3.31 version

```
uint8_t aare::ZmqHeader::version {0}
```

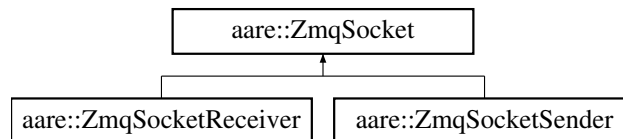
The documentation for this struct was generated from the following files:

- [network\\_io/include/aare/network\\_io/ZmqHeader.hpp](#)
- [network\\_io/src/ZmqHeader.cpp](#)

## 7.26 aare::ZmqSocket Class Reference

```
#include <ZmqSocket.hpp>
```

Inheritance diagram for aare::ZmqSocket:



### Public Member Functions

- [ZmqSocket](#) ()=default
- [~ZmqSocket](#) ()
- [ZmqSocket](#) (const [ZmqSocket](#) &)=delete
- [ZmqSocket operator=](#) (const [ZmqSocket](#) &)=delete
- [ZmqSocket](#) ([ZmqSocket](#) &&)=delete
- void [disconnect](#) ()
- void [set\\_zmq\\_hwm](#) (int hwm)
- void [set\\_timeout\\_ms](#) (int n)
- void [set\\_potential\\_frame\\_size](#) (size\_t size)

### Protected Attributes

- void \* [m\\_context](#) {nullptr}
- void \* [m\\_socket](#) {nullptr}
- std::string [m\\_endpoint](#)
- int [m\\_zmq\\_hwm](#) {1000}
- int [m\\_timeout\\_ms](#) {1000}
- size\_t [m\\_potential\\_frame\\_size](#) {1024 \* 1024}
- char \* [m\\_header\\_buffer](#) = new char[[m\\_max\\_header\\_size](#)]

### Static Protected Attributes

- static constexpr size\_t [m\\_max\\_header\\_size](#) = 1024

## 7.26.1 Constructor & Destructor Documentation

### 7.26.1.1 ZmqSocket() [1/3]

```
aare::ZmqSocket::ZmqSocket ( ) [default]
```

### 7.26.1.2 ~ZmqSocket()

```
aare::ZmqSocket::~~ZmqSocket ( )
```

### 7.26.1.3 ZmqSocket() [2/3]

```
aare::ZmqSocket::ZmqSocket (
    const ZmqSocket & ) [delete]
```

### 7.26.1.4 ZmqSocket() [3/3]

```
aare::ZmqSocket::ZmqSocket (
    ZmqSocket && ) [delete]
```

## 7.26.2 Member Function Documentation

### 7.26.2.1 disconnect()

```
void aare::ZmqSocket::disconnect ( )
```

### 7.26.2.2 operator=()

```
ZmqSocket aare::ZmqSocket::operator= (
    const ZmqSocket & ) [delete]
```

### 7.26.2.3 set\_potential\_frame\_size()

```
void aare::ZmqSocket::set_potential_frame_size (
    size_t size )
```

### 7.26.2.4 set\_timeout\_ms()

```
void aare::ZmqSocket::set_timeout_ms (
    int n )
```

### 7.26.2.5 set\_zmq\_hwm()

```
void aare::ZmqSocket::set_zmq_hwm (
    int hwm )
```

## 7.26.3 Field Documentation

### 7.26.3.1 m\_context

```
void* aare::ZmqSocket::m_context {nullptr} [protected]
```

### 7.26.3.2 m\_endpoint

```
std::string aare::ZmqSocket::m_endpoint [protected]
```

### 7.26.3.3 m\_header\_buffer

```
char* aare::ZmqSocket::m_header_buffer = new char[m_max_header_size] [protected]
```

### 7.26.3.4 m\_max\_header\_size

```
constexpr size_t aare::ZmqSocket::m_max_header_size = 1024 [static], [constexpr], [protected]
```

### 7.26.3.5 m\_potential\_frame\_size

```
size_t aare::ZmqSocket::m_potential_frame_size {1024 * 1024} [protected]
```

### 7.26.3.6 m\_socket

```
void* aare::ZmqSocket::m_socket {nullptr} [protected]
```

### 7.26.3.7 m\_timeout\_ms

```
int aare::ZmqSocket::m_timeout_ms {1000} [protected]
```

### 7.26.3.8 m\_zmq\_hwm

```
int aare::ZmqSocket::m_zmq_hwm {1000} [protected]
```

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

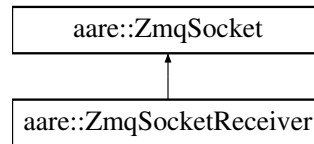
- [network\\_io/include/aare/network\\_io/ZmqSocket.hpp](#)
- [network\\_io/src/ZmqSocket.cpp](#)



## 7.27 aare::ZmqSocketReceiver Class Reference

```
#include <ZmqSocketReceiver.hpp>
```

Inheritance diagram for aare::ZmqSocketReceiver:



### Public Member Functions

- [ZmqSocketReceiver](#) (const std::string &endpoint)  
*Construct a new [ZmqSocketReceiver](#) object.*
- void [connect](#) ()  
*Connect to the given endpoint subscribe to a Zmq published.*
- std::vector< [ZmqFrame](#) > [receive\\_n](#) ()
- void [disconnect](#) ()
- void [set\\_zmq\\_hwm](#) (int hwm)
- void [set\\_timeout\\_ms](#) (int n)
- void [set\\_potential\\_frame\\_size](#) (size\_t size)

### Protected Attributes

- void \* [m\\_context](#) {nullptr}
- void \* [m\\_socket](#) {nullptr}
- std::string [m\\_endpoint](#)
- int [m\\_zmq\\_hwm](#) {1000}
- int [m\\_timeout\\_ms](#) {1000}
- size\_t [m\\_potential\\_frame\\_size](#) {1024 \* 1024}
- char \* [m\\_header\\_buffer](#) = new char[[m\\_max\\_header\\_size](#)]

### Static Protected Attributes

- static constexpr size\_t [m\\_max\\_header\\_size](#) = 1024

### Private Member Functions

- int [receive\\_data](#) (std::byte \*data, size\_t size)  
*receive data following a [ZmqHeader](#)*
- [ZmqFrame](#) [receive\\_zmqframe](#) ()
- [ZmqHeader](#) [receive\\_header](#) ()  
*receive a [ZmqHeader](#)*

## 7.27.1 Constructor & Destructor Documentation

### 7.27.1.1 ZmqSocketReceiver()

```
aare::ZmqSocketReceiver::ZmqSocketReceiver (
    const std::string & endpoint )
```

Construct a new [ZmqSocketReceiver](#) object.

## 7.27.2 Member Function Documentation

### 7.27.2.1 connect()

```
void aare::ZmqSocketReceiver::connect ( )
```

Connect to the given endpoint subscribe to a Zmq published.

### 7.27.2.2 disconnect()

```
void aare::ZmqSocket::disconnect ( ) [inherited]
```

### 7.27.2.3 receive\_data()

```
int aare::ZmqSocketReceiver::receive_data (
    std::byte * data,
    size_t size ) [private]
```

receive data following a [ZmqHeader](#)

#### Parameters

<i>data</i>	pointer to data
<i>size</i>	size of data

#### Returns

[ZmqHeader](#)

### 7.27.2.4 receive\_header()

```
ZmqHeader aare::ZmqSocketReceiver::receive_header ( ) [private]
```

receive a [ZmqHeader](#)

#### Returns

[ZmqHeader](#)

### 7.27.2.5 receive\_n()

```
std::vector< ZmqFrame > aare::ZmqSocketReceiver::receive_n ( )
```

### 7.27.2.6 receive\_zmqframe()

```
ZmqFrame aare::ZmqSocketReceiver::receive_zmqframe ( ) [private]
```

### 7.27.2.7 set\_potential\_frame\_size()

```
void aare::ZmqSocket::set_potential_frame_size (
    size_t size ) [inherited]
```

### 7.27.2.8 set\_timeout\_ms()

```
void aare::ZmqSocket::set_timeout_ms (
    int n ) [inherited]
```

### 7.27.2.9 set\_zmq\_hwm()

```
void aare::ZmqSocket::set_zmq_hwm (
    int hwm ) [inherited]
```

## 7.27.3 Field Documentation

### 7.27.3.1 m\_context

```
void* aare::ZmqSocket::m_context {nullptr} [protected], [inherited]
```

### 7.27.3.2 m\_endpoint

```
std::string aare::ZmqSocket::m_endpoint [protected], [inherited]
```

### 7.27.3.3 m\_header\_buffer

```
char* aare::ZmqSocket::m_header_buffer = new char[m_max_header_size] [protected], [inherited]
```

### 7.27.3.4 m\_max\_header\_size

```
constexpr size_t aare::ZmqSocket::m_max_header_size = 1024 [static], [constexpr], [protected], [inherited]
```

### 7.27.3.5 m\_potential\_frame\_size

```
size_t aare::ZmqSocket::m_potential_frame_size {1024 * 1024} [protected], [inherited]
```

### 7.27.3.6 m\_socket

```
void* aare::ZmqSocket::m_socket {nullptr} [protected], [inherited]
```

### 7.27.3.7 m\_timeout\_ms

```
int aare::ZmqSocket::m_timeout_ms {1000} [protected], [inherited]
```

### 7.27.3.8 m\_zmq\_hwm

```
int aare::ZmqSocket::m_zmq_hwm {1000} [protected], [inherited]
```

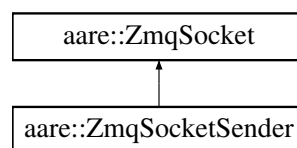
The documentation for this class was generated from the following files:

- [network\\_io/include/aare/network\\_io/ZmqSocketReceiver.hpp](#)
- [network\\_io/src/ZmqSocketReceiver.cpp](#)

## 7.28 aare::ZmqSocketSender Class Reference

```
#include <ZmqSocketSender.hpp>
```

Inheritance diagram for aare::ZmqSocketSender:



### Public Member Functions

- [ZmqSocketSender](#) (const std::string &endpoint)
- void [bind](#) ()
- size\_t [send](#) (const [ZmqHeader](#) &header, const std::byte \*data, size\_t size)
- size\_t [send](#) (const [ZmqFrame](#) &zmq\_frame)
- size\_t [send](#) (const std::vector< [ZmqFrame](#) > &zmq\_frames)
- void [disconnect](#) ()
- void [set\\_zmq\\_hwm](#) (int hwm)
- void [set\\_timeout\\_ms](#) (int n)
- void [set\\_potential\\_frame\\_size](#) (size\_t size)

### Protected Attributes

- void \* [m\\_context](#) {nullptr}
- void \* [m\\_socket](#) {nullptr}
- std::string [m\\_endpoint](#)
- int [m\\_zmq\\_hwm](#) {1000}
- int [m\\_timeout\\_ms](#) {1000}
- size\_t [m\\_potential\\_frame\\_size](#) {1024 \* 1024}
- char \* [m\\_header\\_buffer](#) = new char[[m\\_max\\_header\\_size](#)]

### Static Protected Attributes

- static constexpr size\_t [m\\_max\\_header\\_size](#) = 1024

## 7.28.1 Constructor & Destructor Documentation

### 7.28.1.1 ZmqSocketSender()

```
aare::ZmqSocketSender::ZmqSocketSender (
    const std::string & endpoint )
```

Constructor

Parameters

<i>endpoint</i>	ZMQ endpoint
-----------------	--------------

## 7.28.2 Member Function Documentation

### 7.28.2.1 bind()

```
void aare::ZmqSocketSender::bind ( )
```

bind to the given port

### 7.28.2.2 disconnect()

```
void aare::ZmqSocket::disconnect ( ) [inherited]
```

### 7.28.2.3 send() [1/3]

```
size_t aare::ZmqSocketSender::send (
    const std::vector< ZmqFrame > & zmq_frames )
```

Send a vector of headers and frames

## Parameters

<i>zmq_frames</i>	vector of <a href="#">ZmqFrame</a>
-------------------	------------------------------------

## Returns

number of bytes sent

**7.28.2.4 send() [2/3]**

```
size_t aare::ZmqSocketSender::send (
    const ZmqFrame & zmq_frame )
```

Send a frame with a header

## Parameters

<a href="#">ZmqFrame</a>	that contains a header and a frame
--------------------------	------------------------------------

## Returns

number of bytes sent

**7.28.2.5 send() [3/3]**

```
size_t aare::ZmqSocketSender::send (
    const ZmqHeader & header,
    const std::byte * data,
    size_t size )
```

send a header and data

## Parameters

<i>header</i>	
<i>data</i>	pointer to data
<i>size</i>	size of data

## Returns

number of bytes sent

**7.28.2.6 set\_potential\_frame\_size()**

```
void aare::ZmqSocket::set_potential_frame_size (
    size_t size ) [inherited]
```

### 7.28.2.7 set\_timeout\_ms()

```
void aare::ZmqSocket::set_timeout_ms (
    int n ) [inherited]
```

### 7.28.2.8 set\_zmq\_hwm()

```
void aare::ZmqSocket::set_zmq_hwm (
    int hwm ) [inherited]
```

## 7.28.3 Field Documentation

### 7.28.3.1 m\_context

```
void* aare::ZmqSocket::m_context {nullptr} [protected], [inherited]
```

### 7.28.3.2 m\_endpoint

```
std::string aare::ZmqSocket::m_endpoint [protected], [inherited]
```

### 7.28.3.3 m\_header\_buffer

```
char* aare::ZmqSocket::m_header_buffer = new char[m_max_header_size] [protected], [inherited]
```

### 7.28.3.4 m\_max\_header\_size

```
constexpr size_t aare::ZmqSocket::m_max_header_size = 1024 [static], [constexpr], [protected], [inherited]
```

### 7.28.3.5 m\_potential\_frame\_size

```
size_t aare::ZmqSocket::m_potential_frame_size {1024 * 1024} [protected], [inherited]
```

### 7.28.3.6 m\_socket

```
void* aare::ZmqSocket::m_socket {nullptr} [protected], [inherited]
```

### 7.28.3.7 m\_timeout\_ms

```
int aare::ZmqSocket::m_timeout_ms {1000} [protected], [inherited]
```

### 7.28.3.8 m\_zmq\_hwm

```
int aare::ZmqSocket::m_zmq_hwm {1000} [protected], [inherited]
```

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

- [network\\_io/include/aare/network\\_io/ZmqSocketSender.hpp](#)
- [network\\_io/src/ZmqSocketSender.cpp](#)





# Chapter 8

## File Documentation

### 8.1 core/include/aare/core/CircularFifo.hpp File Reference

```
#include <chrono>
#include <fmt/color.h>
#include <fmt/format.h>
#include <memory>
#include <thread>
#include "aare/core/ProducerConsumerQueue.hpp"
```

#### Data Structures

- class [aare::CircularFifo< ItemType >](#)

#### Namespaces

- namespace [aare](#)

*[Frame](#) class to represent a single frame of data model class should be able to work with streams coming from files or network.*

### 8.2 CircularFifo.hpp

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include <chrono>
00004 #include <fmt/color.h>
00005 #include <fmt/format.h>
00006 #include <memory>
00007 #include <thread>
00008
00009 #include "aare/core/ProducerConsumerQueue.hpp"
00010
00011 namespace aare {
00012
00013 template <class ItemType> class CircularFifo {
00014     uint32_t fifo_size;
00015     folly::ProducerConsumerQueue<ItemType> free_slots;
00016     folly::ProducerConsumerQueue<ItemType> filled_slots;
00017 }
```

```

00018 public:
00019     CircularFifo() : CircularFifo(100){};
00020     CircularFifo(uint32_t size) : fifo_size(size), free_slots(size + 1), filled_slots(size + 1) {
00021         // TODO! how do we deal with alignment for writing? alignas???
00022         // Do we give the user a chance to provide memory locations?
00023         // Templated allocator?
00024         for (size_t i = 0; i < fifo_size; ++i) {
00025             free_slots.write(ItemType{});
00026         }
00027     }
00028 }
00029
00030 bool next() {
00031     // TODO! avoid default constructing ItemType
00032     ItemType it;
00033     if (!filled_slots.read(it))
00034         return false;
00035     if (!free_slots.write(std::move(it)))
00036         return false;
00037     return true;
00038 }
00039
00040 ~CircularFifo() {}
00041
00042 using value_type = ItemType;
00043
00044 auto numFilledSlots() const noexcept { return filled_slots.sizeGuess(); }
00045 auto numFreeSlots() const noexcept { return free_slots.sizeGuess(); }
00046 auto isFull() const noexcept { return filled_slots.isFull(); }
00047
00048 ItemType pop_free() {
00049     ItemType v;
00050     while (!free_slots.read(v))
00051         ;
00052     return std::move(v);
00053     // return v;
00054 }
00055
00056 bool try_pop_free(ItemType &v) { return free_slots.read(v); }
00057
00058 ItemType pop_value(std::chrono::nanoseconds wait, std::atomic<bool> &stopped) {
00059     ItemType v;
00060     while (!filled_slots.read(v) && !stopped) {
00061         std::this_thread::sleep_for(wait);
00062     }
00063     return std::move(v);
00064 }
00065
00066 ItemType pop_value() {
00067     ItemType v;
00068     while (!filled_slots.read(v))
00069         ;
00070     return std::move(v);
00071 }
00072
00073 ItemType *frontPtr() { return filled_slots.frontPtr(); }
00074
00075 // TODO! Add function to move item from filled to free to be used
00076 // with the frontPtr function
00077
00078 template <class... Args> void push_value(Args &&...recordArgs) {
00079     while (!filled_slots.write(std::forward<Args>(recordArgs)...))
00080         ;
00081 }
00082
00083 template <class... Args> bool try_push_value(Args &&...recordArgs) {
00084     return filled_slots.write(std::forward<Args>(recordArgs)...);
00085 }
00086
00087 template <class... Args> void push_free(Args &&...recordArgs) {
00088     while (!free_slots.write(std::forward<Args>(recordArgs)...))
00089         ;
00090 }
00091
00092 template <class... Args> bool try_push_free(Args &&...recordArgs) {
00093     return free_slots.write(std::forward<Args>(recordArgs)...);
00094 }
00095 };
00096
00097 } // namespace aare

```

## 8.3 core/include/aare/core/defs.hpp File Reference

```
#include <array>
#include <stdexcept>
#include <cstdint>
#include <string>
#include <string_view>
#include <variant>
#include <vector>
```

### Data Structures

- struct [aare::sls\\_detector\\_header](#)
- struct [aare::xy](#)
- struct [aare::RawFileConfig](#)

### Namespaces

- namespace [aare](#)  
*[Frame](#) class to represent a single frame of data model class should be able to work with streams coming from files or network.*

### Typedefs

- using [aare::dynamic\\_shape](#) = std::vector< ssize\_t >
- using [aare::DataTypeVariants](#) = std::variant< uint16\_t, uint32\_t >

### Enumerations

- enum class [aare::DetectorType](#) {  
    [aare::Jungfrau](#) , [aare::Eiger](#) , [aare::Mythen3](#) , [aare::Moench](#) ,  
    [aare::ChipTestBoard](#) }
- enum class [aare::TimingMode](#) { [aare::Auto](#) , [aare::Trigger](#) }

### Functions

- template<class T >  
    T [aare::StringTo](#) (std::string sv)
- template<class T >  
    std::string [aare::toString](#) (T sv)
- template<> [DetectorType aare::StringTo](#) (std::string)
- template<> std::string [aare::toString](#) ([DetectorType](#) type)
- template<> [TimingMode aare::StringTo](#) (std::string)

## 8.4 defs.hpp

[Go to the documentation of this file.](#)

```

00001 #pragma once
00002
00003 #include <array>
00004 #include <stdexcept>
00005
00006 #include <cstdint>
00007 #include <string>
00008 #include <string_view>
00009 #include <variant>
00010 #include <vector>
00011
00012 namespace aare {
00013
00014 struct sls_detector_header {
00015     uint64_t frameNumber;
00016     uint32_t expLength;
00017     uint32_t packetNumber;
00018     uint64_t bunchId;
00019     uint64_t timestamp;
00020     uint16_t modId;
00021     uint16_t row;
00022     uint16_t column;
00023     uint16_t reserved;
00024     uint32_t debug;
00025     uint16_t roundRNumber;
00026     uint8_t detType;
00027     uint8_t version;
00028     uint8_t packetMask[64];
00029 };
00030
00031 struct xy {
00032     int row;
00033     int col;
00034     bool operator==(const xy &other) const { return row == other.row && col == other.col; }
00035     bool operator!=(const xy &other) const { return !(*this == other); }
00036 };
00037
00038 // using image_shape = std::array<ssize_t, 2>;
00039 using dynamic_shape = std::vector<ssize_t>;
00040
00041 enum class DetectorType { Jungfrau, Eiger, Mythen3, Moench, ChipTestBoard };
00042
00043 enum class TimingMode { Auto, Trigger };
00044
00045 template <class T> T StringTo(std::string sv) { return T(sv); }
00046
00047 template <class T> std::string toString(T sv) { return T(sv); }
00048
00049 template <> DetectorType StringTo(std::string);
00050 template <> std::string toString(DetectorType type);
00051
00052 template <> TimingMode StringTo(std::string);
00053
00054 using DataTypeVariants = std::variant<uint16_t, uint32_t>;
00055
00056 struct RawFileConfig {
00057     int module_gap_row{};
00058     int module_gap_col{};
00059
00060     bool operator==(const RawFileConfig &other) const {
00061         if (module_gap_col != other.module_gap_col)
00062             return false;
00063         if (module_gap_row != other.module_gap_row)
00064             return false;
00065         return true;
00066     }
00067 };
00068
00069 } // namespace aare

```

## 8.5 network\_io/include/aare/network\_io/defs.hpp File Reference

```

#include "aare/core/Frame.hpp"
#include "aare/network_io/ZmqHeader.hpp"
#include <stdexcept>
#include <string>

```

## Data Structures

- struct [aare::ZmqFrame](#)  
*ZmqFrame* structure wrapper class to contain a *ZmqHeader* and a *Frame*.
- class [aare::network\\_io::NetworkError](#)  
*NetworkError* exception class.

## Namespaces

- namespace [aare](#)  
*Frame* class to represent a single frame of data model class should be able to work with streams coming from files or network.
- namespace [aare::network\\_io](#)

## 8.6 defs.hpp

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002 #include "aare/core/Frame.hpp"
00003 #include "aare/network_io/ZmqHeader.hpp"
00004
00005 #include <stdexcept>
00006 #include <string>
00007
00008 namespace aare {
00013     struct ZmqFrame {
00014         ZmqHeader header;
00015         Frame frame;
00016     };
00017
00018 namespace network_io {
00022     class NetworkError : public std::runtime_error {
00023     private:
00024         const char *m_msg;
00025
00026     public:
00027         NetworkError(const char *msg) : std::runtime_error(msg), m_msg(msg) {}
00028         NetworkError(const std::string msg) : std::runtime_error(msg) { m_msg = strdup(msg.c_str()); }
00029         virtual const char *what() const noexcept override { return m_msg; }
00030     };
00031
00032 } // namespace network_io
00033
00034 } // namespace aare
```

## 8.7 core/include/aare/core/DType.hpp File Reference

```
#include <cstdint>
#include <string>
#include <typeinfo>
```

## Data Structures

- class [aare::DType](#)

## Namespaces

- namespace `aare`

*Frame* class to represent a single frame of data model class should be able to work with streams coming from files or network.

## Enumerations

- enum class `aare::endian` { `aare::little` = `__ORDER_LITTLE_ENDIAN__` , `aare::big` = `__ORDER_BIG_ENDIAN__` , `aare::native` = `__BYTE_ORDER__` }

## 8.8 DType.hpp

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002 #include <stdint>
00003 #include <string>
00004 #include <typeinfo>
00005
00006 namespace aare {
00007
00008     enum class endian {
00009 #ifdef _WIN32
00010         little = 0,
00011         big = 1,
00012         native = little
00013 #else
00014         little = __ORDER_LITTLE_ENDIAN__,
00015         big = __ORDER_BIG_ENDIAN__,
00016         native = __BYTE_ORDER__
00017 #endif
00018     };
00019
00020     class DType {
00021     // TODO! support for non native endianess?
00022         static_assert(sizeof(long) == sizeof(int64_t), "long should be 64bits");
00023
00024     public:
00025         enum TypeIndex { INT8, UINT8, INT16, UINT16, INT32, UINT32, INT64, UINT64, FLOAT, DOUBLE, ERROR };
00026
00027         uint8_t bitdepth() const;
00028
00029         explicit DType(const std::type_info &t);
00030         explicit DType(std::string_view sv);
00031
00032         // not explicit to allow conversions form enum to DType
00033         DType(DType::TypeIndex ti);
00034
00035         bool operator==(const DType &other) const noexcept;
00036         bool operator!=(const DType &other) const noexcept;
00037         bool operator==(const std::type_info &t) const;
00038         bool operator!=(const std::type_info &t) const;
00039
00040         // bool operator==(DType::TypeIndex ti) const;
00041         // bool operator!=(DType::TypeIndex ti) const;
00042         std::string str() const;
00043
00044     private:
00045         TypeIndex m_type{TypeIndex::ERROR};
00046     };
00047
00048 } // namespace aare
```

## 8.9 core/include/aare/core/Frame.hpp File Reference

```
#include "aare/core/NDAarray.hpp"
#include "aare/core/defs.hpp"
#include <cstdint>
```

```
#include <stdint>
#include <memory>
#include <sys/types.h>
#include <vector>
```

## Data Structures

- class [aare::Frame](#)

## Namespaces

- namespace [aare](#)

*Frame* class to represent a single frame of data model class should be able to work with streams coming from files or network.

## 8.10 Frame.hpp

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002 #include "aare/core/NDAarray.hpp"
00003 #include "aare/core/defs.hpp"
00004 #include <cstdint>
00005 #include <stdint>
00006 #include <memory>
00007 #include <sys/types.h>
00008 #include <vector>
00009
00016 namespace aare {
00017
00018 class Frame {
00019     ssize_t m_rows;
00020     ssize_t m_cols;
00021     ssize_t m_bitdepth;
00022     std::byte *m_data;
00023
00024 public:
00025     Frame(ssize_t rows, ssize_t cols, ssize_t m_bitdepth);
00026     Frame(std::byte *fp, ssize_t rows, ssize_t cols, ssize_t m_bitdepth);
00027     std::byte *get(int row, int col);
00028
00029     // TODO! can we, or even want to remove the template?
00030     template <typename T> void set(int row, int col, T data) {
00031         assert(sizeof(T) == m_bitdepth / 8);
00032         if (row < 0 || row >= m_rows || col < 0 || col >= m_cols) {
00033             throw std::out_of_range("Invalid row or column index");
00034         }
00035         std::memcpy(m_data + (row * m_cols + col) * (m_bitdepth / 8), &data, m_bitdepth / 8);
00036     }
00037
00038     ssize_t rows() const { return m_rows; }
00039     ssize_t cols() const { return m_cols; }
00040     ssize_t bitdepth() const { return m_bitdepth; }
00041     ssize_t size() const { return m_rows * m_cols * m_bitdepth / 8; }
00042     std::byte *data() const { return m_data; }
00043
00044     Frame &operator=(Frame &other) {
00045         m_rows = other.rows();
00046         m_cols = other.cols();
00047         m_bitdepth = other.bitdepth();
00048         m_data = new std::byte[m_rows * m_cols * m_bitdepth / 8];
00049         std::memcpy(m_data, other.m_data, m_rows * m_cols * m_bitdepth / 8);
00050         return *this;
00051     }
00052     // add move constructor
00053     Frame(Frame &&other) {
00054         m_rows = other.rows();
00055         m_cols = other.cols();
00056         m_bitdepth = other.bitdepth();
00057         m_data = other.m_data;
```

```

00058         other.m_data = nullptr;
00059         other.m_rows = other.m_cols = other.m_bitdepth = 0;
00060     }
00061     // copy constructor
00062     Frame(const Frame &other) {
00063         m_rows = other.rows();
00064         m_cols = other.cols();
00065         m_bitdepth = other.bitdepth();
00066         m_data = new std::byte[m_rows * m_cols * m_bitdepth / 8];
00067         std::memcpy(m_data, other.m_data, m_rows * m_cols * m_bitdepth / 8);
00068     }
00069
00070     template <typename T> NDView<T> view() {
00071         std::vector<ssize_t> shape = {m_rows, m_cols};
00072         T *data = reinterpret_cast<T *>(m_data);
00073         return NDView<T>(data, shape);
00074     }
00075
00076     template <typename T> NDArray<T> image() { return NDArray<T>(this->view<T>()); }
00077
00078     ~Frame() { delete[] m_data; }
00079 };
00080
00081 } // namespace aare

```

## 8.11 core/include/aare/core/NDArray.hpp File Reference

```

#include "aare/core/NDView.hpp"
#include <algorithm>
#include <array>
#include <cmath>
#include <fmt/format.h>
#include <fstream>
#include <iomanip>
#include <iostream>
#include <numeric>

```

### Data Structures

- class [aare::NDArray< T, Ndim >](#)

### Namespaces

- namespace [aare](#)

*Frame* class to represent a single frame of data model class should be able to work with streams coming from files or network.

### Functions

- template<typename T, ssize\_t Ndim>  
void [aare::save](#) (NDArray< T, Ndim > &img, std::string pathname)
- template<typename T, ssize\_t Ndim>  
[NDArray< T, Ndim > aare::load](#) (const std::string &pathname, std::array< ssize\_t, Ndim > shape)



## 8.12 NDArray.hpp

[Go to the documentation of this file.](#)

```

00001 #pragma once
00002 /*
00003 Container holding image data, or a time series of image data in contiguous
00004 memory.
00005
00006
00007 TODO! Add expression templates for operators
00008
00009 */
00010 #include "aare/core/NDView.hpp"
00011
00012 #include <algorithm>
00013 #include <array>
00014 #include <cmath>
00015 #include <fmt/format.h>
00016 #include <fstream>
00017 #include <iomanip>
00018 #include <iostream>
00019 #include <numeric>
00020
00021 namespace aare {
00022
00023 template <typename T, ssize_t Ndim = 2> class NDArray {
00024 public:
00025     NDArray() : shape_(), strides_(c_strides<Ndim>(shape_)), size_(0), data_(nullptr){};
00026
00027     explicit NDArray(std::array<ssize_t, Ndim> shape)
00028         : shape_(shape), strides_(c_strides<Ndim>(shape_)),
00029           size_(std::accumulate(shape_.begin(), shape_.end(), 1, std::multiplies<ssize_t>())),
00030           data_(new T[size_]){};
00031
00032     NDArray(std::array<ssize_t, Ndim> shape, T value) : NDArray(shape) { this->operator=(value); }
00033
00034     /* When constructing from a NDView we need to copy the data since
00035     NDArray expect to own its data, and span is just a view*/
00036     NDArray(NDView<T, Ndim> span) : NDArray(span.shape()) {
00037         std::copy(span.begin(), span.end(), begin());
00038         // fmt::print("NDArray(NDView<T, Ndim> span)\n");
00039     }
00040
00041     // Move constructor
00042     NDArray(NDArray &&other)
00043         : shape_(other.shape_), strides_(c_strides<Ndim>(shape_)), size_(other.size_), data_(nullptr)
00044     {
00045         data_ = other.data_;
00046         other.reset();
00047         // fmt::print("NDArray(NDArray &&other)\n");
00048     }
00049
00050     // Copy constructor
00051     NDArray(const NDArray &other)
00052         : shape_(other.shape_), strides_(c_strides<Ndim>(shape_)), size_(other.size_), data_(new
00053           T[size_]) {
00054         std::copy(other.data_, other.data_ + size_, data_);
00055         // fmt::print("NDArray(const NDArray &other)\n");
00056     }
00057
00058     ~NDArray() { delete[] data_; }
00059
00060     auto begin() { return data_; }
00061     auto end() { return data_ + size_; }
00062
00063     using value_type = T;
00064
00065     NDArray &operator=(NDArray &&other); // Move assign
00066     NDArray &operator=(const NDArray &other); // Copy assign
00067
00068     NDArray operator+(const NDArray &other);
00069     NDArray operator+=(const NDArray &other);
00070     NDArray operator-(const NDArray &other);
00071     NDArray operator-=(const NDArray &other);
00072     NDArray operator*(const NDArray &other);
00073     NDArray operator*=(const NDArray &other);
00074     NDArray operator/(const NDArray &other);
00075     // NDArray& operator/=(const NDArray& other);
00076     template <typename V> NDArray &operator/=(const NDArray<V, Ndim> &other) {
00077         // check shape
00078         if (shape_ == other.shape()) {
00079             for (int i = 0; i < size_; ++i) {
00080                 data_[i] /= other(i);
00081             }
00082             return *this;
00083         }
00084     }

```

```

00080         } else {
00081             throw(std::runtime_error("Shape of NDAarray must match"));
00082         }
00083     }
00084
00085     NDAarray<bool, Ndim> operator>(const NDAarray &other);
00086
00087     bool operator==(const NDAarray &other) const;
00088     bool operator!=(const NDAarray &other) const;
00089
00090     NDAarray &operator=(const T &);
00091     NDAarray &operator+=(const T &);
00092     NDAarray operator+(const T &);
00093     NDAarray &operator-=(const T &);
00094     NDAarray operator-(const T &);
00095     NDAarray &operator*=(const T &);
00096     NDAarray operator*(const T &);
00097     NDAarray &operator/=(const T &);
00098     NDAarray operator/(const T &);
00099
00100     NDAarray &operator&=(const T &);
00101
00102     void sqrt() {
00103         for (int i = 0; i < size_; ++i) {
00104             data_[i] = std::sqrt(data_[i]);
00105         }
00106     }
00107
00108     NDAarray &operator++(); // pre inc
00109
00110     template <typename... Ix> typename std::enable_if<sizeof...(Ix) == Ndim, T &>::type
00111     operator()(Ix... index) {
00112         return data_[element_offset(strides_, index...)];
00113     }
00114
00115     template <typename... Ix> typename std::enable_if<sizeof...(Ix) == Ndim, T &>::type
00116     operator()(Ix... index) const {
00117         return data_[element_offset(strides_, index...)];
00118     }
00119
00120     template <typename... Ix> typename std::enable_if<sizeof...(Ix) == Ndim, T>::type value(Ix...
00121     index) {
00122         return data_[element_offset(strides_, index...)];
00123     }
00124
00125     T &operator()(int i) { return data_[i]; }
00126     const T &operator()(int i) const { return data_[i]; }
00127
00128     T *data() { return data_; }
00129     std::byte *buffer() { return reinterpret_cast<std::byte *>(data_); }
00130     ssize_t size() const { return size_; }
00131     size_t total_bytes() const { return size_ * sizeof(T); }
00132     std::array<ssize_t, Ndim> shape() const noexcept { return shape_; }
00133     ssize_t shape(ssize_t i) const noexcept { return shape_[i]; }
00134     std::array<ssize_t, Ndim> strides() const noexcept { return strides_; }
00135     std::array<ssize_t, Ndim> byte_strides() const noexcept {
00136         auto byte_strides = strides_;
00137         for (auto &val : byte_strides)
00138             val *= sizeof(T);
00139         return byte_strides;
00140     }
00141     // return strides_;
00142
00143     NDView<T, Ndim> span() const { return NDView<T, Ndim>{data_, shape_}; }
00144
00145     void Print();
00146     void Print_all();
00147     void Print_some();
00148
00149     void reset() {
00150         data_ = nullptr;
00151         size_ = 0;
00152         std::fill(shape_.begin(), shape_.end(), 0);
00153         std::fill(strides_.begin(), strides_.end(), 0);
00154     }
00155
00156 private:
00157     std::array<ssize_t, Ndim> shape_;
00158     std::array<ssize_t, Ndim> strides_;
00159     ssize_t size_;
00160     T *data_;
00161 };
00162
00163 // Move assign
00164 template <typename T, ssize_t Ndim> NDAarray<T, Ndim> &NDAarray<T, Ndim>::operator=(NDAarray<T, Ndim>
00165 &&other) {
00166     if (this != &other) {

```

```

00163         delete[] data_;
00164         data_ = other.data_;
00165         shape_ = other.shape_;
00166         size_ = other.size_;
00167         strides_ = other.strides_;
00168         other.reset();
00169     }
00170     return *this;
00171 }
00172
00173 template <typename T, ssize_t Ndim> NDArray<T, Ndim> NDArray<T, Ndim>::operator+(const NDArray &other)
00174 {
00175     NDArray result(*this);
00176     result += other;
00177     return result;
00178 }
00179 template <typename T, ssize_t Ndim> NDArray<T, Ndim> &NDArray<T, Ndim>::operator+=(const
00180 NDArray<T, Ndim> &other) {
00181     // check shape
00182     if (shape_ == other.shape_) {
00183         for (int i = 0; i < size_; ++i) {
00184             data_[i] += other.data_[i];
00185         }
00186         return *this;
00187     } else {
00188         throw(std::runtime_error("Shape of ImageDatas must match"));
00189     }
00190 }
00191 template <typename T, ssize_t Ndim> NDArray<T, Ndim> NDArray<T, Ndim>::operator-(const NDArray &other)
00192 {
00193     NDArray result(*this);
00194     result -= other;
00195     return result;
00196 }
00197 template <typename T, ssize_t Ndim> NDArray<T, Ndim> &NDArray<T, Ndim>::operator-=(const
00198 NDArray<T, Ndim> &other) {
00199     // check shape
00200     if (shape_ == other.shape_) {
00201         for (int i = 0; i < size_; ++i) {
00202             data_[i] -= other.data_[i];
00203         }
00204         return *this;
00205     } else {
00206         throw(std::runtime_error("Shape of ImageDatas must match"));
00207     }
00208 }
00209 template <typename T, ssize_t Ndim> NDArray<T, Ndim> NDArray<T, Ndim>::operator*(const NDArray &other)
00210 {
00211     NDArray result = *this;
00212     result *= other;
00213     return result;
00214 }
00215 template <typename T, ssize_t Ndim> NDArray<T, Ndim> &NDArray<T, Ndim>::operator*=(const
00216 NDArray<T, Ndim> &other) {
00217     // check shape
00218     if (shape_ == other.shape_) {
00219         for (int i = 0; i < size_; ++i) {
00220             data_[i] *= other.data_[i];
00221         }
00222         return *this;
00223     } else {
00224         throw(std::runtime_error("Shape of ImageDatas must match"));
00225     }
00226 }
00227 template <typename T, ssize_t Ndim> NDArray<T, Ndim> NDArray<T, Ndim>::operator/(const NDArray &other)
00228 {
00229     NDArray result = *this;
00230     result /= other;
00231     return result;
00232 }
00233 template <typename T, ssize_t Ndim> NDArray<T, Ndim> &NDArray<T, Ndim>::operator&=(const T &mask) {
00234     for (auto it = begin(); it != end(); ++it)
00235         *it &= mask;
00236     return *this;
00237 }
00238 // template <typename T, ssize_t Ndim>
00239 // NDArray<T, Ndim> &NDArray<T, Ndim>::operator/=(const NDArray<T, Ndim> &
00240 // other)
00241 // {
00242 //     //check shape
00243 //     if (shape_ == other.shape_) {

```

```

00243 //         for (int i = 0; i < size_; ++i) {
00244 //             data_[i] /= other.data_[i];
00245 //         }
00246 //         return *this;
00247 //     } else {
00248 //         throw(std::runtime_error("Shape of ImageDatas must match"));
00249 //     }
00250 // }
00251
00252 template <typename T, ssize_t Ndim> NDAarray<bool, Ndim> NDAarray<T, Ndim>::operator>(const NDAarray
&other) {
00253     if (shape_ == other.shape_) {
00254         NDAarray<bool> result(shape_);
00255         for (int i = 0; i < size_; ++i) {
00256             result(i) = (data_[i] > other.data_[i]);
00257         }
00258         return result;
00259     } else {
00260         throw(std::runtime_error("Shape of ImageDatas must match"));
00261     }
00262 }
00263
00264 template <typename T, ssize_t Ndim> NDAarray<T, Ndim> &NDAarray<T, Ndim>::operator=(const
NDAarray<T, Ndim> &other) {
00265     if (this != &other) {
00266         delete[] data_;
00267         shape_ = other.shape_;
00268         strides_ = other.strides_;
00269         size_ = other.size_;
00270         data_ = new T[size_];
00271         std::copy(other.data_, other.data_ + size_, data_);
00272     }
00273     return *this;
00274 }
00275
00276 template <typename T, ssize_t Ndim> bool NDAarray<T, Ndim>::operator==(const NDAarray<T, Ndim> &other)
const {
00277     if (shape_ != other.shape_)
00278         return false;
00279
00280     for (int i = 0; i != size_; ++i)
00281         if (data_[i] != other.data_[i])
00282             return false;
00283
00284     return true;
00285 }
00286
00287 template <typename T, ssize_t Ndim> bool NDAarray<T, Ndim>::operator!=(const NDAarray<T, Ndim> &other)
const {
00288     return !((*this) == other);
00289 }
00290 template <typename T, ssize_t Ndim> NDAarray<T, Ndim> &NDAarray<T, Ndim>::operator++() {
00291     for (int i = 0; i < size_; ++i)
00292         data_[i] += 1;
00293     return *this;
00294 }
00295 template <typename T, ssize_t Ndim> NDAarray<T, Ndim> &NDAarray<T, Ndim>::operator=(const T &value) {
00296     std::fill_n(data_, size_, value);
00297     return *this;
00298 }
00299
00300 template <typename T, ssize_t Ndim> NDAarray<T, Ndim> &NDAarray<T, Ndim>::operator+=(const T &value) {
00301     for (int i = 0; i < size_; ++i)
00302         data_[i] += value;
00303     return *this;
00304 }
00305
00306 template <typename T, ssize_t Ndim> NDAarray<T, Ndim> NDAarray<T, Ndim>::operator+(const T &value) {
00307     NDAarray result = *this;
00308     result += value;
00309     return result;
00310 }
00311 template <typename T, ssize_t Ndim> NDAarray<T, Ndim> &NDAarray<T, Ndim>::operator-=(const T &value) {
00312     for (int i = 0; i < size_; ++i)
00313         data_[i] -= value;
00314     return *this;
00315 }
00316 template <typename T, ssize_t Ndim> NDAarray<T, Ndim> NDAarray<T, Ndim>::operator-(const T &value) {
00317     NDAarray result = *this;
00318     result -= value;
00319     return result;
00320 }
00321
00322 template <typename T, ssize_t Ndim> NDAarray<T, Ndim> &NDAarray<T, Ndim>::operator/=(const T &value) {
00323     for (int i = 0; i < size_; ++i)
00324         data_[i] /= value;
00325     return *this;

```

```

00326 }
00327 template <typename T, ssize_t Ndim> NDArray<T, Ndim> NDArray<T, Ndim>::operator/(const T &value) {
00328     NDArray result = *this;
00329     result /= value;
00330     return result;
00331 }
00332 template <typename T, ssize_t Ndim> NDArray<T, Ndim> &NDArray<T, Ndim>::operator*=(const T &value) {
00333     for (int i = 0; i < size_; ++i)
00334         data_[i] *= value;
00335     return *this;
00336 }
00337 template <typename T, ssize_t Ndim> NDArray<T, Ndim> NDArray<T, Ndim>::operator*(const T &value) {
00338     NDArray result = *this;
00339     result *= value;
00340     return result;
00341 }
00342 template <typename T, ssize_t Ndim> void NDArray<T, Ndim>::Print() {
00343     if (shape_[0] < 20 && shape_[1] < 20)
00344         Print_all();
00345     else
00346         Print_some();
00347 }
00348 template <typename T, ssize_t Ndim> void NDArray<T, Ndim>::Print_all() {
00349     for (auto row = 0; row < shape_[0]; ++row) {
00350         for (auto col = 0; col < shape_[1]; ++col) {
00351             std::cout << std::setw(3);
00352             std::cout << (*this)(row, col) << " ";
00353         }
00354         std::cout << "\n";
00355     }
00356 }
00357 template <typename T, ssize_t Ndim> void NDArray<T, Ndim>::Print_some() {
00358     for (auto row = 0; row < 5; ++row) {
00359         for (auto col = 0; col < 5; ++col) {
00360             std::cout << std::setw(7);
00361             std::cout << (*this)(row, col) << " ";
00362         }
00363         std::cout << "\n";
00364     }
00365 }
00366
00367 template <typename T, ssize_t Ndim> void save(NDArray<T, Ndim> &img, std::string pathname) {
00368     std::ofstream f;
00369     f.open(pathname, std::ios::binary);
00370     f.write(img.buffer(), img.size() * sizeof(T));
00371     f.close();
00372 }
00373
00374 template <typename T, ssize_t Ndim>
00375 NDArray<T, Ndim> load(const std::string &pathname, std::array<ssize_t, Ndim> shape) {
00376     NDArray<T, Ndim> img{shape};
00377     std::ifstream f;
00378     f.open(pathname, std::ios::binary);
00379     f.read(img.buffer(), img.size() * sizeof(T));
00380     f.close();
00381     return img;
00382 }
00383
00384 } // namespace aare

```

## 8.13 core/include/aare/core/NDView.hpp File Reference

```

#include <algorithm>
#include <array>
#include <cassert>
#include <cstdint>
#include <numeric>
#include <stdexcept>
#include <vector>

```

### Data Structures

- class [aare::NDView< T, Ndim >](#)

## Namespaces

- namespace `aare`

*Frame* class to represent a single frame of data model class should be able to work with streams coming from files or network.

## Typedefs

- `template<ssize_t Ndim>`  
using `aare::Shape` = `std::array< ssize_t, Ndim >`

## Functions

- `template<ssize_t Ndim>`  
`Shape< Ndim > aare::make_shape` (const `std::vector< size_t > &shape`)
- `template<ssize_t Dim = 0, typename Strides >`  
`ssize_t aare::element_offset` (const `Strides &`)
- `template<ssize_t Dim = 0, typename Strides, typename... Ix>`  
`ssize_t aare::element_offset` (const `Strides &strides, ssize_t i, Ix... index`)
- `template<ssize_t Ndim>`  
`std::array< ssize_t, Ndim > aare::c_strides` (const `std::array< ssize_t, Ndim > &shape`)
- `template<ssize_t Ndim>`  
`std::array< ssize_t, Ndim > aare::make_array` (const `std::vector< ssize_t > &vec`)

## 8.14 NDView.hpp

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002 #include <algorithm>
00003 #include <array>
00004 #include <cassert>
00005 #include <cstdint>
00006 #include <numeric>
00007 #include <stdexcept>
00008 #include <vector>
00009
00010 namespace aare {
00011
00012 template <ssize_t Ndim> using Shape = std::array<ssize_t, Ndim>;
00013
00014 // TODO! fix mismatch between signed and unsigned
00015 template <ssize_t Ndim> Shape<Ndim> make_shape(const std::vector<size_t> &shape) {
00016     if (shape.size() != Ndim)
00017         throw std::runtime_error("Shape size mismatch");
00018     Shape<Ndim> arr;
00019     std::copy_n(shape.begin(), Ndim, arr.begin());
00020     return arr;
00021 }
00022
00023 template <ssize_t Dim = 0, typename Strides> ssize_t element_offset(const Strides &) { return 0; }
00024
00025 template <ssize_t Dim = 0, typename Strides, typename... Ix>
00026 ssize_t element_offset(const Strides &strides, ssize_t i, Ix... index) {
00027     return i * strides[Dim] + element_offset<Dim + 1>(strides, index...);
00028 }
00029
00030 template <ssize_t Ndim> std::array<ssize_t, Ndim> c_strides(const std::array<ssize_t, Ndim> &shape) {
00031     std::array<ssize_t, Ndim> strides;
00032     std::fill(strides.begin(), strides.end(), 1);
00033     for (ssize_t i = Ndim - 1; i > 0; --i) {
00034         strides[i - 1] = strides[i] * shape[i];
00035     }
00036     return strides;
00037 }
00038
00039 template <ssize_t Ndim> std::array<ssize_t, Ndim> make_array(const std::vector<ssize_t> &vec) {
```

```

00040     assert(vec.size() == Ndim);
00041     std::array<ssize_t, Ndim> arr;
00042     std::copy_n(vec.begin(), Ndim, arr.begin());
00043     return arr;
00044 }
00045
00046 template <typename T, ssize_t Ndim = 2> class NDView {
00047 public:
00048     NDView(){};
00049
00050     NDView(T *buffer, std::array<ssize_t, Ndim> shape) {
00051         buffer_ = buffer;
00052         strides_ = c_strides<Ndim>(shape);
00053         shape_ = shape;
00054         size_ = std::accumulate(std::begin(shape), std::end(shape), 1, std::multiplies<ssize_t>());
00055     }
00056
00057     NDView(T *buffer, const std::vector<ssize_t> &shape) {
00058         buffer_ = buffer;
00059         strides_ = c_strides<Ndim>(make_array<Ndim>(shape));
00060         shape_ = make_array<Ndim>(shape);
00061         size_ = std::accumulate(std::begin(shape), std::end(shape), 1, std::multiplies<ssize_t>());
00062     }
00063
00064     template <typename... Ix> typename std::enable_if<sizeof...(Ix) == Ndim, T &::type>
00065     operator()(Ix... index) {
00066         return buffer_[element_offset(strides_, index...)];
00067     }
00068
00069     template <typename... Ix> typename std::enable_if<sizeof...(Ix) == Ndim, T &::type>
00070     operator()(Ix... index) const {
00071         return buffer_[element_offset(strides_, index...)];
00072     }
00073
00074     ssize_t size() const { return size_; }
00075
00076     NDView(const NDView &) = default;
00077     NDView(NDView &&) = default;
00078
00079     T *begin() { return buffer_; }
00080     T *end() { return buffer_ + size_; }
00081     T &operator()(ssize_t i) { return buffer_[i]; }
00082     T &operator[] (ssize_t i) { return buffer_[i]; }
00083
00084     bool operator==(const NDView &other) const {
00085         if (size_ != other.size_)
00086             return false;
00087         for (ssize_t i = 0; i != size_; ++i) {
00088             if (buffer_[i] != other.buffer_[i])
00089                 return false;
00090         }
00091         return true;
00092     }
00093
00094     NDView &operator+=(const T val) { return elementwise(val, std::plus<T>()); }
00095     NDView &operator-=(const T val) { return elementwise(val, std::minus<T>()); }
00096     NDView &operator*=(const T val) { return elementwise(val, std::multiplies<T>()); }
00097     NDView &operator/=(const T val) { return elementwise(val, std::divides<T>()); }
00098
00099     NDView &operator/=(const NDView &other) { return elementwise(other, std::divides<T>()); }
00100
00101     NDView &operator=(const T val) {
00102         for (auto it = begin(); it != end(); ++it)
00103             *it = val;
00104         return *this;
00105     }
00106
00107     NDView &operator=(const NDView &other) {
00108         shape_ = other.shape_;
00109         strides_ = other.strides_;
00110         size_ = other.size_;
00111         buffer_ = other.buffer_;
00112         return *this;
00113     }
00114
00115     auto &shape() { return shape_; }
00116     auto shape(ssize_t i) const { return shape_[i]; }
00117
00118     T *data() { return buffer_; }
00119
00120 private:
00121     T *buffer_{nullptr};
00122     std::array<ssize_t, Ndim> strides_{};
00123     std::array<ssize_t, Ndim> shape_{};
00124     ssize_t size_{};
00125
00126     template <class BinaryOperation> NDView &elementwise(T val, BinaryOperation op) {
00127         for (ssize_t i = 0; i != size_; ++i) {

```

```

00125         buffer_[i] = op(buffer_[i], val);
00126     }
00127     return *this;
00128 }
00129 template <class BinaryOperation> NDView &elementwise(const NDView &other, BinaryOperation op) {
00130     for (ssize_t i = 0; i != size_; ++i) {
00131         buffer_[i] = op(buffer_[i], other.buffer_[i]);
00132     }
00133     return *this;
00134 }
00135 };
00136
00137 template class NDView<uint16_t, 2>;
00138
00139 } // namespace aare

```

## 8.15 core/include/aare/core/ProducerConsumerQueue.hpp File Reference

```

#include <atomic>
#include <cassert>
#include <cstdlib>
#include <memory>
#include <stdexcept>
#include <type_traits>
#include <utility>

```

### Data Structures

- struct [folly::ProducerConsumerQueue< T >](#)

### Namespaces

- namespace [folly](#)

### Variables

- constexpr std::size\_t [hardware\\_destructive\\_interference\\_size](#) = 128

### 8.15.1 Variable Documentation

#### 8.15.1.1 hardware\_destructive\_interference\_size

```
constexpr std::size_t hardware_destructive_interference_size = 128 [constexpr]
```



## 8.16 ProducerConsumerQueue.hpp

[Go to the documentation of this file.](#)

```

00001 /*
00002  * Copyright (c) Meta Platforms, Inc. and affiliates.
00003  *
00004  * Licensed under the Apache License, Version 2.0 (the "License");
00005  * you may not use this file except in compliance with the License.
00006  * You may obtain a copy of the License at
00007  *
00008  * http://www.apache.org/licenses/LICENSE-2.0
00009  *
00010  * Unless required by applicable law or agreed to in writing, software
00011  * distributed under the License is distributed on an "AS IS" BASIS,
00012  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
00013  * See the license for the specific language governing permissions and
00014  * limitations under the License.
00015  */
00016
00017 // @author Bo Hu (bhu@fb.com)
00018 // @author Jordan DeLong (delong.j@fb.com)
00019
00020 // Changes made by PSD Detector Group:
00021 // Copied: Line 34 constexpr std::size_t hardware_destructive_interference_size = 128; from
    folly/lang/Align.h
00022 // Changed extension to .hpp
00023
00024 #pragma once
00025
00026 #include <atomic>
00027 #include <cassert>
00028 #include <cstdlib>
00029 #include <memory>
00030 #include <stdexcept>
00031 #include <type_traits>
00032 #include <utility>
00033
00034 constexpr std::size_t hardware_destructive_interference_size = 128;
00035 namespace folly {
00036
00037 /*
00038  * ProducerConsumerQueue is a one producer and one consumer queue
00039  * without locks.
00040  */
00041 template <class T> struct ProducerConsumerQueue {
00042     typedef T value_type;
00043
00044     ProducerConsumerQueue(const ProducerConsumerQueue &) = delete;
00045     ProducerConsumerQueue &operator=(const ProducerConsumerQueue &) = delete;
00046
00047     // size must be >= 2.
00048     //
00049     // Also, note that the number of usable slots in the queue at any
00050     // given time is actually (size-1), so if you start with an empty queue,
00051     // isFull() will return true after size-1 insertions.
00052     explicit ProducerConsumerQueue(uint32_t size)
00053         : size_(size), records_(static_cast<T *>(std::malloc(sizeof(T) * size))), readIndex_(0),
    writeIndex_(0) {
00054         assert(size >= 2);
00055         if (!records_) {
00056             throw std::bad_alloc();
00057         }
00058     }
00059
00060     ~ProducerConsumerQueue() {
00061         // We need to destruct anything that may still exist in our queue.
00062         // (No real synchronization needed at destructor time: only one
00063         // thread can be doing this.)
00064         if (!std::is_trivially_destructible<T::value>) {
00065             size_t readIndex = readIndex_;
00066             size_t endIndex = writeIndex_;
00067             while (readIndex != endIndex) {
00068                 records_[readIndex].~T();
00069                 if (++readIndex == size_) {
00070                     readIndex = 0;
00071                 }
00072             }
00073         }
00074
00075         std::free(records_);
00076     }
00077
00078     template <class... Args> bool write(Args &&...recordArgs) {
00079         auto const currentWrite = writeIndex_.load(std::memory_order_relaxed);
00080         auto nextRecord = currentWrite + 1;

```

```

00081         if (nextRecord == size_) {
00082             nextRecord = 0;
00083         }
00084         if (nextRecord != readIndex_.load(std::memory_order_acquire)) {
00085             new (&records_[currentWrite]) T(std::forward<Args>(recordArgs)...);
00086             writeIndex_.store(nextRecord, std::memory_order_release);
00087             return true;
00088         }
00089
00090         // queue is full
00091         return false;
00092     }
00093
00094     // move (or copy) the value at the front of the queue to given variable
00095     bool read(T &record) {
00096         auto const currentRead = readIndex_.load(std::memory_order_relaxed);
00097         if (currentRead == writeIndex_.load(std::memory_order_acquire)) {
00098             // queue is empty
00099             return false;
00100         }
00101
00102         auto nextRecord = currentRead + 1;
00103         if (nextRecord == size_) {
00104             nextRecord = 0;
00105         }
00106         record = std::move(records_[currentRead]);
00107         records_[currentRead].~T();
00108         readIndex_.store(nextRecord, std::memory_order_release);
00109         return true;
00110     }
00111
00112     // pointer to the value at the front of the queue (for use in-place) or
00113     // nullptr if empty.
00114     T *frontPtr() {
00115         auto const currentRead = readIndex_.load(std::memory_order_relaxed);
00116         if (currentRead == writeIndex_.load(std::memory_order_acquire)) {
00117             // queue is empty
00118             return nullptr;
00119         }
00120         return &records_[currentRead];
00121     }
00122
00123     // queue must not be empty
00124     void popFront() {
00125         auto const currentRead = readIndex_.load(std::memory_order_relaxed);
00126         assert(currentRead != writeIndex_.load(std::memory_order_acquire));
00127
00128         auto nextRecord = currentRead + 1;
00129         if (nextRecord == size_) {
00130             nextRecord = 0;
00131         }
00132         records_[currentRead].~T();
00133         readIndex_.store(nextRecord, std::memory_order_release);
00134     }
00135
00136     bool isEmpty() const {
00137         return readIndex_.load(std::memory_order_acquire) ==
00138             writeIndex_.load(std::memory_order_acquire);
00139     }
00140
00141     bool isFull() const {
00142         auto nextRecord = writeIndex_.load(std::memory_order_acquire) + 1;
00143         if (nextRecord == size_) {
00144             nextRecord = 0;
00145         }
00146         if (nextRecord != readIndex_.load(std::memory_order_acquire)) {
00147             return false;
00148         }
00149         // queue is full
00150         return true;
00151     }
00152
00153     // * If called by consumer, then true size may be more (because producer may
00154     //   be adding items concurrently).
00155     // * If called by producer, then true size may be less (because consumer may
00156     //   be removing items concurrently).
00157     // * It is undefined to call this from any other thread.
00158     size_t sizeGuess() const {
00159         int ret = writeIndex_.load(std::memory_order_acquire) -
00160             readIndex_.load(std::memory_order_acquire);
00161         if (ret < 0) {
00162             ret += size_;
00163         }
00164         return ret;
00165     }
00166
00167     // maximum number of items in the queue.

```

```

00166     size_t capacity() const { return size_ - 1; }
00167
00168 private:
00169     using AtomicIndex = std::atomic<unsigned int>;
00170
00171     char pad0_[hardware_destructive_interference_size];
00172     const uint32_t size_;
00173     T *const records_;
00174
00175     alignas(hardware_destructive_interference_size) AtomicIndex readIndex_;
00176     alignas(hardware_destructive_interference_size) AtomicIndex writeIndex_;
00177
00178     char pad1_[hardware_destructive_interference_size - sizeof(AtomicIndex)];
00179 };
00180
00181 } // namespace folly

```

## 8.17 core/include/aare/core/VariableSizeClusterFinder.hpp File Reference

```

#include <algorithm>
#include <map>
#include <unordered_map>
#include <vector>
#include "aare/core/NDArray.hpp"

```

### Data Structures

- class [aare::ClusterFinder< T >](#)
- struct [aare::ClusterFinder< T >::Hit](#)

### Namespaces

- namespace [aare](#)

*Frame class to represent a single frame of data model class should be able to work with streams coming from files or network.*

### Variables

- const int [MAX\\_CLUSTER\\_SIZE](#) = 200

## 8.17.1 Variable Documentation

### 8.17.1.1 MAX\_CLUSTER\_SIZE

```
const int MAX_CLUSTER_SIZE = 200
```

## 8.18 VariableSizeClusterFinder.hpp

[Go to the documentation of this file.](#)

```

00001 #pragma once
00002
00003 #include <algorithm>
00004 #include <map>
00005 #include <unordered_map>
00006 #include <vector>
00007
00008 #include "aare/core/NDArray.hpp"
00009
00010 const int MAX_CLUSTER_SIZE = 200;
00011 namespace aare {
00012
00013 template <typename T> class ClusterFinder {
00014 public:
00015     struct Hit {
00016         int16_t size{};
00017         int16_t row{};
00018         int16_t col{};
00019         uint16_t reserved{}; // for alignment
00020         T energy{};
00021         T max{};
00022
00023         // std::vector<int16_t> rows{};
00024         // std::vector<int16_t> cols{};
00025         int16_t rows[MAX_CLUSTER_SIZE] = {0};
00026         int16_t cols[MAX_CLUSTER_SIZE] = {0};
00027         double enes[MAX_CLUSTER_SIZE] = {0};
00028     };
00029
00030 private:
00031     const std::array<ssize_t, 2> shape_;
00032     NDView<T, 2> original_;
00033     NDArray<int, 2> labeled_;
00034     NDArray<int, 2> peripheral_labeled_;
00035     NDArray<bool, 2> binary_; // over threshold flag
00036     T threshold_;
00037     NDView<T, 2> noiseMap_;
00038     bool use_noise_map = false;
00039     int peripheralThresholdFactor_ = 5;
00040     int current_label;
00041     const std::array<int, 4> di_{{0, -1, -1, -1}}; // row ### 8-neighbour by scanning from
left to right
00042     const std::array<int, 4> dj_{{-1, -1, 0, 1}}; // col ### 8-neighbour by scanning from
top to bottom
00043     const std::array<int, 8> di_{{0, 0, -1, 1, -1, 1, -1, 1}}; // row
00044     const std::array<int, 8> dj_{{-1, 1, 0, 0, 1, -1, -1, 1}}; // col
00045     std::map<int, int> child; // heirachy: key: child; val: parent
00046     std::unordered_map<int, Hit> h_size;
00047     std::vector<Hit> hits;
00048     // std::vector<std::vector<int16_t>> row
00049     int check_neighbours(int i, int j);
00050
00051 public:
00052     ClusterFinder(image_shape shape, T threshold)
00053         : shape_(shape), labeled_(shape, 0), peripheral_labeled_(shape, 0), binary_(shape),
threshold_(threshold) {
00054         hits.reserve(2000);
00055     }
00056
00057     NDArray<int, 2> labeled() { return labeled_; }
00058
00059     void set_noiseMap(NDView<T, 2> noise_map) {
00060         noiseMap_ = noise_map;
00061         use_noise_map = true;
00062     }
00063     void set_peripheralThresholdFactor(int factor) { peripheralThresholdFactor_ = factor; }
00064     void find_clusters(NDView<T, 2> img);
00065     void find_clusters_X(NDView<T, 2> img);
00066     void rec_FillHit(int clusterIndex, int i, int j);
00067     void single_pass(NDView<T, 2> img);
00068     void first_pass();
00069     void second_pass();
00070     void store_clusters();
00071
00072     std::vector<Hit> steal_hits() {
00073         std::vector<Hit> tmp;
00074         std::swap(tmp, hits);
00075         return tmp;
00076     };
00077     void clear_hits() { hits.clear(); };
00078
00079     void print_connections() {

```

```

00080         fmt::print("Connections:\n");
00081         for (auto it = child.begin(); it != child.end(); ++it) {
00082             fmt::print("{} -> {}\n", it->first, it->second);
00083         }
00084     }
00085     size_t total_clusters() const {
00086         // TODO! fix for stealing
00087         return hits.size();
00088     }
00089 private:
00090     void add_link(int from, int to) {
00091         // we want to add key from -> value to
00092         // fmt::print("add_link({}, {})\n", from, to);
00093         auto it = child.find(from);
00094         if (it == child.end()) {
00095             child[from] = to;
00096         } else {
00097             // found need to disambiguate
00098             if (it->second == to)
00099                 return;
00100             else {
00101                 if (it->second > to) {
00102                     // child[from] = to;
00103                     auto old = it->second;
00104                     it->second = to;
00105                     add_link(old, to);
00106                 } else {
00107                     // found value is smaller than what we want to link
00108                     add_link(to, it->second);
00109                 }
00110             }
00111         }
00112     }
00113 }
00114 };
00115 template <typename T> int ClusterFinder<T>::check_neighbours(int i, int j) {
00116     std::vector<int> neighbour_labels;
00117     for (int k = 0; k < 4; ++k) {
00118         const auto row = i + di[k];
00119         const auto col = j + dj[k];
00120         if (row >= 0 && col >= 0 && row < shape_[0] && col < shape_[1]) {
00121             auto tmp = labeled_.value(i + di[k], j + dj[k]);
00122             if (tmp != 0)
00123                 neighbour_labels.push_back(tmp);
00124         }
00125     }
00126 }
00127
00128 if (neighbour_labels.size() == 0) {
00129     return 0;
00130 } else {
00131     // need to sort and add to union field
00132     std::sort(neighbour_labels.rbegin(), neighbour_labels.rend());
00133     auto first = neighbour_labels.begin();
00134     auto last = std::unique(first, neighbour_labels.end());
00135     if (last - first == 1)
00136         return *neighbour_labels.begin();
00137     for (auto current = first; current != last - 1; ++current) {
00138         auto next = current + 1;
00139         add_link(*current, *next);
00140     }
00141     return neighbour_labels.back(); // already sorted
00142 }
00143 }
00144 }
00145 }
00146
00147 template <typename T> void ClusterFinder<T>::find_clusters(NDView<T, 2> img) {
00148     original_ = img;
00149     labeled_ = 0;
00150     peripheral_labeled_ = 0;
00151     current_label = 0;
00152     child.clear();
00153     first_pass();
00154     // print_connections();
00155     second_pass();
00156     store_clusters();
00157 }
00158
00159 template <typename T> void ClusterFinder<T>::find_clusters_X(NDView<T, 2> img) {
00160     original_ = img;
00161     int clusterIndex = 0;
00162     for (int i = 0; i < shape_[0]; ++i) {
00163         for (int j = 0; j < shape_[1]; ++j) {
00164             if (use_noise_map)
00165                 threshold_ = 5 * noiseMap(i, j);
00166             if (original_(i, j) > threshold_) {

```

```

00167         // printf("==== Cluster index: %d\n", clusterIndex);
00168         rec_FillHit(clusterIndex, i, j);
00169         clusterIndex++;
00170     }
00171 }
00172 }
00173 for (const auto &h : h_size)
00174     hits.push_back(h.second);
00175 h_size.clear();
00176 }
00177
00178 template <typename T> void ClusterFinder<T>::rec_FillHit(int clusterIndex, int i, int j) {
00179     // printf("original_(%d, %d)=%f\n", i, j, original_(i,j));
00180     // printf("h_size[%d].size=%d\n", clusterIndex, h_size[clusterIndex].size);
00181     if (h_size[clusterIndex].size < MAX_CLUSTER_SIZE) {
00182         h_size[clusterIndex].rows[h_size[clusterIndex].size] = i;
00183         h_size[clusterIndex].cols[h_size[clusterIndex].size] = j;
00184         h_size[clusterIndex].enes[h_size[clusterIndex].size] = original_(i, j);
00185     }
00186     h_size[clusterIndex].size += 1;
00187     h_size[clusterIndex].energy += original_(i, j);
00188     if (h_size[clusterIndex].max < original_(i, j)) {
00189         h_size[clusterIndex].row = i;
00190         h_size[clusterIndex].col = j;
00191         h_size[clusterIndex].max = original_(i, j);
00192     }
00193     original_(i, j) = 0;
00194
00195     for (int k = 0; k < 8; ++k) { // 8 for 8-neighbour
00196         const auto row = i + di_[k];
00197         const auto col = j + dj_[k];
00198         if (row >= 0 && col >= 0 && row < shape_[0] && col < shape_[1]) {
00199             if (use_noise_map)
00200                 threshold_ = peripheralThresholdFactor_ * noiseMap(row, col);
00201             if (original_(row, col) > threshold_) {
00202                 rec_FillHit(clusterIndex, row, col);
00203             } else {
00204                 // if (h_size[clusterIndex].size < MAX_CLUSTER_SIZE){
00205                 //     h_size[clusterIndex].size += 1;
00206                 //     h_size[clusterIndex].rows[h_size[clusterIndex].size] = row;
00207                 //     h_size[clusterIndex].cols[h_size[clusterIndex].size] = col;
00208                 //     h_size[clusterIndex].enes[h_size[clusterIndex].size] = original_(row, col);
00209                 // } // ? weather to include peripheral pixels
00210                 original_(row, col) = 0; // remove peripheral pixels, to avoid potential influence for
00211                 pedestal updating
00212             }
00213         }
00214     }
00215
00216 template <typename T> void ClusterFinder<T>::single_pass(NDView<T, 2> img) {
00217     original_ = img;
00218     labeled_ = 0;
00219     current_label = 0;
00220     child.clear();
00221     first_pass();
00222     // print_connections();
00223     // second_pass();
00224     // store_clusters();
00225 }
00226
00227 template <typename T> void ClusterFinder<T>::first_pass() {
00228
00229     for (int i = 0; i < original_.size(); ++i) {
00230         if (use_noise_map)
00231             threshold_ = 5 * noiseMap(i);
00232         binary_(i) = (original_(i) > threshold_);
00233     }
00234
00235     for (int i = 0; i < shape_[0]; ++i) {
00236         for (int j = 0; j < shape_[1]; ++j) {
00237
00238             // do we have something to process?
00239             if (binary_(i, j)) {
00240                 auto tmp = check_neighbours(i, j);
00241                 if (tmp != 0) {
00242                     labeled_(i, j) = tmp;
00243                 } else {
00244                     labeled_(i, j) = ++current_label;
00245                 }
00246             }
00247         }
00248     }
00249 }
00250
00251 template <typename T> void ClusterFinder<T>::second_pass() {
00252

```

```

00253     for (ssize_t i = 0; i != labeled_.size(); ++i) {
00254         auto current_label = labeled_(i);
00255         if (current_label != 0) {
00256             auto it = child.find(current_label);
00257             while (it != child.end()) {
00258                 current_label = it->second;
00259                 it = child.find(current_label);
00260                 // do this once before doing the second pass?
00261                 // all values point to the final one...
00262             }
00263             labeled_(i) = current_label;
00264         }
00265     }
00266 }
00267
00268 template <typename T> void ClusterFinder<T>::store_clusters() {
00269
00270     // Accumulate hit information in a map
00271     // Do we always have monotonic increasing
00272     // labels? Then vector?
00273     // here the translation is label -> Hit
00274     std::unordered_map<int, Hit> h_size;
00275     for (int i = 0; i < shape_[0]; ++i) {
00276         for (int j = 0; j < shape_[1]; ++j) {
00277             if (labeled_(i, j) != 0 or false
00278                 // (i-1 >= 0 and labeled_(i-1, j) != 0) or // another circle of peripheral pixels
00279                 // (j-1 >= 0 and labeled_(i, j-1) != 0) or
00280                 // (i+1 < shape_[0] and labeled_(i+1, j) != 0) or
00281                 // (j+1 < shape_[1] and labeled_(i, j+1) != 0)
00282             ) {
00283                 Hit &record = h_size[labeled_(i, j)];
00284                 if (record.size < MAX_CLUSTER_SIZE) {
00285                     record.rows[record.size] = i;
00286                     record.cols[record.size] = j;
00287                     record.enes[record.size] = original_(i, j);
00288                 } else {
00289                     continue;
00290                 }
00291                 record.size += 1;
00292                 record.energy += original_(i, j);
00293
00294                 if (record.max < original_(i, j)) {
00295                     record.row = i;
00296                     record.col = j;
00297                     record.max = original_(i, j);
00298                 }
00299             }
00300         }
00301     }
00302
00303     for (const auto &h : h_size)
00304         hits.push_back(h.second);
00305 }
00306
00307 } // namespace aare

```

## 8.19 core/src/defs.cpp File Reference

#include "aare/core/defs.hpp"

### Namespaces

- namespace [aare](#)

*Frame* class to represent a single frame of data model class should be able to work with streams coming from files or network.

### Functions

- template<> std::string [aare::toString](#) (DetectorType type)
- template<> [DetectorType aare::StringTo](#) (std::string name)
- template<> [TimingMode aare::StringTo](#) (std::string mode)

## 8.20 core/src/DType.cpp File Reference

```
#include "aare/core/DType.hpp"
#include "aare/utils/logger.hpp"
#include <fmt/format.h>
```

### Namespaces

- namespace [aare](#)

*[Frame](#) class to represent a single frame of data model class should be able to work with streams coming from files or network.*

## 8.21 core/src/Frame.cpp File Reference

```
#include "aare/core/Frame.hpp"
#include "aare/utils/logger.hpp"
#include <cassert>
#include <iostream>
```

### Namespaces

- namespace [aare](#)

*[Frame](#) class to represent a single frame of data model class should be able to work with streams coming from files or network.*

## 8.22 core/test/CircularFifo.test.cpp File Reference

```
#include <catch2/catch_all.hpp>
#include "aare/core/CircularFifo.hpp"
```

### Data Structures

- struct [MoveOnlyInt](#)

### Functions

- [TEST\\_CASE](#) ("CircularFifo can be default constructed")
- [TEST\\_CASE](#) ("Newly constructed fifo has the right size")
- [TEST\\_CASE](#) ("Can fit size number of objects")
- [TEST\\_CASE](#) ("Push move only type")
- [TEST\\_CASE](#) ("Push pop")
- [TEST\\_CASE](#) ("Pop free and then push")
- [TEST\\_CASE](#) ("Skip the first value")
- [TEST\\_CASE](#) ("Use in place and move to free")



## 8.22.1 Function Documentation

### 8.22.1.1 TEST\_CASE() [1/8]

```
TEST_CASE (
    "Can fit size number of objects" )
```

### 8.22.1.2 TEST\_CASE() [2/8]

```
TEST_CASE (
    "CircularFifo can be default constructed" )
```

### 8.22.1.3 TEST\_CASE() [3/8]

```
TEST_CASE (
    "Newly constructed fifo has the right size" )
```

### 8.22.1.4 TEST\_CASE() [4/8]

```
TEST_CASE (
    "Pop free and then push" )
```

### 8.22.1.5 TEST\_CASE() [5/8]

```
TEST_CASE (
    "Push move only type" )
```

### 8.22.1.6 TEST\_CASE() [6/8]

```
TEST_CASE (
    "Push pop" )
```

### 8.22.1.7 TEST\_CASE() [7/8]

```
TEST_CASE (
    "Skip the first value" )
```

### 8.22.1.8 TEST\_CASE() [8/8]

```
TEST_CASE (
    "Use in place and move to free" )
```

## 8.23 core/test/defs.test.cpp File Reference

```
#include "aare/core/defs.hpp"
#include <catch2/catch_test_macros.hpp>
#include <string>
```

### Functions

- [TEST\\_CASE](#) ("Enum to string conversion")

### 8.23.1 Function Documentation

#### 8.23.1.1 TEST\_CASE()

```
TEST_CASE (
    "Enum to string conversion" )
```

## 8.24 core/test/DType.test.cpp File Reference

```
#include "aare/core/DType.hpp"
#include <catch2/catch_test_macros.hpp>
```

### Functions

- [TEST\\_CASE](#) ("Construct from typeid")
- [TEST\\_CASE](#) ("Construct from string")
- [TEST\\_CASE](#) ("Construct from string with endianness")
- [TEST\\_CASE](#) ("Convert to string")

### 8.24.1 Function Documentation

#### 8.24.1.1 TEST\_CASE() [1/4]

```
TEST_CASE (
    "Construct from string with endianness" )
```

#### 8.24.1.2 TEST\_CASE() [2/4]

```
TEST_CASE (
    "Construct from string" )
```

#### 8.24.1.3 TEST\_CASE() [3/4]

```
TEST_CASE (
    "Construct from typeid" )
```

#### 8.24.1.4 TEST\_CASE() [4/4]

```
TEST_CASE (
    "Convert to string" )
```

## 8.25 core/test/Frame.test.cpp File Reference

```
#include "aare/core/Frame.hpp"
#include <catch2/catch_test_macros.hpp>
```

### Functions

- [TEST\\_CASE](#) ("Construct a frame")
- [TEST\\_CASE](#) ("Set a value in a 8 bit frame")
- [TEST\\_CASE](#) ("Set a value in a 64 bit frame")
- [TEST\\_CASE](#) ("Move construct a frame")

### 8.25.1 Function Documentation

#### 8.25.1.1 TEST\_CASE() [1/4]

```
TEST_CASE (
    "Construct a frame" )
```

#### 8.25.1.2 TEST\_CASE() [2/4]

```
TEST_CASE (
    "Move construct a frame" )
```

#### 8.25.1.3 TEST\_CASE() [3/4]

```
TEST_CASE (
    "Set a value in a 64 bit frame" )
```

#### 8.25.1.4 TEST\_CASE() [4/4]

```
TEST_CASE (
    "Set a value in a 8 bit frame" )
```

## 8.26 core/test/NDArray.test.cpp File Reference

```
#include "aare/core/NDArray.hpp"
#include <array>
#include <catch2/catch_test_macros.hpp>
```

### Functions

- [TEST\\_CASE](#) ("Initial size is zero if no size is specified")
- [TEST\\_CASE](#) ("Construct from a DataSpan")
- [TEST\\_CASE](#) ("1D image")
- [TEST\\_CASE](#) ("Accessing a const object")
- [TEST\\_CASE](#) ("Indexing of a 2D image")
- [TEST\\_CASE](#) ("Indexing of a 3D image")
- [TEST\\_CASE](#) ("Divide double by int")
- [TEST\\_CASE](#) ("Elementwise multiplication of 3D image")
- [TEST\\_CASE](#) ("Compare two images")
- [TEST\\_CASE](#) ("Size and shape matches")
- [TEST\\_CASE](#) ("Initial value matches for all elements")
- [TEST\\_CASE](#) ("Data layout of 3D image, fast index last")
- [TEST\\_CASE](#) ("Bitwise and on data")
- [TEST\\_CASE](#) ("Elementwise operations on images")

### 8.26.1 Function Documentation

#### 8.26.1.1 TEST\_CASE() [1/14]

```
TEST_CASE (
    "1D image" )
```

#### 8.26.1.2 TEST\_CASE() [2/14]

```
TEST_CASE (
    "Accessing a const object" )
```

#### 8.26.1.3 TEST\_CASE() [3/14]

```
TEST_CASE (
    "Bitwise and on data" )
```

#### 8.26.1.4 TEST\_CASE() [4/14]

```
TEST_CASE (
    "Compare two images" )
```

**8.26.1.5 TEST\_CASE()** [5/14]

```
TEST_CASE (
    "Construct from a DataSpan" )
```

**8.26.1.6 TEST\_CASE()** [6/14]

```
TEST_CASE (
    "Data layout of 3D image,
    fast index last" )
```

**8.26.1.7 TEST\_CASE()** [7/14]

```
TEST_CASE (
    "Divide double by int" )
```

**8.26.1.8 TEST\_CASE()** [8/14]

```
TEST_CASE (
    "Elementwise multiplication of 3D image" )
```

**8.26.1.9 TEST\_CASE()** [9/14]

```
TEST_CASE (
    "Elementwise operations on images" )
```

**8.26.1.10 TEST\_CASE()** [10/14]

```
TEST_CASE (
    "Indexing of a 2D image" )
```

**8.26.1.11 TEST\_CASE()** [11/14]

```
TEST_CASE (
    "Indexing of a 3D image" )
```

**8.26.1.12 TEST\_CASE()** [12/14]

```
TEST_CASE (
    "Initial size is zero if no size is specified" )
```

**8.26.1.13 TEST\_CASE()** [13/14]

```
TEST_CASE (
    "Initial value matches for all elements" )
```

#### 8.26.1.14 TEST\_CASE() [14/14]

```
TEST_CASE (
    "Size and shape matches" )
```

### 8.27 core/test/NDView.test.cpp File Reference

```
#include "aare/core/NDView.hpp"
#include <catch2/catch_test_macros.hpp>
#include <iostream>
#include <vector>
```

#### Functions

- [TEST\\_CASE](#) ("Element reference 1D")
- [TEST\\_CASE](#) ("Element reference 2D")
- [TEST\\_CASE](#) ("Element reference 3D")
- [TEST\\_CASE](#) ("Plus and miuns with single value")
- [TEST\\_CASE](#) ("Multiply and divide with single value")
- [TEST\\_CASE](#) ("elementwise assign")
- [TEST\\_CASE](#) ("iterators")
- [TEST\\_CASE](#) ("shape from vector")
- [TEST\\_CASE](#) ("divide with another span")
- [TEST\\_CASE](#) ("Retrieve shape")
- [TEST\\_CASE](#) ("compare two views")

#### 8.27.1 Function Documentation

##### 8.27.1.1 TEST\_CASE() [1/11]

```
TEST_CASE (
    "compare two views" )
```

##### 8.27.1.2 TEST\_CASE() [2/11]

```
TEST_CASE (
    "divide with another span" )
```

##### 8.27.1.3 TEST\_CASE() [3/11]

```
TEST_CASE (
    "Element reference 1D" )
```

**8.27.1.4 TEST\_CASE()** [4/11]

```
TEST_CASE (
    "Element reference 2D" )
```

**8.27.1.5 TEST\_CASE()** [5/11]

```
TEST_CASE (
    "Element reference 3D" )
```

**8.27.1.6 TEST\_CASE()** [6/11]

```
TEST_CASE (
    "elementwise assign" )
```

**8.27.1.7 TEST\_CASE()** [7/11]

```
TEST_CASE (
    "iterators" )
```

**8.27.1.8 TEST\_CASE()** [8/11]

```
TEST_CASE (
    "Multiply and divide with single value" )
```

**8.27.1.9 TEST\_CASE()** [9/11]

```
TEST_CASE (
    "Plus and miuns with single value" )
```

**8.27.1.10 TEST\_CASE()** [10/11]

```
TEST_CASE (
    "Retrieve shape" )
```

**8.27.1.11 TEST\_CASE()** [11/11]

```
TEST_CASE (
    "shape from vector" )
```

**8.28 core/test/ProducerConsumerQueue.test.cpp File Reference**

```
#include "aare/core/ProducerConsumerQueue.hpp"
#include <catch2/catch_all.hpp>
```

## Functions

- [TEST\\_CASE](#) ("push pop")
- [TEST\\_CASE](#) ("Cannot push to a full queue")
- [TEST\\_CASE](#) ("Cannot pop from an empty queue")

### 8.28.1 Function Documentation

#### 8.28.1.1 TEST\_CASE() [1/3]

```
TEST_CASE (
    "Cannot pop from an empty queue" )
```

#### 8.28.1.2 TEST\_CASE() [2/3]

```
TEST_CASE (
    "Cannot push to a full queue" )
```

#### 8.28.1.3 TEST\_CASE() [3/3]

```
TEST_CASE (
    "push pop" )
```

## 8.29 core/test/wrappers.test.cpp File Reference

```
#include <aare/core/Frame.hpp>
#include <aare/core/NDView.hpp>
#include <catch2/catch_test_macros.hpp>
#include <cstdint>
```

## Functions

- [TEST\\_CASE](#) ("Frame")
- [TEST\\_CASE](#) ("NDView")
- [TEST\\_CASE](#) ("NDArray")

### 8.29.1 Function Documentation

#### 8.29.1.1 TEST\_CASE() [1/3]

```
TEST_CASE (
    "Frame" )
```



### 8.29.1.2 TEST\_CASE() [2/3]

```
TEST_CASE (
    "NDArray" )
```

### 8.29.1.3 TEST\_CASE() [3/3]

```
TEST_CASE (
    "NDView" )
```

## 8.30 data/jungfrau/read\_frame.py File Reference

### Namespaces

- namespace [read\\_frame](#)

### Variables

- [read\\_frame.header\\_dt](#)
- int [read\\_frame.rows](#) = 512
- int [read\\_frame.cols](#) = 1024
- int [read\\_frame.frames](#) = 10
- [read\\_frame.data](#) = np.zeros((frames,rows,cols), dtype = np.uint16)
- [read\\_frame.header](#) = np.zeros(frames, dtype = header\_dt)
- str [read\\_frame.file\\_name](#) = 'jungfrau\_single\_d0\_f{}\_0.raw'.format(file\_id)
- [read\\_frame.f](#)
- [read\\_frame.dtype](#)
- [read\\_frame.count](#)
- [read\\_frame.uint16](#)

## 8.31 python/example/read\_frame.py File Reference

### Namespaces

- namespace [example](#)
- namespace [example.read\\_frame](#)

### Variables

- [example.read\\_frame.root\\_dir](#) = Path(os.environ.get("PROJECT\_ROOT\_DIR"))
- [example.read\\_frame.data\\_path](#) = str([root\\_dir](#) / "data"/"jungfrau\_single\_master\_0.json")
- [example.read\\_frame.file](#) = [File](#)([data\\_path](#))
- [example.read\\_frame.frame](#) = [file.get\\_frame](#)(0)
- [example.read\\_frame.arr](#) = np.array([frame.get\\_array](#)())

## 8.32 data/numpy/write\_test\_files.py File Reference

### Namespaces

- namespace [write\\_test\\_files](#)

### Variables

- [write\\_test\\_files.arr](#) = np.arange(10, dtype = np.int32)
- [write\\_test\\_files.arr2](#) = np.zeros((3,2,5), dtype = np.float64)

## 8.33 data/scripts/read\_first\_frame\_number.py File Reference

### Namespaces

- namespace [read\\_first\\_frame\\_number](#)

### Variables

- [read\\_first\\_frame\\_number.header\\_dt](#)
- [read\\_first\\_frame\\_number.frame\\_number](#) = np.fromfile(f, dtype=[header\\_dt](#), count=1)[["Frame Number"](#)][0]

## 8.34 data/jungfrau/read\_multiport.py File Reference

### Namespaces

- namespace [read\\_multiport](#)

### Variables

- [read\\_multiport.header\\_dt](#)
- int [read\\_multiport.frames](#) = 1
- int [read\\_multiport.parts](#) = 2
- int [read\\_multiport.frame\\_cols](#) = 1024
- int [read\\_multiport.frame\\_rows](#) = 512
- int [read\\_multiport.part\\_cols](#) = 1024
- int [read\\_multiport.part\\_rows](#) = 256
- [read\\_multiport.parts\\_data](#) = np.zeros((frames,parts,part\_rows,part\_cols), dtype = np.uint16)
- [read\\_multiport.data](#) = np.zeros((frames,frame\_rows,frame\_cols), dtype = np.uint16)
- [read\\_multiport.header](#) = np.zeros((frames,parts), dtype = [header\\_dt](#))
- str [read\\_multiport.file\\_name](#) = f'jungfrau\_double\_d{part}\_f{frame}\_{0}.raw'
- [read\\_multiport.f](#)
- [read\\_multiport.dtype](#)
- [read\\_multiport.count](#)
- [read\\_multiport.uint16](#)
- [read\\_multiport.axis](#)

## 8.35 data/scripts/read\_multiport.py File Reference

### Namespaces

- namespace [read\\_multiport](#)

## 8.36 examples/json\_example.cpp File Reference

```
#include "aare/file_io/File.hpp"
#include "aare/utils/logger.hpp"
#include <iostream>
```

### Macros

- #define [AARE\\_ROOT\\_DIR\\_VAR](#) "PROJECT\_ROOT\_DIR"

### Functions

- void [test](#) ([File](#) &f, int frame\_number)
- int [main](#) ()

### 8.36.1 Macro Definition Documentation

#### 8.36.1.1 AARE\_ROOT\_DIR\_VAR

```
#define AARE_ROOT_DIR_VAR "PROJECT_ROOT_DIR"
```

### 8.36.2 Function Documentation

#### 8.36.2.1 main()

```
int main ( )
```

#### 8.36.2.2 test()

```
void test (
    File & f,
    int frame_number )
```

## 8.37 examples/logger\_example.cpp File Reference

```
#include "aare/utils/logger.hpp"
#include <fstream>
#include <iostream>
```

## Functions

- int [main](#) ()

### 8.37.1 Function Documentation

#### 8.37.1.1 main()

```
int main ( )
```

## 8.38 examples/multiport\_example.cpp File Reference

```
#include "aare/file_io/File.hpp"  
#include "aare/utils/logger.hpp"  
#include <iostream>
```

## Macros

- #define [AARE\\_ROOT\\_DIR\\_VAR](#) "PROJECT\_ROOT\_DIR"

## Functions

- void [test](#) ([File](#) &*f*, int *frame\_number*)
- int [main](#) ()

### 8.38.1 Macro Definition Documentation

#### 8.38.1.1 AARE\_ROOT\_DIR\_VAR

```
#define AARE_ROOT_DIR_VAR "PROJECT_ROOT_DIR"
```

### 8.38.2 Function Documentation

#### 8.38.2.1 main()

```
int main ( )
```

#### 8.38.2.2 test()

```
void test (  
    File & f,  
    int frame_number )
```

## 8.39 examples/mythen\_example.cpp File Reference

```
#include "aare/file_io/File.hpp"
#include "aare/utils/logger.hpp"
#include <iostream>
```

### Macros

- #define [AARE\\_ROOT\\_DIR\\_VAR](#) "PROJECT\_ROOT\_DIR"

### Functions

- void [test1](#) ([File](#) &*f*, int *frame\_number*)
- void [test2](#) ([File](#) &*f*, int *frame\_number*)
- int [main](#) ()

### 8.39.1 Macro Definition Documentation

#### 8.39.1.1 AARE\_ROOT\_DIR\_VAR

```
#define AARE_ROOT_DIR_VAR "PROJECT_ROOT_DIR"
```

### 8.39.2 Function Documentation

#### 8.39.2.1 main()

```
int main ( )
```

#### 8.39.2.2 test1()

```
void test1 (
    File & f,
    int frame_number )
```

#### 8.39.2.3 test2()

```
void test2 (
    File & f,
    int frame_number )
```

## 8.40 examples/numpy\_read\_example.cpp File Reference

```
#include "aare/file_io/File.hpp"
#include <iostream>
```

## Macros

- `#define AARE_ROOT_DIR_VAR "PROJECT_ROOT_DIR"`

## Functions

- `void test (File &f, int frame_number)`
- `int main ()`

## 8.40.1 Macro Definition Documentation

### 8.40.1.1 AARE\_ROOT\_DIR\_VAR

```
#define AARE_ROOT_DIR_VAR "PROJECT_ROOT_DIR"
```

## 8.40.2 Function Documentation

### 8.40.2.1 main()

```
int main ( )
```

### 8.40.2.2 test()

```
void test (
    File & f,
    int frame_number )
```

## 8.41 examples/numpy\_write\_example.cpp File Reference

```
#include "aare/core/Frame.hpp"
#include "aare/file_io/File.hpp"
#include <iostream>
```

## Macros

- `#define AARE_ROOT_DIR_VAR "PROJECT_ROOT_DIR"`

## Functions

- `int main ()`

## 8.41.1 Macro Definition Documentation

### 8.41.1.1 AARE\_ROOT\_DIR\_VAR

```
#define AARE_ROOT_DIR_VAR "PROJECT_ROOT_DIR"
```

## 8.41.2 Function Documentation

### 8.41.2.1 main()

```
int main ( )
```

## 8.42 examples/raw\_example.cpp File Reference

```
#include "aare/file_io/File.hpp"  
#include "aare/utils/logger.hpp"  
#include <iostream>
```

### Macros

- #define [AARE\\_ROOT\\_DIR\\_VAR](#) "PROJECT\_ROOT\_DIR"

### Functions

- void [test](#) ([File](#) &*f*, int *frame\_number*)
- int [main](#) ()

## 8.42.1 Macro Definition Documentation

### 8.42.1.1 AARE\_ROOT\_DIR\_VAR

```
#define AARE_ROOT_DIR_VAR "PROJECT_ROOT_DIR"
```

## 8.42.2 Function Documentation

### 8.42.2.1 main()

```
int main ( )
```

### 8.42.2.2 test()

```
void test (  
    File & f,  
    int frame_number )
```

## 8.43 examples/zmq\_receiver\_example.cpp File Reference

```
#include "aare/network_io/ZmqSocketReceiver.hpp"
#include "aare/network_io/defs.hpp"
#include <cassert>
#include <fmt/core.h>
#include <string>
```

### Functions

- int [main](#) ()

#### 8.43.1 Function Documentation

##### 8.43.1.1 main()

```
int main ( )
```

## 8.44 examples/zmq\_restream\_example.cpp File Reference

```
#include <chrono>
#include <thread>
#include "aare/file_io/File.hpp"
#include "aare/network_io/ZmqSocketSender.hpp"
#include <boost/program_options.hpp>
```

### Functions

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

#### 8.44.1 Function Documentation

##### 8.44.1.1 main()

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



## 8.45 examples/zmq\_sender\_example.cpp File Reference

```
#include "aare/core/Frame.hpp"
#include "aare/network_io/ZmqHeader.hpp"
#include "aare/network_io/ZmqSocketSender.hpp"
#include "aare/network_io/defs.hpp"
#include "aare/utils/logger.hpp"
#include <ctime>
#include <fmt/core.h>
#include <string>
#include <unistd.h>
```

### Functions

- int [main](#) ()

#### 8.45.1 Function Documentation

##### 8.45.1.1 main()

```
int main ( )
```

## 8.46 file\_io/include/aare/file\_io/File.hpp File Reference

```
#include "aare/file_io/FileInterface.hpp"
```

### Data Structures

- class [aare::File](#)

*RAII [File](#) class for reading and writing image files in various formats wrapper on a [FileInterface](#) to abstract the underlying file format.*

### Namespaces

- namespace [aare](#)

*[Frame](#) class to represent a single frame of data model class should be able to work with streams coming from files or network.*

## 8.47 File.hpp

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002 #include "aare/file_io/FileInterface.hpp"
00003
00004 namespace aare {
00005
00011 class File {
00012     private:
00013         FileInterface *file_impl;
00014
00015     public:
00025         File(std::filesystem::path fname, std::string mode, FileConfig cfg = {});
00026         void write(Frame &frame);
00027         Frame read();
00028         Frame iread(size_t frame_number);
00029         std::vector<Frame> read(size_t n_frames);
00030         void read_into(std::byte *image_buf);
00031         void read_into(std::byte *image_buf, size_t n_frames);
00032         size_t frame_number(size_t frame_index);
00033         size_t bytes_per_frame();
00034         size_t pixels();
00035         void seek(size_t frame_number);
00036         size_t tell() const;
00037         size_t total_frames() const;
00038         ssize_t rows() const;
00039         ssize_t cols() const;
00040         ssize_t bitdepth() const;
00041
00046         File(File &&other);
00047
00051         ~File();
00052 };
00053
00054 } // namespace aare
```

## 8.48 file\_io/include/aare/file\_io/FileInterface.hpp File Reference

```
#include "aare/core/DType.hpp"
#include "aare/core/Frame.hpp"
#include "aare/core/defs.hpp"
#include <filesystem>
#include <vector>
```

### Data Structures

- struct [aare::FileConfig](#)  
*FileConfig* structure to store the configuration of a file dtype: data type of the file rows: number of rows in the file cols: number of columns in the file geometry: geometry of the file.
- class [aare::FileInterface](#)  
*FileInterface* class to define the interface for file operations.

### Namespaces

- namespace [aare](#)  
*Frame* class to represent a single frame of data model class should be able to work with streams coming from files or network.

## 8.49 FileInterface.hpp

[Go to the documentation of this file.](#)

```

00001 #pragma once
00002 #include "aare/core/DType.hpp"
00003 #include "aare/core/Frame.hpp"
00004 #include "aare/core/defs.hpp"
00005 #include <filesystem>
00006 #include <vector>
00007
00008 namespace aare {
00009
00010 struct FileConfig {
00011     aare::DType dtype = aare::DType(typeid(uint16_t));
00012     uint64_t rows;
00013     uint64_t cols;
00014     xy geometry{1, 1};
00015     bool operator==(const FileConfig &other) const {
00016         return dtype == other.dtype && rows == other.rows && cols == other.cols && geometry ==
00017             other.geometry;
00018     }
00019     bool operator!=(const FileConfig &other) const { return !(*this == other); }
00020 };
00021
00022 class FileInterface {
00023 public:
00024     virtual void write(Frame &frame) = 0;
00025
00026     // virtual void write(std::vector<Frame> &frames) = 0;
00027
00028     virtual Frame read() = 0;
00029
00030     virtual std::vector<Frame> read(size_t n_frames) = 0; // Is this the right interface?
00031
00032     virtual void read_into(std::byte *image_buf) = 0;
00033
00034     virtual void read_into(std::byte *image_buf, size_t n_frames) = 0;
00035
00036     virtual size_t frame_number(size_t frame_index) = 0;
00037
00038     virtual size_t bytes_per_frame() = 0;
00039
00040     virtual size_t pixels() = 0;
00041
00042     virtual void seek(size_t frame_number) = 0;
00043
00044     virtual size_t tell() = 0;
00045
00046     virtual size_t total_frames() const = 0;
00047     virtual ssize_t rows() const = 0;
00048     virtual ssize_t cols() const = 0;
00049     virtual ssize_t bitdepth() const = 0;
00050
00051     Frame iread(size_t frame_number) {
00052         auto old_pos = tell();
00053         seek(frame_number);
00054         Frame tmp = read();
00055         seek(old_pos);
00056         return tmp;
00057     };
00058
00059     std::vector<Frame> iread(size_t frame_number, size_t n_frames) {
00060         auto old_pos = tell();
00061         seek(frame_number);
00062         std::vector<Frame> tmp = read(n_frames);
00063         seek(old_pos);
00064         return tmp;
00065     }
00066
00067     // function to query the data type of the file
00068     /*virtual DataType dtype = 0; */
00069
00070     virtual ~FileInterface() {
00071     };
00072
00073 public:
00074     std::string m_mode;
00075     std::filesystem::path m_fname;
00076     std::filesystem::path m_base_path;
00077     std::string m_base_name, m_ext;
00078     int m_findex;
00079     size_t m_total_frames{};
00080     size_t max_frames_per_file{};
00081     std::string version;

```

```

00174     DetectorType m_type;
00175     ssize_t m_rows{};
00176     ssize_t m_cols{};
00177     ssize_t m_bitdepth{};
00178     size_t current_frame{};
00179 };
00180
00181 } // namespace aare

```

## 8.50 file\_io/include/aare/file\_io/NumpyFile.hpp File Reference

```

#include "aare/core/DType.hpp"
#include "aare/core/defs.hpp"
#include "aare/file_io/FileInterface.hpp"
#include "aare/file_io/NumpyHelpers.hpp"
#include <filesystem>
#include <iostream>
#include <numeric>

```

### Data Structures

- class [aare::NumpyFile](#)  
*NumpyFile class to read and write numpy files.*

### Namespaces

- namespace [aare](#)  
*Frame class to represent a single frame of data model class should be able to work with streams coming from files or network.*

## 8.51 NumpyFile.hpp

[Go to the documentation of this file.](#)

```

00001 #pragma once
00002 #include "aare/core/DType.hpp"
00003 #include "aare/core/defs.hpp"
00004 #include "aare/file_io/FileInterface.hpp"
00005 #include "aare/file_io/NumpyHelpers.hpp"
00006 #include <filesystem>
00007 #include <iostream>
00008 #include <numeric>
00009
00010 namespace aare {
00011
00012     class NumpyFile : public FileInterface {
00013     public:
00014         NumpyFile(const std::filesystem::path &fname, const std::string &mode = "r", FileConfig cfg = {});
00015
00016         void write(Frame &frame) override;
00017         Frame read() override { return get_frame(this->current_frame++); }
00018
00019         std::vector<Frame> read(size_t n_frames) override;
00020         void read_into(std::byte *image_buf) override { return get_frame_into(this->current_frame++,
00021 image_buf); }
00022         void read_into(std::byte *image_buf, size_t n_frames) override;
00023         size_t frame_number(size_t frame_index) override { return frame_index; };
00024         size_t bytes_per_frame() override;
00025         size_t pixels() override;
00026         void seek(size_t frame_number) override { this->current_frame = frame_number; }
00027         size_t tell() override { return this->current_frame; }
00028
00029     };
00030
00031 }

```

```

00040     size_t total_frames() const override { return m_header.shape[0]; }
00041     ssize_t rows() const override { return m_header.shape[1]; }
00042     ssize_t cols() const override { return m_header.shape[2]; }
00043     ssize_t bitdepth() const override { return m_header.dtype.bitdepth(); }
00044
00049     DType dtype() const { return m_header.dtype; }
00050
00055     std::vector<size_t> shape() const { return m_header.shape; }
00056
00063     template <typename T, size_t NDim> NArray<T, NDim> load() {
00064         NArray<T, NDim> arr(make_shape<NDim>(m_header.shape));
00065         fseek(fp, header_size, SEEK_SET);
00066         fread(arr.data(), sizeof(T), arr.size(), fp);
00067         return arr;
00068     }
00069
00070     ~NumpyFile();
00071
00072 private:
00073     FILE *fp = nullptr;
00074     size_t initial_header_len = 0;
00075     size_t current_frame{};
00076     uint32_t header_len{};
00077     uint8_t header_len_size{};
00078     size_t header_size{};
00079     NumpyHeader m_header;
00080     uint8_t major_ver{};
00081     uint8_t minor_ver{};
00082
00083     void load_metadata();
00084     void get_frame_into(size_t, std::byte *);
00085     Frame get_frame(size_t frame_number);
00086 };
00087
00088 } // namespace aare

```

## 8.52 file\_io/include/aare/file\_io/NumpyHelpers.hpp File Reference

```

#include <algorithm>
#include <array>
#include <filesystem>
#include <fstream>
#include <iostream>
#include <numeric>
#include <sstream>
#include <string>
#include <unordered_map>
#include <vector>
#include "aare/core/DType.hpp"
#include "aare/core/defs.hpp"

```

### Data Structures

- struct [aare::NumpyHeader](#)

### Namespaces

- namespace [aare](#)

*Frame* class to represent a single frame of data model class should be able to work with streams coming from files or network.

- namespace [aare::NumpyHelpers](#)

## Typedefs

- using `aare::shape_t` = `std::vector< size_t >`

## Functions

- `std::string aare::NumpyHelpers::parse_str` (const `std::string` &in)
- `std::string aare::NumpyHelpers::trim` (const `std::string` &str)
- `std::vector< std::string > aare::NumpyHelpers::parse_tuple` (std::string in)
- `bool aare::NumpyHelpers::parse_bool` (const `std::string` &in)
- `std::string aare::NumpyHelpers::get_value_from_map` (const `std::string` &mapstr)
- `std::unordered_map< std::string, std::string > aare::NumpyHelpers::parse_dict` (std::string in, const `std::vector< std::string > &keys`)
- `template<typename T, size_t N>`  
`bool aare::NumpyHelpers::in_array` (T val, const `std::array< T, N > &arr`)
- `bool aare::NumpyHelpers::is_digits` (const `std::string` &str)
- `aare::DType aare::NumpyHelpers::parse_descr` (std::string tpestring)
- `size_t aare::NumpyHelpers::write_header` (std::filesystem::path fname, const `NumpyHeader` &header)
- `size_t aare::NumpyHelpers::write_header` (std::ostream &out, const `NumpyHeader` &header)

## Variables

- `const constexpr std::array< char, 6 > aare::NumpyHelpers::magic_str` {"\x93", 'N', 'U', 'M', 'P', 'Y'}
- `const uint8_t aare::NumpyHelpers::magic_string_length` {6}

## 8.53 NumpyHelpers.hpp

[Go to the documentation of this file.](#)

```

00001
00002 #pragma once
00003 #include <algorithm>
00004 #include <array>
00005 #include <filesystem>
00006 #include <fstream>
00007 #include <iostream>
00008 #include <numeric>
00009 #include <sstream>
00010 #include <string>
00011 #include <unordered_map>
00012 #include <vector>
00013
00014 #include "aare/core/DType.hpp"
00015 #include "aare/core/defs.hpp"
00016
00017 namespace aare {
00018
00019 using shape_t = std::vector<size_t>;
00020
00021 struct NumpyHeader {
00022     DType dtype{aare::DType::ERROR};
00023     bool fortran_order{false};
00024     shape_t shape{};
00025
00026     std::string to_string() const;
00027 };
00028
00029 namespace NumpyHelpers {
00030
00031 const constexpr std::array<char, 6> magic_str{'\x93', 'N', 'U', 'M', 'P', 'Y'};
00032 const uint8_t magic_string_length{6};
00033
00034 std::string parse_str(const std::string &in);
00038 std::string trim(const std::string &str);
00039
00040 std::vector<std::string> parse_tuple(std::string in);

```

```

00041
00042 bool parse_bool(const std::string &in);
00043
00044 std::string get_value_from_map(const std::string &mapstr);
00045
00046 std::unordered_map<std::string, std::string> parse_dict(std::string in, const std::vector<std::string>
&keys);
00047
00048 template <typename T, size_t N> bool in_array(T val, const std::array<T, N> &arr) {
00049     return std::find(std::begin(arr), std::end(arr), val) != std::end(arr);
00050 }
00051 bool is_digits(const std::string &str);
00052
00053 aare::DType parse_descr(std::string typestring);
00054 size_t write_header(std::filesystem::path fname, const NumpyHeader &header);
00055 size_t write_header(std::ostream &out, const NumpyHeader &header);
00056
00057 } // namespace NumpyHelpers
00058 } // namespace aare

```

## 8.54 file\_io/include/aare/file\_io/RawFile.hpp File Reference

```

#include "aare/core/Frame.hpp"
#include "aare/file_io/FileInterface.hpp"
#include "aare/file_io/SubFile.hpp"

```

### Data Structures

- class [aare::RawFile](#)  
*RawFile class to read .raw and .json files.*

### Namespaces

- namespace [aare](#)  
*Frame class to represent a single frame of data model class should be able to work with streams coming from files or network.*

## 8.55 RawFile.hpp

[Go to the documentation of this file.](#)

```

00001 #pragma once
00002 #include "aare/core/Frame.hpp"
00003 #include "aare/file_io/FileInterface.hpp"
00004 #include "aare/file_io/SubFile.hpp"
00005
00006 namespace aare {
00007
00013 class RawFile : public FileInterface {
00014     public:
00021     RawFile(const std::filesystem::path &fname, const std::string &mode = "r", const FileConfig &cfg =
{});
00022
00027     void write(Frame &frame) override { throw std::runtime_error("Not implemented"); };
00028     Frame read() override { return get_frame(this->current_frame++); };
00029     std::vector<Frame> read(size_t n_frames) override;
00030     void read_into(std::byte *image_buf) override { return get_frame_into(this->current_frame++,
image_buf); };
00031     void read_into(std::byte *image_buf, size_t n_frames) override;
00032     size_t frame_number(size_t frame_index) override;
00033
00038     size_t bytes_per_frame() override { return m_rows * m_cols * m_bitdepth / 8; }
00039
00044     size_t pixels() override { return m_rows * m_cols; }

```

```

00045
00046 // goto frame number
00047 void seek(size_t frame_number) override { this->current_frame = frame_number; };
00048
00049 // return the position of the file pointer (in number of frames)
00050 size_t tell() override { return this->current_frame; };
00051
00052 static bool is_master_file(std::filesystem::path fpath);
00053
00054 inline void set_config(int row, int col) {
00055     cfg.module_gap_row = row;
00056     cfg.module_gap_col = col;
00057 }
00058 // TODO! Deal with fast quad and missing files
00059
00060 void find_number_of_subfiles();
00061
00062 inline std::filesystem::path master_fname();
00063 inline std::filesystem::path data_fname(int mod_id, int file_id);
00064
00065 ~RawFile();
00066
00067 size_t total_frames() const override { return m_total_frames; }
00068 ssize_t rows() const override { return m_rows; }
00069 ssize_t cols() const override { return m_cols; }
00070 ssize_t bitdepth() const override { return m_bitdepth; }
00071
00072 private:
00073 void get_frame_into(size_t frame_number, std::byte *image_buf);
00074
00075 Frame get_frame(size_t frame_number);
00076
00077 void parse_fname();
00078
00079 void parse_metadata();
00080
00081 void parse_raw_metadata();
00082
00083 void parse_json_metadata();
00084
00085 void find_geometry();
00086
00087 sls_detector_header read_header(const std::filesystem::path &fname);
00088
00089 void open_subfiles();
00090
00091 size_t n_subfiles;
00092 size_t n_subfile_parts;
00093 std::vector<std::vector<SubFile *>> subfiles;
00094 int subfile_rows, subfile_cols;
00095 xy geometry;
00096 std::vector<xy> positions;
00097 RawFileConfig cfg{0, 0};
00098 TimingMode timing_mode;
00099 bool quad{false};
00100 };
00101
00102 } // namespace aare

```

## 8.56 file\_io/include/aare/file\_io/SubFile.hpp File Reference

```

#include "aare/core/defs.hpp"
#include <cstdint>
#include <filesystem>
#include <map>
#include <variant>

```

### Data Structures

- class [aare::SubFile](#)

*Class to read a subfile from a [RawFile](#).*



## Namespaces

- namespace `aare`

*Frame* class to represent a single frame of data model class should be able to work with streams coming from files or network.

## 8.57 SubFile.hpp

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002 #include "aare/core/defs.hpp"
00003 #include <stdint>
00004 #include <filesystem>
00005 #include <map>
00006 #include <variant>
00007
00008 namespace aare {
00009
00013 class SubFile {
00014     protected:
00020         using pfunc = size_t (SubFile::*)(std::byte *);
00021         pfunc read_impl = nullptr;
00028         std::map<std::pair<DetectorType, int>, pfunc> read_impl_map = {
00029             {{DetectorType::Moench, 16}, &SubFile::read_impl_reorder<uint16_t>},
00030             {{DetectorType::Jungfrau, 16}, &SubFile::read_impl_normal},
00031             {{DetectorType::ChipTestBoard, 16}, &SubFile::read_impl_normal},
00032             {{DetectorType::Mythen3, 32}, &SubFile::read_impl_normal},
00033             {{DetectorType::Eiger, 32}, &SubFile::read_impl_normal},
00034             {{DetectorType::Eiger, 16}, &SubFile::read_impl_normal}
00035         };
00036     };
00037
00038     public:
00048         SubFile(std::filesystem::path fname, DetectorType detector, ssize_t rows, ssize_t cols, uint16_t
bitdepth);
00049
00055         size_t read_impl_normal(std::byte *buffer);
00056
00062         template <typename DataType> size_t read_impl_flip(std::byte *buffer);
00063
00069         template <typename DataType> size_t read_impl_reorder(std::byte *buffer);
00070
00077         size_t get_part(std::byte *buffer, int frame_number);
00078         size_t frame_number(int frame_index);
00079
00080         // TODO: define the inlines as variables and assign them in constructor
00081         inline size_t bytes_per_part() { return (m_bitdepth / 8) * m_rows * m_cols; }
00082         inline size_t pixels_per_part() { return m_rows * m_cols; }
00083
00084     protected:
00085         FILE *fp = nullptr;
00086         ssize_t m_bitdepth;
00087         std::filesystem::path m_fname;
00088         ssize_t m_rows{};
00089         ssize_t m_cols{};
00090         ssize_t n_frames{};
00091         int m_sub_file_index_{};
00092 };
00093
00094 } // namespace aare
```

## 8.58 file\_io/src/File.cpp File Reference

```
#include "aare/file_io/File.hpp"
#include "aare/file_io/NumpyFile.hpp"
#include "aare/file_io/RawFile.hpp"
#include "aare/utils/logger.hpp"
#include <fmt/format.h>
```

## Namespaces

- namespace [aare](#)

*[Frame](#) class to represent a single frame of data model class should be able to work with streams coming from files or network.*

## 8.59 file\_io/src/NumpyFile.cpp File Reference

```
#include "aare/file_io/NumpyFile.hpp"
#include "aare/utils/logger.hpp"
```

## Namespaces

- namespace [aare](#)

*[Frame](#) class to represent a single frame of data model class should be able to work with streams coming from files or network.*

## 8.60 file\_io/src/NumpyHelpers.cpp File Reference

```
#include "aare/file_io/NumpyHelpers.hpp"
#include <iterator>
```

## Namespaces

- namespace [aare](#)

*[Frame](#) class to represent a single frame of data model class should be able to work with streams coming from files or network.*

- namespace [aare::NumpyHelpers](#)

## Functions

- `std::unordered_map< std::string, std::string > aare::NumpyHelpers::parse\_dict (std::string in, const std::vector< std::string > &keys)`
- `aare::DType aare::NumpyHelpers::parse\_descr (std::string tpestring)`
- `bool aare::NumpyHelpers::parse\_bool (const std::string &in)`
- `std::string aare::NumpyHelpers::get\_value\_from\_map (const std::string &mapstr)`
- `bool aare::NumpyHelpers::is\_digits (const std::string &str)`
- `std::vector< std::string > aare::NumpyHelpers::parse\_tuple (std::string in)`
- `std::string aare::NumpyHelpers::trim (const std::string &str)`
- `std::string aare::NumpyHelpers::parse\_str (const std::string &in)`
- `void aare::NumpyHelpers::write\_magic (std::ostream &ostream, int version_major, int version_minor)`
- `template<typename T >  
std::string aare::NumpyHelpers::write\_tuple (const std::vector< T > &v)`
- `std::string aare::NumpyHelpers::write\_boolean (bool b)`
- `std::string aare::NumpyHelpers::write\_header\_dict (const std::string &descr, bool fortran_order, const shape\_t &shape)`
- `size_t aare::NumpyHelpers::write\_header (std::filesystem::path fname, const NumpyHeader &header)`
- `size_t aare::NumpyHelpers::write\_header (std::ostream &out, const NumpyHeader &header)`

## 8.61 file\_io/src/RawFile.cpp File Reference

```
#include "aare/file_io/RawFile.hpp"
#include "aare/core/defs.hpp"
#include "aare/utils/logger.hpp"
#include <fmt/format.h>
#include <nlohmann/json.hpp>
```

### Namespaces

- namespace [aare](#)  
*[Frame](#) class to represent a single frame of data model class should be able to work with streams coming from files or network.*

### Typedefs

- using [json](#) = nlohmann::json

### 8.61.1 Typedef Documentation

#### 8.61.1.1 json

```
using json = nlohmann::json
```

## 8.62 file\_io/src/SubFile.cpp File Reference

```
#include "aare/file_io/SubFile.hpp"
#include "aare/utils/logger.hpp"
#include <cstring>
#include <fmt/core.h>
#include <iostream>
```

### Namespaces

- namespace [aare](#)  
*[Frame](#) class to represent a single frame of data model class should be able to work with streams coming from files or network.*

## 8.63 file\_io/test/NumpyFile.test.cpp File Reference

```
#include "aare/file_io/NumpyFile.hpp"
#include "aare/core/NDArray.hpp"
#include <catch2/catch_test_macros.hpp>
#include "test_config.hpp"
```

## Functions

- [TEST\\_CASE](#) ("Read a 1D numpy file with int32 data type")
- [TEST\\_CASE](#) ("Read a 3D numpy file with np.double data type")

### 8.63.1 Function Documentation

#### 8.63.1.1 TEST\_CASE() [1/2]

```
TEST_CASE (
    "Read a 1D numpy file with int32 data type" )
```

#### 8.63.1.2 TEST\_CASE() [2/2]

```
TEST_CASE (
    "Read a 3D numpy file with np.double data type" )
```

## 8.64 file\_io/test/NumpyHelpers.test.cpp File Reference

```
#include "aare/file_io/NumpyHelpers.hpp"
#include <catch2/catch_test_macros.hpp>
```

## Functions

- [TEST\\_CASE](#) ("is\_digits with a few standard cases")
- [TEST\\_CASE](#) ("Check for quotes and return stripped string")
- [TEST\\_CASE](#) ("parsing a string without quotes throws")
- [TEST\\_CASE](#) ("trim whitespace")
- [TEST\\_CASE](#) ("parse data type descriptions")
- [TEST\\_CASE](#) ("is element in array")
- [TEST\\_CASE](#) ("Parse numpy dict")

### 8.64.1 Function Documentation

#### 8.64.1.1 TEST\_CASE() [1/7]

```
TEST_CASE (
    "Check for quotes and return stripped string" )
```

#### 8.64.1.2 TEST\_CASE() [2/7]

```
TEST_CASE (
    "is element in array" )
```

#### 8.64.1.3 TEST\_CASE() [3/7]

```
TEST_CASE (
    "is_digits with a few standard cases" )
```

#### 8.64.1.4 TEST\_CASE() [4/7]

```
TEST_CASE (
    "parse data type descriptions" )
```

#### 8.64.1.5 TEST\_CASE() [5/7]

```
TEST_CASE (
    "Parse numpy dict" )
```

#### 8.64.1.6 TEST\_CASE() [6/7]

```
TEST_CASE (
    "parsing a string without quotes throws" )
```

#### 8.64.1.7 TEST\_CASE() [7/7]

```
TEST_CASE (
    "trim whitespace" )
```

### 8.65 file\_io/test/RawFile.test.cpp File Reference

```
#include "aare/file_io/File.hpp"
#include "aare/utils/logger.hpp"
#include <catch2/catch_test_macros.hpp>
#include <filesystem>
#include "test_config.hpp"
```

#### Functions

- [TEST\\_CASE](#) ("Read number of frames from a jungfrau raw file")
- [TEST\\_CASE](#) ("Read frame numbers from a jungfrau raw file")
- [TEST\\_CASE](#) ("Read data from a jungfrau 500k single port raw file")
- [TEST\\_CASE](#) ("Read frame numbers from a raw file")
- [TEST\\_CASE](#) ("Compare reading from a numpy file with a raw file")

## 8.65.1 Function Documentation

### 8.65.1.1 TEST\_CASE() [1/5]

```
TEST_CASE (
    "Compare reading from a numpy file with a raw file" )
```

### 8.65.1.2 TEST\_CASE() [2/5]

```
TEST_CASE (
    "Read data from a jungfrau 500k single port raw file" )
```

### 8.65.1.3 TEST\_CASE() [3/5]

```
TEST_CASE (
    "Read frame numbers from a jungfrau raw file" )
```

### 8.65.1.4 TEST\_CASE() [4/5]

```
TEST_CASE (
    "Read frame numbers from a raw file" )
```

### 8.65.1.5 TEST\_CASE() [5/5]

```
TEST_CASE (
    "Read number of frames from a jungfrau raw file" )
```

## 8.66 include/aare/aare.hpp File Reference

## 8.67 aare.hpp

[Go to the documentation of this file.](#)

```
00001 // This is the top level header to include and what most users will use
```

## 8.68 network\_io/include/aare/network\_io/ZmqHeader.hpp File Reference

```
#include "aare/core/Frame.hpp"
#include "aare/utils/logger.hpp"
#include "simdjson.h"
#include <array>
#include <cstdint>
#include <map>
#include <string>
```

## Data Structures

- struct [aare::ZmqHeader](#)

## Namespaces

- namespace [simdjson](#)
- namespace [aare](#)

*Frame* class to represent a single frame of data model class should be able to work with streams coming from files or network.

## 8.69 ZmqHeader.hpp

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002 #include "aare/core/Frame.hpp"
00003 #include "aare/utils/logger.hpp"
00004
00005 #include "simdjson.h"
00006 #include <array>
00007 #include <stdint>
00008 #include <map>
00009 #include <string>
00010 namespace simdjson {
00011 template <> simdjson_inline simdjson::simdjson_result<std::array<int, 4>
00012     simdjson::ondemand::value::get() noexcept {
00013     ondemand::array array;
00014     auto error = get_array().get(array);
00015     if (error) {
00016         return error;
00017     }
00018     std::array<int, 4> arr;
00019     int i = 0;
00020     for (auto v : array) {
00021         int64_t val;
00022         error = v.get_int64().get(val);
00023         if (error) {
00024             return error;
00025         }
00026         arr[i++] = val;
00027     }
00028     return arr;
00029 }
00030
00031 template <> simdjson_inline simdjson::simdjson_result<uint32_t> simdjson::ondemand::value::get()
00032     noexcept {
00033     size_t val;
00034     auto error = get_uint64().get(val);
00035     if (error) {
00036         return error;
00037     }
00038     if (val > std::numeric_limits<uint32_t>::max()) {
00039         return 1;
00040     }
00041     return static_cast<uint32_t>(val);
00042 }
00043
00044 template <>
00045 simdjson_inline simdjson::simdjson_result<std::map<std::string, std::string>
00046     simdjson::ondemand::value::get() noexcept {
00047     std::map<std::string, std::string> map;
00048     ondemand::object obj;
00049     auto error = get_object().get(obj);
00050     if (error) {
00051         return error;
00052     }
00053     for (auto field : obj) {
00054         simdjson::ondemand::raw_json_string tmp;
00055         error = field.key().get(tmp);
00056         if (error) {
00057             return error;
00058         }
00059         error = field.value().get(tmp);
00060         if (error) {
00061             return error;
00062         }
00063     }
00064 }
```

```

00071         return error;
00072     }
00073     std::string_view key_view = field.unescaped_key();
00074     std::string key_str(key_view.data(), key_view.size());
00075     std::string_view value_view = field.value().get_string();
00076     map[key_str] = {value_view.data(), value_view.size()};
00077 }
00078 return map;
00079 }
00080
00081 } // namespace simdjson
00082
00083 namespace aare {
00084
00085     struct ZmqHeader {
00086         bool data{true};
00087         uint32_t jsonversion{0};
00088         uint32_t dynamicRange{0};
00089         uint64_t fileIndex{0};
00090         uint32_t ndetx{0};
00091         uint32_t ndety{0};
00092         uint32_t npixelsx{0};
00093         uint32_t npixelsy{0};
00094         uint32_t imageSize{0};
00095         uint64_t acqIndex{0};
00096         uint64_t frameIndex{0};
00097         double progress{0};
00098         std::string fname;
00099         uint64_t frameNumber{0};
00100         uint32_t expLength{0};
00101         uint32_t packetNumber{0};
00102         uint64_t detSpec1{0};
00103         uint64_t timestamp{0};
00104         uint16_t modId{0};
00105         uint16_t row{0};
00106         uint16_t column{0};
00107         uint16_t detSpec2{0};
00108         uint32_t detSpec3{0};
00109         uint16_t detSpec4{0};
00110         uint8_t detType{0};
00111         uint8_t version{0};
00112         int flipRows{0};
00113         uint32_t quad{0};
00114         bool completeImage{false};
00115         std::map<std::string, std::string> addJsonHeader;
00116         std::array<int, 4> rx_roi{};
00117
00118         std::string to_string() const;
00119         void from_string(std::string &s);
00120         // compare operator
00121         bool operator==(const ZmqHeader &other) const;
00122     };
00123
00124 } // namespace aare

```

## 8.70 network\_io/include/aare/network\_io/ZmqSocket.hpp File Reference

```
#include <string>
```

### Data Structures

- class [aare::ZmqSocket](#)

### Namespaces

- namespace [aare](#)

*Frame* class to represent a single frame of data model class should be able to work with streams coming from files or network.



## 8.71 ZmqSocket.hpp

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include <string>
00004
00005 // Socket to receive data from a ZMQ publisher
00006 // needs to be in sync with the main library (or maybe better use the versioning in the header)
00007
00008 // forward declare zmq_msg_t to avoid including zmq.h in the header
00009 class zmq_msg_t;
00010
00011 namespace aare {
00012
00013 class ZmqSocket {
00014     protected:
00015         void *m_context{nullptr};
00016         void *m_socket{nullptr};
00017         std::string m_endpoint;
00018         int m_zmq_hwm{1000};
00019         int m_timeout_ms{1000};
00020         size_t m_potential_frame_size{1024 * 1024};
00021         constexpr static size_t m_max_header_size = 1024;
00022         char *m_header_buffer = new char[m_max_header_size];
00023
00024     public:
00025         ZmqSocket() = default;
00026         ~ZmqSocket();
00027
00028         ZmqSocket(const ZmqSocket &) = delete;
00029         ZmqSocket operator=(const ZmqSocket &) = delete;
00030         ZmqSocket(ZmqSocket &&) = delete;
00031
00032         void disconnect();
00033         void set_zmq_hwm(int hwm);
00034         void set_timeout_ms(int n);
00035         void set_potential_frame_size(size_t size);
00036 };
00037
00038 } // namespace aare
```

## 8.72 network\_io/include/aare/network\_io/ZmqSocketReceiver.hpp File Reference

```
#include "aare/core/Frame.hpp"
#include "aare/network_io/ZmqHeader.hpp"
#include "aare/network_io/ZmqSocket.hpp"
#include "aare/network_io/defs.hpp"
#include <stdint>
#include <string>
```

### Data Structures

- class [aare::ZmqSocketReceiver](#)

### Namespaces

- namespace [aare](#)

*Frame* class to represent a single frame of data model class should be able to work with streams coming from files or network.

## 8.73 ZmqSocketReceiver.hpp

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include "aare/core/Frame.hpp"
00004 #include "aare/network_io/ZmqHeader.hpp"
00005 #include "aare/network_io/ZmqSocket.hpp"
00006 #include "aare/network_io/defs.hpp"
00007
00008 #include <cstdint>
00009 #include <string>
00010
00011 // Socket to receive data from a ZMQ publisher
00012 // needs to be in sync with the main library (or maybe better use the versioning in the header)
00013
00014 // forward declare zmq_msg_t to avoid including zmq.h in the header
00015 class zmq_msg_t;
00016
00017 namespace aare {
00018
00019 class ZmqSocketReceiver : public ZmqSocket {
00020 public:
00021     ZmqSocketReceiver(const std::string &endpoint);
00022     void connect();
00023     std::vector<ZmqFrame> receive_n();
00024
00025 private:
00026     int receive_data(std::byte *data, size_t size);
00027     ZmqFrame receive_zmqframe();
00028     ZmqHeader receive_header();
00029 };
00030
00031 } // namespace aare
```

## 8.74 network\_io/include/aare/network\_io/ZmqSocketSender.hpp File Reference

```
#include "aare/core/Frame.hpp"
#include "aare/network_io/ZmqHeader.hpp"
#include "aare/network_io/ZmqSocket.hpp"
#include "aare/network_io/defs.hpp"
```

### Data Structures

- class [aare::ZmqSocketSender](#)

### Namespaces

- namespace [aare](#)

*Frame* class to represent a single frame of data model class should be able to work with streams coming from files or network.

## 8.75 ZmqSocketSender.hpp

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002 #include "aare/core/Frame.hpp"
00003 #include "aare/network_io/ZmqHeader.hpp"
00004 #include "aare/network_io/ZmqSocket.hpp"
00005 #include "aare/network_io/defs.hpp"
00006
00007 namespace aare {
00008 class ZmqSocketSender : public ZmqSocket {
00009 public:
00010     ZmqSocketSender(const std::string &endpoint);
00011     void bind();
00012     size_t send(const ZmqHeader &header, const std::byte *data, size_t size);
00013     size_t send(const ZmqFrame &zmq_frame);
00014     size_t send(const std::vector<ZmqFrame> &zmq_frames);
00015 };
00016 } // namespace aare
```

## 8.76 network\_io/src/ZmqHeader.cpp File Reference

```
#include "aare/network_io/ZmqHeader.hpp"
#include "simdjson.h"
```

### Namespaces

- namespace [aare](#)

*Frame* class to represent a single frame of data model class should be able to work with streams coming from files or network.

### Functions

- `template<typename T >`  
void [write\\_digit](#) (std::string &s, const std::string &key, const T &value)  
*write a digit to a string takes key and value and outputs->"key": value,*
- void [write\\_str](#) (std::string &s, const std::string &key, const std::string &value)
- void [write\\_map](#) (std::string &s, const std::string &key, const std::map< std::string, std::string > &value)
- void [write\\_array](#) (std::string &s, const std::string &key, const std::array< int, 4 > &value)

### 8.76.1 Function Documentation

#### 8.76.1.1 write\_array()

```
void write_array (
    std::string & s,
    const std::string & key,
    const std::array< int, 4 > & value )
```

#### 8.76.1.2 write\_digit()

```
template<typename T >
void write_digit (
    std::string & s,
    const std::string & key,
    const T & value )
```

write a digit to a string takes key and value and outputs->"key": value,

## Template Parameters

<i>T</i>	type of value (int, uint32_t, ...)
----------	------------------------------------

## Parameters

<i>s</i>	string to append to
<i>key</i>	key to write
<i>value</i>	value to write

## Returns

void

## Note

- can't use concepts here because we are using c++17

## 8.76.1.3 write\_map()

```
void write_map (
    std::string & s,
    const std::string & key,
    const std::map< std::string, std::string > & value )
```

## 8.76.1.4 write\_str()

```
void write_str (
    std::string & s,
    const std::string & key,
    const std::string & value )
```

## 8.77 network\_io/src/ZmqSocket.cpp File Reference

```
#include "aare/network_io/ZmqSocket.hpp"
#include <zmq.h>
```

## Namespaces

- namespace [aare](#)

*Frame* class to represent a single frame of data model class should be able to work with streams coming from files or network.

## 8.78 network\_io/src/ZmqSocketReceiver.cpp File Reference

```
#include "aare/network_io/ZmqSocketReceiver.hpp"
#include "aare/utils/logger.hpp"
#include <fmt/core.h>
#include <zmq.h>
```

### Namespaces

- namespace [aare](#)

*[Frame](#) class to represent a single frame of data model class should be able to work with streams coming from files or network.*

## 8.79 network\_io/src/ZmqSocketSender.cpp File Reference

```
#include "aare/network_io/ZmqSocketSender.hpp"
#include <cassert>
#include <zmq.h>
```

### Namespaces

- namespace [aare](#)

*[Frame](#) class to represent a single frame of data model class should be able to work with streams coming from files or network.*

## 8.80 network\_io/test/ZmqHeader.test.cpp File Reference

```
#include "aare/network_io/ZmqHeader.hpp"
#include "aare/utils/logger.hpp"
#include <catch2/catch_test_macros.hpp>
```

### Functions

- [TEST\\_CASE](#) ("Test ZmqHeader")

### 8.80.1 Function Documentation

#### 8.80.1.1 TEST\_CASE()

```
TEST_CASE (
    "Test ZmqHeader" )
```

## 8.81 python/aare/File.py File Reference

### Data Structures

- class [aare.File.File](#)

### Namespaces

- namespace [aare](#)  
*[Frame](#) class to represent a single frame of data model class should be able to work with streams coming from files or network.*
- namespace [aare.File](#)

## 8.82 python/aare/Frame.py File Reference

### Data Structures

- class [aare.Frame.Frame](#)

### Namespaces

- namespace [aare](#)  
*[Frame](#) class to represent a single frame of data model class should be able to work with streams coming from files or network.*
- namespace [aare.Frame](#)

## 8.83 python/aare/\_\_init\_\_.py File Reference

### Namespaces

- namespace [aare](#)  
*[Frame](#) class to represent a single frame of data model class should be able to work with streams coming from files or network.*

## 8.84 python/example/\_\_init\_\_.py File Reference

## 8.85 python/src/bindings.cpp File Reference

```
#include <cstdint>
#include <filesystem>
#include <pybind11/pybind11.h>
#include <pybind11/stl.h>
#include <string>
#include "aare/FileHandler.hpp"
#include "aare/core/Frame.hpp"
#include "aare/core/defs.hpp"
```

## Functions

- [PYBIND11\\_MODULE](#) (`_aare, m`)

### 8.85.1 Function Documentation

#### 8.85.1.1 PYBIND11\_MODULE()

```
PYBIND11_MODULE (
    _aare ,
    m )
```

## 8.86 README.md File Reference

### 8.87 tests/test.cpp File Reference

```
#include <catch2/catch_test_macros.hpp>
#include <filesystem>
#include <fstream>
#include "test_config.hpp"
```

## Functions

- [TEST\\_CASE](#) ("Test suite can find data assets")
- [TEST\\_CASE](#) ("Test suite can open data assets")

### 8.87.1 Function Documentation

#### 8.87.1.1 TEST\_CASE() [1/2]

```
TEST_CASE (
    "Test suite can find data assets" )
```

#### 8.87.1.2 TEST\_CASE() [2/2]

```
TEST_CASE (
    "Test suite can open data assets" )
```

### 8.88 utils/include/aare/utils/logger.hpp File Reference

```
#include <filesystem>
#include <fstream>
#include <iostream>
#include <map>
#include <vector>
```

## Data Structures

- class [aare::logger::Logger](#)

## Namespaces

- namespace [aare](#)  
*Frame class to represent a single frame of data model class should be able to work with streams coming from files or network.*
- namespace [aare::logger](#)
- namespace [aare::logger::internal](#)

## Macros

- #define [LOCATION](#) std::string(\_\_FILE\_\_) + std::string(" ") + std::to\_string(\_\_LINE\_\_) + " " + std::string(\_\_func\_\_) + " "

## Enumerations

- enum [aare::logger::LOGGING\\_LEVEL](#) { [aare::logger::DEBUG](#) = 0 , [aare::logger::INFO](#) = 1 , [aare::logger::WARNING](#) = 2 , [aare::logger::ERROR](#) = 3 }

## Functions

- template<typename T >  
std::ostream & [operator<<](#) (std::ostream &out, const std::vector< T > &v)
- template<typename T , size\_t N>  
std::ostream & [operator<<](#) (std::ostream &out, const std::array< T, N > &v)
- template<typename K , typename V >  
std::ostream & [operator<<](#) (std::ostream &out, const std::map< K, V > &v)
- template<[LOGGING\\_LEVEL](#) level, typename... Strings>  
void [aare::logger::log](#) (const Strings... s)
- template<typename... Strings>  
void [aare::logger::debug](#) (const Strings... s)
- template<typename... Strings>  
void [aare::logger::info](#) (const Strings... s)
- template<typename... Strings>  
void [aare::logger::warn](#) (const Strings... s)
- template<typename... Strings>  
void [aare::logger::error](#) (const Strings... s)
- void [aare::logger::set\\_streams](#) (std::streambuf \*out, std::streambuf \*err)
- void [aare::logger::set\\_streams](#) (std::streambuf \*out)
- void [aare::logger::set\\_verbosity](#) ([LOGGING\\_LEVEL](#) level)
- void [aare::logger::set\\_output\\_file](#) (std::string filename)
- [Logger](#) & [aare::logger::get\\_logger\\_instance](#) ()

## Variables

- [aare::logger::Logger](#) [aare::logger::internal::logger\\_instance](#) = [aare::logger::Logger](#)()



## 8.88.1 Macro Definition Documentation

### 8.88.1.1 LOCATION

```
#define LOCATION std::string(__FILE__) + std::string(":") + std::to_string(__LINE__) + ":" +
std::string(__func__) + ":"
```

## 8.88.2 Function Documentation

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

```
template<typename T , size_t N>
std::ostream & operator<< (
    std::ostream & out,
    const std::array< T, N > & v )
```

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

```
template<typename K , typename V >
std::ostream & operator<< (
    std::ostream & out,
    const std::map< K, V > & v )
```

### 8.88.2.3 operator<<() [3/3]

```
template<typename T >
std::ostream & operator<< (
    std::ostream & out,
    const std::vector< T > & v )
```

## 8.89 logger.hpp

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002 #include <filesystem>
00003 #include <fstream>
00004 #include <iostream>
00005 #include <map>
00006 #include <vector>
00007
00008 #define LOCATION std::string(__FILE__) + std::string(":") + std::to_string(__LINE__) + ":" +
std::string(__func__) + ":"
00009
00010 // operator overload to print vectors
00011 // typename T must be printable (i.e. have the « operator)
00012 template <typename T> std::ostream &operator<<(std::ostream &out, const std::vector<T> &v) {
00013     out << "[";
00014     size_t last = v.size() - 1;
00015     for (size_t i = 0; i < v.size(); ++i) {
00016         out << v[i];
00017         if (i != last)
00018             out << ", ";
00019     }
00020     out << "]";
00021     return out;
00022 }
00023
```

```

00024 // operator overload for std::array
00025 template <typename T, size_t N> std::ostream &operator<<(std::ostream &out, const std::array<T, N> &v)
00026 {
00027     out << "[";
00028     size_t last = N - 1;
00029     for (size_t i = 0; i < N; ++i) {
00030         out << v[i];
00031         if (i != last)
00032             out << ", ";
00033     }
00034     out << "]";
00035     return out;
00036 }
00037 // operator overload for std::map
00038 template <typename K, typename V> std::ostream &operator<<(std::ostream &out, const std::map<K, V> &v)
00039 {
00040     out << "{";
00041     size_t i = 0;
00042     for (auto &kv : v) {
00043         out << kv.first << ": " << kv.second << ((++i != v.size()) ? ", " : " ");
00044     }
00045     out << "}";
00046     return out;
00047 }
00048 namespace aare {
00049     namespace logger {
00050         enum LOGGING_LEVEL {
00051             DEBUG = 0,
00052             INFO = 1,
00053             WARNING = 2,
00054             ERROR = 3
00055         };
00056     };
00057     class Logger {
00058     public:
00059         std::streambuf *standard_buf = std::cout.rdbuf();
00060         std::streambuf *error_buf = std::cerr.rdbuf();
00061         std::ostream *standard_output;
00062         std::ostream *error_output;
00063         LOGGING_LEVEL VERBOSITY_LEVEL = LOGGING_LEVEL::INFO;
00064         std::ofstream out_file;
00065         void set_output_file(std::string filename) {
00066             if (out_file.is_open())
00067                 out_file.close();
00068             out_file.open(filename);
00069             set_streams(out_file.rdbuf());
00070         }
00071         void set_streams(std::streambuf *out, std::streambuf *err) {
00072             delete standard_output;
00073             delete error_output;
00074             standard_output = new std::ostream(out);
00075             error_output = new std::ostream(err);
00076         }
00077         void set_streams(std::streambuf *out) { set_streams(out, out); }
00078         void set_verbosity(LOGGING_LEVEL level) { VERBOSITY_LEVEL = level; }
00079         Logger() {
00080             standard_output = new std::ostream(standard_buf);
00081             error_output = new std::ostream(error_buf);
00082         }
00083         ~Logger() {
00084             if (out_file.is_open())
00085                 out_file.close();
00086             standard_output->flush();
00087             error_output->flush();
00088             delete standard_output;
00089             delete error_output;
00090         }
00091         template <LOGGING_LEVEL level, typename... Strings> void log(const Strings... s) {
00092             if (level >= VERBOSITY_LEVEL)
00093                 log<level>(s...);
00094         }
00095         template <typename... Strings> void debug(const Strings... s) {
00096             log<LOGGING_LEVEL::DEBUG>("[DEBUG]", s...);
00097         }
00098         template <typename... Strings> void info(const Strings... s) { log<LOGGING_LEVEL::INFO>("[INFO]", s...); }
00099         template <typename... Strings> void warn(const Strings... s) {
00100             log<LOGGING_LEVEL::WARNING>("[WARN]", s...);
00101         }
00102         template <typename... Strings> void error(const Strings... s) {

```

```

00106     log<LOGGING_LEVEL::ERROR>("[ERROR]", s...); }
00107 private:
00108     template <LOGGING_LEVEL level> void log_() {
00109         if (level == LOGGING_LEVEL::ERROR) {
00110             *error_output << std::endl;
00111         } else {
00112             *standard_output << std::endl;
00113         }
00114     }
00115     template <LOGGING_LEVEL level, typename First, typename... Strings> void log_(First arg, const
Strings... s) {
00116         if (level == LOGGING_LEVEL::ERROR) {
00117             *error_output << (arg) << ' ';
00118             error_output->flush();
00119         } else {
00120             *standard_output << (arg) << ' ';
00121             standard_output->flush();
00122         }
00123         log_<level>(s...);
00124     }
00125 };
00126
00127 namespace internal {
00128
00129 extern aare::logger::Logger logger_instance;
00130 } // namespace internal
00131
00132 template <LOGGING_LEVEL level, typename... Strings> void log(const Strings... s) {
00133     internal::logger_instance.log<level>(s...);
00134 }
00135 template <typename... Strings> void debug(const Strings... s) { internal::logger_instance.debug(s...); }
00136 template <typename... Strings> void info(const Strings... s) { internal::logger_instance.info(s...); }
00137 template <typename... Strings> void warn(const Strings... s) { internal::logger_instance.warn(s...); }
00138 template <typename... Strings> void error(const Strings... s) { internal::logger_instance.error(s...); }
00139
00140 extern void set_streams(std::streambuf *out, std::streambuf *err);
00141 extern void set_streams(std::streambuf *out);
00142 extern void set_verbosity(LOGGING_LEVEL level);
00143 extern void set_output_file(std::string filename);
00144 extern Logger &get_logger_instance();
00145
00146 } // namespace logger
00147
00148 } // namespace aare

```

## 8.90 utils/src/logger.cpp File Reference

```
#include "aare/utils/logger.hpp"
```

### Namespaces

- namespace [aare](#)
  - Frame* class to represent a single frame of data model class should be able to work with streams coming from files or network.
- namespace [aare::logger](#)
- namespace [aare::logger::internal](#)

### Functions

- void [aare::logger::set\\_streams](#) (std::streambuf \*out, std::streambuf \*err)
- void [aare::logger::set\\_streams](#) (std::streambuf \*out)
- void [aare::logger::set\\_verbosity](#) (LOGGING\_LEVEL level)
- [Logger & aare::logger::get\\_logger\\_instance](#) ()
- void [aare::logger::set\\_output\\_file](#) (std::string filename)

## 8.91 utils/test/logger.test.cpp File Reference

```
#include "aare/utils/logger.hpp"  
#include <catch2/catch_test_macros.hpp>
```

# Index

- `__getattrattribute__`
    - `aare.File.File`, [44](#)
    - `aare.Frame.Frame`, [58](#)
  - `__init__`
    - `aare.File.File`, [44](#)
    - `aare.Frame.Frame`, [57](#)
  - `_file`
    - `aare.File.File`, [44](#)
  - `_frameImpl`
    - `aare.Frame.Frame`, [58](#)
  - `~CircularFifo`
    - `aare::CircularFifo< ItemType >`, [30](#)
  - `~File`
    - `aare::File`, [41](#)
  - `~FileInterface`
    - `aare::FileInterface`, [47](#)
  - `~Frame`
    - `aare::Frame`, [55](#)
  - `~Logger`
    - `aare::logger::Logger`, [60](#)
  - `~NDArray`
    - `aare::NDArray< T, Ndim >`, [67](#)
  - `~NumpyFile`
    - `aare::NumpyFile`, [82](#)
  - `~ProducerConsumerQueue`
    - `folly::ProducerConsumerQueue< T >`, [92](#)
  - `~RawFile`
    - `aare::RawFile`, [97](#)
  - `~ZmqSocket`
    - `aare::ZmqSocket`, [123](#)
- `aare`, [1](#), [11](#)
  - `Auto`, [14](#)
  - `big`, [14](#)
  - `c_strides`, [14](#)
  - `ChipTestBoard`, [13](#)
  - `DataTypeVariants`, [13](#)
  - `DetectorType`, [13](#)
  - `dynamic_shape`, [13](#)
  - `Eiger`, [13](#)
  - `element_offset`, [14](#)
  - `endian`, [13](#)
  - `Jungfrau`, [13](#)
  - `little`, [14](#)
  - `load`, [14](#)
  - `make_array`, [14](#)
  - `make_shape`, [15](#)
  - `Moench`, [13](#)
  - `Mythen3`, [13](#)
  - `native`, [14](#)
  - `save`, [15](#)
  - `Shape`, [13](#)
  - `shape_t`, [13](#)
  - `StringTo`, [15](#)
  - `TimingMode`, [14](#)
  - `toString`, [16](#)
  - `Trigger`, [14](#)
- `aare.File`, [16](#)
- `aare.File.File`, [43](#)
  - `__getattrattribute__`, [44](#)
  - `__init__`, [44](#)
  - `_file`, [44](#)
  - `path`, [44](#)
- `aare.Frame`, [16](#)
- `aare.Frame.Frame`, [57](#)
  - `__getattrattribute__`, [58](#)
  - `__init__`, [57](#)
  - `_frameImpl`, [58](#)
- `aare::CircularFifo< ItemType >`, [29](#)
  - `~CircularFifo`, [30](#)
  - `CircularFifo`, [30](#)
  - `fifo_size`, [32](#)
  - `filled_slots`, [32](#)
  - `free_slots`, [32](#)
  - `frontPtr`, [30](#)
  - `isFull`, [30](#)
  - `next`, [30](#)
  - `numFilledSlots`, [30](#)
  - `numFreeSlots`, [31](#)
  - `pop_free`, [31](#)
  - `pop_value`, [31](#)
  - `push_free`, [31](#)
  - `push_value`, [31](#)
  - `try_pop_free`, [31](#)
  - `try_push_free`, [31](#)
  - `try_push_value`, [32](#)
  - `value_type`, [30](#)
- `aare::ClusterFinder< T >`, [32](#)
  - `add_link`, [34](#)
  - `binary_`, [36](#)
  - `check_neighbours`, [34](#)
  - `child`, [36](#)
  - `clear_hits`, [34](#)
  - `ClusterFinder`, [33](#)
  - `current_label`, [36](#)
  - `di`, [36](#)
  - `di_`, [36](#)
  - `dj`, [36](#)
  - `dj_`, [36](#)

- find\_clusters, 34
- find\_clusters\_X, 34
- first\_pass, 34
- h\_size, 36
- hits, 37
- labeled, 34
- labeled\_, 37
- noiseMap, 37
- original\_, 37
- peripheral\_labeled\_, 37
- peripheralThresholdFactor\_, 37
- print\_connections, 34
- rec\_FillHit, 35
- second\_pass, 35
- set\_noiseMap, 35
- set\_peripheralThresholdFactor, 35
- shape\_, 37
- single\_pass, 35
- steal\_hits, 35
- store\_clusters, 35
- threshold\_, 37
- total\_clusters, 35
- use\_noise\_map, 37
- aare::ClusterFinder< T >::Hit, 58
  - col, 58
  - cols, 58
  - energy, 59
  - enes, 59
  - max, 59
  - reserved, 59
  - row, 59
  - rows, 59
  - size, 59
- aare::DType, 38
  - bitdepth, 39
  - DOUBLE, 38
  - DType, 39
  - ERROR, 38
  - FLOAT, 38
  - INT16, 38
  - INT32, 38
  - INT64, 38
  - INT8, 38
  - m\_type, 40
  - operator!=, 39
  - operator==, 39
  - str, 39
  - TypeIndex, 38
  - UINT16, 38
  - UINT32, 38
  - UINT64, 38
  - UINT8, 38
- aare::File, 40
  - ~File, 41
  - bitdepth, 42
  - bytes\_per\_frame, 42
  - cols, 42
  - File, 41
  - file\_impl, 43
  - frame\_number, 42
  - iread, 42
  - pixels, 42
  - read, 42
  - read\_into, 42
  - rows, 43
  - seek, 43
  - tell, 43
  - total\_frames, 43
  - write, 43
- aare::FileConfig, 45
  - cols, 45
  - dtype, 45
  - geometry, 45
  - operator!=, 45
  - operator==, 45
  - rows, 46
- aare::FileInterface, 46
  - ~FileInterface, 47
  - bitdepth, 48
  - bytes\_per\_frame, 48
  - cols, 48
  - current\_frame, 53
  - frame\_number, 48
  - iread, 49
  - m\_base\_name, 53
  - m\_base\_path, 53
  - m\_bitdepth, 53
  - m\_cols, 53
  - m\_ext, 53
  - m\_findex, 53
  - m\_fframe, 53
  - m\_mode, 53
  - m\_rows, 53
  - m\_total\_frames, 54
  - m\_type, 54
  - max\_frames\_per\_file, 54
  - pixels, 49
  - read, 49, 50
  - read\_into, 50, 51
  - rows, 51
  - seek, 51
  - tell, 52
  - total\_frames, 52
  - version, 54
  - write, 52
- aare::Frame, 54
  - ~Frame, 55
  - bitdepth, 55
  - cols, 55
  - data, 56
  - Frame, 55
  - get, 56
  - image, 56
  - m\_bitdepth, 57
  - m\_cols, 57
  - m\_data, 57

- m\_rows, 57
- operator=, 56
- rows, 56
- set, 56
- size, 56
- view, 56
- aare::logger, 16
  - DEBUG, 17
  - debug, 17
  - ERROR, 17
  - error, 17
  - get\_logger\_instance, 18
  - INFO, 17
  - info, 18
  - log, 18
  - LOGGING\_LEVEL, 17
  - set\_output\_file, 18
  - set\_streams, 18
  - set\_verbosity, 18
  - warn, 18
  - WARNING, 17
- aare::logger::internal, 19
  - logger\_instance, 19
- aare::logger::Logger, 60
  - ~Logger, 60
  - debug, 61
  - error, 61
  - error\_buf, 62
  - error\_output, 62
  - info, 61
  - log, 61
  - log\_, 61
  - Logger, 60
  - out\_file, 62
  - set\_output\_file, 61
  - set\_streams, 61, 62
  - set\_verbosity, 62
  - standard\_buf, 62
  - standard\_output, 62
  - VERBOSITY\_LEVEL, 62
  - warn, 62
- aare::NDArray< T, Ndim >, 64
  - ~NDArray, 67
  - begin, 67
  - buffer, 67
  - byte\_strides, 67
  - data, 67
  - data\_, 73
  - end, 67
  - NDArray, 66
  - operator!=, 67
  - operator>, 71
  - operator(), 68
  - operator+, 69
  - operator++, 69
  - operator+=, 69
  - operator-, 69
  - operator-=, 70
  - operator/, 70
  - operator/=: 70
  - operator=, 70, 71
  - operator==, 71
  - operator&=, 67
  - operator\*, 68
  - operator\*=, 68, 69
  - Print, 71
  - Print\_all, 71
  - Print\_some, 71
  - reset, 71
  - shape, 72
  - shape\_, 73
  - size, 72
  - size\_, 73
  - span, 72
  - sqrt, 72
  - strides, 72
  - strides\_, 73
  - total\_bytes, 72
  - value, 72
  - value\_type, 66
- aare::NDView< T, Ndim >, 73
  - begin, 75
  - buffer\_, 78
  - data, 75
  - elemenwise, 75
  - end, 76
  - NDView, 74, 75
  - operator(), 76
  - operator+=, 76
  - operator-=, 76
  - operator/=: 77
  - operator=, 77
  - operator==, 77
  - operator[], 77
  - operator\*=, 76
  - shape, 77
  - shape\_, 78
  - size, 78
  - size\_, 78
  - strides\_, 78
- aare::network\_io, 19
- aare::network\_io::NetworkError, 78
  - m\_msg, 79
  - NetworkError, 79
  - what, 79
- aare::NumpyFile, 80
  - ~NumpyFile, 82
  - bitdepth, 82
  - bytes\_per\_frame, 82
  - cols, 82
  - current\_frame, 88
  - dtype, 83
  - fp, 88
  - frame\_number, 83
  - get\_frame, 83
  - get\_frame\_into, 83

- header\_len, 88
- header\_len\_size, 88
- header\_size, 88
- initial\_header\_len, 88
- iread, 83, 84
- load, 84
- load\_metadata, 84
- m\_base\_name, 88
- m\_base\_path, 88
- m\_bitdepth, 89
- m\_cols, 89
- m\_ext, 89
- m\_findex, 89
- m\_fname, 89
- m\_header, 89
- m\_mode, 89
- m\_rows, 89
- m\_total\_frames, 89
- m\_type, 89
- major\_ver\_, 90
- max\_frames\_per\_file, 90
- minor\_ver\_, 90
- NumpyFile, 82
- pixels, 85
- read, 85
- read\_into, 86
- rows, 86
- seek, 86
- shape, 87
- tell, 87
- total\_frames, 87
- version, 90
- write, 87
- aare::NumpyHeader, 90
  - dtype, 91
  - fortran\_order, 91
  - shape, 91
  - to\_string, 90
- aare::NumpyHelpers, 19
  - get\_value\_from\_map, 20
  - in\_array, 20
  - is\_digits, 20
  - magic\_str, 22
  - magic\_string\_length, 22
  - parse\_bool, 20
  - parse\_descr, 20
  - parse\_dict, 20
  - parse\_str, 20
  - parse\_tuple, 20
  - trim, 21
  - write\_boolean, 21
  - write\_header, 21
  - write\_header\_dict, 21
  - write\_magic, 21
  - write\_tuple, 21
- aare::RawFile, 94
  - ~RawFile, 97
  - bitdepth, 97
  - bytes\_per\_frame, 97
  - cfg, 105
  - cols, 98
  - current\_frame, 105
  - data\_fname, 98
  - find\_geometry, 98
  - find\_number\_of\_subfiles, 98
  - frame\_number, 99
  - geometry, 105
  - get\_frame, 99
  - get\_frame\_into, 99
  - iread, 100
  - is\_master\_file, 100
  - m\_base\_name, 105
  - m\_base\_path, 105
  - m\_bitdepth, 105
  - m\_cols, 105
  - m\_ext, 105
  - m\_findex, 105
  - m\_fname, 106
  - m\_mode, 106
  - m\_rows, 106
  - m\_total\_frames, 106
  - m\_type, 106
  - master\_fname, 101
  - max\_frames\_per\_file, 106
  - n\_subfile\_parts, 106
  - n\_subfiles, 106
  - open\_subfiles, 101
  - parse\_fname, 101
  - parse\_json\_metadata, 101
  - parse\_metadata, 101
  - parse\_raw\_metadata, 101
  - pixels, 101
  - positions, 106
  - quad, 106
  - RawFile, 97
  - read, 101, 102
  - read\_header, 102
  - read\_into, 102, 103
  - rows, 103
  - seek, 103
  - set\_config, 104
  - subfile\_cols, 107
  - subfile\_rows, 107
  - subfiles, 107
  - tell, 104
  - timing\_mode, 107
  - total\_frames, 104
  - version, 107
  - write, 104
- aare::RawFileConfig, 107
  - module\_gap\_col, 108
  - module\_gap\_row, 108
  - operator==, 108
- aare::sls\_detector\_header, 108
  - bunchId, 108
  - column, 108



- debug, [109](#)
- detType, [109](#)
- expLength, [109](#)
- frameNumber, [109](#)
- modId, [109](#)
- packetMask, [109](#)
- packetNumber, [109](#)
- reserved, [109](#)
- roundRNumber, [109](#)
- row, [109](#)
- timestamp, [110](#)
- version, [110](#)
- aare::SubFile, [110](#)
  - bytes\_per\_part, [112](#)
  - fp, [114](#)
  - frame\_number, [112](#)
  - get\_part, [112](#)
  - m\_bitdepth, [114](#)
  - m\_cols, [114](#)
  - m\_fname, [114](#)
  - m\_rows, [114](#)
  - m\_sub\_file\_index\_, [114](#)
  - n\_frames, [114](#)
  - pfunc, [111](#)
  - pixels\_per\_part, [112](#)
  - read\_impl, [114](#)
  - read\_impl\_flip, [112](#)
  - read\_impl\_map, [114](#)
  - read\_impl\_normal, [113](#)
  - read\_impl\_reorder, [113](#)
  - SubFile, [111](#)
- aare::xy, [115](#)
  - col, [116](#)
  - operator!=, [115](#)
  - operator==, [115](#)
  - row, [116](#)
- aare::ZmqFrame, [116](#)
  - frame, [116](#)
  - header, [116](#)
- aare::ZmqHeader, [117](#)
  - acqIndex, [118](#)
  - addJsonHeader, [118](#)
  - column, [118](#)
  - completeImage, [118](#)
  - data, [118](#)
  - detSpec1, [119](#)
  - detSpec2, [119](#)
  - detSpec3, [119](#)
  - detSpec4, [119](#)
  - detType, [119](#)
  - dynamicRange, [119](#)
  - expLength, [119](#)
  - fileIndex, [119](#)
  - flipRows, [119](#)
  - fname, [119](#)
  - frameIndex, [120](#)
  - frameNumber, [120](#)
  - from\_string, [118](#)
  - imageSize, [120](#)
  - jsonversion, [120](#)
  - modId, [120](#)
  - ndetx, [120](#)
  - ndety, [120](#)
  - npixelsx, [120](#)
  - npixelsy, [121](#)
  - operator==, [118](#)
  - packetNumber, [121](#)
  - progress, [121](#)
  - quad, [121](#)
  - row, [121](#)
  - rx\_roi, [121](#)
  - timestamp, [121](#)
  - to\_string, [118](#)
  - version, [121](#)
- aare::ZmqSocket, [122](#)
  - ~ZmqSocket, [123](#)
  - disconnect, [123](#)
  - m\_context, [124](#)
  - m\_endpoint, [124](#)
  - m\_header\_buffer, [124](#)
  - m\_max\_header\_size, [124](#)
  - m\_potential\_frame\_size, [124](#)
  - m\_socket, [124](#)
  - m\_timeout\_ms, [124](#)
  - m\_zmq\_hwm, [124](#)
  - operator=, [123](#)
  - set\_potential\_frame\_size, [123](#)
  - set\_timeout\_ms, [123](#)
  - set\_zmq\_hwm, [123](#)
  - ZmqSocket, [123](#)
- aare::ZmqSocketReceiver, [125](#)
  - connect, [126](#)
  - disconnect, [126](#)
  - m\_context, [127](#)
  - m\_endpoint, [127](#)
  - m\_header\_buffer, [127](#)
  - m\_max\_header\_size, [127](#)
  - m\_potential\_frame\_size, [127](#)
  - m\_socket, [128](#)
  - m\_timeout\_ms, [128](#)
  - m\_zmq\_hwm, [128](#)
  - receive\_data, [126](#)
  - receive\_header, [126](#)
  - receive\_n, [126](#)
  - receive\_zmqframe, [127](#)
  - set\_potential\_frame\_size, [127](#)
  - set\_timeout\_ms, [127](#)
  - set\_zmq\_hwm, [127](#)
  - ZmqSocketReceiver, [126](#)
- aare::ZmqSocketSender, [128](#)
  - bind, [129](#)
  - disconnect, [129](#)
  - m\_context, [131](#)
  - m\_endpoint, [131](#)
  - m\_header\_buffer, [131](#)
  - m\_max\_header\_size, [131](#)

- m\_potential\_frame\_size, [131](#)
  - m\_socket, [131](#)
  - m\_timeout\_ms, [131](#)
  - m\_zmq\_hwm, [131](#)
  - send, [129](#), [130](#)
  - set\_potential\_frame\_size, [130](#)
  - set\_timeout\_ms, [130](#)
  - set\_zmq\_hwm, [131](#)
  - ZmqSocketSender, [129](#)
- AARE\_ROOT\_DIR\_VAR
  - json\_example.cpp, [167](#)
  - multiport\_example.cpp, [168](#)
  - mythen\_example.cpp, [169](#)
  - numpy\_read\_example.cpp, [170](#)
  - numpy\_write\_example.cpp, [171](#)
  - raw\_example.cpp, [171](#)
- acqIndex
  - aare::ZmqHeader, [118](#)
- add\_link
  - aare::ClusterFinder< T >, [34](#)
- addJsonHeader
  - aare::ZmqHeader, [118](#)
- arr
  - example.read\_frame, [22](#)
  - write\_test\_files, [28](#)
- arr2
  - write\_test\_files, [28](#)
- AtomicIndex
  - folly::ProducerConsumerQueue< T >, [92](#)
- Auto
  - aare, [14](#)
- axis
  - read\_multiport, [26](#)
- begin
  - aare::NDArray< T, Ndim >, [67](#)
  - aare::NDView< T, Ndim >, [75](#)
- big
  - aare, [14](#)
- binary\_
  - aare::ClusterFinder< T >, [36](#)
- bind
  - aare::ZmqSocketSender, [129](#)
- bindings.cpp
  - PYBIND11\_MODULE, [195](#)
- bitdepth
  - aare::DType, [39](#)
  - aare::File, [42](#)
  - aare::FileInterface, [48](#)
  - aare::Frame, [55](#)
  - aare::NumpyFile, [82](#)
  - aare::RawFile, [97](#)
- buffer
  - aare::NDArray< T, Ndim >, [67](#)
- buffer\_
  - aare::NDView< T, Ndim >, [78](#)
- bunchId
  - aare::sls\_detector\_header, [108](#)
- byte\_strides
  - aare::NDArray< T, Ndim >, [67](#)
- bytes\_per\_frame
  - aare::File, [42](#)
  - aare::FileInterface, [48](#)
  - aare::NumpyFile, [82](#)
  - aare::RawFile, [97](#)
- bytes\_per\_part
  - aare::SubFile, [112](#)
- c\_strides
  - aare, [14](#)
- capacity
  - folly::ProducerConsumerQueue< T >, [92](#)
- cfg
  - aare::RawFile, [105](#)
- check\_neighbours
  - aare::ClusterFinder< T >, [34](#)
- child
  - aare::ClusterFinder< T >, [36](#)
- ChipTestBoard
  - aare, [13](#)
- CircularFifo
  - aare::CircularFifo< ItemType >, [30](#)
- CircularFifo.test.cpp
  - TEST\_CASE, [157](#)
- clear\_hits
  - aare::ClusterFinder< T >, [34](#)
- ClusterFinder
  - aare::ClusterFinder< T >, [33](#)
- col
  - aare::ClusterFinder< T >::Hit, [58](#)
  - aare::xy, [116](#)
- cols
  - aare::ClusterFinder< T >::Hit, [58](#)
  - aare::File, [42](#)
  - aare::FileConfig, [45](#)
  - aare::FileInterface, [48](#)
  - aare::Frame, [55](#)
  - aare::NumpyFile, [82](#)
  - aare::RawFile, [98](#)
  - read\_frame, [24](#)
- column
  - aare::sls\_detector\_header, [108](#)
  - aare::ZmqHeader, [118](#)
- completeImage
  - aare::ZmqHeader, [118](#)
- connect
  - aare::ZmqSocketReceiver, [126](#)
- core/include/aare/core/CircularFifo.hpp, [133](#)
- core/include/aare/core/defs.hpp, [135](#), [136](#)
- core/include/aare/core/DType.hpp, [137](#), [138](#)
- core/include/aare/core/Frame.hpp, [138](#), [139](#)
- core/include/aare/core/NDArray.hpp, [140](#), [141](#)
- core/include/aare/core/NDView.hpp, [145](#), [146](#)
- core/include/aare/core/ProducerConsumerQueue.hpp, [148](#), [149](#)
- core/include/aare/core/VariableSizeClusterFinder.hpp, [151](#), [152](#)
- core/src/defs.cpp, [155](#)

- core/src/DType.cpp, 156
- core/src/Frame.cpp, 156
- core/test/CircularFifo.test.cpp, 156
- core/test/defs.test.cpp, 158
- core/test/DType.test.cpp, 158
- core/test/Frame.test.cpp, 159
- core/test/NDArray.test.cpp, 160
- core/test/NDView.test.cpp, 162
- core/test/ProducerConsumerQueue.test.cpp, 163
- core/test/wrappers.test.cpp, 164
- count
  - read\_frame, 24
  - read\_multiport, 26
- current\_frame
  - aare::FileInterface, 53
  - aare::NumpyFile, 88
  - aare::RawFile, 105
- current\_label
  - aare::ClusterFinder< T >, 36
- data
  - aare::Frame, 56
  - aare::NDArray< T, Ndim >, 67
  - aare::NDView< T, Ndim >, 75
  - aare::ZmqHeader, 118
  - read\_frame, 24
  - read\_multiport, 26
- data/jungfrau/read\_frame.py, 165
- data/jungfrau/read\_multiport.py, 166
- data/numpy/write\_test\_files.py, 166
- data/scripts/read\_first\_frame\_number.py, 166
- data/scripts/read\_multiport.py, 167
- data\_
  - aare::NDArray< T, Ndim >, 73
- data\_fname
  - aare::RawFile, 98
- data\_path
  - example.read\_frame, 22
- DataTypeVariants
  - aare, 13
- DEBUG
  - aare::logger, 17
- debug
  - aare::logger, 17
  - aare::logger::Logger, 61
  - aare::sls\_detector\_header, 109
- defs.test.cpp
  - TEST\_CASE, 158
- DetectorType
  - aare, 13
- detSpec1
  - aare::ZmqHeader, 119
- detSpec2
  - aare::ZmqHeader, 119
- detSpec3
  - aare::ZmqHeader, 119
- detSpec4
  - aare::ZmqHeader, 119
- detType
  - aare::sls\_detector\_header, 109
  - aare::ZmqHeader, 119
- di
  - aare::ClusterFinder< T >, 36
- di\_
  - aare::ClusterFinder< T >, 36
- disconnect
  - aare::ZmqSocket, 123
  - aare::ZmqSocketReceiver, 126
  - aare::ZmqSocketSender, 129
- dj
  - aare::ClusterFinder< T >, 36
- dj\_
  - aare::ClusterFinder< T >, 36
- DOUBLE
  - aare::DType, 38
- DType
  - aare::DType, 39
- dtype
  - aare::FileConfig, 45
  - aare::NumpyFile, 83
  - aare::NumpyHeader, 91
  - read\_frame, 24
  - read\_multiport, 26
- DType.test.cpp
  - TEST\_CASE, 158, 159
- dynamic\_shape
  - aare, 13
- dynamicRange
  - aare::ZmqHeader, 119
- Eiger
  - aare, 13
- element\_offset
  - aare, 14
- elemenwise
  - aare::NDView< T, Ndim >, 75
- end
  - aare::NDArray< T, Ndim >, 67
  - aare::NDView< T, Ndim >, 76
- endian
  - aare, 13
- energy
  - aare::ClusterFinder< T >::Hit, 59
- enes
  - aare::ClusterFinder< T >::Hit, 59
- ERROR
  - aare::DType, 38
  - aare::logger, 17
- error
  - aare::logger, 17
  - aare::logger::Logger, 61
- error\_buf
  - aare::logger::Logger, 62
- error\_output
  - aare::logger::Logger, 62
- example, 22
- example.read\_frame, 22
- arr, 22

- data\_path, [22](#)
- file, [22](#)
- frame, [23](#)
- root\_dir, [23](#)
- examples/json\_example.cpp, [167](#)
- examples/logger\_example.cpp, [167](#)
- examples/multiport\_example.cpp, [168](#)
- examples/mythen\_example.cpp, [169](#)
- examples/numpy\_read\_example.cpp, [169](#)
- examples/numpy\_write\_example.cpp, [170](#)
- examples/raw\_example.cpp, [171](#)
- examples/zmq\_receiver\_example.cpp, [172](#)
- examples/zmq\_restream\_example.cpp, [172](#)
- examples/zmq\_sender\_example.cpp, [173](#)
- expLength
  - aare::sls\_detector\_header, [109](#)
  - aare::ZmqHeader, [119](#)
- f
  - read\_frame, [24](#)
  - read\_multiport, [26](#)
- fifo\_size
  - aare::CircularFifo< ItemType >, [32](#)
- File
  - aare::File, [41](#)
- file
  - example.read\_frame, [22](#)
- file\_impl
  - aare::File, [43](#)
- file\_io/include/aare/file\_io/File.hpp, [173](#), [174](#)
- file\_io/include/aare/file\_io/FileInterface.hpp, [174](#), [175](#)
- file\_io/include/aare/file\_io/NumpyFile.hpp, [176](#)
- file\_io/include/aare/file\_io/NumpyHelpers.hpp, [177](#), [178](#)
- file\_io/include/aare/file\_io/RawFile.hpp, [179](#)
- file\_io/include/aare/file\_io/SubFile.hpp, [180](#), [181](#)
- file\_io/src/File.cpp, [181](#)
- file\_io/src/NumpyFile.cpp, [182](#)
- file\_io/src/NumpyHelpers.cpp, [182](#)
- file\_io/src/RawFile.cpp, [183](#)
- file\_io/src/SubFile.cpp, [183](#)
- file\_io/test/NumpyFile.test.cpp, [183](#)
- file\_io/test/NumpyHelpers.test.cpp, [184](#)
- file\_io/test/RawFile.test.cpp, [185](#)
- file\_name
  - read\_frame, [24](#)
  - read\_multiport, [26](#)
- fileIndex
  - aare::ZmqHeader, [119](#)
- filled\_slots
  - aare::CircularFifo< ItemType >, [32](#)
- find\_clusters
  - aare::ClusterFinder< T >, [34](#)
- find\_clusters\_X
  - aare::ClusterFinder< T >, [34](#)
- find\_geometry
  - aare::RawFile, [98](#)
- find\_number\_of\_subfiles
  - aare::RawFile, [98](#)
- first\_pass
  - aare::ClusterFinder< T >, [34](#)
- flipRows
  - aare::ZmqHeader, [119](#)
- FLOAT
  - aare::DType, [38](#)
- fname
  - aare::ZmqHeader, [119](#)
- folly, [23](#)
- folly::ProducerConsumerQueue< T >, [91](#)
  - ~ProducerConsumerQueue, [92](#)
  - AtomicIndex, [92](#)
  - capacity, [92](#)
  - frontPtr, [92](#)
  - isEmpty, [93](#)
  - isFull, [93](#)
  - operator=, [93](#)
  - pad0\_, [94](#)
  - pad1\_, [94](#)
  - popFront, [93](#)
  - ProducerConsumerQueue, [92](#)
  - read, [93](#)
  - readIndex\_, [94](#)
  - records\_, [94](#)
  - size\_, [94](#)
  - sizeGuess, [93](#)
  - value\_type, [92](#)
  - write, [93](#)
  - writeIndex\_, [94](#)
- fortran\_order
  - aare::NumpyHeader, [91](#)
- fp
  - aare::NumpyFile, [88](#)
  - aare::SubFile, [114](#)
- Frame
  - aare::Frame, [55](#)
- frame
  - aare::ZmqFrame, [116](#)
  - example.read\_frame, [23](#)
- Frame.test.cpp
  - TEST\_CASE, [159](#)
- frame\_cols
  - read\_multiport, [26](#)
- frame\_number
  - aare::File, [42](#)
  - aare::FileInterface, [48](#)
  - aare::NumpyFile, [83](#)
  - aare::RawFile, [99](#)
  - aare::SubFile, [112](#)
  - read\_first\_frame\_number, [23](#)
- frame\_rows
  - read\_multiport, [26](#)
- frameIndex
  - aare::ZmqHeader, [120](#)
- frameNumber
  - aare::sls\_detector\_header, [109](#)
  - aare::ZmqHeader, [120](#)
- frames
  - read\_frame, [24](#)

- read\_multiport, 26
- free\_slots
  - aare::CircularFifo< ItemType >, 32
- from\_string
  - aare::ZmqHeader, 118
- frontPtr
  - aare::CircularFifo< ItemType >, 30
  - folly::ProducerConsumerQueue< T >, 92
- geometry
  - aare::FileConfig, 45
  - aare::RawFile, 105
- get
  - aare::Frame, 56
- get\_frame
  - aare::NumpyFile, 83
  - aare::RawFile, 99
- get\_frame\_into
  - aare::NumpyFile, 83
  - aare::RawFile, 99
- get\_logger\_instance
  - aare::logger, 18
- get\_part
  - aare::SubFile, 112
- get\_value\_from\_map
  - aare::NumpyHelpers, 20
- h\_size
  - aare::ClusterFinder< T >, 36
- hardware\_destructive\_interference\_size
  - ProducerConsumerQueue.hpp, 148
- header
  - aare::ZmqFrame, 116
  - read\_frame, 24
  - read\_multiport, 26
- header\_dt
  - read\_first\_frame\_number, 23
  - read\_frame, 25
  - read\_multiport, 26
- header\_len
  - aare::NumpyFile, 88
- header\_len\_size
  - aare::NumpyFile, 88
- header\_size
  - aare::NumpyFile, 88
- hits
  - aare::ClusterFinder< T >, 37
- image
  - aare::Frame, 56
- imageSize
  - aare::ZmqHeader, 120
- in\_array
  - aare::NumpyHelpers, 20
- include/aare/aare.hpp, 186
- INFO
  - aare::logger, 17
- info
  - aare::logger, 18
- aare::logger::Logger, 61
- initial\_header\_len
  - aare::NumpyFile, 88
- INT16
  - aare::DType, 38
- INT32
  - aare::DType, 38
- INT64
  - aare::DType, 38
- INT8
  - aare::DType, 38
- iread
  - aare::File, 42
  - aare::FileInterface, 49
  - aare::NumpyFile, 83, 84
  - aare::RawFile, 100
- is\_digits
  - aare::NumpyHelpers, 20
- is\_master\_file
  - aare::RawFile, 100
- isEmpty
  - folly::ProducerConsumerQueue< T >, 93
- isFull
  - aare::CircularFifo< ItemType >, 30
  - folly::ProducerConsumerQueue< T >, 93
- json
  - RawFile.cpp, 183
- json\_example.cpp
  - AARE\_ROOT\_DIR\_VAR, 167
  - main, 167
  - test, 167
- jsonversion
  - aare::ZmqHeader, 120
- Jungfrau
  - aare, 13
- labeled
  - aare::ClusterFinder< T >, 34
- labeled\_
  - aare::ClusterFinder< T >, 37
- little
  - aare, 14
- load
  - aare, 14
  - aare::NumpyFile, 84
- load\_metadata
  - aare::NumpyFile, 84
- LOCATION
  - logger.hpp, 197
- log
  - aare::logger, 18
  - aare::logger::Logger, 61
- log\_
  - aare::logger::Logger, 61
- Logger
  - aare::logger::Logger, 60
- logger.hpp
  - LOCATION, 197

- operator<<, 197
- logger\_example.cpp
  - main, 168
- logger\_instance
  - aare::logger::internal, 19
- LOGGING\_LEVEL
  - aare::logger, 17
- m\_base\_name
  - aare::FileInterface, 53
  - aare::NumpyFile, 88
  - aare::RawFile, 105
- m\_base\_path
  - aare::FileInterface, 53
  - aare::NumpyFile, 88
  - aare::RawFile, 105
- m\_bitdepth
  - aare::FileInterface, 53
  - aare::Frame, 57
  - aare::NumpyFile, 89
  - aare::RawFile, 105
  - aare::SubFile, 114
- m\_cols
  - aare::FileInterface, 53
  - aare::Frame, 57
  - aare::NumpyFile, 89
  - aare::RawFile, 105
  - aare::SubFile, 114
- m\_context
  - aare::ZmqSocket, 124
  - aare::ZmqSocketReceiver, 127
  - aare::ZmqSocketSender, 131
- m\_data
  - aare::Frame, 57
- m\_endpoint
  - aare::ZmqSocket, 124
  - aare::ZmqSocketReceiver, 127
  - aare::ZmqSocketSender, 131
- m\_ext
  - aare::FileInterface, 53
  - aare::NumpyFile, 89
  - aare::RawFile, 105
- m\_findex
  - aare::FileInterface, 53
  - aare::NumpyFile, 89
  - aare::RawFile, 105
- m\_fname
  - aare::FileInterface, 53
  - aare::NumpyFile, 89
  - aare::RawFile, 106
  - aare::SubFile, 114
- m\_header
  - aare::NumpyFile, 89
- m\_header\_buffer
  - aare::ZmqSocket, 124
  - aare::ZmqSocketReceiver, 127
  - aare::ZmqSocketSender, 131
- m\_max\_header\_size
  - aare::ZmqSocket, 124
- aare::ZmqSocketReceiver, 127
- aare::ZmqSocketSender, 131
- m\_mode
  - aare::FileInterface, 53
  - aare::NumpyFile, 89
  - aare::RawFile, 106
- m\_msg
  - aare::network\_io::NetworkError, 79
- m\_potential\_frame\_size
  - aare::ZmqSocket, 124
  - aare::ZmqSocketReceiver, 127
  - aare::ZmqSocketSender, 131
- m\_rows
  - aare::FileInterface, 53
  - aare::Frame, 57
  - aare::NumpyFile, 89
  - aare::RawFile, 106
  - aare::SubFile, 114
- m\_socket
  - aare::ZmqSocket, 124
  - aare::ZmqSocketReceiver, 128
  - aare::ZmqSocketSender, 131
- m\_sub\_file\_index\_
  - aare::SubFile, 114
- m\_timeout\_ms
  - aare::ZmqSocket, 124
  - aare::ZmqSocketReceiver, 128
  - aare::ZmqSocketSender, 131
- m\_total\_frames
  - aare::FileInterface, 54
  - aare::NumpyFile, 89
  - aare::RawFile, 106
- m\_type
  - aare::DType, 40
  - aare::FileInterface, 54
  - aare::NumpyFile, 89
  - aare::RawFile, 106
- m\_zmq\_hwm
  - aare::ZmqSocket, 124
  - aare::ZmqSocketReceiver, 128
  - aare::ZmqSocketSender, 131
- magic\_str
  - aare::NumpyHelpers, 22
- magic\_string\_length
  - aare::NumpyHelpers, 22
- main
  - json\_example.cpp, 167
  - logger\_example.cpp, 168
  - multiport\_example.cpp, 168
  - mythen\_example.cpp, 169
  - numpy\_read\_example.cpp, 170
  - numpy\_write\_example.cpp, 171
  - raw\_example.cpp, 171
  - zmq\_receiver\_example.cpp, 172
  - zmq\_restream\_example.cpp, 172
  - zmq\_sender\_example.cpp, 173
- major\_ver\_
  - aare::NumpyFile, 90

- make\_array
  - aare, 14
- make\_shape
  - aare, 15
- master\_fname
  - aare::RawFile, 101
- max
  - aare::ClusterFinder< T >::Hit, 59
- MAX\_CLUSTER\_SIZE
  - VariableSizeClusterFinder.hpp, 151
- max\_frames\_per\_file
  - aare::FileInterface, 54
  - aare::NumpyFile, 90
  - aare::RawFile, 106
- minor\_ver\_
  - aare::NumpyFile, 90
- modld
  - aare::sls\_detector\_header, 109
  - aare::ZmqHeader, 120
- module\_gap\_col
  - aare::RawFileConfig, 108
- module\_gap\_row
  - aare::RawFileConfig, 108
- Moench
  - aare, 13
- MoveOnlyInt, 63
  - MoveOnlyInt, 63
  - operator=, 64
  - operator==, 64
  - value, 64
- multiport\_example.cpp
  - AARE\_ROOT\_DIR\_VAR, 168
  - main, 168
  - test, 168
- Mythen3
  - aare, 13
- mythen\_example.cpp
  - AARE\_ROOT\_DIR\_VAR, 169
  - main, 169
  - test1, 169
  - test2, 169
- n\_frames
  - aare::SubFile, 114
- n\_subfile\_parts
  - aare::RawFile, 106
- n\_subfiles
  - aare::RawFile, 106
- native
  - aare, 14
- NDArray
  - aare::NDArray< T, Ndim >, 66
- NDArray.test.cpp
  - TEST\_CASE, 160, 161
- ndetx
  - aare::ZmqHeader, 120
- ndety
  - aare::ZmqHeader, 120
- NDView
  - aare::NDView< T, Ndim >, 74, 75
- NDView.test.cpp
  - TEST\_CASE, 162, 163
- network\_io/include/aare/network\_io/defs.hpp, 136, 137
- network\_io/include/aare/network\_io/ZmqHeader.hpp, 186, 187
- network\_io/include/aare/network\_io/ZmqSocket.hpp, 188, 189
- network\_io/include/aare/network\_io/ZmqSocketReceiver.hpp, 189, 190
- network\_io/include/aare/network\_io/ZmqSocketSender.hpp, 190, 191
- network\_io/src/ZmqHeader.cpp, 191
- network\_io/src/ZmqSocket.cpp, 192
- network\_io/src/ZmqSocketReceiver.cpp, 193
- network\_io/src/ZmqSocketSender.cpp, 193
- network\_io/test/ZmqHeader.test.cpp, 193
- NetworkError
  - aare::network\_io::NetworkError, 79
- next
  - aare::CircularFifo< ItemType >, 30
- noiseMap
  - aare::ClusterFinder< T >, 37
- npixelsx
  - aare::ZmqHeader, 120
- npixelsy
  - aare::ZmqHeader, 121
- numFilledSlots
  - aare::CircularFifo< ItemType >, 30
- numFreeSlots
  - aare::CircularFifo< ItemType >, 31
- numpy\_read\_example.cpp
  - AARE\_ROOT\_DIR\_VAR, 170
  - main, 170
  - test, 170
- numpy\_write\_example.cpp
  - AARE\_ROOT\_DIR\_VAR, 171
  - main, 171
- NumpyFile
  - aare::NumpyFile, 82
- NumpyFile.test.cpp
  - TEST\_CASE, 184
- NumpyHelpers.test.cpp
  - TEST\_CASE, 184, 185
- open\_subfiles
  - aare::RawFile, 101
- operator!=
  - aare::DType, 39
  - aare::FileConfig, 45
  - aare::NDArray< T, Ndim >, 67
  - aare::xy, 115
- operator<<
  - logger.hpp, 197
- operator>
  - aare::NDArray< T, Ndim >, 71
- operator()
  - aare::NDArray< T, Ndim >, 68
  - aare::NDView< T, Ndim >, 76

- operator+
  - aare::NDArray< T, Ndim >, 69
- operator++
  - aare::NDArray< T, Ndim >, 69
- operator+=
  - aare::NDArray< T, Ndim >, 69
  - aare::NDView< T, Ndim >, 76
- operator-
  - aare::NDArray< T, Ndim >, 69
- operator-=
  - aare::NDArray< T, Ndim >, 70
  - aare::NDView< T, Ndim >, 76
- operator/
  - aare::NDArray< T, Ndim >, 70
- operator/=
  - aare::NDArray< T, Ndim >, 70
  - aare::NDView< T, Ndim >, 77
- operator=
  - aare::Frame, 56
  - aare::NDArray< T, Ndim >, 70, 71
  - aare::NDView< T, Ndim >, 77
  - aare::ZmqSocket, 123
  - folly::ProducerConsumerQueue< T >, 93
  - MoveOnlyInt, 64
- operator==
  - aare::DType, 39
  - aare::FileConfig, 45
  - aare::NDArray< T, Ndim >, 71
  - aare::NDView< T, Ndim >, 77
  - aare::RawFileConfig, 108
  - aare::xy, 115
  - aare::ZmqHeader, 118
  - MoveOnlyInt, 64
- operator&=
  - aare::NDArray< T, Ndim >, 67
- operator[]
  - aare::NDView< T, Ndim >, 77
- operator\*
  - aare::NDArray< T, Ndim >, 68
- operator\*=
  - aare::NDArray< T, Ndim >, 68, 69
  - aare::NDView< T, Ndim >, 76
- original\_
  - aare::ClusterFinder< T >, 37
- out\_file
  - aare::logger::Logger, 62
- packetMask
  - aare::sls\_detector\_header, 109
- packetNumber
  - aare::sls\_detector\_header, 109
  - aare::ZmqHeader, 121
- pad0\_
  - folly::ProducerConsumerQueue< T >, 94
- pad1\_
  - folly::ProducerConsumerQueue< T >, 94
- parse\_bool
  - aare::NumpyHelpers, 20
- parse\_descr
  - aare::NumpyHelpers, 20
- parse\_dict
  - aare::NumpyHelpers, 20
- parse\_fname
  - aare::RawFile, 101
- parse\_json\_metadata
  - aare::RawFile, 101
- parse\_metadata
  - aare::RawFile, 101
- parse\_raw\_metadata
  - aare::RawFile, 101
- parse\_str
  - aare::NumpyHelpers, 20
- parse\_tuple
  - aare::NumpyHelpers, 20
- part\_cols
  - read\_multiport, 27
- part\_rows
  - read\_multiport, 27
- parts
  - read\_multiport, 27
- parts\_data
  - read\_multiport, 27
- path
  - aare.File.File, 44
- peripheral\_labeled\_
  - aare::ClusterFinder< T >, 37
- peripheralThresholdFactor\_
  - aare::ClusterFinder< T >, 37
- pfunc
  - aare::SubFile, 111
- pixels
  - aare::File, 42
  - aare::FileInterface, 49
  - aare::NumpyFile, 85
  - aare::RawFile, 101
- pixels\_per\_part
  - aare::SubFile, 112
- pop\_free
  - aare::CircularFifo< ItemType >, 31
- pop\_value
  - aare::CircularFifo< ItemType >, 31
- popFront
  - folly::ProducerConsumerQueue< T >, 93
- positions
  - aare::RawFile, 106
- Print
  - aare::NDArray< T, Ndim >, 71
- Print\_all
  - aare::NDArray< T, Ndim >, 71
- print\_connections
  - aare::ClusterFinder< T >, 34
- Print\_some
  - aare::NDArray< T, Ndim >, 71
- ProducerConsumerQueue
  - folly::ProducerConsumerQueue< T >, 92
- ProducerConsumerQueue.hpp
  - hardware\_destructive\_interference\_size, 148



ProducerConsumerQueue.test.cpp  
     TEST\_CASE, 164  
 progress  
     aare::ZmqHeader, 121  
 push\_free  
     aare::CircularFifo< ItemType >, 31  
 push\_value  
     aare::CircularFifo< ItemType >, 31  
 PYBIND11\_MODULE  
     bindings.cpp, 195  
 python/aare/\_\_init\_\_.py, 194  
 python/aare/File.py, 194  
 python/aare/Frame.py, 194  
 python/example/\_\_init\_\_.py, 194  
 python/example/read\_frame.py, 165  
 python/src/bindings.cpp, 194  
  
 quad  
     aare::RawFile, 106  
     aare::ZmqHeader, 121  
  
 raw\_example.cpp  
     AARE\_ROOT\_DIR\_VAR, 171  
     main, 171  
     test, 171  
 RawFile  
     aare::RawFile, 97  
 RawFile.cpp  
     json, 183  
 RawFile.test.cpp  
     TEST\_CASE, 186  
 read  
     aare::File, 42  
     aare::FileInterface, 49, 50  
     aare::NumpyFile, 85  
     aare::RawFile, 101, 102  
     folly::ProducerConsumerQueue< T >, 93  
 read\_first\_frame\_number, 23  
     frame\_number, 23  
     header\_dt, 23  
 read\_frame, 24  
     cols, 24  
     count, 24  
     data, 24  
     dtype, 24  
     f, 24  
     file\_name, 24  
     frames, 24  
     header, 24  
     header\_dt, 25  
     rows, 25  
     uint16, 25  
 read\_header  
     aare::RawFile, 102  
 read\_impl  
     aare::SubFile, 114  
 read\_impl\_flip  
     aare::SubFile, 112  
 read\_impl\_map  
     aare::SubFile, 114  
 read\_impl\_normal  
     aare::SubFile, 113  
 read\_impl\_reorder  
     aare::SubFile, 113  
 read\_into  
     aare::File, 42  
     aare::FileInterface, 50, 51  
     aare::NumpyFile, 86  
     aare::RawFile, 102, 103  
 read\_multiport, 25  
     axis, 26  
     count, 26  
     data, 26  
     dtype, 26  
     f, 26  
     file\_name, 26  
     frame\_cols, 26  
     frame\_rows, 26  
     frames, 26  
     header, 26  
     header\_dt, 26  
     part\_cols, 27  
     part\_rows, 27  
     parts, 27  
     parts\_data, 27  
     uint16, 27  
 readIndex\_  
     folly::ProducerConsumerQueue< T >, 94  
 README.md, 195  
 rec\_FillHit  
     aare::ClusterFinder< T >, 35  
 receive\_data  
     aare::ZmqSocketReceiver, 126  
 receive\_header  
     aare::ZmqSocketReceiver, 126  
 receive\_n  
     aare::ZmqSocketReceiver, 126  
 receive\_zmqframe  
     aare::ZmqSocketReceiver, 127  
 records\_  
     folly::ProducerConsumerQueue< T >, 94  
 reserved  
     aare::ClusterFinder< T >::Hit, 59  
     aare::sls\_detector\_header, 109  
 reset  
     aare::NDArray< T, Ndim >, 71  
 root\_dir  
     example.read\_frame, 23  
 roundRNumber  
     aare::sls\_detector\_header, 109  
 row  
     aare::ClusterFinder< T >::Hit, 59  
     aare::sls\_detector\_header, 109  
     aare::xy, 116  
     aare::ZmqHeader, 121  
 rows  
     aare::ClusterFinder< T >::Hit, 59

- aare::File, [43](#)
- aare::FileConfig, [46](#)
- aare::FileInterface, [51](#)
- aare::Frame, [56](#)
- aare::NumpyFile, [86](#)
- aare::RawFile, [103](#)
- read\_frame, [25](#)
- rx\_roi
  - aare::ZmqHeader, [121](#)
- save
  - aare, [15](#)
- second\_pass
  - aare::ClusterFinder< T >, [35](#)
- seek
  - aare::File, [43](#)
  - aare::FileInterface, [51](#)
  - aare::NumpyFile, [86](#)
  - aare::RawFile, [103](#)
- send
  - aare::ZmqSocketSender, [129](#), [130](#)
- set
  - aare::Frame, [56](#)
- set\_config
  - aare::RawFile, [104](#)
- set\_noiseMap
  - aare::ClusterFinder< T >, [35](#)
- set\_output\_file
  - aare::logger, [18](#)
  - aare::logger::Logger, [61](#)
- set\_peripheralThresholdFactor
  - aare::ClusterFinder< T >, [35](#)
- set\_potential\_frame\_size
  - aare::ZmqSocket, [123](#)
  - aare::ZmqSocketReceiver, [127](#)
  - aare::ZmqSocketSender, [130](#)
- set\_streams
  - aare::logger, [18](#)
  - aare::logger::Logger, [61](#), [62](#)
- set\_timeout\_ms
  - aare::ZmqSocket, [123](#)
  - aare::ZmqSocketReceiver, [127](#)
  - aare::ZmqSocketSender, [130](#)
- set\_verbosity
  - aare::logger, [18](#)
  - aare::logger::Logger, [62](#)
- set\_zmq\_hwm
  - aare::ZmqSocket, [123](#)
  - aare::ZmqSocketReceiver, [127](#)
  - aare::ZmqSocketSender, [131](#)
- Shape
  - aare, [13](#)
- shape
  - aare::NDArray< T, Ndim >, [72](#)
  - aare::NDView< T, Ndim >, [77](#)
  - aare::NumpyFile, [87](#)
  - aare::NumpyHeader, [91](#)
- shape\_
  - aare::ClusterFinder< T >, [37](#)
- aare::NDArray< T, Ndim >, [73](#)
- aare::NDView< T, Ndim >, [78](#)
- shape\_t
  - aare, [13](#)
- simdjson, [27](#)
- single\_pass
  - aare::ClusterFinder< T >, [35](#)
- size
  - aare::ClusterFinder< T >::Hit, [59](#)
  - aare::Frame, [56](#)
  - aare::NDArray< T, Ndim >, [72](#)
  - aare::NDView< T, Ndim >, [78](#)
- size\_
  - aare::NDArray< T, Ndim >, [73](#)
  - aare::NDView< T, Ndim >, [78](#)
  - folly::ProducerConsumerQueue< T >, [94](#)
- sizeGuess
  - folly::ProducerConsumerQueue< T >, [93](#)
- span
  - aare::NDArray< T, Ndim >, [72](#)
- sqrt
  - aare::NDArray< T, Ndim >, [72](#)
- standard\_buf
  - aare::logger::Logger, [62](#)
- standard\_output
  - aare::logger::Logger, [62](#)
- steal\_hits
  - aare::ClusterFinder< T >, [35](#)
- store\_clusters
  - aare::ClusterFinder< T >, [35](#)
- str
  - aare::DType, [39](#)
- strides
  - aare::NDArray< T, Ndim >, [72](#)
- strides\_
  - aare::NDArray< T, Ndim >, [73](#)
  - aare::NDView< T, Ndim >, [78](#)
- StringTo
  - aare, [15](#)
- SubFile
  - aare::SubFile, [111](#)
- subfile\_cols
  - aare::RawFile, [107](#)
- subfile\_rows
  - aare::RawFile, [107](#)
- subfiles
  - aare::RawFile, [107](#)
- tell
  - aare::File, [43](#)
  - aare::FileInterface, [52](#)
  - aare::NumpyFile, [87](#)
  - aare::RawFile, [104](#)
- test
  - json\_example.cpp, [167](#)
  - multiport\_example.cpp, [168](#)
  - numpy\_read\_example.cpp, [170](#)
  - raw\_example.cpp, [171](#)
- test.cpp

- TEST\_CASE, 195
- test1
  - mythen\_example.cpp, 169
- test2
  - mythen\_example.cpp, 169
- TEST\_CASE
  - CircularFifo.test.cpp, 157
  - defs.test.cpp, 158
  - DType.test.cpp, 158, 159
  - Frame.test.cpp, 159
  - NDArray.test.cpp, 160, 161
  - NDView.test.cpp, 162, 163
  - NumpyFile.test.cpp, 184
  - NumpyHelpers.test.cpp, 184, 185
  - ProducerConsumerQueue.test.cpp, 164
  - RawFile.test.cpp, 186
  - test.cpp, 195
  - wrappers.test.cpp, 164, 165
  - ZmqHeader.test.cpp, 193
- tests/test.cpp, 195
- threshold\_
  - aare::ClusterFinder< T >, 37
- timestamp
  - aare::sls\_detector\_header, 110
  - aare::ZmqHeader, 121
- timing\_mode
  - aare::RawFile, 107
- TimingMode
  - aare, 14
- to\_string
  - aare::NumpyHeader, 90
  - aare::ZmqHeader, 118
- toString
  - aare, 16
- total\_bytes
  - aare::NDArray< T, Ndim >, 72
- total\_clusters
  - aare::ClusterFinder< T >, 35
- total\_frames
  - aare::File, 43
  - aare::FileInterface, 52
  - aare::NumpyFile, 87
  - aare::RawFile, 104
- Trigger
  - aare, 14
- trim
  - aare::NumpyHelpers, 21
- try\_pop\_free
  - aare::CircularFifo< ItemType >, 31
- try\_push\_free
  - aare::CircularFifo< ItemType >, 31
- try\_push\_value
  - aare::CircularFifo< ItemType >, 32
- TypeIndex
  - aare::DType, 38
- UINT16
  - aare::DType, 38
- uint16
  - read\_frame, 25
  - read\_multiport, 27
- UINT32
  - aare::DType, 38
- UINT64
  - aare::DType, 38
- UINT8
  - aare::DType, 38
- use\_noise\_map
  - aare::ClusterFinder< T >, 37
- utils/include/aare/utils/logger.hpp, 195, 197
- utils/src/logger.cpp, 199
- utils/test/logger.test.cpp, 200
- value
  - aare::NDArray< T, Ndim >, 72
  - MoveOnlyInt, 64
- value\_type
  - aare::CircularFifo< ItemType >, 30
  - aare::NDArray< T, Ndim >, 66
  - folly::ProducerConsumerQueue< T >, 92
- VariableSizeClusterFinder.hpp
  - MAX\_CLUSTER\_SIZE, 151
- VERBOSITY\_LEVEL
  - aare::logger::Logger, 62
- version
  - aare::FileInterface, 54
  - aare::NumpyFile, 90
  - aare::RawFile, 107
  - aare::sls\_detector\_header, 110
  - aare::ZmqHeader, 121
- view
  - aare::Frame, 56
- warn
  - aare::logger, 18
  - aare::logger::Logger, 62
- WARNING
  - aare::logger, 17
- what
  - aare::network\_io::NetworkError, 79
- wrappers.test.cpp
  - TEST\_CASE, 164, 165
- write
  - aare::File, 43
  - aare::FileInterface, 52
  - aare::NumpyFile, 87
  - aare::RawFile, 104
  - folly::ProducerConsumerQueue< T >, 93
- write\_array
  - ZmqHeader.cpp, 191
- write\_boolean
  - aare::NumpyHelpers, 21
- write\_digit
  - ZmqHeader.cpp, 191
- write\_header
  - aare::NumpyHelpers, 21
- write\_header\_dict
  - aare::NumpyHelpers, 21

- write\_magic
  - aare::NumpyHelpers, [21](#)
- write\_map
  - ZmqHeader.cpp, [192](#)
- write\_str
  - ZmqHeader.cpp, [192](#)
- write\_test\_files, [27](#)
  - arr, [28](#)
  - arr2, [28](#)
- write\_tuple
  - aare::NumpyHelpers, [21](#)
- writeIndex\_
  - folly::ProducerConsumerQueue< T >, [94](#)
  
- zmq\_receiver\_example.cpp
  - main, [172](#)
- zmq\_restream\_example.cpp
  - main, [172](#)
- zmq\_sender\_example.cpp
  - main, [173](#)
- ZmqHeader.cpp
  - write\_array, [191](#)
  - write\_digit, [191](#)
  - write\_map, [192](#)
  - write\_str, [192](#)
- ZmqHeader.test.cpp
  - TEST\_CASE, [193](#)
- ZmqSocket
  - aare::ZmqSocket, [123](#)
- ZmqSocketReceiver
  - aare::ZmqSocketReceiver, [126](#)
- ZmqSocketSender
  - aare::ZmqSocketSender, [129](#)